

A TRIDENT SCHOLAR PROJECT REPORT

NO. 506

Drone Swarming Tactics using Reinforcement Learning and Policy Optimization

by

Midshipman 1/C Elizabeth K. Gergal, USN



UNITED STATES NAVAL ACADEMY
ANNAPOLIS, MARYLAND

This document has been approved for public
release and sale; its distribution is unlimited.

USNA-1531-2

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.				
1. REPORT DATE (DD-MM-YYYY) 7/12/21		2. REPORT TYPE		3. DATES COVERED (From - To)
4. TITLE AND SUBTITLE Drone Swarming Tactics using Reinforcement Learning and Policy Optimization		5a. CONTRACT NUMBER		
		5b. GRANT NUMBER		
		5c. PROGRAM ELEMENT NUMBER		
6. AUTHOR(S) Gergal, Elizabeth K.		5d. PROJECT NUMBER		
		5e. TASK NUMBER		
		5f. WORK UNIT NUMBER		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U.S. Naval Academy Annapolis, MD 21402		10. SPONSOR/MONITOR'S ACRONYM(S)		
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) Trident Scholar Report no. 506 (2021)		
12. DISTRIBUTION / AVAILABILITY STATEMENT This document has been approved for public release; its distribution is UNLIMITED.				
13. SUPPLEMENTARY NOTES				
14. ABSTRACT This project aims to develop defensive drone swarming tactics using reinforcement learning (RL). Swarming is a military tactic where many individually operated units maneuver as one mass to attack an enemy. Defensive swarm tactics are current topics of interest for the US military as other countries and non-state actors are gaining advantages because swarm agents are usually simple, inexpensive, and easy to implement. Current work has already developed the means of flying (drones), communicating, and swarming. However, swarms do not yet have the ability to coordinate an attack against an enemy swarm. We simulated drone battles between two swarms of military fixed wings drones using pre-programmed tactics. Even when outnumbered by up to 100%, there were effective tactics that could overcome the difference in size. When used in defense of a ship, these programmed tactics, on average, allowed between 0 and 0.5 drones to pass the defense and hit the ship which outperforms the current defenses on an Arleigh Burke Class Destroyer and other researched drone swarm defenses. This research shows that it is possible to gain a tactical advantage over an enemy swarm using certain maneuvers and tactics. In order to develop even more effective tactics, we trained an "Agent" tactic using RL. RL is a branch of machine learning that allows an agent to learn an environment, train, and learn which actions that will result in success. The "Agent" tactic does not exhibit emergent behavior yet, but it does kill some enemy drones and outperform other researched RL trained drone swarm tactics.				
15. SUBJECT TERMS Drone swarming tactics, reinforcement learning, policy optimization, unmanned aerial vehicles, ship defense, military swarming				
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES 56
a. REPORT	b. ABSTRACT	c. THIS PAGE		
				19a. NAME OF RESPONSIBLE PERSON
				19b. TELEPHONE NUMBER (include area code)

U.S.N.A. --- Trident Scholar project report; no. 506 (2021)

**DRONE SWARMING TACTICS USING REINFORCEMENT
LEARNING AND POLICY OPTIMIZATION**

by

Midshipman 1/C Elizabeth K. Gergal
United States Naval Academy
Annapolis, Maryland

Certification of Adviser Approval

Professor Frederick L. Crabbe
Computer Science Department

Acceptance for the Trident Scholar Committee

Professor Maria J. Schroeder
Associate Director of Midshipman Research

Abstract

This project aims to develop defensive drone swarming tactics using reinforcement learning (RL). Swarming is a military tactic where many individually operated units maneuver as one mass to attack an enemy. Defensive swarm tactics are current topics of interest for the US military as other countries and non-state actors are gaining advantages over the US Military because swarm agents are usually simple, inexpensive, and easy to implement. Current work has already developed the means of flying (drones), communicating, and swarming. However, swarms do not yet have the ability to coordinate an attack against an enemy swarm. We simulated drone battles between two swarms of military fixed wings drones using pre-programmed tactics. Even when outnumbered by up to 100%, there were effective tactics that could overcome the difference in size. When used in defense of a ship, these programmed tactics, on average, allowed between 0 and .5 drones to pass the defense and hit the ship which outperforms the current defenses on an Arleigh Burke Class Destroyer and other researched drone swarm defenses. This research shows that it is possible to gain a tactical advantage over an enemy swarm using certain maneuvers and tactics. In order to develop even more effective tactics, we trained an “Agent” tactic using RL. RL is a branch of machine learning that allows an agent to learn an environment, train, and learn which actions that will result in success. The “Agent” tactic does not exhibit emergent behavior yet, but it does kill some enemy drones and outperform other researched RL trained drone swarm tactics. Continuing to implement RL into the development of swarm and counter-swarm tactics will aid the US in maintaining our military advantage over our enemies and protecting America’s national interests.

Keywords: Drone Swarming Tactics, Reinforcement Learning, Policy Optimization, Unmanned Aerial Vehicles, Ship Defense, Military Swarming

Acknowledgments

I would like to thank the United States Naval Academy, the Trident Scholar Program, and Professor Crabbe for giving me the opportunity to pursue this research project. I have learned a great deal about academic research and have been able to mentally challenge myself through this program. I would also like to thank my family, mentors, and friends in 13th Company, Drum and Bugle Corps, the Electrical and Computer Engineering department, and the Computer Science department. Without their support, I would not have even applied for the Trident Scholar Program or completed this project. Specifically, I would like to thank Professor Gavin Taylor from the USNA Computer Science Department for his unofficial guidance and counsel on this project. Special thank you to fellow midshipmen researchers Lillian Usadi, Phil Gatbonton, and Matt Alese for their much appreciated feedback and assistance on the papers, posters, and presentations for this project. Lastly, I could not possibly be the student, midshipman, and person I am today without the constant love, support, and inspiration of my parents and siblings, Andrew and Katie, and I will be forever grateful for everything they do for me.

Contents

1	Introduction	7
1.1	Motivation	7
1.2	Organization of this Report	8
2	Current Events	8
3	Problem Statement	9
4	Drone Swarming Background	10
4.1	Progression of Tactics	10
4.2	Swarming in Nature	11
4.3	Swarm vs. Single Entity	12
4.4	Swarm vs. Swarm	12
4.5	Military Swarm Applications	12
5	Reinforcement Learning Background	13
5.1	Markov Decision Processes	13
5.2	Reinforcement Learning	15
5.2.1	Q-Learning	17
5.2.2	Policy Gradient Algorithms	18
5.3	Cutting Edge RL Algorithms	19
6	Levels of Autonomy	21
6.1	Automation	21
6.2	Semi-Autonomous Systems	21
6.3	Autonomous Systems	21
7	Networks	22
8	Related Works	23
8.1	Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning	23
8.2	On Coordination in Multiple Aerial Engagement	23
8.3	Emergent Tool Use from Multi-Agent Interaction	24
8.4	Responding to Unmanned Aerial Swarm Saturation Attacks with Autonomous Counter-Swarms	24
8.5	Previous Work at USNA	25
9	Environment Description	26
9.1	Unity Video Game Engine	26
9.2	MLAgents Plug-in	26
9.3	The ARC Cluster	27
9.4	Game Assets	27
9.5	Swarm vs. Swarm Scene	30

9.6	Ship Attack and Defense Scene	30
10	Pre-Programmed Tactics	30
10.1	Description of Tactics	30
10.1.1	Swarm vs. Swarm Scene	30
10.1.2	Ship Attack	31
10.1.3	Ship Defense	33
10.2	Swarm vs. Swarm Comparison of Tactics	33
10.3	Ship Attack and Defense Comparison of Tactics	36
10.4	Analysis of Pre-Programmed Tactics	38
11	RL Agent Tactics	38
11.1	Agent Design	38
11.1.1	Distributed Network	38
11.1.2	Inputs	39
11.1.3	Reward Structure	40
11.1.4	Outputs	40
11.2	Agent Training	40
11.2.1	K-frame Updates	41
11.2.2	Imitation Learning	41
11.2.3	Curiosity Reward	42
11.2.4	Curriculum Learning	42
11.2.5	RL Algorithm Configuration	43
11.3	Swarm vs. Swarm Results	45
11.4	Ship Attack and Defense Results	46
11.5	Analysis of RL Tactics	47
12	Ethical Implications	47
12.1	Safety Precautions	48
12.2	Necessity of Continued Research	48
13	Future Work	48
13.1	Implement Programmed Tactics	48
13.2	Use RL to Learn Tactics vs. Programmed Tactics	48
13.3	Use RL to Learn Tactics vs. RL Learned Tactics	48
14	Conclusion	49
A	Pre-Programmed Tactics Results	54
A.1	Swarm vs. Swarm- Figure 10.3	54
A.2	Ship Attack and Defense- Figure 10.4	55
B	RL Agent Results	56
B.1	Swarm vs. Swarm- Figure 11.5 and Figure 11.7	56

Nomenclature

γ : discount factor, programmer controlled hyper-parameter that dictates how much future rewards are diminished by

$\hat{Q}(s, a)$: estimated value of the state-action pair

$\nabla \pi_{\theta_t}(a|s)$: gradient of the current policy

$\pi_{\theta}(a|s)$: probability that the action will be taken in the given state

θ : parameters or weights on the policy

a : action taken

$J(\theta)$: performance measure that is used to update the policy

$R(s, a)$: reward function

s : current state

s' : next state

$T(s, a, s')$: transition function, probability of moving from state s to state s' if action a is taken

$V(s)$: value of current state

Actions: Steps taken to move to another state

Agent: Entity in the environment that takes actions to accomplish the goal

Environment: Universe and anything inside that is relevant to the problem

MDP: Markov Decision Process

Policy Gradient: method of changing a policy in the direction of the optimal policy

PPO: Proximal Policy Optimization

Q-Learning: method of reinforcement learning that calculates and store that values of each state-action pair

RL: Reinforcement Learning

SAC: Soft Actor Critic

States: Representation of every possible attribute of the environment

UAV: Unmanned Aerial Vehicles

List of Figures

2.1	Formations of Chinese Unmanned Boats	8
4.1	Progression of Combat	10
4.2	Living Ant Bridges	11
4.3	Unmanned Fixed Wing Aircraft	13
5.1	RL Basics	14
5.2	Q-Learning 1	16
5.3	Q-Learning 2	16
5.4	Q-Learning 3	17
5.5	Policy Optimization with Deep Neural Network	19
5.6	Policy Optimization Algorithms	20
7.1	Types of Networks	22
8.1	Multiple Agent Tactics	23
8.2	Multi-Agent Hide and Seek Self-Play Simulation	24
8.3	Python Simulation of Drone Swarming Battles	25
9.1	Unity Game Objects	28
9.2	Swarm vs. Swarm Scene	30
10.1	Swarm vs. Swarm Tactics	32
10.2	Ship Attack and Defense Tactics	33
10.3	Swarm vs. Swarm Results	34
10.5	Ship Attack and Defense Scene	36
10.4	Ship Attack and Defense Results	37
11.1	Relative Bearing Example	39
11.2	Unity MLAGents- GAIL	41
11.3	Curriculum- Starting Positions	43
11.4	RL Configurations	44
11.6	Swarm vs. Swarm RL Agent Training	45
11.5	Swarm vs. Swarm RL Agent Kills	45
11.7	Ship Attack and Defense RL Agent Kills	46
11.8	Ship Attack and Defense RL Agent Training	47

1 Introduction

The problems that modern day computer scientists are trying to solve are becoming increasingly complex. For large scale problems, it is impossible for a human to think of every possible situation, determine the desired action for each situation, and then code those actions for the computer to execute. If computers were able to write their own instructions, the world of computer science could expand even more to accomplish more difficult tasks. This is field of machine learning. Recent work has brought the world various photo classifiers, computer vision, search engines, recommender systems, and more. Using machine learning, computers are even able to learn and master games like Snake, Chess, and Go. With this technology, self-driving cars, intelligent robots, and autonomous machinery do not seem impossible anymore.

The US military has consistently pushed the bounds of technology to give us a tactical advantage over our enemies. Using machine learning to aid the US in battle will increase our military capabilities and ultimately protect our people. Recent developments in unconventional warfare have given rise to unmanned vehicles and tactical swarming of autonomous agents like drones. It is imperative that the US military create defenses against this type of enemy technology and develop ways to use swarming to our advantage. Implementing machine learning methods to the problem of multi-agent drone swarming could provide the US military the ability to fight and counter enemy swarms.

1.1 Motivation

The US military is always exploring the latest technology advances in order to maintain a competitive edge over our enemies. Swarming tactics is currently a major area of military research. The US and other countries are finding new ways to use drones, boats, and vehicles with the existing swarming techniques. For example, Russia is developing impressive drone swarming capabilities. [Reid 2018] Iran has created massive boat swarms. [Osburn 2019] China can create swarms of armed helicopter drones. [Peck 2019] Countries, big and small, and even non-state actors are using the current swarming technologies to increase their military power and contend with the US. This possible threat to US security and the opportunity to gain advantages over other powers are the motivating factors of this research project. If the US does not develop defenses and tactics to counter enemy swarms, our people, assets, and national interests are in danger. In this research project, we aim to use the state of the art RL algorithms to develop drone swarming tactics and defensive counter tactics. Studying the current RL algorithms and learning how to apply them to real-world problems are important next steps for the world of computer science as well as the military. This project is designed to combine the available RL tools with drone swarming in order to find swarming tactics and counter tactics that will defeat enemy swarms, improve military doctrine, and protect US national interests.

1.2 Organization of this Report

This report begins with the current events that motivate the need for drone swarming tactics as well as a definition of the problem that we are trying to solve. The next sections provide background on drones, military swarming, reinforcement learning, and the policy optimization algorithms that are used in this research project. We also include previous work relating to the power of RL and how it has been used in combination with the current drone and swarming technologies. The section also details the work of two midshipman who have worked with Professor Crabbe on the early stages of this research project in the past two years. The next section describes the environment/simulation that we built. Following this information is the start of our current results. We built two different scenarios and ran similar tests on each one. The first is the Swarm vs. Swarm scene and the second is the Ship Attack and Defense scene. For both scenes, we describe the programmed tactics that we implemented and present the results of the comparison of those tactics. Next, we describe the RL Agent design and RL training and test its effectiveness. After all of the results are presented, we analyze our findings and describe the ethics and future direction of this research project.

2 Current Events

Multiple nations and non-state actors have developed small drones and drone swarms with offensive capabilities. For example, the Chinese military has recently developed drone swarm capabilities that could threaten US security if the US military cannot counter these attacks. According to the Chinese newspaper Global Times, “China’s domestically developed helicopter drones carrying proximity explosive mortar shells, grenade launchers and machine guns can now form swarms and engage in coordinated strikes.” [Peck 2019] Additionally, China has demonstrated the capabilities to launch unmanned boats. Figure 2.1 shows an image from a Chinese news video demonstrating unmanned boats moving in formations, avoiding a ships, and shifting to form various patterns. [Barnes 2018]



Figure 2.1: Formations of Chinese Unmanned Boats

Iran is building their military capabilities and expanding their technological abilities to challenge international powers. They specifically have developed the ability to swarm small, fast boats in attack against large ships similar to the US Navy Fleet. These capabilities give Iran the opportunity to compete with large militaries like the US despite having less funding, manning, and resources. The US ships’ defenses are designed to fight against large vessels, not a large number of small boats. Even if the US ship can eliminate 1 small boat, there are many

more still on the attack. The Navy Fleet must develop tactics to save the ship against these enemy tactics. [Osborn 2019]

Small countries, non-state actors, and individual rebels are even using drones and swarming tactics as a way to compete with developed countries and strong militaries. Drones can be made with cheap materials and limited programming skills. They are also quick and easy to operate. Compared to the cost and manning requirements of a US aircraft carrier, drone swarms are easily accessible for small budgets. For example, in January 2018, a swarm of 10 low-tech DIY drones armed with explosive materials attacked a Russian military base. Russia blamed “Syrian rebels backed by Turkey” which shows how drone swarming technology can even the playing field between developed militaries and small forces. [Reid 2018]

“UAV Swarm are Attack: Protection System Alternatives for Destroyers” was a Naval Postgraduate School Capstone Project by Loc V. Pham. In this study, the team simulated drone swarms armed with improvised explosive devices (IEDs) attacking a US Arleigh Burke Class Destroyer. Their simulations showed that when a swarm of only 8 drones attacked a ship, on average 3.82 drones will penetrate the ship’s defenses and hit the ship. The research shows that drones have a small radar signature making them difficult to locate using US ship’s current defense systems. The incoming drones are usually too close to the ship by the time they are detected and the ship’s defensive weapons are not effective. The US Navy and military as a whole is currently designed to fight against large and technologically advanced enemy ships, aircraft, and ground forces, not inexpensive and small unmanned aerial vehicles. Through this NPS research project, the group was able to develop and simulate better defenses that only allow 1.12 drones of an 8 drone swarm to hit the ship. Although this is an improvement, the idea of 1 in 8 drones packed with explosives hitting a US Navy warship motivates the need for the US military to develop drone swarming counter tactics in order to protect our military and national interests. [Pham 2012]

3 Problem Statement

The goal of this project is to use state-of-the-art reinforcement learning algorithms to develop drone swarm tactics and counter tactics. We are building the Swarm vs. Swarm Scene and the Ship Attack and Defense Scene in a computer graphic simulation. The simulated drones represent the fixed wing unmanned aerial vehicles that the US military is researching and implementing into their military doctrine. We are using both pre-programmed tactics and RL agent learned tactics to develop defensive drone swarming capabilities. The reinforcement algorithms as detailed in Section 5.3 allow the swarms to determine the most effective way to accomplish the goal rather than rely on human pilots or computer programmers. Using all the tools at our disposal, we show that it is possible to gain an advantage of an enemy using defensive drone swarming tactics.

4 Drone Swarming Background

4.1 Progression of Tactics

In order to understand the role of swarming in combat, it is helpful to look at it in the context of various styles of combat. Melee is the first battle tactic that was used in war. A melee is group of individuals fighting as individuals against the enemy individuals. There is little order or instruction in this type of fighting. Before the advent of technology and modern communication, people were forced to fight as individuals because they had no means of coordinating with the others. This made melees very chaotic, disorganized, and especially dangerous. [Arquilla 2000]

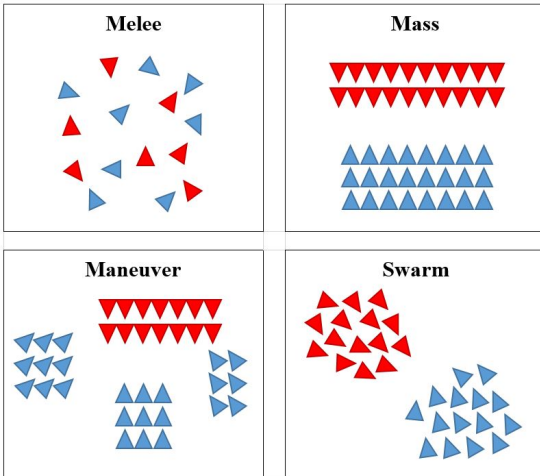


Figure 4.1: Progression of Combat

Massing is the next iteration of group fighting tactics. When a group participated in melees against another group, individuals would cluster with their friendly teammates. This provided strength in numbers and some sense of protection against the enemy. Here, groups would be set in formation and fight from their designated positions. There is still some element of individual fighting skills, but the group is organized and general kept together as a mass. The group leaders could use hand signals, signs, and verbal communication. The individuals in the mass are not only relying on their fighting skills, but also the skills of the people nearby, the strength of the formation they are in, and the amount of protection offered by their specific position in the mass. [Arquilla 2000]

Maneuver is the next progression in the development of effective group fighting tactics. As massing continued to develop in battle, group leaders would discover different ways to use their masses to gain an advantage over the enemy. These new found techniques grew into the maneuver warfare tactics that are still used today. This type of warfare heavily relies on a strong and tactical leader and efficient communication between all members of the group. In order to coordinate the plan of action throughout the duration of the battle, especially in the chaos of battles and in times of expected changes to the intended plan, the leaders must have open lines of communication with their people. The hierarchy of leadership allows the overall leaders to communicate with the leaders under them so that they may then communicate with the people under them. Some of the important tactics in maneuver warfare are surprising the enemy, penetrating the enemy front, and outflanking the enemy formations. All of these techniques are designed and implemented to give one side a tactical advantage over the enemy that can outweigh any size, strength, or resource disadvantages. [Arquilla 2000]

The swarming tactic is the most recent development in military fighting techniques and tactics. A swarm is a group of autonomous or semi-autonomous agents that can communicate with each other. The swarm's goal is to converge, maneuver as one entity and pursue a common target. It tries to overwhelm the target's ability to fight back because of the sheer size of the swarm. The agents in a swarm are typically small, inexpensive, and simple. Additionally, swarming agents are usually autonomous (unmanned aerial vehicles or unmanned small boats). This can lead to a decrease in the loss of life based on these new fighting techniques. Compare the danger to human life of launching a swarm of drones against the landing on Normandy Beach in World War 2. Although this is not the same scenario, there is a considerable decrease in the amount of human lines on the front line when swarms are used. There are many advantages to swarming and tactics are still being developed today to increase its effectiveness in modern warfare. [Arquilla 2000]

4.2 Swarming in Nature

Swarming has been used by many parts of nature to provide protection to the animals, accomplish complex tasks, and suppress any enemy that is trying to kill them. For example, bees, ants, and other insects use swarming techniques when they feel attacked by a predator, find food they want to pursue, or need to protect themselves. When these animals pursue a common target, they form a swarm and attack as a group. Instinctively, one single bee knows it will not stand a chance against a large animal or a human. However, a massive swarm of thousands of bees working together will overwhelm and incapacitate any other being that opposes them. This is similar to ants swarming around a food source or resource. They can attack from all sides and work as a group to control the target object. Additionally, ants are known to conduct war-like swarm attack other groups of ants to extend their territory. Similarly, small animals like wolves operate in packs and use many of the swarming advantages in their attacks. These coordinated attacks in nature can influence the way we develop tactics for battle and war.



Figure 4.2: Living Ant Bridges

Researchers at Princeton University worked with “army ants of the species *Eciton hamatum*” to analyze the way they build living bridges made entirely of ants [Kelley 2015]. The ants could collaborate and link themselves together in a line long enough and strong enough to span a gap and support other ants as they transport across the living. These ant are operating as a swarm; they are individual agents with an individual brain making decisions on their own. The ants are operating as a distributed network as described in Section 7. Much like drones in a drone swarm, the ants were adapting to the

situation around them and playing their role without explicit instructions from a command center or leader. Additionally, the researchers experimented with techniques to improve the

intelligent behavior of the ants. They would begin with a relatively easy task of crossing a small gap with around 10 ants and then increase the difficulty until the ant bridges contained thousands of ants. Human often create designs based on what they see in nature and swarming is a great example of people learning battle tactics from the natural world [Kelley 2015].

4.3 Swarm vs. Single Entity

A swarm of agents is very effective at attacking and overwhelming a single entity. When a single ship attacks an enemy ship, the fighting is focus on one front and each ship can manage their defenses and weapons to target the enemy it is fighting against. On the other hand, if a swarm of drones attacks a ship, then the ship must eliminate hundreds of different entities in order to win the fight. When the ship successfully shoots down one drone, there are a hundred more swarming and still attacking. This is a hard problem to defend against because of the amount of agents that are in the swarm. The ship will also realize that this swarm will attack from every side and every angle making it even more difficult to defend itself. [Arquilla 2000]

4.4 Swarm vs. Swarm

In response to the development of swarm tactics, various organizations are experimenting with swarm versus swarm battles. It is theorized that launching a counter swarm would be an effective antidote to an incoming swarm. Currently, swarm on swarm battles rely heavily on the numerical or physical advantage of the swarm. When a swarm is forced to fight another swarm, the advantages that swarms have against single, more capable opponents are not available. The swarm does not have the power of number to overwhelm an entity because it now must fights a large number of agents like itself. New tactics are being developed and studied in many projects including this research proposal. Example tactics could be breaking the large swarm into two smaller swarms or sending multiple swarm waves. Studies in this field are continuously evolving and new techniques and tactics are being developed to give friendly swarms competitive advantages over any enemy swarm that is trying to attack it.

4.5 Military Swarm Applications

Unmanned aerial vehicles are widely used for surveillance and reconnaissance. Drones can capture a view of the battle space from above. These agents are very small and can go places quickly and undetected. Drones have the ability to gather information and communicate back to a host or electronic center of the swarm. Swarm agents can collect information from the enemy using signals intelligence and data collection tactics. [Edwards 2000]

The US Military and militaries around the world are using swarming as an offensive threat. Drones, boats, and even vehicles can operate unmanned and swarm as a unit in order to attack an enemy. Using small and cheap agents in large quantities can give smaller militaries an advantage against the power of the US military. For example, small boats or drones can converge on a single ship and cause massive amounts of damage, like destroying

the ship's radar. As an offensive technique, swarms are powerful assets that can be used as a offensive war fighting scheme. [Edwards 2000]

In response to the offensive swarming techniques, militaries are starting to research and use swarms as defensive mechanisms against incoming swarms and other threats. The other defensive weapons systems are not built to fight a large mass of small drones, so launching a counter swarm could be a viable defense against the newest swarm tactics. Swarms could also be used to defend single entities against incoming weapons system. Researchers are creating new ways to build, arm, and train small unmanned vehicles so that they can be reliable assets to the US military. [Edwards 2000]



Figure 4.3: Unmanned Fixed Wing Aircraft

5 Reinforcement Learning Background

This section describes the progression of reinforcement learning algorithms that we will use to develop counter swarm tactics. Each topic begins with a short summary followed by a detailed technical explanation.

5.1 Markov Decision Processes

The Markov Decision Process (MDP) is one of the foundations of machine learning and artificial intelligence. In an MDP, the world is broken up into “states”, where each state represents a possible configuration of the environment. For example, in a navigation task, the state would be the location of the agent, whereas in a cargo unloading task, the state would be the locations of the cargo, how it is stacked, and what available locations exist on shore. In any state, an agent has an array of actions to choose from. Each action changes the environment, so triggers a transition from the current state to a new state. The transitions are probabilistic, governed by the transition function $T(s, a, s')$ which is the probability of ending up in state s' given that the current state is s and the agent takes action a , i.e. $T(s, a, s') = P(s'|s, a)$.

The goal of a agent in an MDP is to maximize the “reward” it receives during its time completing its task. The reward is a signal from the environment indicating how well the agent is doing. It can come at the end of the task (with success or failure) or along the way, depending on the type of task.



Figure 5.1: These 8 squares represent the environment in this simple example. The purple circle is the agent. The picture on the left represents State 1 and the picture on the right represents State 2. If the agent is in State 1, it can take the action of 'Move Right' and will reach State 2. The goal state or end state is when the agent is on the yellow square labelled Goal.

MDP Formal Definition:

- **Environment:** The environment is the universe and everything inside of it that is applicable to solving the problem at hand.
- **States:** A state is a snapshot of the agent and its environment. The state represents every possible attribute of the environment including the position and characteristics of the agent and every other object in the environment. The state space is the set of all possible states. The current state is the environment at any point in time. The goal state is the state in which the desired task is completed or the problem is solved.
- **Agent:** The agent is an entity in the environment that takes actions that modify the state. It takes a sequence of actions in order to complete the goal or task.
- **Actions:** Actions are steps that the agent can take in order to move to a different state. An action will create a change in the environment or in the agent's position in the environment. These actions should move towards the desired or goal state.

Rewards allow each state to be given a certain value. States that produce a positive reward or are close to states that produce a positive reward have a higher value than states that are far away from any positive reward. If all of the adjacent states' values are known, then the agent will take the action that moves to the state with the highest value. This state value is represented by $V(s)$ and calculated by the Bellman Equation below.

$$V(s) = \max_a (R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s')) \quad (1)$$

where,

- $V(s)$: value of current state
- s : current state
- a : action taken

- s' : next state
- $T(s, a, s')$: transition function, probability of moving from state s to state s' if action a is taken,
- $V(s')$: value of next state, recursively calculated
- γ : programmer controlled discount factor between 0 and 1, amount that future reward are diminished by, a higher γ values future rewards while a lower γ focuses on rewards in the present and near future
- $R(s, a)$: reward function, amount of reward the agent receives for taking action a while in state s
- $\max_a$: maximize $V(s)$ by choosing the action a that produces the highest reward and value for state s

The Bellman Equation says that the value of a state is equal to the reward for taking the most rewarding action in the current state plus a discounted sum of the future rewards for moving to the next state. This equation calculates $V(s)$ recursively with the summation of all possible future states. The γ is less than 1, so the summation will converge to a value. Once the value of every state is known, the agent will decide to take the actions that lead to the next possible state with the highest value and will move towards the goal state thus completing the task. [Sutton 2018]

5.2 Reinforcement Learning

There are many ways to approach solving MDPs, each requiring more or less knowledge about the environment. An approach that requires almost no knowledge about the environment is Reinforcement Learning (RL). RL can be compared to a young child learning how to play basketball with the help of a coach. At first, the child has no skills or knowledge about how to correctly play the sport or what techniques will be effective. The child will begin experimenting with the ball and running all over the court. After every shot, mistake, or play, the coach will offer feedback to the player based on what he observes. This advice is similar to the positive or negative rewards that the agent receives. The child is gradually learning what actions he should take in various situations while the coach is judging the effectiveness of those actions and correcting the player accordingly. After years of this training cycle, the player will develop the policy of actions that he should take and the strategies that will allow him to effectively play basketball.

RL only requires that the environment be able to provide a reinforcement signal to the agent to indicate reward or punishment for certain outcomes. Before training, the agent usually does not know the reward function, the transition function, or the exact values for every state. If these values were able to be calculated by a human, then this problem would be solvable through traditional programming practices.

The typical reinforcement learning approach is to use a training stage to learn the values for each state. This is done through sampling and experimentation. The agent will begin with no information forcing it to randomly take actions. Each action the agent takes will lead to a positive, negative, or no reward. Using this information, the agent will update the value for the nearby states. As training progresses, the agent will learn more of the values for the states and improve the accuracy of these values. Eventually, the random actions will become carefully calculated actions that lead the agent to solving the problems at hand. This method of solving Markov Decision Processes is illustrated in Figure 5.2-5.4.

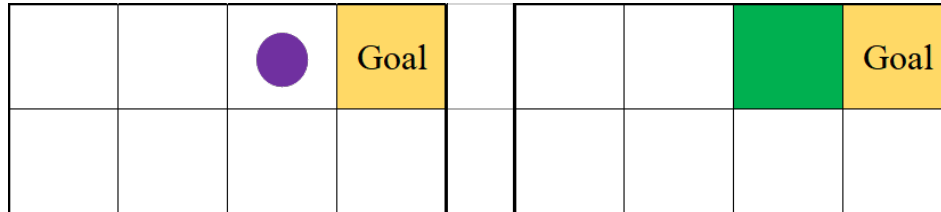


Figure 5.2: The left picture represents an environment with an agent at the beginning of training. The agent does not know the value of each state (evidenced by the white squares). The agent only receives a positive reward when it reaches the yellow Goal square making the yellow state valuable. If the agent is in State 1 as shown in the left picture and takes the action 'Move Right', the agent will receive a positive reward. This teaches the agent that the state it just left is also valuable because it leads to a future reward. The agent updates the value of the squares (shown by the green color) and continues training.

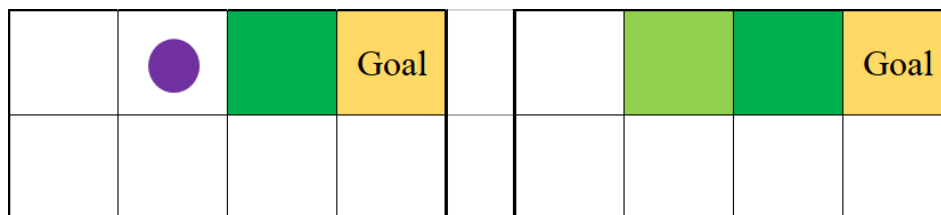


Figure 5.3: The left picture represents the same environment from Figure 5.2 after some training has taken place. The square adjacent to the goal square has a high value because the agent learned that from there, it can reach the goal square and get a reward in 1 action. If the agent is in the state shown on the left and takes the action 'Move Right', it will reach the green high value square. The agent now knows that the square it just left is also valuable. It is not as valuable as the green square because the potential reward is further in the future, but it is helpful to reach this square. The future rewards are discounted by the γ value. Therefore, the square that the agent just left gets updated with a light green shade to represent its value.



Figure 5.4: These pictures represent the same environment from Figure 5.2 after training has taken place. After many rounds of training, the agents will learn all the values of each state as represented by the colored diagrams. The agent learned that it should move to the right and the top every action in order to reach the goal state. The agent will move towards the states with higher values (darker green squares) and until it reaches the goal state to receive its reward.

5.2.1 Q-Learning

Q-learning (Quality Learning) is the method of calculating and storing the value of each state directly. If the state space is sufficiently small and the reward and transition functions are known, then the value of every state can be calculated. The agent can also experimentally learn the value of each state by randomly taking actions as described in the reinforcement learning section above. Once the agent has calculated and stored the value of every state, it will take the action that leads to the highest value state of the current reachable states. Each action will bring the agent to a more valuable state until it reaches the states that offer rewards or the goal state that accomplishes the desired task.

One possible problem with this approach of learning the states' values is that the agent may not have access to the transition function. Without the transition function, the agent cannot determine an action in a state because it does not know how likely it is to end up in each of the possible next states. Another branch of RL can solve this potential problem. Rather than calculating the value of each state, the agent calculates a value Q for each state-action pair using this formula below.

$$Q(s, a) = \sum_{s'} T(s, a, s') (R(s, a) + \gamma V(s')) \quad (2)$$

and then determines the best action by,

$$a_{best} = \arg \max_a Q(s, a) \quad (3)$$

The agent begins exploring its environment and uses the rewards it receives to estimate the value of the state-action pairs. Because the agent is sampling the data, the exact transition function ($T(s, a, s')$) does not need to be known. The more actions that agent takes, the more rewards it earn and the more accurate the value for each state-action pair will be. As the agent gains more information, it will learn to choose the desired actions that lead to the goal state.

Q-learning works in both theory and practice but does have some limitations. Many hard problems have too complex environments to calculate, store, and access the values for each possible state or state-action pair. When the state space is large, using Q-learning might be possible, but it is very slow given the amount of calculations. The number of these pairs can be very large even in seemingly simple problems. Some of these issues can be mitigated by approximating the Q function with a continuous parameterized function such as a polynomial or a neural network. Unfortunately, this approach can still result in a function that is large and unsmooth, resulting in taking exceedingly long to converge, if it does at all. [Sutton 2018] The problem of swarm counter tactics has a continuous environment with many states and many possible actions, so Q-learning alone is not the most efficient RL algorithm for this problem. [Sutton 2018]

5.2.2 Policy Gradient Algorithms

In contrast to value-based algorithms such as Q-learning, there is an alternate approach, that of policy-optimization. A policy $\pi(a|s, \theta)$ is a mapping from the state s and a set of parameters θ to a probability that the agent should take action a . This policy is like an instructions manual for the agent. For example, if the agent is in state A, then it will perform action 1 75% of the time, action 2 15% of the time, and action 3 10 % of the time. The parameters θ are the weights that determine the probability that each action in each state will be taken. At the beginning of training, the parameters are randomly initialized. As the agent explore its environment and learns more about the state space, the parameters will converge towards the optimal policy, or the set of parameters that maps the state to the actions that will most effectively accomplish the task. $J(\theta)$ is a performance measure that is used to update the policy in the direction that provides a better performing policy. This is known as policy gradient ascent and follows the equation below.

$$\theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \quad (4)$$

$J(\theta)$ can be broken down further into a function of the value of the state action pairs and the current policy.

$$\theta_{t+1} = \theta_t + \alpha \frac{\hat{Q}(s, a) \nabla \pi_{\theta_t}(a|s)}{\pi_{\theta_t}(a|s)} \quad (5)$$

where,

- θ_{t+1} : Updated parameters
- θ_t : Current parameters
- α : Programmer-controlled hyper-parameter that determines how large of a step the policy moves, usually starts large and decreases as training continues in order to fine-tune policy
- $\hat{Q}(s, a)$: estimated value of the state-action pair, estimation of how good this action is
- $\nabla \pi_{\theta_t}(a|s)$: Gradient of the current policy, direction that moves towards a better policy

- $\pi_\theta(a|s)$: probability that the action will be taken in the given state, how often the policy is being updated

The parameters (θ_t) can be modelled as a neural network. There are inputs, layers of nodes with connections between each node, and outputs. The weights on each connection are the parameters from Equation 5. In addition to the policy, the neural network also produces an estimated value of the current state. Using the policy, the agent can calculate the actual reward from that state. By subtracting the estimated value from the actual value, we get a loss or error function. If the error is positive, then the current policy did better than expected and the probability of that action should be increased because that action had a high value. If the error is negative, then the current policy performed worse than expected and the probability of that action should be decreased. Additionally, each update to the policy also changes the estimated values that the neural net returns. As the training progresses, the policy should improve and the estimated values should become more accurate.

Policy Optimization is more effective and faster than Q-learning in large complex environments. Although there are many advantages, Policy Optimization can sometimes be unstable and create noisy gradients, meaning the policy may not converge as it approaches the optimal policy. Another potential problem with Policy Optimization is that the high variance in the gradient may cause a slow convergence or no convergence at all. [Yoon 2019] If the policy changes too rapidly, then the agent be in very different states at each step and unable to learn the environment. Despite these limitations, policy optimization has proved to be effective in large and complex environments and certain variations of Policy Optimization can minimize these issues. [Ecoffet 2019] [Proximal 2018]

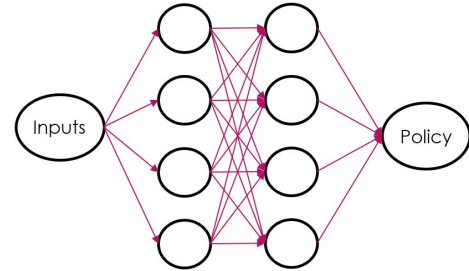


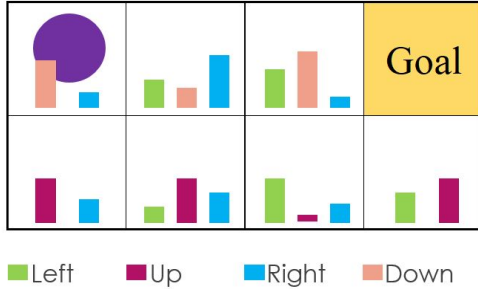
Figure 5.5: Policy Optimization with Deep Neural Network

5.3 Cutting Edge RL Algorithms

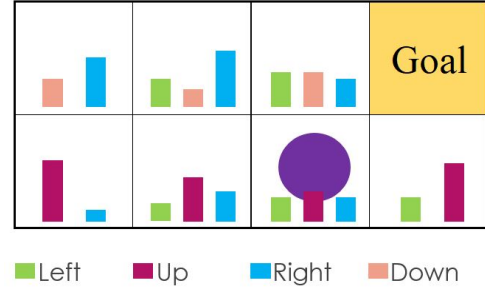
Proximal Policy Optimization and Soft Actor Critic are both algorithms that create and improve on a policy as described in Section 5.2.2.

Proximal Policy Optimization (PPO) is based on another type of Policy Optimization called Trust Region Policy Optimization. PPO aims to improve performance and ensure convergence by limiting the step size of the policy so that the change in policy is based on the information on hand. If the step size is too large, then the policy could be changing too quick and unable to truly learn the environment. PPO is a stochastic on-policy algorithm which means it has some amount of randomness in its decision making but mostly stays with actions that are close to current policy. As the training progresses, the actions get less random and follow the policy more closely in order to approach convergence on the optimal policy. [Proximal 2018]

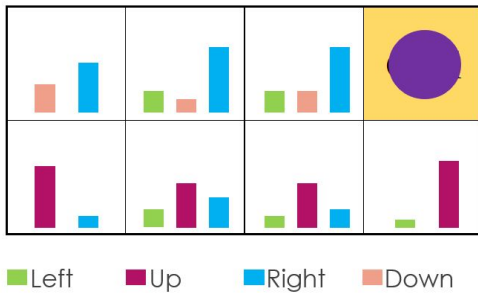
Soft Actor Critic (SAC) is another type of Policy Optimization algorithm that is stochastic off policy. Stochastic off policy means that the agent will randomly choose actions that



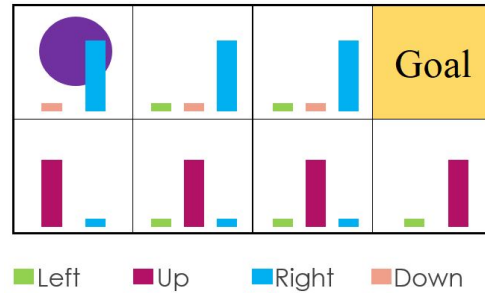
(a) This picture represents the policy of the agent at the beginning of training. All the parameters are randomly initialized so the probability of the agent taking each action is also randomized. The colored bars in each square represent the probability of that action being taken. The higher the bar, the higher the chance that that action is taken.



(b) As the agent takes actions that are proportional to the policy, the policy gets updated according to the performance function $\nabla J(\theta_t)$. The bars have increased or decreased in height in order to represent this change in policy.



(c) The training continues and the agent receives more feedback on its actions and updated its policy accordingly. This picture shows that, in the top 3 squares, the 'Move Right' action is becoming more probable. Similarly, in the bottom 4 squares, the 'Move Up' action is growing probability.



(d) After many iterations of training, the policy will converge towards the optimal policy that accomplishes the goal. At this point, the policy of the best action will be close to 1 and the probability of the non-optimal actions will be close to 0. This picture shows the optimal policy of the agent moving up and to the right to reach the goal state.

Figure 5.6: Policy Optimization Algorithms

are not determined by the current policy in order to facilitate exploration of the environment and curiosity. SAC aims to include entropy (randomness) regularization into the algorithm so that the agent does not converge on a bad policy too quickly and get stuck there. In the early stages of training SAC encourages more exploration which gives the agent more information to use to improve the policy as the training progresses. This algorithm can lead to convergence on an optimal policy in a shorter amount of time due to the early exploration and increased entropy. [Soft 2018]

Policy Optimization has been effective for complex environment where Q-Learning is too computationally expensive. Both PPO and SAC seem like possible ways to develop swarm tactics and train an agent to perform intelligent behaviors.

6 Levels of Autonomy

6.1 Automation

The dictionary definition of automation is the “automatically controlled operation of an apparatus, process, or system ” [Automation 2021]. This is where a human maintains control over the machine and makes the decisions of how the technology is used. In terms of drone swarming, automation would include the use of remotely-piloted aircraft. There is no pilot in the actual cockpit of the aircraft, but there is a human controlling the drone or drone swarm and consciously deciding where the drone(s) will go and what they will do. The machinery is increasing the effective of the human and decreasing the risks, but the human is responsible for all of the actions.

6.2 Semi-Autonomous Systems

The next level of autonomy is the use of semi-autonomous systems. These systems are more advanced and capable than automated systems but still rely on periodic human input. The systems can have certain routine that they can complete without assistance once given they command to execute. Deploying a drone swarm to survey an area to collect intelligence is an example of a semi-autonomous routine. The human is responsible for sending the drone swarm out in a certain area at a certain time and then recalling the drones when the human decides to end the collection. The semi-autonomous systems do not have decision-making capabilities, but they are able to operate independently in specific situations.

6.3 Autonomous Systems

Autonomous is defined as “self-directing freedom and especially moral independence” [Autonomy 2021]. This means that autonomous systems are given information and allowed to decide what to do to best accomplish a task or goal. The push to build autonomous systems is what bridges the gap between drone swarming in Section 4 and the reinforcement learning algorithms in Section 5. If drone swarms could train and develop their own intelligent behavior, they could more effectively destroy other enemy swarms and protect the US troops. Autonomous drone swarm technology is the next progression of swarming tactics and the topic of interest for this research project.

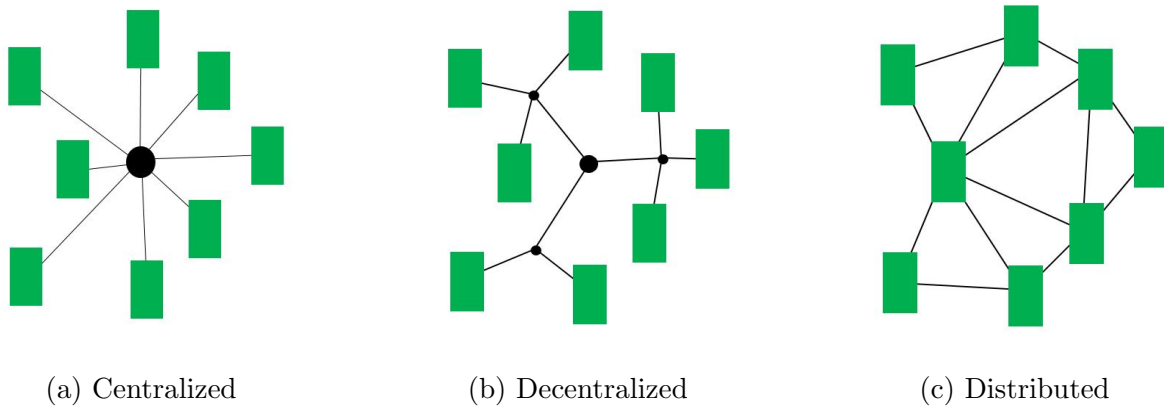


Figure 7.1: Types of Networks

7 Networks

The network of different agents fall into 3 categories based on the way the agents are connected. Centralized, decentralized, and distributed each have advantages and disadvantages with the information shared, the speed of connections, and the reliability of the network. These networks are pictured in Figure 7.1.

Centralized networks have one central command center that is connected to all of the other agents. This allows for fast point-to-point connections and information passing between each agent and the center node, but slower communication between different agent node as it must go through the command center first. Additionally, centralized network are susceptible to an attack at the command center that could easily destroy the entire network's ability to communicate.

Decentralized networks have multiple command centers and subgroups of agents each connected to a command center. This allows agents to quickly get the information that is most relevant to them while maintaining the ability to communicate with all of the agents. There is a natural hierarchy in decentralized networks because each agent responds to the command center that is directly connected to it. Additionally, decentralized networks are more resilient than centralized networks because one hit to a command center will only affect a portion of the agents rather than the entire network.

Distributed networks do not have any command centers but rather each agent is in control of itself and communicates with the nearby agents. This type of network is very resilient to attacks as any agents can be eliminated without lowering the effectiveness of any other agent. For this reason, distributed networks encompass the idea of drone swarming by making them difficult to defend against and easily scaled to large swarms. Additionally, giving the agents on the most relevant information can decrease the amount of computing power needed on each agent. The different types of networks are important to understand when working with drone swarming, and they can even produce different swarm behaviors. [Berty 2019]

8 Related Works

This section will describe some recent work in reinforcement learning with groups of agents and drone swarms.

8.1 Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning

Cano Lopez et al. used current reinforcement algorithms to train a quadcopter drone to fly, hover, and move to designated locations [G. Cano Lopes 2018]. The system used Markov Decision Processes, and implemented the Actor-Critic method of reinforcement learning to train the agent in a flight simulator. These reinforcement learning methods are similar to the methods we hope to apply to the problem of drone swarm tactics.

Using the Virtual Experimentation Platform (V-REP) from Coppelia Robotics as the simulation to train the drone to fly. They were able to achieve fast convergence on their training policy. At the end of the training, they were able to stay in flight and move to different locations in the simulations.

This work showed that reinforcement learning is an effective way to train drones to operate. The methods that we hope to use in this project can be implemented with the current technology. We will extend the experimentation from this paper by applying the RL algorithms to fixed wing unmanned aircraft in a similar simulation. However, rather than only getting the drones to fly and move, we will be training them to work together and strategically plan where to fly and how to operate.

8.2 On Coordination in Multiple Aerial Engagement

Strickland et al. used simulations to test various unmanned aerial vehicle tactics and test what the deciding factors of winning the battle could be. They programmed a swarm with tactics and allowed this swarm to fight an enemy swarm. The agents attempted to coordinate attacks against the enemy drones using the bracket and sandwich approach shown in Figure 8.1. These tactics were more effective than solo shooting at the enemy only when there were two drones against one enemy drone with enough space between them and other pairs of drones. Secondly, some agents would fly away from the enemy as a means to protect themselves and never use any offensive tactics against the enemy. [Strickland 2019]

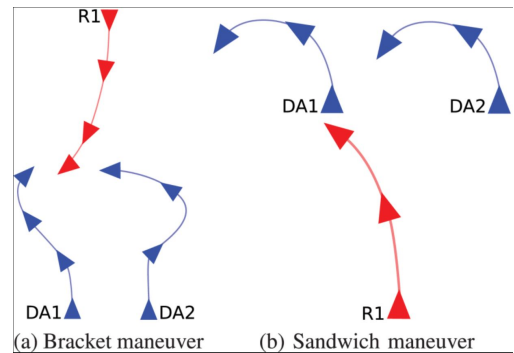


Figure 8.1: Multiple Agent Tactics

8.3 Emergent Tool Use from Multi-Agent Interaction

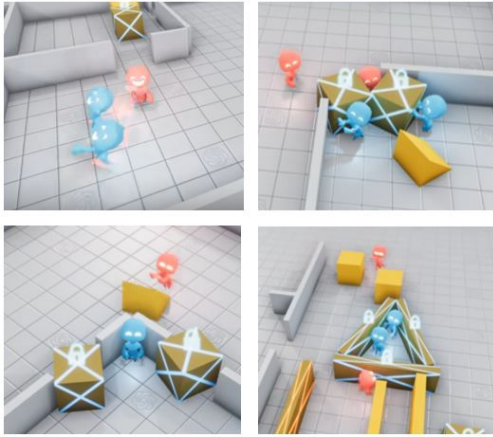


Figure 8.2: Multi-Agent Hide and Seek Self-Play Simulation

This project used PPO to train multiple agents in a game of hide and seek using reinforcement learning. Two red agents were a team designated as seekers and two blue agents were a team designated as hiders. The agents were play in an open environment with a few walls and maybe some blocks as seen in Figure 8.2. The agents could run around and put a force on the movable blocks. The red team got a reward for seeing the blue team and the blue team received a reward for the time they were not hidden. Both agents were trained with self-play and policy optimization algorithms. The teams competed for millions of training iterations and develop tactics and techniques to counter the others actions. At first, both teams ran around aimlessly, but they eventually develop some intelligent behaviors to help them gain rewards. The blue team learned how to block doors, create shelter for themselves, and hide the other objects from the red team. The red team chased the blue agents, used ramps to infiltrate their shelters, hopped on top of the blocks to see over walls. Some of the tactics that these agents developed were even more creative than what a human programmer would have instructed them to do. Most importantly, the agents taught themselves how to work together and assign each agent a specific role in order to accomplish the team goal.

The results of this study show the power of reinforcement learning and the self-play method of learning. Both agents were able to develop intelligent behavior because of the competition between them. We will use the framework of this project to solve our problem of drone swarming tactics. Expanding on the hide and seek game to a drone swarming battle will improve the capabilities of reinforcement learning. The self-play techniques is featured in the RL swarm vs. RL swarm section of the future work for this project which is detailed in Section 13.3. [Baker 2018]

The results of this study show the power of reinforcement learning and the self-play method of learning. Both agents were able to develop intelligent behavior because of the competition between them. We will use the framework of this project to solve our problem of drone swarming tactics. Expanding on the hide and seek game to a drone swarming battle will improve the capabilities of reinforcement learning. The self-play techniques is featured in the RL swarm vs. RL swarm section of the future work for this project which is detailed in Section 13.3. [Baker 2018]

8.4 Responding to Unmanned Aerial Swarm Saturation Attacks with Autonomous Counter-Swarms

In this study, researchers simulated and tested drone swarming tactics using computer programming and reinforcement learning. The group created a list of possible swarm tactics including a simple shooter, an evader to draw the enemy towards a teammates, and a herder that breaks the enemy swarm into sub-swarms. The researchers then created a simulator to test these battle tactics. They collected data on which tactics were the most effective and even tested these algorithms on real-life fixed wing drones. We will be implementing some of these tactics in the first stages of our research and expanding the current doctrine of programmable swarming tactics.

The second aspect of this research paper was implementing reinforcement learning methods to allow the agents to develop their own swarming tactics. The allied drones received a positive reward for killing an enemy drone and a negative reward for being killed by the enemy. The enemy swarm was pre-programmed with the successful single shooter from the first stage of the research. The goal of this project was to allow the agent to develop tactics against the enemy swarm. However, the allied drones learned that they should flee and simply fly away from the enemy to avoid the negative reward of being killed. Because the enemy was too effective, the allied drones were not able to gain enough positive rewards to learn how to attack the enemy swarm. We will be using reinforcement learning to train the agents in a similar way, however we hope to gain more conclusive results. In order to prevent the allied drones fleeing from the enemy, we will reward the agents more for attacking and killing the enemy than the risk of dying. We can also program the agent to defend an asset like a ship or base. This research project provided a good foundation for the research we have done. [Strickland, Day, et al. 2018]

8.5 Previous Work at USNA

This research project is a continuation of recent work in reinforcement learning and drone swarming. The computer science field has been developing the state of the art reinforcement learning algorithms like PPO and SAC which this project aims to apply the current military problem of drone swarm tactics.

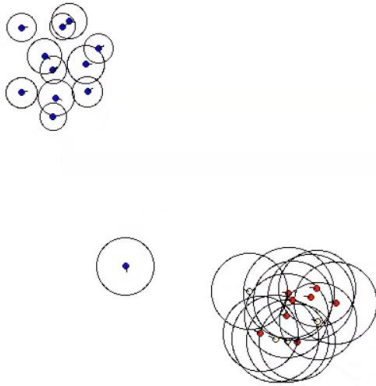


Figure 8.3: Python Simulation of Drone Swarming Battles

MIDN 1/C Abramoff (Class of 2019) studied drone swarming tactics and simulated miniature swarm on swarm battles in Python. He created a 2-D space and represented each agent in the swarm with a dot. Each agent could shoot forward (in the direction it is moving and facing). Agents that were hit by “bullets” from another agent were assumed to be dead and were deleted from the simulation. Abramoff created the swarms and wrote a flocking algorithm so that the agents would swarm and fly as one mass without colliding, separating, or breaking the swarm. Once the agents swarmed realistically, Abramoff explored various drone swarm tactics such as Select-Nearest and Assign-Nearest and tested their effectiveness against an enemy swarm. Select-Nearest allows each agent to aim at the closest enemy to itself. As the swarms move towards each other, the agent will change its target based on which enemy drone is closest at each moment in time.

Assign Nearest gives each agent an assignment to neutralize a different enemy drone. The assignments are decided based on which enemy drones are closest to the friendly swarm and are re-updated at every frame. Abramoff experimented with the simulated battles of two swarms into test which swarm tactics were most effective. He also attempted to fight

using counter swarm tactics such as sending a “rabbit” agent in front of swarm and breaking into subswarms. In conclusion, Abramoff found that Assign-Nearest was most effective in his experiments and some of the counter swarm tactics were successful. The results were not conclusive but showed progress towards the development of swarm and counter swarm military tactics. This research proposal will expand on MIDN 1/C Abramoff’s work by creating a 3-D environment simulation and improving the agent capabilities to represent a realistic drone swarm battle. This environment of this research proposal will have a larger battlespace and more actions that the agents can take including altitude changes, weapon aiming, and inter swarm communication/teamwork.

MIDN 1/C Thompson (Class of 2020) built a 3-Dimensional environment that he used to simulate more tactics. This environment is more realistic simulation of a real-world battlespace than MIDN 1/C Abramoff used. The swarms are to move freely in the 3-D space and act accordingly with real-world physics, meaning gravity and altitude as well as the feasible turning rate on the aircraft. Figure 8.3 shows Thompson’s Python environment simulation. The swarm of drones in the upper left hand corner are colored in blue and represent the allied drone swarm. The swarm in the bottom right hand corner are colored in red and represent the enemy drone swarm. The circle around each drone represents the altitude despite the 2-D display. The bigger circles show a higher altitude which means the enemy swarm is higher than the allied swarm in Figure 8.3. MIDN 1/C Thompson fixed the 3-D aspects of the environment and integrate the drone into that space. He also worked with the turning rate of each drone to ensure the simulation meets the real-life drone specifications.

9 Environment Description

9.1 Unity Video Game Engine

We chose the Unity Video Game Engine to run the drone swarm simulator. Unity is an open source, 3D environment designed to allow video game makers to build and run scripts. Unity allows us to create and see the drones in 3 dimensions as well as use the camera system to view the battles in real time. These features are useful during the programming, training, and testing stages of this project as well as for the presentation of our results at the end of the year. Secondly, Unity has machine learning plug-ins that can be easily downloaded and integrated with the objects in the Unity scene. The Unity MLAgents program is described in Section 9.2. Finally, Unity is widely used for projects containing drone simulations, ML agents, and gathering data so there is a plethora of information, forums, and advice on the internet. These resources allow us to fix issues and develop the behavior we want easily because of the research available. [Haas 2014]

9.2 MLAgents Plug-in

The Unity MLAgents is a downloadable package, open source on Github, that comes with the Python code to create ML agents in Unity and train their brains using either the Proximal

Policy Optimization or the Soft Actor-Critic RL algorithms. The length of the training, gamma value, epsilon value, number of nodes and layers in the neural net, and other options can be changed in a config file in order to alter the training. There are also additional methods to better train agents including imitation learning, curriculum learning, and self play. MLAGents contains a C# script that connects the Python RL algorithms with the agent in the Unity scene. The script contains the following functions:

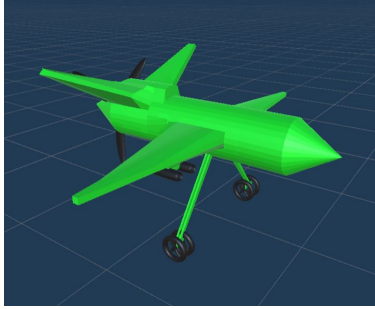
- Initialize: Sets up the Academy and the Environment for the Python scripts to run.
- Collect Observations: Takes information from the scene, like agent's position or other objects nearby, and send it to the Python ML scripts as the inputs to the neural network.
- On Actions Received: Receives a list of actions from the Python ML scripts that are the output of the neural network. The brain makes the agents perform these actions in the Unity environment. The actions are performed and the Unity environment is changed before the next call to Collect Observations.
- Set Reward: Sets the positive reward or negative penalty based on the results of the actions performed. This is used in the training to develop the neural network that produces the highest cumulative reward.
- Reset Parameters: Resets the scene after each run or episode of the scene. The scene ends when the agent completes the goal or dies from an action.
- Heuristic: Allows the agent's to be controlled by a human using the keyboard or by the a pre-programmed set of actions rather than by the output of the agent's brain.

9.3 The ARC Cluster

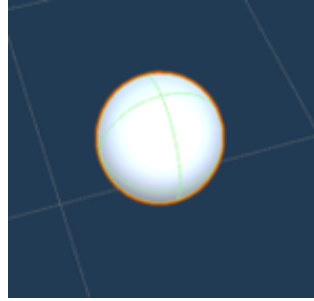
In order to run drone battle simulations in large quantities and to train the drone agents, we used the ARC Cluster. This supercomputer is run by the USNA Computer Science department and allows us to run jobs using Slurm script. There are 35 high performance computing nodes with 40 cores and 192 GB of memory each. We were able to run multiple simulations at the same time in order to collect the data of the most effective drone swarm tactics. We also decrease the time needed to train agents by parallelizing the computations needed to update the weights on the neural network. Overall, running the code on the ARC Cluster provided more data in a reasonable amount of time and and try more configurations in the agent training stage of the project.

9.4 Game Assets

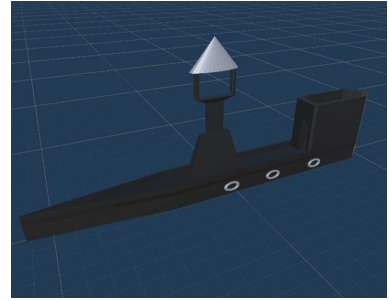
In Unity, game objects are “the fundamental objects in Unity that represent characters, props and scenery” [Technologies 2015]. The Unity environment for this project contains various game objects and scripts which are described below.



(a) Drone



(b) Bullet



(c) Ship

Figure 9.1: Unity Game Objects

- Scene: The scene contains a 500 x 500 meter floor. Objects (drones and bullets) will die when hitting the floor or leaving the bounds of the playing field. The Ship Attack and Defense Scene also contains a ship in the center of the playing field and resting on the water (floor).
- Drone: A drone (Figure 9.1(a)) is represented by a 1 x 1 x .5 meter rectangular prism. A 3D fixed wing drone was modelled by Andrew Gergal and attached to the drone game object that contains the scripts and mechanisms needed to control the drone. The model is for visual purposes and does not affect the flight pattern or interactions with other game objects. The drone flies forward at various speeds. It can change directions by turning or rotating to the left and right. It can change altitude by lifting or lowering the front of the drone up or down. The drone object will die and instantly disappear from the scene when any object, including the floor, a bullet, another drone, or the ship, collides with the rectangular prism that represents its body. Each drone can shoot bullet out of the bottom itself in the direction that it is moving at the time of the shot. A drone can be a part of a swarm or act independently. In either case, the drone can be assigned an individual target where the drone aims and flies towards that target.
- Bullet: A bullet (Figure 9.1(b)) is .1 meter radius sphere. When a drone shoots, a bullet will appear directly beneath the drone. The bullet travels with a speed of 2 m/s in the direction that the drone was traveling at the moment of the shot. The bullet will kill a drone on the collision and disappear or die when it reaches the floor or the edge of the playing field.
- Ship: The ship (Figure 9.1(c)) is a 20 x 20 x 100 meter capsule shape that sits in the middle of the playing field on the Ship Attack and Defense Scene. This ship roughly simulates an Arleigh Burke Class Destroyer in the water. A 3D ship was modelled by Andrew Gergal and attached to the ship game object that contains the scripts and mechanisms needed to count the number of drone hits on the ship. The model is for visual purposes and does not affect its interactions with other game objects. The ship does not move or have any weapons. The only feature of this ship is to count the number of hits that the offensive team achieves in the Ship Attack and Defense

Scene. A hit is counted when one of the offensive drones collides with the ship. This is designed to simulate the kamikaze style drone attacks where the drone is filled with explosive and intend to fly into the ship. Although important, the bullet hits on the ship are ignored in this project.

- **Swarm:** A swarm game object is a list of drones from the same team. It is important to clarify that in the other sections of this paper, the word swarm refers to all the drones on a team no matter how they operate or interact with one another. Here the swarm game object is a part of the pre-programmed tactics where the drones that are designated as a member of the swarm game object take instructions for the swarm game object and move collectively. When a team has multiple swarm game objects or drones that are not a part of the swarm game object, it is a decentralized network, described in Section 7. Each swarm game object acts independently from others, but gives the instructions to each of its drones. The drones in the swarm game object will continually move towards the center of the swarm game object to maintain cohesion. Simultaneously, the drones will keep a certain distance from the other drones in the swarm game object to maintain separation. The swarm game object is given a target and will, as a group, aim and fly towards that target. The drones in the swarm game object will generally fly in the same direction with the same speed with minor exceptions needed to maintain the structure of the swarm or individually target different drones. Even drone that are operating as a swarm game object can aim at different points in space or different targets according to the drone's individual assignment/task.
- **Tactic:** Pre-programmed tactics direct the drones on the team. The tactic can be thought of as a centralized network as described in Section 7 because the tactics direct each of the drones on the team to behave a certain way. The various programmed tactics available are listed and described in sections 10.1 and 10.1.2. The tactic dictates where the drones fly and when they shoot. The drone will use the flocking tactic, where they fly towards the center of the enemy team's drones, until the center of the friendly team's drones are less than 250 meters away from the center of the enemy team's drones. Within 250 meters, the team will operate using the tactic specified in the config file.
- **Simulation:** The simulation creates two teams of drones, one green and one red. The config file allows us to input a tactic (listed in sections 10.1 and 10.1.2) and a number of drones for each team. The simulation will create any swarms necessary and indicate the roles for each drone. Then, the simulation will make both teams take off, battle according to their prescribed tactics, and end when one team eliminates all the drones on the other teams. If the battle is not ended after 60 seconds, the battle will timeout and result in a tie. The simulation repeats for as many rounds as listed in the config file.

9.5 Swarm vs. Swarm Scene

The simulation begins with two swarms of any number of drones. The drones on each team are initialized in a random location on their respective side of the playing field. The simulation starts as both teams take off. Each team executes the given tactic, either pre-programmed or the agent learned. Wins, losses, and ties are counted for each team if multiple rounds are run.

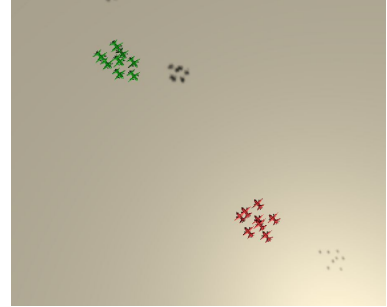


Figure 9.2: Swarm vs. Swarm Scene

9.6 Ship Attack and Defense Scene

The simulation begins with two swarms of any number of drones. The defensive team is initialized in the center of the ship that is placed in the center of the playing field. The ship is stationary and does not return fire, but it will count the number of drone hits it receives. The offensive team is initialized in a random location of the playing field that is at least 200 meter away from the center of the ship. The simulation starts as both teams take off. Each team executes the given tactic, either pre-programmed or the agent learned. The total number of drone hits on the ship and the number of defensive drones remaining are counted for each team if multiple rounds are run.

10 Pre-Programmed Tactics

10.1 Description of Tactics

After the environment, drones, and swarms were created in Unity, we began programming various tactics that each team could use in simulated battles against the enemy. We implemented the tactics that MIDN Abramoff and Thompson had used in their Python simulator as well other researched tactics and new tactics.

10.1.1 Swarm vs. Swarm Scene

The descriptions below are from the perspective of the friendly team that is using the given tactic. The tactics are pictured in Figure 10.1. The enemy team in these descriptions are independent and can be using any of the available tactics. The list of tactics that are implemented include:

- **Flocking:** All drones on the friendly team form a swarm. The swarm aims at the center of all the enemy drones.
- **Select Nearest:** All drones on the friendly team form a swarm. The swarm aims at the center of all the enemy drones. Within the friendly swarm, each drone individually targets the closest enemy drone to itself. This individual target is updated every frame.

- **Assign Nearest:** All drones on the friendly team form a swarm. The friendly swarm aims at the center of all the enemy drones. Within the friendly swarm, each drone is assigned one enemy drone to individually target. This assignment is updated every frame. The enemy drones are ordered from closest to farthest (with respect to the center of the friendly swarm). The closest enemy drone is assigned to the friendly drone with the minimum distance to that enemy. The next closest enemy drone is assigned to the friendly drone with the minimum distance to that enemy that does not have an assignment yet. This process continues until there are no more friendly drones without assignments. If there are fewer friendly drones than enemy drones, some enemy drones will not be targeted. If there are more friendly drones than enemy drones, the additional friendly drones will target the closest enemy drone.
- **Rabbit Hold And Wait:** One friendly drone is designated as the rabbit which flies towards the center of the enemy drones at a higher speed than the remaining friendly drones. When the rabbit is within a certain distance of the enemy drones, it will turn to the left up to 90 degrees. All other drones (besides the rabbit) on the friendly team form a swarm. The friendly swarm aims at the center of all the enemy drones. Within the friendly swarm, each drone is assigned one enemy drone to individually target according to the Assign Nearest behavior. As the rabbit and the swarm approach the enemy drones, the swarm will slow its speed to remain out of shooting range while the rabbit creates the diversion with the left turn. After the diversion is made, the swarm will continue towards the enemy drones and engage with Assign Nearest behavior.
- **Double Swarm:** The friendly drones are split into 2 smaller swarms. One swarm flies to the left of the enemy drones and the other swarm flies to the right of the enemy drones. When both swarms have turned 50 degrees in their respective directions, the swarms turn inward and aim at the center of the enemy swarm. The two smaller swarms converge on the enemy simultaneously and shoot as they fly towards the center of the enemy drones.
- **Circle Flock:** Each of the friendly drones are assigned a spot along a 100 meter circle that surrounds the enemy swarm. The friendly drones fly directly to their assigned spot. When all drones arrive, they will form a 100 meter circle around the center of the enemy drones and turn inward to converge on the enemy drones. As the friendly drones fly towards the center of the enemy drones, they shoot and aim at the enemy drones in the circle.

10.1.2 Ship Attack

We implemented the tactics from the Swarm vs. Swarm Scene (Section 10.1.1) and modified them to work within the Ship Scene (Figure 10.2). The descriptions below are from the perspective of the friendly team that is using the given tactic from the offensive position. The tactics are pictured in Figure 10.2. The enemy team in these descriptions are independent and can be using any of the available tactics. The list of tactics that are implemented include:

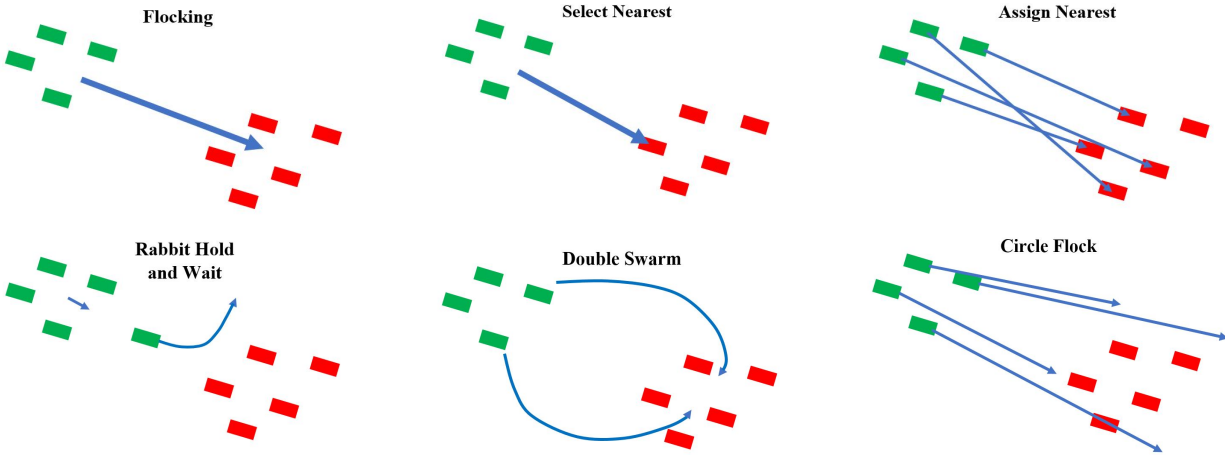


Figure 10.1: Swarm vs. Swarm Tactics

- **Offensive Flocking:** The offensive enemy drones will attack the closest portion of the ship and will disregard the enemy defense in the way.
- **Offensive Select Nearest:** The offensive enemy drones will attack the closest portion of the ship if any of the following criteria are met. Ship attack criteria- distance between the center of offensive drones and the closest portion of the ship is less than 10 meters, distance between the center of the offensive drones and the closest portion of the ship is less than the distance between enemy defensive drones and the closest portion of the ship, distance between the center of the offensive drones and the closest portion of the ship is less than the distance between the center of both teams, or the number of offensive drones minus the number of defensive drones is more than the number of defensive drones divided by three. If the ship attack criteria is not met, then the offensive drones will execute the Select Nearest tactic. The ship attack criteria is re-evaluated at each frame update.
- **Offensive Assign Nearest:** The offensive enemy drones will attack the closest portion of the ship if any of the ship attack criteria are met. If the ship attack criteria (defined in the Select Nearest description) is not met, then the offensive drones will execute the Assign Nearest tactic. The ship attack criteria is re-evaluated at each frame update.
- **Offensive Rabbit Hold and Wait:** The offensive enemy drones will attack the closest portion of the ship if any of the ship attack criteria are met. If the ship attack criteria (defined in the Select Nearest description) is not met, then the offensive drones will execute the Rabbit Hold and Wait tactic. The rabbit drone will turn away from the ship to pull the defense away. The ship attack criteria is re-evaluated at each frame update.
- **Offensive Double Swarm:** The offensive enemy drones will attack the closest portion of the ship if any of the ship attack criteria are met. If the ship attack criteria (defined in the Select Nearest description) is not met, then the offensive drones will execute the Double Swarm tactic. The ship attack criteria is re-evaluated at each frame update.

- **Offensive Circle Flock:** The offensive enemy drones will attack the closest portion of the ship if any of the ship attack criteria are met. If the ship attack criteria (defined in the Select Nearest description) is not met, then the offensive drones will execute the Circle Flock tactic. The ship attack criteria is re-evaluated at each frame update.

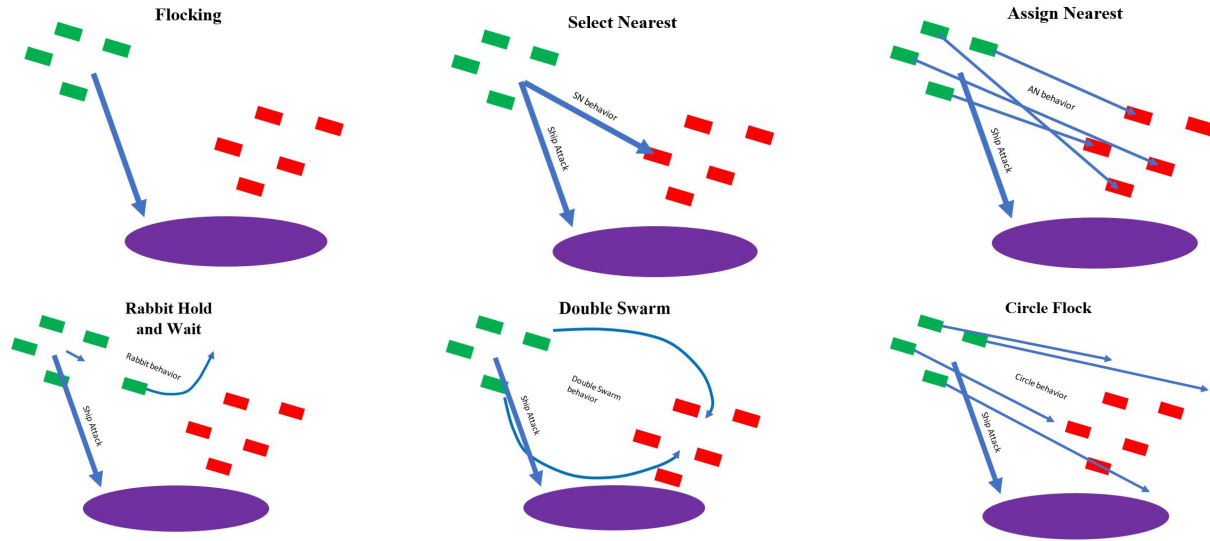


Figure 10.2: Ship Attack and Defense Tactics

10.1.3 Ship Defense

In the Ship Attack and Defense Scene, the defensive team can use any of the tactics in order to eliminate the offensive drones. The execution of the tactics is the same as the drones in the Swarm vs. Swarm scene (described in Section 10.1.1). The defense disregards the ship and is set to destroy all of the offensive drones as fast as possible.

10.2 Swarm vs. Swarm Comparison of Tactics

The plots in Figure 10.3 show the results of the comparison of the six different tactics described in Section 10.1. The previous research showed that the size of the swarm was the predominant factor in their effectiveness. Bigger swarms mean more wins and the team with less drones could not use tactics to overcome their disadvantage. In order to determine whether pre-programmed tactics were effective, we tested each of the tactics against each of the other tactics and found the average behavior of 1000 rounds of battle.

One goal of this research project is to determine if drone swarm tactics are effective against other swarms and if research in drone counter-swarm tactics could be fruitful. The results show that there are effective tactics that can be used to beat other drone swarms. Analyzing the top left plot in Figure 10.3 shows the effectiveness of each tactic in battle. Overall, Assign Nearest is the most effective tactic out of the 6 we implemented and tested. When Assign Nearest faces a similarly sized swarm, it will win at least 66%, and up to 110%,

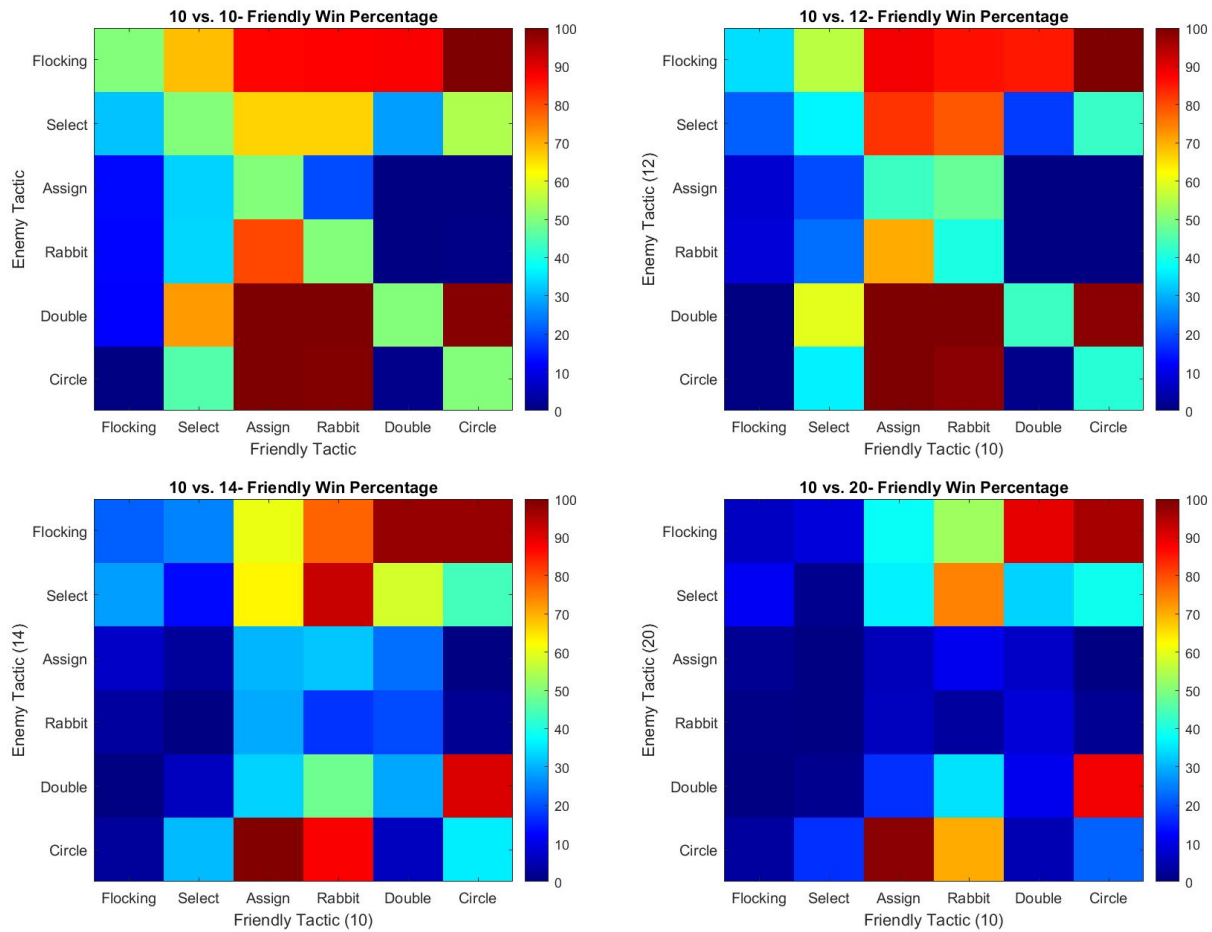


Figure 10.3: Swarm vs. Swarm Results- The friendly swarm always had 10 drones and ran the tactic labelled on the X axis. The enemy swarm had 10, 12, 14, or 20 drones which is labelled in the title of the graph. The color of the square matches the legend and represents the percentage that the friendly tactic (X-axis) would win against the enemy tactic (Y-axis). A win is defined as the friendly team killing all of the enemy drones while keeping at least 1 friendly drone alive. The number of ties were so small that half of the tied rounds were added to the number of rounds won in order to make these graphs. These results were the average performances of 1000 rounds.

of the time. The next most effective is Rabbit Hold and Wait. It has a low win percentage when competing against an Assign Nearest swarm but will otherwise win at least 59% of its battles. The Rabbit Hold and Wait can even beat Double Swarm and Circle Flock 100% of the time. In general, Select Nearest was the third most effective tactic as it can provide at least a decent chance of surviving an attack from any of the other tactics. Select Nearest has a high win percentage against Flocking and Double Swarm, has a 45% chance of beating a Circle Flock swarm, and can win against Assign Nearest or Rabbit 30% of the time. The Circle Flock tactic is especially effective against Flocking or any tactic that targets the center of the opposing swarm as none of the drones are actually in the center. The Circle Flock and Double Swarm are more specialized tactics that can significantly beat a simple swarm and can be used in specific circumstances. These tactics are not as versatile as the others but can still be useful in the development of an arsenal of drone swarm tactics. Flocking is not an effective tactic when compared to the other tactics, but is simple to implement. This may be used by an enemy who does not have the resources to use other tactics, and it is beneficial to compare other tactics against a Flocking swarm baseline.

Another question we wanted to test is whether any of these drone swarm tactics were still effective if the swarm was outnumbered by the enemy. In order to do this, we ran the same test but increased the number of enemy drones to 12, 14, and 20, respectively. These results are plotted in Figure 10.3. The plot on the top right shows that even when the enemy swarm is 20% larger, we still see similar win percentages. The percentages are slightly lower with some tactics, but the friendly (smaller) swarm is still very effective. Assign Nearest and Rabbit Hold and Wait tactic still win over 60% of their battles. They can still win almost 100% of the time against Double Swarm or Circle Flock. Additionally, the other tactics are able to have some success in beating the bigger swarm and are only slightly affected by the additional enemies. The lower left graph in Figure 10.3 displays the results from a 10 vs. 14 drone swarm battles. Despite the enemy swarm starting with 40% more drones, the friendly team is able to defeat an enemy swarm that is using the Flocking or Select Nearest over 50% of the time. There are also specific tactic comparisons that result in 90% or 100% despite the difference in size. Lastly, we tested the 6 tactics against a swarm that was twice its size. If the compared size of the swarms was the most important factor in the outcome of the battle, then we would expect the 20 drone enemy swarm to beat the 10 drone every time. However, the bottom right plot in Figure 10.3 shows otherwise. Some of the tactics are plotted in orange, red, and maroon which represents a win percentage of over 70% even though the enemy is twice as large. Additionally, the tactics can beat Flocking and Select Nearest over 30% of the time. These results prove that the size of the swarm can affect the swarm's win percentage, but this difference can be overruled with effective tactics.

Based on the results of this research, Assign Nearest is the most effective tactic in most situations. However, if the drone swarm could change tactics given prior knowledge of the situation, the enemy's size, and the enemy's tactic, then the drone swarm could be even more effective. It would be difficult to have the level of foresight needed to pick the best pre-programmed tactic for the battle. It would be more realistic and effective to have an agent that can adapt in the midst of a battle. An adaptable agent could use the aforementioned pre-programmed tactics in the best situations and reap the benefits of all of the tactics used together. This is the next step in the development of drone swarm tactics and the idea behind the RL trained agent tactic featured in Section 11.3.

10.3 Ship Attack and Defense Comparison of Tactics



Figure 10.5: Ship Attack and Defense Scene

The plots in Figure 10.4 show the results of the comparison of the six different tactics described in Section 10.1.2 and represents the average behavior of 1000 rounds of battle. The NPS research project that was mentioned in Section 2 tested the current defenses on an Arleigh Burke Class Destroyer against an offensive swarm of 8 drones. In order to compare the effectiveness of counter-swarm tactics with the current defenses, we ran a similar test. The top left plot in Figure 10.4 shows that most of the graph is represented in blue which corresponds to less than 1 drone passing the defense and hitting the ship. There are a few offensive/defensive tactic pairs that result in around 4 ship hits, but in a majority of the situations the defensive swarm protects the ship better than the current Arleigh Burke

defenses (3.82 ship hits) and the simulated improved defenses from that project (1.12 ship hits). Even more impressive is the fact that for many of the rounds, including almost all of the rounds with the defense running Assign Nearest or Select Nearest the number of ship hits is 0 (or less than .25). These results show that drone swarms are an effective defense against incoming drone swarms.

Assign Nearest, Select Nearest, and the Rabbit Hold and Wait tactic were the most effective defensive strategies. It is not surprising that Assign Nearest is effective in the Ship Attack and Defense scene as it was also effective in the Swarm vs. Swarm scene. The job of the defense is to eliminate the enemy drones as quickly as possible and Assign Nearest does that well. The same is true for the Select Nearest tactic. Rabbit Hold and Wait seemed to be effective as a defensive tactic because the majority of the swarm stayed close to the ship and waited for the offense to approach. This meant that it was less likely for the offense to get “behind”, closer to the ship, than the defense. The idea of staying in between the offense and the object that must be protected is a fundamental aspect of defense in most sports and should be incorporated into future development of defensive tactics.

Another consideration that we tested is whether these programmed defensive tactics were effectiveness against larger offensive swarms. Could the tactical advantage overcome the challenge of being outnumbered? The other 3 graphs in Figure 10.4 show the number of ship hits when a 10 drone defensive swarm is attacked by a 12, 14, and 20 drone offensive swarm (as noted by the title of the plot). The majority of these plots are colored dark blue which means most of the time less than 1 drone is getting past the defensive team and hitting the ship. This is even more impressive since the offensive swarm is 20%, 40%, and 100% larger than the defensive swarm. Many of the rounds even results in 0 (or less than .25) ship hits. This is very promising considering other enemies are building swarms with

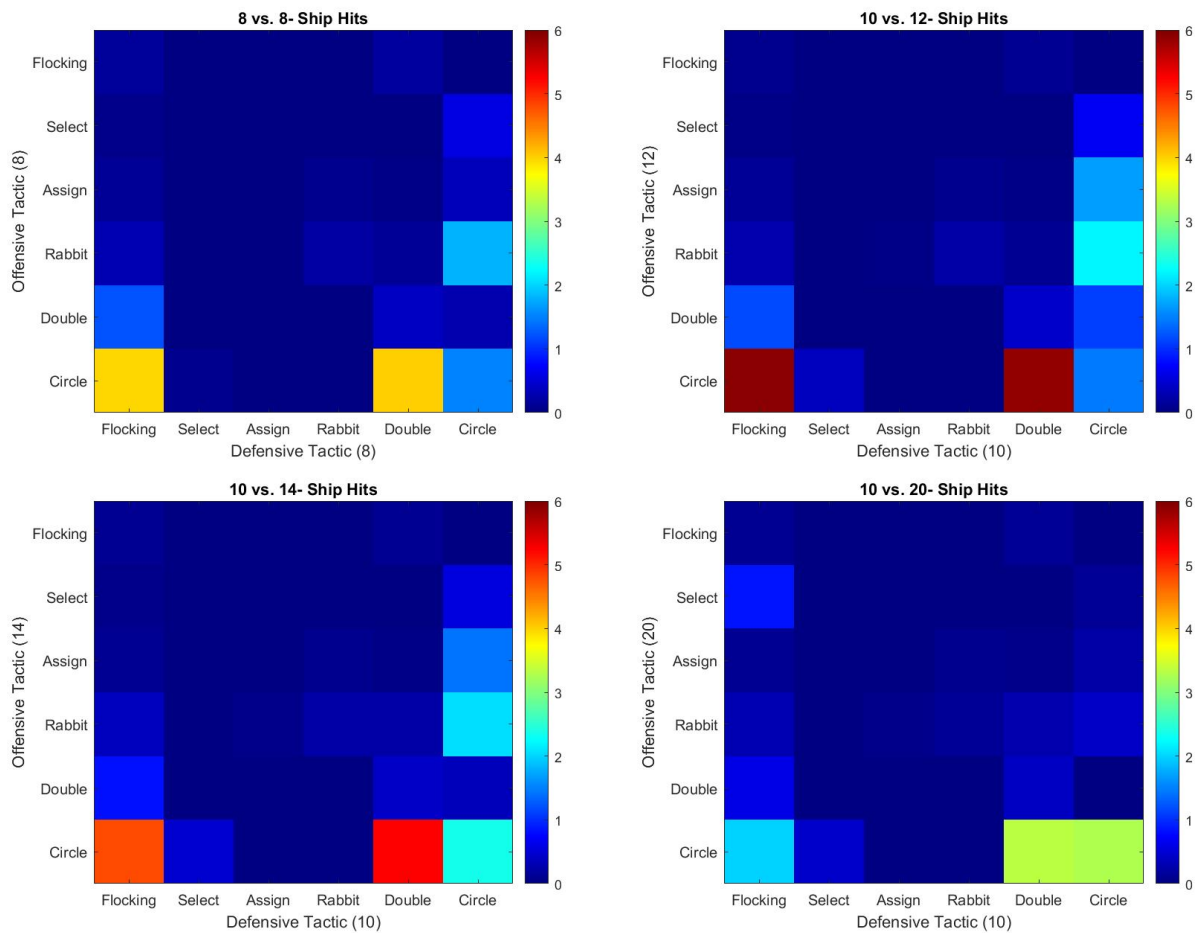


Figure 10.4: Ship Attack and Defense Results- The defensive swarm had either 8 or 10 drones and ran the tactic labelled on the X axis. The offensive swarm had 8, 12, 14, or 20 drones which is labelled in the title of the graph. The color of the square matches the legend and represents the number of offensive drones (Y-axis) that pass the defensive team (X-axis) and hit the ship. The scale of the legend is from 0 to the number of offensive drones at the beginning of each round. These results were the average performances of 1000 rounds.

hundreds or thousands of drones. It is interesting to analyze the graphs for the tactics that allowed a greater number of drones to hit the ship. We thought that the more drones on the offense, the more ship hits they could achieve. In some cases, the defense was more effective against larger swarms. This could be the results of the offense tactic operating differently with more drones. For example, the circle created in the Circle Flock tactic was too large to be effective. More analysis and tests could be beneficial in determining the reason that tactics are effective and how to improve them.

These pre-programmed tactics are effective in defending the ship against these 6 pre-programmed tactics. As enemies continue to develop new offensive capabilities, it is important that the US continue to develop defenses. The next step in the development of defensive swarm tactics is to build a RL trained agent that can adapt to the current situation and actions of the enemy in real time.

10.4 Analysis of Pre-Programmed Tactics

The results of the pre-programmed tactics tests show that it is possible to use tactics to gain an advantage over an enemy swarm even when outnumbered by 20%-100%. Pre-programmed drone swarm tactics are effective in both defeating enemy swarms and protecting an area like a ship. In order to use the advantages that come from programmed tactics and to improve upon these capabilities, we should develop a tactic/method that make the swarm adaptable to the current environment and resilient in battle. This is where reinforcement learning can be used to train a new “Agent” tactic that can beat all of the pre-programmed tactics and better our drone swarming capabilities.

11 RL Agent Tactics

11.1 Agent Design

Many different designs were trained and tested over the course of this project. The agent takes input from the environment, runs it through a neural network using Proximal Policy Optimization, and makes decisions (output) to take certain actions. The reward structure is used to alter the policy to better the reward as the training progresses. The design of the most effective agent is described in the following sections.

11.1.1 Distributed Network

The current agent uses a distributed network to make decisions. Previous agents would use a centralized network where there was 1 agent with 1 brain telling each of the 10 drones what to do. This was ineffective because there were so many inputs and outputs to the neural network that the problem was too complex and the network failed to converge on a good policy. We found a better approach is to give each drone an individual network which is the exact same network as all of its teammates. This allows the agent to take in only the most relevant information and make a decision based on its situation and what is happening around it. In the long run, this is more reliable because there is no large command center controlling everything that can be destroyed or disrupted by the enemy. It also allows

individual drones to die without causing major problems for the rest of the drones that are still alive. When 1 drone dies, the other drone continue operating with 1 less teammate instead of losing communication, breaking the network, ending the mission.

11.1.2 Inputs

The inputs to the agents are observations from the environment. The observations are the inputs of the neural network of the reinforcement learning algorithm. This is one of the areas that we can adjust in order to train the agent tactic. Over the experimentation phase of this project, we used agents that took absolute positions, number of drone remaining, centers of the swarms, the yaw/pitch/speed of the drone, whether the drone was alive or dead, relative 3D positions as input. These agents were not able to develop intelligent behavior.

The current and most effective agent has 63 inputs to the neural network for the Swarm vs. Swarm Scene and 66 inputs to the neural network for the Ship Attack and Defense Scene. The agent takes the following inputs:

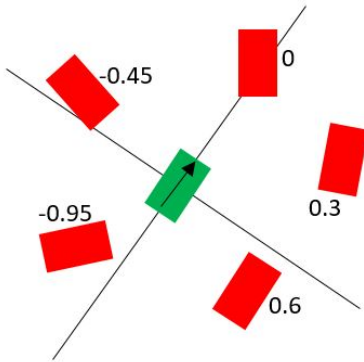


Figure 11.1: Relative Bearing Example

- Ship Attack and Defense Scene Only
 - Relative bearing to the closest point on ship
 - Range to the closest point on the ship
- For the 3 closest boundaries/walls
 - Relative bearing to the closest point on that boundary/wall
 - Range to the closest point on that boundary/wall
- For the 10 closest friendly drones (on same team, in order from closest to farthest)
 - Relative bearing to the friendly drone
 - Range to the friendly drone
- For the 10 closest enemy drones (on opposing team, in order from closest to farthest)
 - Relative bearing to the enemy drone
 - Range to the enemy drone

All inputs are relative to the individual drone as shown in Figure 11.1. Any drones that are dead or farther away than the 10 drone limit are unseen by the agent. If there are less than 10 friendly or enemy drones still alive, then an observation of 1 is input as a placeholder. All inputs are normalized to be between -1 and 1.

11.1.3 Reward Structure

The reward structure gives positive and negative feedback that changes the policy as the agent training progresses. The goal of the RL algorithm is to change the parameters of the neural network so that they produce the highest cumulative reward possible. This reward structure encourages good behavior, penalizes bad behavior, and defines wins and losses. This is one of the main ways to influence the agent's training and produce intelligent behavior from the RL algorithms. The current reward structure is as follows:

- Kill (one enemy drone is killed): +10 per drone
- Teammate's Kill (the teammates of the drone that successfully kills an enemy each receive a small reward): +1 per drone
- Die (self drone dies): -5 per drone
- Friendly Fire (occurs when a drone fires and hits its teammate): -5 per drone
- Boundary (one friendly drone hits the edge boundary or the floor of the environment): -50 per drone
- Angle (the yaw and pitch of a friendly drone are each pointed at an enemy drone plus or minus 5 degrees): +0.2 per update frame per drone
- Timeout (at least one drones remains on each team and the 500 second timer runs out): -30 and end of round

11.1.4 Outputs

The outputs of the agent's neural network are translated into the actions the agents will take. There are two types of outputs from the neural net, discrete and continuous. We have experimented building agents with both types of outputs and found that continuous outputs provide the best results.

The continuous outputs from the neural net provide a list of floats between -1 and 1. Similar to the inputs, the outputs are the relative bearing to the position to which the drone should aim. The drone shoots automatically every 3 frame updates. The outputs were normalized, so they are multiplied by 180 and 90 respectively to create to angle that the drone should target. Because the agent's brain is only controlling 1 drone, there are 2 floats output from the neural network. Currently, the continuous outputs of the agents are as follows:

- Relative bearing on horizontal plane (yaw)
- Relative bearing on vertical plane (pitch)

11.2 Agent Training

The RL training techniques that we used to train our current and highest performing agent are listed below.

11.2.1 K-frame Updates

First, the agent requests a decision every k frames instead of every frame. At the beginning of the training, the k is initialized to 5 and decreases to 4 and eventually 3 as the agent improves. If the agent requests a decision at every frame, then the agent has to collect observations, compute the output through the neural network, and update the parameters of the neural network based on the rewards received. This is a great deal of work and does not need to occur at every frame update. The agent trains faster when the k value is higher because the agent can play/experience the world more for the same amount of work. Additionally, the output for the agent is a relative bearing to aim towards. When there was a new output/target at every frame update, the drone agent did not have enough time to turn towards or make any progress towards the intended target. This made it very difficult for the agent to learn because it did not have a chance to see the results or rewards of each output. In the early training stages especially, the outputs are very random and the agent needed a few frame to turn towards the output target in order to distinguish between good and bad actions. In the later training stages, the outputs were less random and more refined, so the k value could be smaller and the agent could have more control over its intended target. The effects of the k value could play a larger role in future research and should be analyzed more.

11.2.2 Imitation Learning

Imitation learning stems from the method typically used to teach a child a new task. First, an expert will demonstrate the task to the child, and the child will only watch the demonstration. The expert will show the child various situations that they may find themselves in and what actions they should take in those situation. Then the child can begin copying the expert's actions until the child learns enough to make their own decisions and learn from the effectiveness of their own actions. This system of training is paralleled in the agent training for this project.

First, we recorded various demonstrations of successful drone swarm tactics. The most effective agent used the recordings of the Select Nearest tactic as there were negative consequences from too much random information. The demonstrations were varied in the starting positions of both swarms, the starting rotation of both swarms, and the frequency of shots fired from the agent swarm. We recorded separate demonstrations in the Swarm vs. Swarm Scene and the Ship Attack and Defense Scene.

When the agent trained, it received a reward for Behavior Cloning. This reward was added to the cumulative reward with a half of the strength as the external reward (reward

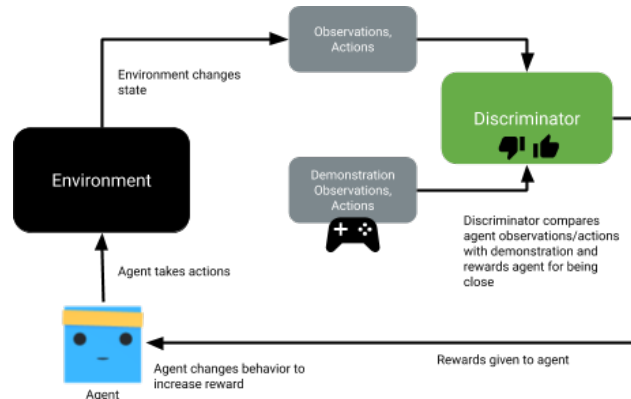


Figure 11.2: Unity MLAgents- GAIL

structure from in environment as described in Section 11.1.3). The agent receives a higher reward for copying or cloning the actions of the demonstrations. The demonstrations are like a bank of inputs with their corresponding outputs, so the agent can find the closest entry in that bank and take the corresponding action.

In order to improve on the results of copying the demonstrations, we incorporated Generative Adversarial Imitation Learning (GAIL). In GAIL, a second neural network called discriminator is trained to predict whether an action was the result of the agent's decision making or the output of the demonstration. The discriminator will reward the agent for making decisions similar to the demonstration at a strength of three-quarters of the external reward. The discriminator will get better at determining the difference between demonstration actions and agent's decisions. As the training progresses, the discriminator will improve and the agent will improve from the GAIL and external (environmental) rewards it is receiving. The agents is performing better by acting like the demonstrations until it can learn more from the environment itself. This is how the agent can outperform the demonstrations. [Teng 2020]

Lastly, the agent built on the results of Behavior Cloning and GAIL by using the external rewards (based on the programmer's reward structure as described in Section 11.1.3). When the agents completes an action that accomplishes something something good in the environment, it gets a positive reward and reinforces that action. The agent is also penalizes for completing actions that result in bad outcomes in the scene. Over the training, the agent can find and complete more positive actions and increase its cumulative reward. In this project, the positive rewards like killing an enemy drone are quite sparse which would cause difficulty training if the agents can never find the reward in the first place. External rewards may not be enough to incentivize intelligent behavior, but when built on top of demonstration learning, it is very effective.

11.2.3 Curiosity Reward

One of the major trade-offs in RL is between exploration and exploitation. Exploration is when the agent tries new actions and finds new states of the environment, and hopefully new and better positive rewards. Exploitation is when the agent uses the current policy and makes a decision based on the outputs of the neural network. If the agent only continues acting on what it knows, then the agent will be stuck in a local maximum and might miss the opportunity to find bigger and better rewards elsewhere in the environment. In order to prevent this and to promote the most intelligent behavior possible, we incorporated a curiosity reward where the agent gets a small reward just for trying a new action and getting to an environment that has not yet been seen.

11.2.4 Curriculum Learning

Another RL technique known as Curriculum Learning is based on the way an expert would train a child to complete a task. It would be very difficult for a young child who is new to math to begin with Calculus problems. Instead, the training begins with simple tasks like counting before progressing to addition, Algebra, Calculus, and eventually research in Math topics. In the beginning stages of this project, we put untrained agents in a large space against an ef-

fective enemy. The drone agent did not learn anything because it could not even figure out to aim at an enemy drone. The drone agent was just helplessly running into the walls or the floor and being killed by the effective enemy. With the Curriculum Learning, we began training the drone agents in simpler environments and moved to the next level once the previous task was mastered. The level of mastery is measured by the mean reward per decision hitting a certain threshold.

First, we limited the Bounds of the play area. It began at 100 and eventually reached 500 meters. The small area kept everything compact and increased the likelihood that a drone agent would hit an enemy drone and gain the positive reward from that action.

Second, we initially started the drone agent swarm and the enemy swarm close to each other at around 75 meters. As training progressed, the starting positions grew to around 400 meters away from each other.

Third, we started the drone agents facing the enemy swarm. As the agent learned to shoot straight and hit the enemy directly in front, we would point the starting drone agents 10 degrees away from the enemy. The starting rotation eventually reach 120 degrees to either direction away from the enemy drone.

Lastly, we did not train the beginner agent against a drone swarm running one of the effective tactics as described in Section 10.1. We made a new enemy tactic called Flocking Slow. This tactic create a very slow moving and dumb enemy that moves towards the center of the drone agents swarm and does not shoot at them. The next step for future research is to increase the capabilities of the enemy and train the drone agents to defend themselves and fight the enemy simultaneously.

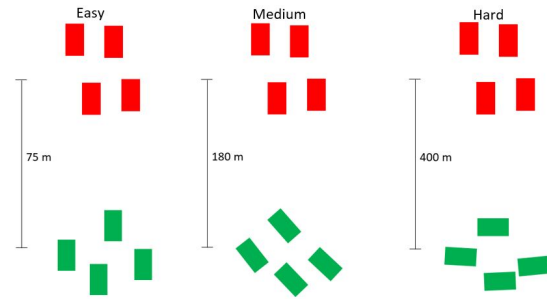


Figure 11.3: Curriculum- Starting Positions

11.2.5 RL Algorithm Configuration

There are two RL algorithms available in the Unity MLAgents plugin, PPO and SAC (described in Section 5.3). Both algorithms have configuration files available in order to change the hyper-parameters, reward structure, and size and shape of the neural network. These changes alter the speed and stability of the training and can affect whether the training is effective in developing intelligent behavior. For this project, we only used the PPO algorithm. The SAC is worth experimenting more in future research but it was not providing better training than PPO, and we decided to focus on only 1 algorithm for the remainder of this project. The current configuration of the PPO algorithm for the Swarm vs. Swarm Scene and the Ship Attack and Defense Scene are shown in Figure 11.4. The configuration files are very similar between the 2 scenes with the only differences being the demonstrations used and the max steps.

```

behaviors:
  DroneML:
    trainer_type: ppo
    hyperparameters:
      batch_size: 64
      buffer_size: 12000
      learning_rate: 0.0003
      beta: 0.01
      epsilon: 0.3
      lambda: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: true
      hidden_units: 64
      num_layers: 3
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
      gail:
        strength: 0.75
        demo_path: TrainEnv/SlowDemos
      curiosity:
        gamma: 0.99
        strength: 0.25
        encoding_size: 256
        learning_rate: 0.0003
    behavioral_cloning:
      strength: 0.5
      demo_path: TrainEnv/SlowDemos
    keep_checkpoints: 5
    max_steps: 1720000
    checkpoint_interval: 50000
    time_horizon: 1000
    summary_freq: 10000
    threaded: true

```

```

curriculum:
  DroneML:
    measure: reward
    thresholds: [2,4,6,8,10,12]
    min_lesson_length: 1000
    signal_smoothing: true
    parameters:
      boundslimit: [100,100,200,300,400,500,500]
      Kvalue: [5,5,5,4,3,3,3]
      Turn: [0,10,20,30,45,90,120]

```

(a) Swarm vs. Swarm Scene

```

behaviors:
  DroneML:
    trainer_type: ppo
    hyperparameters:
      batch_size: 64
      buffer_size: 12000
      learning_rate: 0.0003
      beta: 0.01
      epsilon: 0.3
      lambda: 0.99
      num_epoch: 3
      learning_rate_schedule: linear
    network_settings:
      normalize: true
      hidden_units: 64
      num_layers: 3
      vis_encode_type: simple
    reward_signals:
      extrinsic:
        gamma: 0.99
        strength: 1.0
      gail:
        strength: 0.75
        demo_path: TrainEnv/ShipDemos
      curiosity:
        gamma: 0.99
        strength: 0.25
        encoding_size: 256
        learning_rate: 0.0003
    behavioral_cloning:
      strength: 0.5
      demo_path: TrainEnv/ShipDemos
    keep_checkpoints: 5
    max_steps: 2200000
    checkpoint_interval: 50000
    time_horizon: 1000
    summary_freq: 10000
    threaded: true

```

```

curriculum:
  DroneML:
    measure: reward
    thresholds: [2,4,6,8,10,12]
    min_lesson_length: 1000
    signal_smoothing: true
    parameters:
      boundslimit: [100,100,200,300,400,500,500]
      Kvalue: [5,5,5,4,3,3,3]
      Turn: [0,10,20,30,45,90,120]

```

(b) Ship Attack and Defense Scene

Figure 11.4: RL Configurations

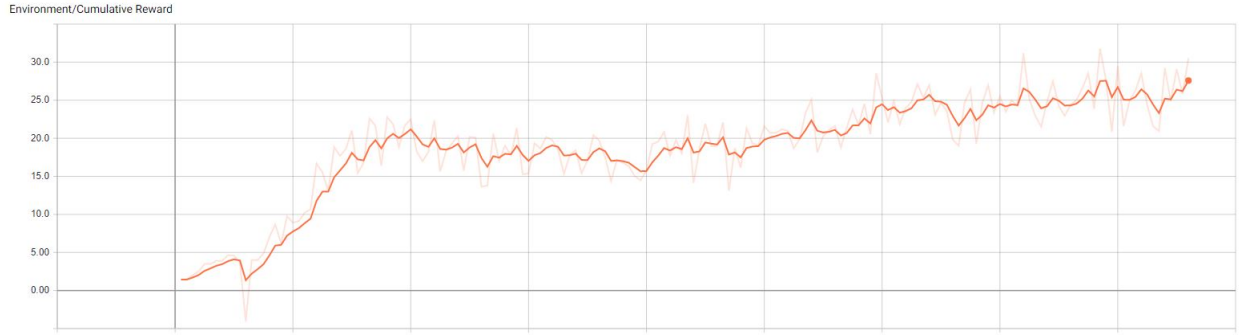


Figure 11.6: Swarm vs. Swarm RL Agent Training- This plot shows the cumulative reward over the duration of the agent’s training.

11.3 Swarm vs. Swarm Results

The next step in the development of effective swarm tactics is to build a RL trained agent that takes input from the environment and provide the actions at each step. The RL agent is built according to the specifications in Section 11.1. So far, the agent has not been effective in terms of defeating the enemy using a pre-programmed tactic, but there have been improvements to the agent’s behavior based on the work he had completed. The cumulative reward of the agent during it trained is plotted in Figure 11.6.

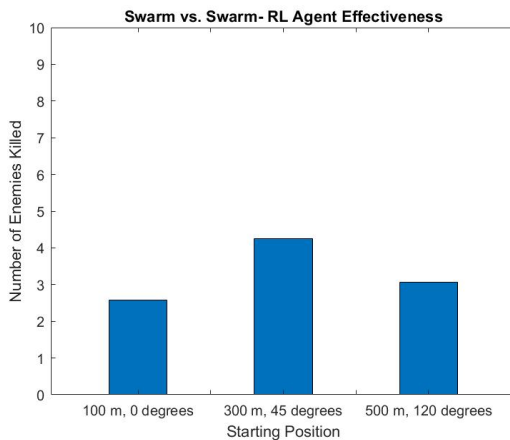


Figure 11.5: Swarm vs. Swarm RL Agent Kills

Previous researchers as mentioned in Section 8.4 attempt to use RL to develop an “Agent” drone swarm tactic. However, their agent drones learned to flee the scene and fly away as fast as possible in order to avoid being killed by the enemy. Similarly, the first agents that we developed in this project learned a policy where they flew to the edge boundary of the playing field and died on impact with the edge. With the reward structure of that agent, being killed by an enemy was penalized more heavily than dying by hitting the boundary. After more training, the drone agent flies around randomly but in the general direction of the enemy drones.

The current and most effective agent attempts to imitate Select Nearest. At the start, the drones in a swarm and they turn towards the enemy swarm. They move forward as a group in the direction of the enemy swarm. The enemy swarm is running the Flocking Slow tactic and is moving towards the drone agents. When the 2 swarms collide the Agent team usually kills off a few of the enemy and flies past the rest of the enemy drones. At this point, the Agent drones will scatter around the play area looking lost and confused. They try to

turn back towards the enemy swarm and fly around aimlessly. The drone avoid the walls, the ceiling, and the floors, but they fly seemingly randomly until the time runs out.

In order to quantify the effectiveness of these RL trained drone agents, we tested the Agent against the Flocking Slow tactic and calculated the average number of enemy drones that the Agent was able to kill. The 3 tests were conducted in an easy, medium, and hard mode according to the start distance and starting rotation with respect to the bearing of the enemy swarm. We ran 100 rounds of testing and found the average performance. Figure 11.5 shows the results of these test. The Agent was able to kill between 2.5 and 4.5 enemy drones. Considering the Agent was not given any direction and it only relies on the inputs to its neural network, it is impressive that the Agent can kill any enemy drones.

Overall, the drone agent is exhibiting beneficial behavior but not an intelligent and effective tactic yet. Future research can add capabilities to this drone agent and improve its effectiveness in battles. When the drone agent's policy is an effective tactic, it can be tested and compared against the pre-programmed tactics in Section 10.1. The most recent drone agent shows that is possible to train and learn drone swarming tactics and future research is needed to fulfill this goal.

11.4 Ship Attack and Defense Results

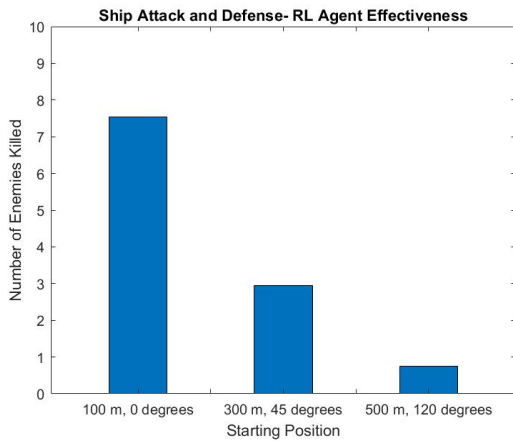


Figure 11.7: Ship Attack and Defense RL Agent Kills

The RL Agent in the Ship Attack and Defense Scene was built and trained like the Swarm vs. Swarm Agent, has the same goal to quickly kill all of the enemy drones, and exhibits similar behavior. The RL Agent is the defensive swarm in this scene. The cumulative reward of the agent during it trained is plotted in Figure 11.8. When the round starts, the drone agents will start as a swarm and turn towards the offensive swarm. The agents attempt to use Select Nearest, move forward towards the offensive swarm, and kill a few enemy drones as the 2 swarms collide. After this point, the defensive RL drone agents fly around randomly looking for an enemy to target. They avoid walls and the floor and do not fly into the ship. They do this unintelligent behavior until the round ends.

Similar to the Swarm vs. Swarm RL agent test, we calculated the average number of enemy kills the defensive RL Agent can get in 1 round in an easy, medium, and hard scene. We varied the starting position and the bearing to the enemy swarm. We ran 100 rounds of testing and found the average performance. We found that the RL Agent is able to kill over 7 offensive enemy drones in the easy mode. As the offensive team starts farther away, the effectiveness decreases. The defensive RL drone agents reach the random behavior before the offensive team reaches the ship. Further analysis and solutions are areas of future research.

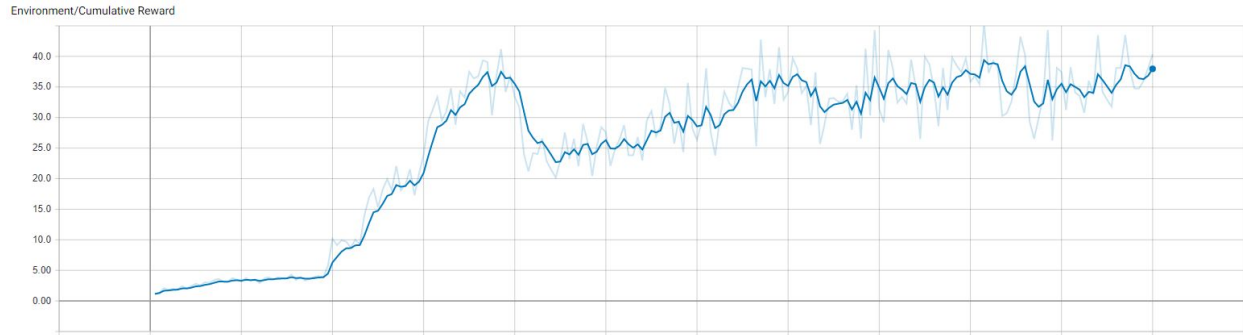


Figure 11.8: Ship Attack and Defense RL Agent Training- This plot shows the cumulative reward over the duration of the agent’s training.

11.5 Analysis of RL Tactics

Overall, the current state of RL Agent bodes well for the future of this research. There are more hyperparameters to the neural network and different factors to experiment with that can lead to more intelligent or even emergent behaviors (patterns emerging from randomness). Due to the time constraints of this research project, not all options were pursued and there is more research to come. Using the current RL agent as a foundation, the drone agents can spend more time training and learning until they formed an RL Agent tactic that can outperform the pre-programmed tactics.

12 Ethical Implications

It would be naive to complete a research project in the area of autonomous armed drone swarming tactics and simply hope that this technology be used ethically and for a good cause. As researchers and military members, it is important to think about the ramifications of our work and justify the need for this research and technology. This specific project is in the early stages of autonomous drone swarm research where the project is built in a simulation. Using these simulations to learn can expand knowledge on drone swarm in general and does not have to directly translate to the physical world. Pushing the bounds of a topic in a simulation or model can lead to other developments that are safer, more effective, and more ethical. We do not know where the research will lead until we pursue it. The unethical implementation of the technology should be stopped, not the research that enables the technology.

If this simulated project is brought onto physical drones and real life drone battles are conducted, then we would have to consider the idea of armed autonomous systems, the safety precautions necessary to protect innocent and friendly people, and the need for the work. There should be rules and regulations on drone swarming technology and the continued research and development will gain the United States the opportunity to write those rules and regulations before our enemies do.

12.1 Safety Precautions

Despite this system being autonomous, it could be made safer with some human input and safety controls. Having a remote pilot or commander who can shutdown or take control of the swarm will add an additional layer of safety. Additionally, this research project focuses on armed drones that attack other drones. This system is meant to be more of a defensive tool rather than an offensive weapon. The drones are also trained for this specific purpose- Attack other drones, eliminate the enemy, and return to the base.

12.2 Necessity of Continued Research

Drone swarming is a popular topic in today's day and age. If the US does not pursue research in autonomous weapon systems and drone swarming, then other countries, groups, and even our enemies will. Not all countries and non-state actors follow the same ethical guidelines as the US, and they are probably more willing to engage in new types of warfare. Research and development in drone swarm tactics will allow us to protect our people and ensure that this technology is used in safe and ethical ways.

13 Future Work

13.1 Implement Programmed Tactics

Future researchers can continue developing pre-programmed tactics and testing these tactics against one another. This will further military doctrine in terms of drone swarming tactics. Secondly, the programmed tactics can be used to train the RL agent and by developing more programmed tactics, they can build a better foundation to use for training. Researching drone tactics and military aircraft dog-fighting techniques can provide more ideas of various drone swarming tactics to implement.

13.2 Use RL to Learn Tactics vs. Programmed Tactics

We will continue to use the RL training program to develop tactics from a computer's perspective. With more data from more programmed tactics, as well as different inputs, reward structures, outputs, K-values, demonstrations, curriculums, and RL algorithm configuration, future work can produce more intelligent behavior and hopefully more effective agents. The size of the swarms, the start positions, and the enemy swarm's tactics can also be varied to train the RL agent in a wide range of possible situations. Eventually, researchers can train large swarms of agent drones vs. large enemy swarms that can use any pre-programmed tactic.

13.3 Use RL to Learn Tactics vs. RL Learned Tactics

If RL is able to develop swarm counter tactics effectively, it is possible that other powers will attempt to use RL in the same way. Future researchers can use RL for both the friendly swarm and the enemy swarm. This technique is known as self-play and was used in the

work described in Section 8.3. Self-play will allow two RL agents to learn and train against each other. This competition will hopefully lead to intelligent and unexpected behavior and stronger drone swarm tactics because both swarms are continuously improving.

14 Conclusion

The goal of this research project is to develop offensive and defensive drone swarming tactics using reinforcement learning. Swarming is a huge area of research for the US Military because other countries and non-states actors are using swarming to their advantage. If the US is unable to keep up with these other powers in terms of drone swarming technology, then our people and the world at large are in danger. Many people question the ethics of using artificial intelligence in the military, but it will be safer for everyone if the US can develop this technology for ethical use and implement strategic deterrence to maintain peace. It is important for our people's safety and the progress of society if the US Military continues to push the bounds of technological advancements. This research project developed drone swarming tactics using traditional programming and reinforcement learning agent training. Developing swarm and counter swarm tactics using RL will improve military doctrine in this field and protect US and our allies' national interests in the future.

References

- Adams, E. 2016. “DARPA’s Developing Tiny Drones that Swarm to and from Motherships.” April 13. Accessed January 7, 2017.
- Arquilla, John, and David F. Ronfeldt. *Swarming and the Future of Conflict*. RAND: National Defense Research Institute, 2000.
- “Automation.” Merriam-Webster.com Dictionary, Merriam-Webster, <https://www.merriam-webster.com/dictionary/automation>. Accessed 14 Apr. 2021.
- “Autonomy.” Merriam-Webster.com Dictionary, Merriam-Webster, <https://www.merriam-webster.com/dictionary/autonomy>. Accessed 14 Apr. 2021.
- Baker, Bowen. “Emergent Tool Use from Multi-Agent Interaction.” OpenAI, OpenAI, 29 Oct. 2019, openai.com/blog/emergent-tool-use/.
- Barnes, Tom. “China Tests Army of Tiny Drone Ships That Can ‘Shark Swarm’ Enemies.” The Independent, Independent Digital News and Media, 7 June 2018, <http://www.independent.co.uk/news/world/asia/china-drone-ships-unmanned-test-video-military-south-sea-shark-swarm-a8387626.html>.
- Berty Technologies, and Manfred Tournon. “Centralized vs Decentralized vs Distributed Systems.” Berty Technologies, NGO Berty Technologies, 20 June 2019, berty.tech/blog/decentralized-distributed-centralized.
- Bogunowicz, Damian. “Project - Reinforcement Learning with Unity 3D: G.E.A.R.” Data Science, AI, Technology, DTRANSPOSED, 10 Mar. 2019, dtransposed.github.io/blog/GEAR.html.
- Burn, Carter B. *Human Aided Reinforcement Learning in Complex Environments*. United States Naval Academy, 2018.
- Hambling, D. 2016b. “If Drone Swarms are the Future, China May Be Winning.” Popular Mechanics, December 23.
- E. Bøhn, E. M. Coates, S. Moe and T. A. Johansen, “Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy optimization,” 2019 International Conference on Unmanned Aircraft Systems (ICUAS), Atlanta, GA, USA, 2019, pp. 523-533. doi: 10.1109/ICUAS.2019.8798254
- Ecoffet, Adrien Lucas. “An Intuitive Explanation of Policy Gradient-Part 1: REINFORCE.” Towards Data Science, Medium, 25 Apr. 2019, towardsdatascience.com/an-intuitive-explanation-of-policy-gradient-part-1-reinforce-aa4392cbfd3c.

Edwards, Sean J. A. *Swarming on the Battlefield: Past, Present, and Future*. Rand, 2000.

G. Cano Lopes, M. Ferreira, A. da Silva Simões and E. Luna Colombini, “Intelligent Control of a Quadrotor with Proximal Policy Optimization Reinforcement Learning,” 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), Joao Pessoa, 2018, pp. 503-508. doi: 10.1109/LARS/SBR/WRE.2018.00094

Haas, J. K. (2014). A history of the unity game engine.

Hauert, Sabine, et al. “Reynolds Flocking in Reality with Fixed-Wing Robots: Communication Range vs. Maximum Turning Rate.” 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, doi:10.1109/iros.2011.6095129.

Hurst, Jules. “Robotic Swarms in Offensive Maneuver.” *Joint Force Quarterly : JFQ* no. 87 (Fourth, 2017): 105-111, <https://search.proquest.com/docview/1952059112?accountid=14748> (accessed January 22, 2020).

Irving Lachow (2017) The upside and downside of swarming drones, *Bulletin of the Atomic Scientists*, 73:2, 96-101, DOI: 10.1080/00963402.2017.1290879.

Juliani, A., Berges, V., Teng, E., Cohen, A., Harper, J., Elion, C., Goy, C., Gao, Y., Henry, H., Mattar, M., Lange, D. (2020). Unity: A General Platform for Intelligent Agents. arXiv preprint arXiv:1809.02627. <https://github.com/Unity-Technologies/ml-agents>.

Juliani, Arthur. “Solving Sparse-Reward Tasks with Curiosity.” Unity Blog, Unity Technologies, 5 June 2020, blogs.unity3d.com/2018/06/26/solving-sparse-reward-tasks-with-curiosity/.

Kelley, Morgan. “Ants Build ‘living’ Bridges with Their Bodies, Speak Volumes about Group Intelligence.” Princeton University, Princeton University, <https://www.princeton.edu/news/2015/11/30/ants-build-living-bridges-their-bodies-speak-volumes-about-group-intelligence>. Accessed 14 Apr. 2021.

Kober, Jens, et al. “Reinforcement Learning in Robotics: A Survey.” *The International Journal of Robotics Research*, vol. 32, no. 11, Sept. 2013, pp. 1238–1274, doi:10.1177/0278364913495721.

Kolon, Carl C. *Stability of Nonlinear Swarms on Flat and Curved Surfaces*. United States Naval Academy, 2018.

N. Jia, Z. Yang and K. Yang, “Operational Effectiveness Evaluation of the Swarming UAVs Combat System Based on a System Dynamics Model,” in *IEEE Access*, vol. 7, pp. 25209-25224, 2019. doi: 10.1109/ACCESS.2019.2898728

Osborn, Kris. “Iran Has Deadly Boat Swarms (And the Navy Has Plans to Kill Them).” *The*

National Interest, The Center for the National Interest, 28 Sept. 2019, nationalinterest.org/blog/buzz/iran-has-deadly-boat-swarms-and-navy-has-plans-kill-them-83951.

Peck, Michael. “China Will Overwhelm Its Enemies With Swarms of Rocket-Armed Heli-Drones.” The National Interest, The Center for the National Interest, 26 May 2019, nationalinterest.org/blog/buzz/china-will-overwhelm-its-enemies-swarms-rocket-armed-heli-drones-59277.

Pham, Loc V. “UAV Swarm Attack: Protection System Alternatives for Destroyers.” Dudley Knox Library, Naval Postgraduate School, 2012, calhoun.nps.edu/bitstream/handle/10945/28669/12Dec_SE_Cohort_311-1120_Team_Crane.pdf.

“Proximal Policy Optimization.” Spinning Up Documentation, OpenAI, 2018, spinningup.openai.com/en/latest/algorithms/ppo.html.

Reid, David. “A Swarm of Armed Drones Attacked a Russian Military Base in Syria.” CNBC, NBC Universal, 11 Jan. 2018, www.cnbc.com/2018/01/11/swarm-of-armed-drones-attacks-russian-military-base-in-syria.html.

Schulman, John, et al. “Proximal Policy Optimization.” OpenAI, OpenAI, 9 Mar. 2019, openai.com/blog/openai-baselines-ppo/.

Simonini, Thomas. “An Introduction to Unity ML-Agents.” Medium, Towards Data Science, 25 Mar. 2020, towardsdatascience.com/an-introduction-to-unity-ml-agents-6238452fcf4c.

“Soft Actor-Critic.” Spinning Up Documentation, OpenAI, 2018, spinningup.openai.com/en/latest/algorithms/sac.html.

Strickland, Laura G. et al. “On Coordination in Multiple Aerial Engagement” 2019 International Conference on Unmanned Aircraft Systems (ICUAS) (2019): 557-562.

Strickland, Laura, Michael A. Day, Kevin DeMarco, Eric Squires, Charles Pippin, “Responding to unmanned aerial swarm saturation attacks with autonomous counter-swarms,” Proc. SPIE 10635, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, 106350Y (4 May 2018); <https://doi.org/10.1117/12.2305086>

Sutton, Richard S., and Andrew G. Barto. Reinforcement Learning: an Introduction. 2nd ed., The MIT Press, 2018.

Technologies, U., 2015. Unity - Scripting API:. [online] Docs.unity3d.com. Available at: <http://docs.unity3d.com/ScriptReference/index.html>;

Teng, Ervin. “Training Your Agents 7 Times Faster With ML-Agents.” Unity Technologies Blog, Unity, 5 June 2020, blogs.unity3d.com/2019/11/11/training-your-agents-7-times-faster-with-ml-agents.

Toma, Valentin-Marian. The Tactics Of Using Fast Patrol Boats In Coastline Combat: A Critical View. Vol. 2. Bucharest: “Carol I” National Defence University, 2018, <https://search.proquest.com/docview/2043183312?accountid=14748> (accessed January 22, 2020).

Trivedi, Chintan. “Proximal Policy Optimization Tutorial (Part 1: Actor-Critic Method).” Medium, Towards Data Science, 2 Sept. 2019, towardsdatascience.com/proximal-policy-optimization-tutorial-part-1-actor-critic-method-d53f9affbf6.

“U.S. Navy Plans to Fly First Drone Swarm This Summer.” Military.com, Defense Tech, 4 Jan. 2016, www.military.com/defensetech/2016/01/04/u-s-navy-plans-to-fly-first-drone-swarm-this-summer.

Vaggalis, Nikos. “AI Beats Human Pilot In DARPA Dogfight.” i-Programmer, U.S. Defense Advanced Research Projects Agency, 2 Sept. 2020, www.i-programmer.info/news/105-artificial-intelligence/13967-ai-beats-human-pilot-in-darpa-dogfight.html.

Wassmuth, Daniel, and Dave Blair. “Loyal Wingman, Flocking, and Swarming: New Models of Distributed Airpower.” War on the Rocks, Texas National Security Review, 21 Feb. 2018, warontherocks.com/2018/02/loyal-wingman-flocking-swarming-new-models-distributed-airpower/.

Yoon, Chris. “Understanding Actor Critic Methods.” Medium, Towards Data Science, 17 July 2019, towardsdatascience.com/understanding-actor-critic-methods-931b97b6df3f.

Zachary Kallenborn and Philipp C. Bleek (2018) Swarming destruction: drone swarms and chemical, biological, radiological, and nuclear weapons, *The Nonproliferation Review*, 25:5-6, 523-543, DOI: 10.1080/10736700.2018.1546902

A Pre-Programmed Tactics Results

A.1 Swarm vs. Swarm- Figure 10.3

The tables below show the actual results that were used to make the graphs in Figure 10.3. Each table represents one friendly tactic (listed in the title). Each row of the table represents the results against the listed enemy tactic. The size of the swarms is listed in the title of the table as well. The value in each cell is the percentage of wins, losses, and tie out of the 1000 rounds that were simulated.

10 v. 10- Flocking			
	Win	Loss	Tie
F	42.4	42.6	15.0
SN	31.3	68.4	0.3
AN	13.0	87.0	0.0
R	12.4	87.3	0.3
DS	0.0	75.2	24.8
CF	0.0	100.0	0.0

10 v. 10- Select Nearest			
	Win	Loss	Tie
F	68.4	31.3	0.3
SN	52.5	47.2	0.3
AN	33.1	66.6	0.3
R	33.2	66.3	0.5
DS	71.8	27.9	0.3
CF	45.6	54.2	0.2

10 v. 10- Assign Nearest			
	Win	Loss	Tie
F	87.0	13.0	0.0
SN	66.6	33.1	0.3
AN	54.8	44.9	0.3
R	80.2	19.4	0.4
DS	100.0	0.0	0.0
CF	100.0	0.0	0.0

10 v. 10- Rabbit			
	Win	Loss	Tie
F	87.3	12.4	0.3
SN	66.3	33.2	0.5
AN	19.4	80.2	0.4
R	59.4	40.0	0.6
DS	99.8	0.2	0.0
CF	99.4	0.6	0.0

10 v. 10- Double Swarm			
	Win	Loss	Tie
F	75.2	0.0	24.8
SN	27.9	71.8	0.3
AN	0.0	100.0	0.0
R	0.2	99.8	0.0
DS	27.2	25.8	47.0
CF	0.3	98.7	1.0

10 v. 10- Circle Flock			
	Win	Loss	Tie
F	100.0	0.0	0.0
SN	54.2	45.6	0.2
AN	0.0	100.0	0.0
R	0.6	99.4	0.0
DS	98.7	0.3	1.0
CF	49.0	45.4	5.6

10 v. 12- Flocking			
	Win	Loss	Tie
F	26.8	57.4	15.8
SN	21.5	78.3	0.2
AN	7.5	92.4	0.1
R	8.5	91.5	0
DS	0	84.5	15.5
CF	0	99.7	0.3

10 v. 12- Select Nearest			
	Win	Loss	Tie
F	55.7	44.1	0.2
SN	36.4	63.3	0.3
AN	19.7	80.1	0.2
R	23.2	76.8	0
DS	59.4	40.5	0.1
CF	36	63.5	0.5

10 v. 12- Assign Nearest			
	Win	Loss	Tie
F	88.4	11.6	0
SN	82.1	17.7	0.2
AN	42.4	56.4	1.2
R	70.1	29.3	0.6
DS	99.9	0.1	0
CF	99.6	0.4	0

10 v. 12- Rabbit			
	Win	Loss	Tie
F	85.5	13.9	0.6
SN	78.7	21	0.3
AN	47.3	52.4	0.3
R	40.2	59.7	0.1
DS	99.9	0.1	0
CF	98.6	1.3	0.1

10 v. 12- Double Swarm			
	Win	Loss	Tie
F	69.9	0	30.1
SN	17.9	81.9	0.2
AN	0	100	0
R	0	100	0
DS	21	34.8	44.2
CF	0.5	98.9	0.6

10 v. 12- Circle Flock			
	Win	Loss	Tie
F	99.7	0.2	0.1
SN	42.8	56.9	0.3
AN	0	100	0
R	0.1	99.9	0
DS	98.4	1.2	0.4
CF	40	57.5	2.5

10 v. 14- Flocking			
	Win	Loss	Tie
F	19	75.4	5.6
SN	28.1	71.9	0
AN	6.8	93.1	0.1
R	2.3	96.5	1.2
DS	0	99.9	0.1
CF	0	94.6	5.4

10 v. 14- Select Nearest			
	Win	Loss	Tie
F	25	74.7	0.3
SN	12.8	86.8	0.4
AN	2.4	97.6	0
R	0.1	99.9	0
DS	5.9	94.1	0
CF	27.4	66.3	6.3

10 v. 14- Assign Nearest			
	Win	Loss	Tie
F	60.3	39.3	0.4
SN	62.5	36.7	0.8
AN	30.2	69.1	0.1
R	29	70.6	0.4
DS	33	66.8	0.2
CF	99.5	0.3	0.2

10 v. 14- Rabbit			
	Win	Loss	Tie
F	70.2	15	14.8
SN	91.6	8.2	2
AN	31.9	67.9	0.2
R	16.9	82.3	0.8
DS	41.8	45.3	12.9
CF	85	10.1	4.9

10 v. 14- Double Swarm			
	Win	Loss	Tie
F	96.8	1.7	1.5
SN	57.9	41.8	0.3
AN	23.3	76.7	0.1
R	17.7	78.5	3.8
DS	22.6	65	12.4
CF	0.2	88.3	11.5

10 v. 14- Circle Flock			
	Win	Loss	Tie
F	94.9	0.2	4.9
SN	41.9	54.3	3.8
AN	0	99.9	0.1
R	1.1	97.9	1
DS	84	2.1	13.9
CF	18.8	47.5	33.7

10 v. 20- Flocking				10 v. 20- Select Nearest				10 v. 20- Assign Nearest			
	Win	Loss	Tie		Win	Loss	Tie		Win	Loss	Tie
F	5.5	92.6	1.9	F	8.7	91.1	0.2	F	38	61.7	0.3
SN	11.1	88.8	0.1	SN	1.1	98.6	0.3	SN	36	63.6	0.4
AN	1.8	98.2	0	AN	0	100	0	AN	5.6	94	0.4
R	0.5	99.2	0.3	R	0.1	99.9	0	R	5.9	94	0.1
DS	0	100	0	DS	1.1	98.7	0.2	DS	16.9	83.1	0
CF	0	93.9	6.1	CF	11.4	77.6	11	CF	98.5	1	0.5

10 v. 20- Rabbit				10 v. 20- Double Swarm				10 v. 20- Circle Flock			
	Win	Loss	Tie		Win	Loss	Tie		Win	Loss	Tie
F	40.7	34.5	24.8	F	86.1	7	6.9	F	92.3	0.3	7.4
SN	74.3	25.5	0.2	SN	32.9	67.1	0	SN	37.1	59.8	3.1
AN	10.6	89.1	0.3	AN	6.7	93.1	0.2	AN	0	100	0
R	3	97	0	R	7.8	91.1	1.1	R	1	97.7	1.3
DS	27.1	57.9	15	DS	6.8	85.6	7.6	DS	80.2	3.2	16.6
CF	66.9	26	7.1	CF	0.1	90.6	9.3	CF	5.2	61.2	33.6

A.2 Ship Attack and Defense- Figure 10.4

The tables below show the actual results that were used to make the graphs in Figure 10.4. Each table represents one friendly tactic (listed in the title). Each row of the table represents the results against the listed enemy tactic. The size of the swarms is listed in the title of the table as well. The value in each cell is the average number of drone hits on the ship and the average number of green (defensive) drones remaining at the end of the 1000 rounds that were simulated. The number of remaining defensive drones is not used for Figure 10.4 or display any patterns, but it is interesting to see the variation in results of different tactics.

8 v. 8- Flocking			8 v. 8- Select Nearest			8 v. 8- Assign Nearest		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0.152	6.501	F	0	6.919	F	0	7.167
SN	0.062	4.734	SN	0	4.111	SN	0	6.266
AN	0.138	2.136	AN	0	1.912	AN	0	2.686
R	0.303	2.157	R	0.014	1.816	R	0.004	2.786
DS	1.237	1.86	DS	0	1.711	DS	0	2.519
CF	3.968	2.08	CF	0.08	4.115	CF	0	5.831

8 v. 8- Rabbit			8 v. 8- Double Swarm			8 v. 8- Circle Flock		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0	7.017	F	0.171	6.351	F	0	7.631
SN	0.004	5.162	SN	0	7.161	SN	0.564	3.398
AN	0.092	3.08	AN	0.024	4.44	AN	0.347	3.254
R	0.205	2.599	R	0.137	4.041	R	1.8	4.034
DS	0.007	4.229	DS	0.389	3.731	DS	0.26	5.589
CF	0.002	5.835	CF	4.022	2.132	CF	1.522	6.12

10 v. 12- Flocking			10 v. 12- Select Nearest			10 v. 12- Assign Nearest		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0.081	8.056	F	0	8.244	F	0	8.654
SN	0.035	5.915	SN	0	4.681	SN	0	7.938
AN	0.135	2.205	AN	0	1.873	AN	0	3.307
R	0.279	2.044	R	0.01	1.808	R	0.026	2.93
DS	1.186	2.337	DS	0	2.043	DS	0	3.147
CF	5.914	2.759	CF	0.368	3.78	CF	0	6.776

10 v. 12- Rabbit			10 v. 12- Double Swarm			10 v. 12- Circle Flock		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0	8.357	F	0.107	7.917	F	0	9.398
SN	0.004	5.564	SN	0.001	8.763	SN	0.673	4.31
AN	0.075	3.122	AN	0.029	4.847	AN	1.675	4.403
R	0.212	2.766	R	0.108	4.351	R	2.197	5.665
DS	0.004	4.499	DS	0.425	4.267	DS	1.117	7.389
CF	0.001	6.688	CF	5.864	2.689	CF	1.473	5.36

10 v. 14- Flocking			10 v. 14- Select Nearest			10 v. 14- Assign Nearest		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0.097	7.929	F	0	8.26	F	0	8.599
SN	0.063	5.496	SN	0	5.421	SN	0	7.946
AN	0.117	2.276	AN	0	2.408	AN	0	4.274
R	0.359	1.914	R	0.003	1.844	R	0.049	3.225
DS	0.838	3.178	DS	0	2.508	DS	0	4.44
CF	4.804	3.414	CF	0.453	4.065	CF	0	6.731

10 v. 14- Rabbit			10 v. 14- Double Swarm			10 v. 14- Circle Flock		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0	8.307	F	0.097	7.804	F	0	9.317
SN	0.006	5.385	SN	0	8.337	SN	0.545	5.515
AN	0.086	3.023	AN	0.027	4.659	AN	1.422	5.785
R	0.211	2.897	R	0.232	3.826	R	2.045	5.649
DS	0.007	4.932	DS	0.406	4.498	DS	0.342	8.271
CF	0.003	6.731	CF	5.235	2.978	CF	2.352	6.832

10 v. 20- Flocking			10 v. 20- Select Nearest			10 v. 20- Assign Nearest		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0.11	7.746	F	0	8.041	F	0	8.442
SN	0.85	4.28	SN	0	5.117	SN	0	7.894
AN	0.112	2.131	AN	0	2.405	AN	0	4.778
R	0.287	4.297	R	0.002	1.902	R	0.054	3.544
DS	0.601	4.84	DS	0	25.66	DS	0	4.949
CF	1.969	4.84	CF	0.43	4.367	CF	0	6.865

10 v. 20- Rabbit			10 v. 20- Double Swarm			10 v. 20- Circle Flock		
	Hits	Green Left		Hits	Green Left		Hits	Green Left
F	0	8.225	F	0.126	7.531	F	0	9.191
SN	0.01	4.785	SN	0.009	7.173	SN	0.121	8.385
AN	0.09	3.175	AN	0.053	4.165	AN	0.226	8.348
R	0.123	3.572	R	0.262	3.733	R	0.415	8.653
DS	0	5.807	DS	0.384	4.553	DS	0.023	9.003
CF	0.005	6.843	CF	3.34	3.893	CF	3.264	7.706

B RL Agent Results

B.1 Swarm vs. Swarm- Figure 11.5 and Figure 11.7

The tables below show the actual results that were used to make the graphs in Figure 11.5 and Figure 11.7. Both the Swarm vs. Swarm Scene and the Ship Attack and Defense Scene are displayed in this table. The rows in the table each represent the difficulty of the starting position of the RL Agent drones. The values in each cell represent the average number of enemy drones that the RL Agents kill in the 100 rounds that were simulated.

RL Agent Kills		
	Swarm vs. Swarm	Ship Attack and Defense
100m, 0°	2.58	7.53
300m, 45°	4.25	2.94
500m, 120°	3.07	0.74