Missile Defense Using Multi-Agent Reinforcement Learning: Simulation, Training Infrastructure, and Coordination Strategies

Pius Lim Zhen Ye, Lim Yew Hao, Cheah Tze Ang, Srinivasan Srivatsan

April 18, 2025

Abstract

This project presents a reinforcement learning-based framework for autonomous aerial base defense using multi-agent coordination. A custom 2D simulation environment was developed to train drones to intercept incoming missile threats using Multi-Agent Proximal Policy Optimization (MAPPO). Each drone operates under decentralized execution with shared policies, while missile assignment is managed through a centralized controller. The environment was implemented using a Gym-compatible architecture and integrated with Ray RLlib to support scalable training.

A key focus of the project was the design and refinement of the training environment, which progressed from basic fixed-length episodes to a continuous, curriculum-based system. The final setup introduced missile respawning, cooldown timing, and gradually increasing difficulty through infinite levels. Combined with velocity-based control, smoothness and pursuit rewards, and urgency-weighted target engagement, this structure enabled agents to develop coordinated behaviors such as adaptive positioning, re-engagement after failed intercepts, and implicit coverage distribution.

The final policy handled both 3v3 and 10v10 scenarios effectively, without relying on hand-crafted strategies. While the simulation remains an abstraction of real-world conditions, the system lays the groundwork for future enhancements including learned assignment policies, adversarial training, and 3D simulation with realistic physics and perception inputs.

The full source code is available at: https://github.com/PiusLim373/missile_defense_marl

Chapter 1

Introduction

As part of the ME5424 module on Swarm and Aerial Robotics, the team proposed a swarm-focused project titled **Missile Defense Multi-Agent Reinforcement Learning (RL) Simulation**. The project explores the application of reinforcement learning to autonomous multi-agent systems, with the specific goal of enabling cooperative aerial defense behaviors in a simulated missile interception scenario. Reinforcement learning was identified as a suitable approach due to its capacity for enabling agents to adaptively learn and improve decision-making in dynamic, adversarial environments.

The project direction was influenced by the research trajectory of Dr. Sutthiphong Srigrarom, whose work spans multi-drone interception and swarm control, as well as task assignment, scheduling, and path planning algorithms for heterogeneous robot teams—including aerial, ground, and legged platforms—in complex operations such as search and rescue. These research efforts underscore the growing potential for multi-agent autonomy in mission-critical settings and served as both a conceptual foundation and motivation for this project.

In parallel, the team engaged in exploratory discussions with the Republic of Singapore Air Force (RSAF) to examine the use of reinforcement learning for autonomous aerial systems. These discussions centered on scenarios involving two teams of drones engaging in airspace denial or aerial dogfights. The core research question was how reinforcement learning could be leveraged to enable drones to make tactical decisions, such as pursuit and evasion, without relying on manually engineered control logic.

The initial project concept was to implement and train these AI-controlled drones in a three-dimensional simulation environment using Unreal Engine. A high-fidelity simulation with realistic physics was considered a critical prerequisite for validating autonomous behaviors prior to any physical deployment.

Chapter 2

Prior Work and Objectives

2.1 Background on Multi-Agent Defense Systems

Autonomous multi-agent systems hold significant promise for future air defense applications. In particular, drone swarms present a compelling solution due to their potential for scalability, decentralized control, and coordinated response to evolving threats. If properly trained, a fleet of autonomous aerial agents could offer rapid interception capabilities without requiring continuous human supervision. Such systems could dynamically adapt to diverse and complex threat scenarios, providing an operational edge over traditional, rule-based defenses. The long-term goal of this line of research is to enable a scalable and intelligent defense framework that is both cost-efficient and strategically robust.

However, this vision poses several fundamental challenges. Threats in real-world scenarios—such as nuisance or adversarial drones—may not behave predictably. Some may fly in coordinated formations, while others may employ evasive maneuvers or attempt to mislead defenders. Designing handcrafted behaviors to account for such a wide behavioral spectrum quickly leads to brittle and overly complex expert systems. While game-theoretic approaches have been studied for aerial conflict modeling, they often rely on idealized assumptions, such as complete information and stable equilibria, which are difficult to uphold in practice. In light of these limitations, this project investigates the use of Multi-Agent Reinforcement Learning (MARL) as a more generalizable and adaptive alternative for swarm-based air defense. The objective is to test whether learning-based agents can develop effective interception strategies through interaction with the environment, rather than through predefined rule sets.

2.2 Project Objectives

Due to time and resource constraints, the implementation was carried out in a simplified two-dimensional simulation environment. This abstraction serves as a conceptual foundation for evaluating reinforcement learning approaches in a constrained but controlled setting.

The primary objectives of the project are:

- To evaluate whether reinforcement learning can enable intelligent and responsive base defense behavior in a multi-agent setting.
- To train agents to learn pursuit and interception strategies without relying on handcrafted control policies.

Achieving these objectives implicitly requires the agents to perform key capabilities such as target prioritization, coordinated interception, and adaptive movement. The system must demonstrate that reinforcement learning is capable of both low-level control and high-level decision-making under multi-agent defense scenarios.

Chapter 3

Experimentation Setup

3.1 Agent and Algorithm Selection

The missile defense problem is modeled as a cooperative multi-agent scenario in which autonomous drones work together to intercept incoming missiles targeting a central base. Each drone acts independently under a decentralized control structure, making decisions based solely on its own observations. This setup naturally aligns with the framework of Multi-Agent Reinforcement Learning (MARL), which supports decentralized execution while enabling shared training that agents to learn coordination strategies without centralized decision-making.

The project employs Multi-Agent Proximal Policy Optimization (MAPPO) as the core learning algorithm. MAPPO extends the well-established Proximal Policy Optimization (PPO) to multi-agent systems. PPO is particularly effective for this task for several reasons:

- It offers strong training stability through its clipped surrogate objective, which prevents destructive policy updates.
- It is sample-efficient, allowing multiple training epochs over the same collected trajectories.
- It integrates seamlessly with Generalized Advantage Estimation (GAE) to produce smoother, lower-variance advantage updates, improving learning in environments with sparse or delayed rewards.

In addition, MAPPO supports parameter sharing across agents, enabling coordination without centralized commands. The actor–critic structure used in PPO allows agents to jointly optimize action selection and value estimation through shared training signals. These properties are particularly suited to dynamic environments involving real-time interactions and adaptive threats. The algorithm was implemented using Ray RLlib, which provides first-class support for multi-agent setups and greatly simplifies the integration of PPO with custom Gym environments. This allows the team to focus on environment development rather than low-level algorithm integration.

3.2 Simulation Model and Environment Design

The original concept envisioned a large-scale 3D simulation implemented in Unreal Engine, where two AI-controlled drone teams would engage in autonomous air-to-air combat. However, training two fully independent policies with realistic flight dynamics required significant resources and simulation fidelity. To maintain feasibility, the environment was scaled down to a 2D setting using a 1000 pixel $\times$ 1000 pixel grid, where 1 pixel represents approximately 1 meter.

To further simplify the simulation while retaining decision-making complexity, the attacking drones were replaced by missiles that follow linear trajectories toward the base. The defending drones were modeled as quadcopters capable of omnidirectional movement. Agents intercept missiles by colliding with them. Initially, interception resulted in the drone's termination, but this behavior was revised to allow drones to survive and continue acting in the environment after an interception.

The system begins with a 3-vs-3 configuration, where the defending drones intercept a single incoming missile each. Once the agents develop competent interception policies, the scenario scales to a 3-vs-N setting to increase complexity and test generalization, where N is the number of missile scaling to infinity.

3.3 Custom Gym Environment

A custom environment was developed in Python, using OpenAI's gymnasium API as the interface and Pygame for rendering. Pygame supports both symbolic visualizations and sprite-based visuals for drones, missiles, and the central base, offering clarity for both debugging and demonstration purposes.

The environment is fully compatible with major reinforcement learning libraries, particularly Ray RLlib, making it easy to integrate with PPO-based multi-agent setups. The simulator supports both episodic and continuous training modes.

The custom gym is structured around several major components that define the learning process:

- The state representation captures observations about the agent, its assigned target, and global threat proximity.
- The action space defines how agents control their motion in the environment.
- The reward system shapes behavior toward efficient and cooperative interception.
- The missile assignment system dynamically allocates threats to agents.
- The training curriculum scales difficulty as agents improve.

Together, these elements were carefully engineered to simulate real-world aerial defense constraints and guide agents to learn intelligent base defense strategies in a cooperative swarm setting.

3.4 State Representation

The original observation space consisted of three fixed missile slots, each intended to provide information about an active missile or padded with default values if fewer threats were present. While this worked well in simple scenarios involving one missile, the agents failed to generalize as more missiles were introduced. Agents could not learn to differentiate between the roles of missiles across slots, leading to confusion and policy collapse. This limitation was observed during an earlier implementation of the environment that used a 3-vs-1 setup, where the fixed-slot observation structure performed adequately. However, when scaling to 3-vs-3 scenarios, the lack of slot differentiation caused significant breakdowns in learning.

To resolve this, the state space was restructured to provide targeted, interpretable information. Each drone now observes:

- Self state: position (x, y), velocity (vx, vy), and acceleration (ax, ay)
- Assigned missile: position, direction unit vector, distance to the agent, and distance to the base
- Globally closest missile to base: same structure as the assigned missile data

This revised format offers two key benefits. First, the velocity and acceleration states enable tracking of smoothness for movement penalties and rewards. Second, the inclusion of both an assigned missile and the globally closest missile allows the agent to balance individual and collective defense priorities. All of this information is concatenated into a single flat vector per agent. The missile assignment system, which determines each drone's assigned missile at every step, is discussed in detail later.

This observation design allows the agents to learn solely from their observations, without needing centralized coordination or hardcoded rules. It supports complex behavior such as threat anticipation and prioritization while maintaining compatibility with standard RL policy architectures.

3.5 Action Space

The drones operate using a discrete velocity-based action space. In each timestep, an agent selects a velocity command that sets its movement along the x and y axes. The velocity in each direction is capped between -5 and 5 pixels per second, effectively allowing a drone to move up to 5 pixels per axis per step. Since each pixel represents approximately one meter, this corresponds to a maximum speed of 5 meters per second per axis.

This velocity control method replaces the earlier position-based movement system. The previous formulation often resulted in unnatural, jerky behavior. By controlling velocity instead, agent movement becomes smoother and easier to regulate.

To further improve realism, the environment also enforces a maximum acceleration limit, which restricts how quickly a drone can change its speed from one timestep to the next. This encourages more stable and natural-looking trajectories and prevents agents from exploiting rapid reversals or unrealistic directional changes.

Overall, the discrete velocity space with acceleration limits creates a more grounded simulation and leads to more learnable, coordinated behaviors during training.

3.6 Reward Structure

The reward structure is composed of several major components, each designed to encourage tactical and coordinated drone behavior.

Interception Reward

Interception is the primary goal of the system, as successfully neutralizing missiles is critical for base protection. A flat reward of +200 is given whenever a drone intercepts a missile, regardless of assignment. This ensures that drones are rewarded for any effective interception and encourages opportunistic behavior.

Smoothness Reward

The smoothness reward is designed to encourage drones to move in a more controlled and stable manner by penalizing abrupt changes in movement. It works as follows:

- The magnitude of change in velocity (Δv) and acceleration (Δa) is computed by comparing the current and previous values.
- These deltas reflect how erratically a drone is maneuvering.
- If the drone is within a critical proximity of any active missile , the penalty is scaled down by 50%.
- This allows agents to make sharp, aggressive movements near high-priority threats without being overly penalized.

This mechanism ensures that agents are encouraged to fly smoothly during patrol or transit, while still retaining the agility to execute sharp interception maneuvers when needed.

Pursuit Reward

Because missiles always travel directly toward the base, the distance between them and the drones tends to decrease over time. If the reward function relied solely on proximity, an agent could game the system by simply waiting passively for the missile to approach.

To address this, a pursuit reward was implemented to promote active engagement. This reward is computed using dot products:

- Between the agent's velocity and the direction unit vector toward the missile.
- Between the agent's acceleration and the same unit vector.

This encourages alignment of the drone's motion with the missile's trajectory, guiding the agent to move toward the target rather than letting it come to them.

Urgency-Weighted Reward

In scenarios involving multiple incoming threats, drones must be able to prioritize higher-threat missiles over those that are less urgent. For example, if the globally closest missile to the base is significantly nearer than a drone's assigned target, it may be optimal to abandon the assignment and intercept the more imminent threat.

To model this, the environment implements a softmax-based urgency weighting between two targets:

- The assigned missile.
- The globally closest missile to the base.

The softmax weights are calculated using the following formula:

$$w_{\text{assigned}} = \frac{\exp\left(\beta \cdot \left(1 - \frac{d_{\text{assigned}}}{d_{\max}}\right)\right)}{\exp\left(\beta \cdot \left(1 - \frac{d_{\text{assigned}}}{d_{\max}}\right)\right) + \exp\left(\beta \cdot \left(1 - \frac{d_{\text{closest}}}{d_{\max}}\right)\right)}$$

Where β is a hyperparameter that determines the sharpness of preference (default $\beta = 5.0$). These weights are then applied to the pursuit rewards of each target to form a final urgency-weighted reward.

The following table illustrates the impact of softmax scaling:

Let's assume MAX_DISTANCE_FROM_BASE = 1000

Table: Example softmax weight for urgency-weighted reward Table:

Softmax Weighting Examples (Assume `MAX_DISTANCE_FROM_BASE` = 1000)

Assigned Dist to Base	Closest Dist to Base	Threats (raw)	Softmax (scale=1)	Softmax (scale=5)	Softmax (scale=10)
300	200	0.7 vs 0.8	0.45 vs 0.55	0.27 vs 0.73	0.12 vs 0.88
100	800	0.9 vs 0.2	0.82 vs 0.18	0.99 vs 0.01	∼1.00 vs ∼0.00
500	500	0.5 vs 0.5	0.50 vs 0.50	0.50 vs 0.50	0.50 vs 0.50
100	150	0.9 vs 0.85	0.51 vs 0.49	0.62 vs 0.38	0.73 vs 0.27

3.7 Environment Mechanics

The environment includes several key systems that support long-term agent learning and simulate escalating complexity.

Continuous Episodes

The simulation is run in continuous mode, where drones remain alive even after intercepting a missile. This diverges from real-world scenarios where drones may self-sacrifice upon impact (e.g., kamikaze tactics). However, this decision improves learnability by allowing agents to recover from mistakes and engage in multiple interception cycles. In practice, this abstraction could also apply to signal-jamming UAVs or non-destructive interceptors, such as those used in drone-vs-drone engagements.

Missile Spawning

The environment begins with three active missiles. As the difficulty level increases, additional missiles are introduced. Missiles are spawned from the boundaries of the field, and each follows a direct path toward the base. Once intercepted, missiles enter a cooldown phase before being respawned. The cooldown period decreases over time to match higher levels of difficulty.

Curriculum Learning

To prevent training failure in high-difficulty scenarios, a curriculum learning system is used. Early training levels involve slower missiles and fewer threats. As agents demonstrate proficiency, the environment increases complexity gradually. This includes:

- Higher missile acceleration and maximum speed
- Occasional increases in the number of simultaneous missiles
- An infinite level progression, allowing difficulty to grow over time and supporting continual adaptation

This system enables the agents to scale their capabilities progressively, from simple pursuit to handling complex, high-speed threat saturation scenarios.

Missile Assignment System

The current environment implements a centralized rule-based missile assignment system to coordinate agents and improve coverage efficiency. At each timestep, a distance matrix of size $N_{\text{agents}} \times N_{\text{missiles}}$ is computed using Euclidean distances. Neutralized missiles are excluded from consideration. The system then iterates through the sorted list of agent–missile pairs and assigns each drone to the closest unassigned missile, skipping already assigned targets.

To avoid excessive switching, reassignment is triggered only when:

- More than half the drones have no assignments, or

- More than half could be reassigned to significantly closer threats

This mechanism ensures consistent target allocation, minimizes collisions or redundancy, and significantly stabilizes the learning process in scenarios involving many simultaneous threats.

3.8 Training Configuration

Training was performed using PPO within Ray RLlib, configured with carefully chosen hyperparameters to balance convergence speed and learning stability:

Parameter	**Value**	**Rationale**
Learning Rate	5e-4	Smooth convergence without overshooting
Epochs per Update	10	Multiple optimization passes per rollout
Mini Batch Size	256	Enables mini-batch PPO updates
Train Batch Size	4000	Sufficient samples per update to ensure stable learning
Gamma (γ)	0.99	Encourages long-term planning and reward accumulation
GAE Lambda (λ)	0.95	Balances bias and variance in advantage estimation
Clipping Parameter	0.2	Prevents large, destabilizing policy updates
Entropy Coefficient	0.01	Starts with higher exploration, annealed to support convergence
Value Loss Coefficient	1.0	Balances policy gradient with value function accuracy

Table 3.1: PPO hyperparameters used during training

The model was trained for over 2500 iterations, with checkpoints saved every 50 iterations for periodic evaluation.

Chapter 4

Experimental Results and Discussion

To fulfill the project objectives defined earlier, the reinforcement learning framework and simulation environment underwent several rounds of iteration and refinement. These changes were crucial in aligning the agent behavior with the desired goals of intelligent pursuit, cooperative interception, and adaptive defense coordination. This chapter outlines key improvements made during development, presents the final evaluation results, and reflects on the limitations and future potential of the current system.

4.1 Iterative Improvements

Throughout the project, several key iterations were made to improve both the agent learning process and the simulation environment. One major refinement involved replacing the original position-based action space with a velocity-based formulation. In the earlier design, agents could directly specify their next position, but this led to jerky and unrealistic movements, as there were no constraints on acceleration. By switching to velocity-based control, it became possible to enforce a maximum acceleration limit, which produced smoother, more physically realistic trajectories. However, even with this change, early training still resulted in jittery agent behavior, particularly during rapid maneuvering. To address this, a smoothness reward was introduced to penalize sudden changes in velocity and acceleration. This reward was scaled down when drones were near active threats, allowing tactical flexibility while still encouraging controlled flight.

Another major improvement involved the observation space. The original design exposed three missile slots to each drone, with padding used when fewer missiles were present. This structure proved ineffective, as agents were unable to distinguish between slot meanings when missile counts varied. In response, a missile assignment system was implemented, dynamically assigning one missile to each drone and simplifying the observation to include only the assigned missile and the globally closest missile to the base. This change improved interpretability and made the learning task more tractable.

This update addressed issues encountered in earlier versions of the environment, where agents were expected to implicitly learn task allocation from fixed-slot observations. In those setups, drones would frequently clump together and pursue the same missile, resulting in poor scalability and inefficient threat coverage. The new rule-based assignment system eliminated this bottleneck, enabling better agent coordination and accelerating policy convergence in scenarios with multiple simultaneous threats.

Additionally, the environment initially terminated drones upon missile interception, mimicking a sacrificial defense behavior. This was later replaced with a continuous training mode, allowing drones to remain active and respond to additional threats after each interception. Combined with the change that removed neutralized missiles entirely from the system, this resulted in a cleaner, more efficient training signal and significantly faster convergence, especially in multi-missile scenarios.

4.2 Performance Evaluation

The team conducted extensive training and testing across multiple environment versions and reward formulations. Each iteration of the environment and learning logic was validated through long-run trials, with training performance used as a key indicator for evaluating the effectiveness of each proposed change.

The final and most successful version of the system was trained under a 5 drones vs. 5 missiles configuration, with curriculum scaling enabled to simulate increasingly complex defense scenarios. Over the course of 2500 iterations, the missile count gradually increased as part of the level-based progression, eventually reaching up to 14 concurrent threats. The policy performance began to stabilize around iteration 1500, where agents demonstrated reliable coordination, interception, and defensive coverage.

Evaluation was also conducted on deployment-like configurations, notably the 3v3 scenario (aligned with project objectives) and an expanded 10v10 setup to stress test scalability. In both cases, the agents showed strong and consistent performance, highlighting the policy's generalization capability.

Key observations from the final policy include:

- Drones generally remain within regions close to the base, forming a spatial buffer against incoming threats
- They consistently intercept incoming missiles without external coordination
- Agents successfully avoid going out of bounds, a common failure mode in early versions
- Movements became increasingly deliberate and reward-maximizing over time

This behavioral improvement is a direct result of PPO's core mechanism: the policy is updated based on cumulative reward improvement, meaning the agents

iteratively adjust their behavior to better align with long-term goals. The PPO framework consistently yielded the most stable and effective behaviors across tested variants, reinforcing its suitability for this class of multi-agent control tasks.

A video demonstration of the final MARL system's performance is available at: https://youtu.be/7Z7UBZj8iI4

Figure 4.1: Agents intercepting multiple incoming threats using learned policy

4.3 Deployment Scalability

A key enabler of training and deployment flexibility was the switch to a missile assignment system, which allowed the use of variable drone and missile counts. Since the project followed a multi-agent RL approach with shared policies and observation-independent logic, the agent model is not tied to a fixed team configuration.

This means the number of drones and missiles at test or deployment time can differ from those seen during training. This was demonstrated by successfully deploying the final model in both 3vN, 5vN, and 10vN configurations. The 10v10 case showed stable behavior, although with occasional signs of inefficiency.

One notable limitation lies in the rule-based nature of the missile assignment system. As the number of drones and missiles increases, the system may become suboptimal, especially when multiple drones converge on overlapping targets. At

high team sizes, assignment decisions may not reflect global optimality or threat coverage.

A promising future direction is the development of a learnable task allocation mechanism, either trained jointly with the primary policy or as a separate policy module. Decentralized learning approaches could also allow agents to autonomously negotiate threat coverage without centralized logic.

4.4 Training Environment Evolution

In the early phases of the project, the environment was structured around a simple assumption: drones would intercept incoming missiles at the cost of their own destruction, simulating a kamikaze-style defense. While conceptually valid, this implementation created a fundamental limitation—if an agent failed or overshot its target, it had no opportunity to recover. In many cases, drones would miss a missile and either idle, fly out of bounds, or simply fail to re-engage.

Recognizing these issues, the team transitioned to a continuous training environment, where drones remained active even after successful interceptions. This change enabled drones to attempt multiple engagements per episode, recover from failed attempts, and gradually learn more flexible and strategic behaviors.

Initially, the simulation also operated on fixed-length episodes, where agents were evaluated over a predefined number of steps with a fixed number of threats. While useful for early-stage debugging, this design was inefficient for multi-missile scenarios and continuous learning.

To support lifelong learning and policy robustness, the team implemented an infinite training gym. This new environment incorporated several critical features:

- Missile respawning: Neutralized missiles are removed and replaced after a cooldown
- Cooldown system: Ensures dynamic pacing without overwhelming the agents
- Curriculum-based leveling: Each new level introduces greater challenge

As training progresses:

- Missiles move faster, with higher acceleration values
- The number of active missiles increases, introducing saturation dynamics
- Drones face an increasingly diverse range of interception angles, timings, and urgency levels

This curriculum structure allows for continual learning, where agents are constantly exposed to new variations and threats. A level-up multiplier of 1.014 is

applied to missile parameters, meaning missile difficulty roughly doubles every 50 levels.

With sufficient training in this evolving setup, agents developed robust defensive strategies:

- They reposition adaptively based on missile spawn patterns
- They re-engage with missed targets and recover from failed attempts
- Target coverage naturally distributes across agents, often overlapping based on threat urgency

  This evolution of the environment—from discrete, fixed-length, single-shot episodes to an infinite, continuously escalating curriculum—was critical to enabling the policy to generalize beyond simple reactive behaviors.

4.5 Simulation Limitations

The current 2D simulation environment was sufficient to evaluate reinforcement learning-based coordination, pursuit behavior, and reward shaping strategies in a multi-agent setting. However, several limitations reduce the fidelity and realism of the results.

First, the environment operates strictly in two dimensions, omitting any modeling of altitude or vertical separation, which are critical for real-world UAV applications. All threats and drone actions are restricted to a flat plane, limiting the complexity of tactical behaviors.

Second, the system does not model terrain, obstacles, or occlusion. In practice, buildings, restricted zones, and varying elevation impact both agent perception and movement planning. The absence of such constraints simplifies the learning task but limits operational generalizability.

Third, the flight dynamics are simplified. While acceleration limits were added for motion smoothness, there is no modeling of thrust, inertia, drag, or air resistance. Agents execute instant velocity commands, which diverge significantly from physical quadcopter behavior.

Additionally, the simulation assumes perfect sensing and communication. Agents receive full missile state data with no latency, noise, or field-of-view restriction. Real systems would instead rely on vision, radar, or LiDAR, all of which introduce uncertainty. These limitations provide clear targets for future system improvements.

4.6 Strategic Discussion and Future Perspectives

Beyond performance metrics and simulation results, the system offers useful insights into the roles of autonomy layers, explainability trade-offs, and future deployment

strategies in more complex mission settings.

Mission vs. Platform Autonomy

The current system separates mission-level autonomy from platform-level control. The RL policy governs high-level decisions — such as which missile to intercept and how to adjust tactics over time — while low-level functions like motion control and stabilization are abstracted away.

One critical design choice is the use of a centralized rule-based missile assignment system. While this works well in small team settings, it does not scale gracefully as team size increases or threat scenarios become more dynamic. A promising improvement is to replace this rule-based logic with a trainable assignment policy, which could be integrated into the MARL framework either as a separate module or a jointly trained head. This would enable autonomous, scalable, and adaptive mission coordination.

Explainability vs. Superhuman Autonomy

The trained agents exhibit strong performance but little transparency into the rationale behind their decisions. This is a common trade-off in reinforcement learning: policies often outperform rule-based systems but sacrifice interpretability.

The current platform offers a foundation for exploring explainable autonomy. Future extensions could:

- Log and visualize trajectories to understand behavior clustering
- Integrate saliency-based attention visualizations to highlight which threats influenced actions
- Compare hybrid (rule + RL) agents to purely learned policies under controlled scenarios

Such studies are critical for real-world deployment, where operator trust, safety assurance, and debugging are equally important as raw performance.

Heterogeneous Fleet Modeling

Future deployment scenarios may involve heterogeneous teams composed of agents with varying roles, modalities, and movement capabilities — for example, UAVs paired with ground vehicles or specialized units for observation, delivery, and engagement.

This would enable missions such as:

- Persistent surveillance using long-range observers
- Search and rescue with coordinated exploration and response
- Multi-role collaboration where some agents detect, others engage, and others transport or extract

To support this, several technical challenges must be addressed:

- Agent-specific observation and action spaces to reflect distinct sensors and dynamics
- Role-based reward shaping to ensure diverse behaviors are incentivized
- Multi-policy training (or shared policies with embedded role identifiers) for behavior specialization
- Flexible communication frameworks for task negotiation and situational awareness
- Trajectory deconfliction across roles to avoid congestion and redundancy

This direction represents a major step toward general-purpose, mission-adaptive autonomous systems operating in mixed domains.

4.7 Future Implementation

While the current system meets its objectives in a simplified environment, several enhancements can be made to bring the platform closer to real-world readiness and unlock more advanced learning dynamics.

Missile Assignment via Reinforcement Learning

Currently, missile assignment is governed by a centralized rule-based system. While effective for small teams, this becomes a bottleneck in larger swarms or more dynamic scenarios. A promising improvement would be to learn this logic using a dedicated reinforcement learning policy, either:

- Trained separately and integrated with the main agent policy, or
- Jointly optimized as part of a multi-policy framework

This would allow agents to perform dynamic, decentralized task allocation, adapt to real-time threat changes, and reduce reliance on static logic. In high-density scenarios, a learned assignment mechanism would scale better and potentially lead to more optimal threat coverage.

3D Simulation and Unreal Engine Integration

A natural progression is to migrate the simulation from 2D to 3D, using engines like Unreal Engine paired with AirSim. This would allow all state and observation components to be extended with an additional z-axis dimension, enabling agents to plan and maneuver vertically. Unreal Engine also supports collision detection, which opens the door to simulating friendly fire, proximity constraints, or physical impact modeling. Agents would have to respect minimum separation buffers and collision boundaries, introducing new coordination challenges.

Smarter Opponents and AI-vs-AI Self-Play

In the current setup, missiles follow predictable linear paths. In future versions, they could be replaced with smart adversarial drones that exhibit evasive behaviors or even use their own RL policy. Once this is in place, the trained defender agents can be used as opponents for training a new generation of attacker agents. This self-play loop can be repeated iteratively to bootstrap a more powerful and general-purpose air defense system, similar to competitive RL seen in strategy games and multi-agent benchmarks.

Graph-Based Reasoning with GNNs

The missile-agent relationship structure is naturally a graph — drones and threats form nodes with spatial relationships as edges. A Graph Neural Network (GNN) could encode these relationships and enable more expressive reasoning over global threat structure, e.g., clustering, threat zones, or joint coverage estimation. This is an exploratory direction that could improve generalization to large-scale swarm scenarios.

Observation Encoding with Vision-Based Models

Another future direction is to replace vector-based observations with image-based inputs, e.g., simulated top-down camera views. This would make the agent more deployable in sensor-driven environments and test the policy's ability to learn from raw visual data. A CNN-based encoder would be required to preprocess the image before feeding it into the PPO network. While this approach increases model complexity and computational cost, it enables more scalable policies that are less dependent on handcrafted features.

Chapter 5

Conclusion

This project developed a multi-agent reinforcement learning framework for autonomous aerial base defense, using a custom 2D simulation environment where drones intercept incoming missile threats. The system was refined through key improvements, including the transition from position-based to velocity-based control, the introduction of smoothness and pursuit rewards, and the integration of a centralized missile assignment system to improve coordination and threat coverage.

A major contributor to training success was the evolution of the training environment. The initial fixed-length episodes were replaced with a continuous, curriculum-based structure, where missiles respawned dynamically and difficulty increased over time. This infinite training gym enabled agents to develop scalable, generalizable behaviors across progressively more complex scenarios.

The final policy demonstrated consistent performance in multiple combinations of agent numbers and missiles numbers, effectively adapting to changing threat patterns and maintaining coverage. These behaviors emerged from the combined impact of structured rewards, curriculum scaling, and the centralized missile assignment system.

Looking forward, the system provides a foundation for several impactful extensions: replacing the rule-based assignment logic with a learned reinforcement learning policy for decentralized task allocation, migrating the simulation into a 3D engine with altitude and collision modeling, training against smarter adversarial agents via self-play, and exploring structured reasoning through Graph Neural Networks (GNNs) or image-based perception using convolutional encoders. These future enhancements would further increase realism, scalability, and the system's potential for deployment in real-world autonomous defense applications.

For reproducibility and further exploration, the full project repository is available at: https://github.com/PiusLim373/missile_defense_marl