Autonomous Drone Control: A Reinforcement Learning Approach

Auteur : Hansen, Julien
Promoteur(s) : Ernst, Damien
Faculté : Faculté des Sciences appliquées
Diplôme : Master en ingénieur civil en informatique, à finalité spécialisée en "intelligent systems"
Année académique : 2024-2025
URI/URL : http://hdl.handle.net/2268.2/23358

UNIVERSITE DE LIEGE
FACULTE DES SCIENCES APPLIQUEES

Autonomous Drone Control: A Reinforcement Learning Approach

Hansen Julien

Thesis presented to obtain the degree of :
Master of Science in Computer Science Engineering

Thesis supervisor :
Ernst Damien

Academic year: **2024 - 2025**

ABSTRACT

Drones have become an essential tools across a wide range of industries, from agriculture to surveillance, and are increasingly deployed in military contexts for detection, recognition, identification, exploration, and combat purposes. While most systems remain controlled by human, the shift toward autonomy is intensifying, driven by breakthroughs in artificial intelligence, notably in reinforcement learning and scalable simulation techniques.

This Master's thesis explores the potential of reinforcement learning for drone control within both single-agent and multi-agent frameworks. Two tasks are addressed: navigation in unknown terrains and adversarial drone combat. Our work focuses on designing simulation environments that model the learning process of agents as they interact with these tasks. Our navigation environment consists of multiple randomly spaced obstacles (spikes), a target, and a drone placed on opposite sides of the terrain. The drone is equipped with a sensor, either a LiDAR or a camera which it uses to explore the environment and reach the target. In the adversarial scenario, the environment includes two drones: an attacker and a defender. The attacker attempts to reach a designated target, while the defender tries to intercept it by colliding with it.

Reinforcement learning is particularly well suited to these tasks due to its ability to learn complex, sequential decision-making policies from interaction with the environment. In scenarios such as drone navigation or combat, where the environment is often partially observable, highly dynamic, and difficult to model analytically, RL offers a flexible and data-driven approach to learning effective control strategies. Furthermore, Reinforcement learning naturally supports learning in multi-agent settings, where agents must coordinate or compete in real time.

To tackle these tasks, *policy gradient* methods such as *Proximal Policy Optimization* , its multi-agent extension *Independent Proximal Policy Optimization* and a variant inspired by self-play methods were explored. To train and evaluate our agents, IsaacLab environments were designed following the formalism of partially observable Markov decision process and stochastic games. Our work highlights the performance of trained agents regarding these tasks and show promising potential for future improvements regarding autonomous drone control.

RESUME

Les drones sont devenus des outils essentiels dans de nombreux secteurs, de l'agriculture à la surveillance, et sont de plus en plus déployés dans des contextes militaires pour des missions de détection, reconnaissance, identification, exploration et combat. Bien que la plupart des systèmes restent contrôlés par des humains, la transition vers l'autonomie s'intensifie, portée par des avancées en intelligence artificielle, notamment dans l'apprentissage par renforcement et les techniques de simulation à grande échelle.

Ce mémoire de Master explore le potentiel de l'apprentissage par renforcement pour le contrôle de drones, dans des cadres à agent unique ainsi qu'à agents multiples. Deux tâches sont abordées : la navigation dans des terrains inconnus et le combat aérien adversarial. Notre travail se concentre sur la conception d'environnements de simulation modélisant le processus d'apprentissage des agents en interaction avec ces tâches. Notre environnement de navigation est composé de multiples obstacles (pics) placés aléatoirement, d'une cible et d'un drone positionnés de part et d'autre du terrain. Le drone est équipé d'un capteur, soit un LiDAR, soit une caméra u'il utilise pour explorer l'environnement et atteindre la cible. Dans le scénario adversarial, l'environnement comprend deux drones : un attaquant et un défenseur. L'attaquant cherche à atteindre une cible désignée, tandis que le défenseur tente de l'intercepter en entrant en collision avec lui.

L'apprentissage par renforcement est particulièrement adapté à ces tâches grâce à sa capacité à apprendre des politiques complexes de prise de décision séquentielle via l'interaction avec l'environnement. Dans des scénarios tels que la navigation ou le combat de drones, où l'environnement est souvent partiellement observable, très dynamique et difficile à modéliser analytiquement, le RL offre une approche flexible et basée sur les données pour apprendre des stratégies de contrôle efficaces. De plus, le RL supporte naturellement l'apprentissage en contextes multi-agents, où les agents doivent coopérer ou s'affronter en temps réel.

Pour relever ces défis, des méthodes dites de *policy-gradient*, telles que *Proximal Policy Optimization*, son extension à plusieurs agents *Independent Proximal Policy Optimization*, ainsi qu'une variante inspirée du paradigme *self-play*, ont été explorées. Pour entraîner et évaluer nos agents, des environnements IsaacLab ont été conçus en suivant le formalisme des processus de décision markoviens et des jeux stochastiques. Notre travail met en lumière les performances des agents entraînés sur ces tâches et révèle un potentiel prometteur pour de futures améliorations en matière de contrôle autonome de drones.

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my thesis advisor, Professor Damien Ernst, whose expertise and guidance have greatly enriched my graduate experience.

I would also like to sincerely thank Research Engineer Leroy Pascal and PhD student Louette Arthur for their continuous support, generosity, and kindness during the years. Their willingness to share their time, knowledge, and experience has been truly invaluable to me. The many discussions, thoughtful advice, and feedback I received from them have not only contributed greatly to the progress of this work, but also made this journey far more enriching and enjoyable.

I am deeply grateful to my family for their unwavering support and encouragement throughout my academic journey. A special mention goes to Quarto Kassandra, whose help and constant presence were particularly meaningful during this period. Their belief in me has been a constant source of motivation.

Finally, I would like to extend my heartfelt thanks to my friends, whose continuous support, encouragement, and good humor made this journey an unforgettable experience.

CONTENTS

CHAPTER 1

INTRODUCTION

Drones have changed heavily since their early use as rudimentary military decoys (Blom, 2010). Nowadays drones are widely deployed across a broad range of civilian, commercial and military applications (Wikipedia contributors, 2025). Modern drones vary in design and functionality. From unmanned combat aerial vehicles (CAV) design for reconnaissance and precision strikes, to medium-sized fixed-wing systems created for surveillance and monitoring, and even small rotary-wing drones that began as tools for photography and are now increasingly weaponized providing strategic advantages, disrupt enemy operations, and gather critical data, marking a significant advancement in drone warfare (Blom, 2010) see Figure 1.1.

(a) Rotary-wing drone

(b) Fixed-wing drone

(c) CAV

Figure 1.1: Different Types of drones.

In the civilian domain, drones are routinely deployed to monitor deforestation (Geoawesome, 2024), measure atmospheric variables for climate research (Stone, 2025), support search-and-rescue efforts in hazardous environments (Public Safety, 2025), capture cinematic aerial footage (Viper Drones, 2025), and inspect infrastructure such as power lines and bridges (LocoRobo, 2025). In logistics, drones are increasingly used for last-mile delivery in both urban and remote areas (Franetic, 2025). In the military sphere, UASs have revolutionized modern warfare. In recent conflicts, such as the ongoing war in Ukraine (Porter & Baker, 2025), the proliferation of low-cost, commercially available drones equipped with improvised sensors and munitions has transformed battlefield tactics. Small quadcopters are now employed for real-time reconnaissance (O'Grady et al., 2025), target designation, and even offensive operations. Meanwhile, larger UASs execute precision-guided strikes without endangering personnel (Loh, 2025).

Despite their increasing capabilities, the control of UASs relies heavily on classical control strategies. Traditional systems employ Proportional-Integral-Derivative (PID) controllers (Åström & Hägglund, 1995), Linear Quadratic Regulators (Kwakernaak & Sivan, 1972), and related model-based approaches to maintain flight stability, execute trajectory tracking, and

compensate for environmental disturbances. These methods process sensor feedback. These sensors include gyroscopic, accelerometric, barometric, and GPS data to continuously adjust motor commands and ensure desired flight behavior. However, classical control approaches remain constrained by their reliance on accurate dynamical models and assumptions of environmental stationarity. In real-world scenarios, particularly in contested environment with GPS denial, electromagnetic interference or rapidly evolving terrain, these assumptions may be to restrictive.

To address these limitations, Reinforcement Learning (RL) has emerged as a promising alternative. RL is a branch of machine learning in which an agent learns to make decisions by interacting with an environment, receiving a scalar feedback signal called reward, and adapting its behavior to maximize long-term cumulative reward. Unlike classical control, which relies on predefined models and control laws, RL enables autonomous agents to learn control policies through trial-and-error, progressively refining their strategies based on observed outcomes. These learned policies are optimized to satisfy mission objectives such as energy efficiency, obstacle avoidance, or adversary neutralization. RL has demonstrated impressive results on multiple domains such as, the games of Chess and Go (Silver et al., 2016), real-time strategy games such as StarCraft II (Vinyals et al., 2019), and Atari-like visual environments (Mnih et al., 2013). In the realm of autonomous flight, RL agents have also outperformed expert human pilots in drone racing competitions (Kaufmann et al., 2023). Despite these breakthroughs, real-world deployment of RL into drones remains constrained by several challenges like sample inefficiency and safety. RL training typically requires millions of interactions, which are impractical and unsafe to perform on physical hardware due to wear, risk of crashes, and operational costs.

To solve these issues, high-fidelity simulators have become central to the development and training of RL-based drone control policies. In this thesis, the IsaacSim simulator and the IsaacLab framework (NVIDIA, 2024) were selected to create scalable, physics-accurate training environments. This manuscript focuses on two primary UAS control tasks: navigation in partially observable, unknown terrains, and adversarial combat in multi-agent settings. These tasks were selected as they represent real-world challenges faced frequently. Navigation in unknown environments under partial observability models the complexity of autonomous flight in unknown area such as battlefields where GPS signals are denied, where the drone must rely solely on sensors data. Our environment composed of various randomly spaced spikes, a target and a drone models these tasks. Regarding the combat environment, as stated, the conflict in Ukraine highlights the growing deployment of weaponized versions of commercial drones. Their small size and speed impose constraints and complexity on designing counter-measures that require minimal resources and time. This environment explores the deployment of interceptor drones as counter-measures to these threats, in it two drones compete against each other, an attacker tries to reach a target while the defender tries to stop it by colliding against. The training of these agents were performed with policy gradient methods, in the single-agent setting, Proximal Policy Optimization (PPO) was selected (Schulman et al., 2017). In the multi-agent setting, training was performed with Independent PPO (Schulman et al., 2017) and a new variant we introduce called S-IPPO based on self-play methods.

The thesis is structured as follows. Chapter 2 introduces the theoretical background in single-agent RL and decision processes. Chapter 3 surveys the related literature addressing UAS control through RL and optimal control. Chapter 4 details our single-agent environment, including the implementation of scenarios within IsaacSim/IsaacLab, the learning algorithms and experimental methodology as well as the results obtained. Chapter 5 presents the multi-agent extension, through our paper entitled "*Autonomous Drone Combat: A Multi-Agent Reinforcement Learning Approach*". Chapter 6 concludes this thesis and outlines future work.

CHAPTER 2

THEORETICAL BACKGROUND

In the first section of this chapter, RL is described, along with the necessary formalism for partially observable environment. In the second section, policy gradient methods are introduced along with some of their limitations.

2.1 Markov Decision Process

Many real-world problems, including autonomous control and robotics, can be formalize as *sequential decision processes* (Sutton & Barto, 2018), where an agent must make a series of decisions over time to achieve a goal. At each time step, the agent observes the current state of the system, takes an action, receives feedback in the form of a reward or cost, and transitions to a new state. The objective is to find a strategy or policy that optimizes the long-term return accumulated over time.

Sequential decision-making are usually model through the formalism of *Markov Decision Process* (MDP) (Sutton & Barto, 2018). An MDP is defined as a tuple $(\mathcal{S}, \mathcal{U}, p, r, \gamma)$, where $\mathcal{S}$ denotes the set of possible states in the environment, and $\mathcal{U}$ represents the set of actions possible of the agent. The transition probability are governed by $p(s'|s, u)$. The reward function $r(s, u)$ is the scalar feedback signal given to each state-action pair. Finally, the discount factor $\gamma \in [0, 1)$ determines the present value of future rewards. It serves two purposes: it guarantees the convergence of the cumulative reward in infinite-horizon settings and consequently prioritizies immediate rewards over distant ones. During a timestep, the agent samples actions from the policy $\pi(\cdot|s_t)$ and transitions to the next state s_{t+1} according to the probabilistic dynamics $p(\cdot|s_t, u_t)$ and receive the reward $r(s_t, u_t)$ (see Figure 2.1).

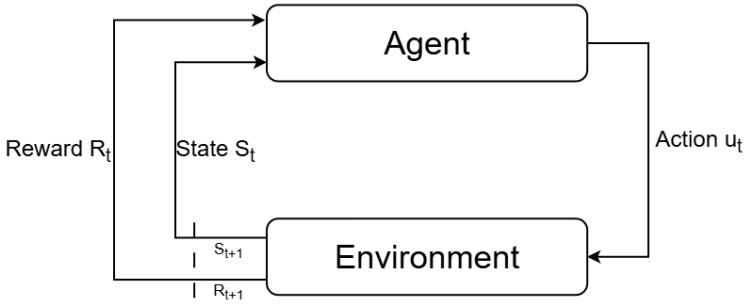

Figure 2.1: MDP

2.2 Reinforcement Learning

RL extensively relies on MDP to formulate problems. The agent's objective is to learn a policy π^*, a mapping from states to actions that maximizes the expected cumulative reward over time. This objective is formalized in Equation 2.1:

$$\pi^* = \max_{\pi} \mathop{\mathbb{E}}_{\substack{u_t \sim \pi(\cdot|s_t) \\ s_{t+1} \sim p(\cdot|s_t, u_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, u_t) \right], \tag{2.1}$$

Nowadays RL has become a widely adopted approach for solving decision-making problems across various domains, including robotics (Kober et al., 2013), power network control (Ernst et al., 2005 ; Ernst et al., 2008), pricing strategies (Vengerov, 2008), and resource management (Barrett et al., 2012), among others. The goal is to maximize long-term cumulative rewards by selecting actions through a learned policy (Wiering & van Otterlo, 2012). As stated previously, the essence of RL lies in optimizing a policy purely through interaction with the environment. These interactions may be generated by the current policy being optimized (on-policy) or by a different policy (off-policy), such as a prior version of the policy or an explicit exploration strategy. In *model-based* RL, a model of the environment's dynamics is first learned or is known. This model can then be used to compute optimal policies using principles from optimal control theory (Moerland et al., 2020).

Another class of methods, *value-based* RL methods aim to learn a value function without explicitly modeling the environment's dynamics (Sutton & Barto, 2018). The value function estimates the expected return of taking a specific action in a given state, and policies are derived by selecting actions that maximize this value. These approaches prove successful on a wide range of tasks, including mastering ATARI games (Mnih et al., 2013). However, value-based methods often struggle in environments with continuous action spaces (Sutton & Barto, 2018).

To address this limitation, *policy-based* methods directly learn a parameterized policy by optimizing it using techniques such as gradient ascent. These algorithms, including recent state-of-the-art approaches, demonstrate strong performance in complex, high-dimensional control tasks (Schulman et al., 2017). Both value-based and policy-based approaches that do not require an explicit model of the environment are commonly referred to as *model-free* methods.

2.2.1 Partially Observable Markov Decision Process

In many real-world environments, agents do not have full access to the true state of the environment. Instead, they rely on partial, noisy, observations to make decisions. This scenario can be modeled by Partially Observable Markov Decision Processes (POMDPs) (Sutton & Barto, 2018), which generalize MDPs. A POMDP introduces an observation space and a stochastic observation function that defines a probability distribution over observations given the current state and action. The decision process in a POMDP can be seen in Figure 4.2. This added complexity enables the modeling of more realistic settings where full observability is not possible. As in the standard MDP case, reinforcement learning in POMDPs train agent to learn an optimal policy purely through interaction, without access to the underlying dynamics or observation model. Planning and control in POMDPs are significantly more challenging due to the need to reason over belief states.

Definition: A POMDP is defined as the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{U}, p, r, \Omega, \mathcal{O}, \gamma, \rho, T)$, where $\mathcal{S}$ is the set of possible states in the environment, $\mathcal{U}$ is the set of actions, and $p : \mathcal{S} \times \mathcal{U} \to \Delta(\mathcal{S})$ is the transition function that defines the probability of transitioning between states given an action. The reward function $r : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$ assigns a scalar reward for each state-action pair. The set

Ω denotes the observation space, and $\mathcal{O} : \mathcal{S} \to \Delta(\Omega)$ is the observation function defining the probability of perceiving each observation given the state. The discount factor $\gamma \in [0, 1)$ weighs future rewards, $\rho \in \Delta(\mathcal{S})$ is the initial state distribution, and $T \in \mathbb{N} \cup \{\infty\}$ denotes the time horizon.

At each time step t, the environment is in a state $s_t \in \mathcal{S}$. The agent takes an action $u_t \in \mathcal{U}$, receives a reward $r(s_t, u_t)$, transitions to a new state $s_{t+1} \sim \mathcal{P}(\cdot|s_t, u_t)$, and receives an observation $o_{t+1} \sim \mathcal{O}(\cdot|s_{t+1})$.

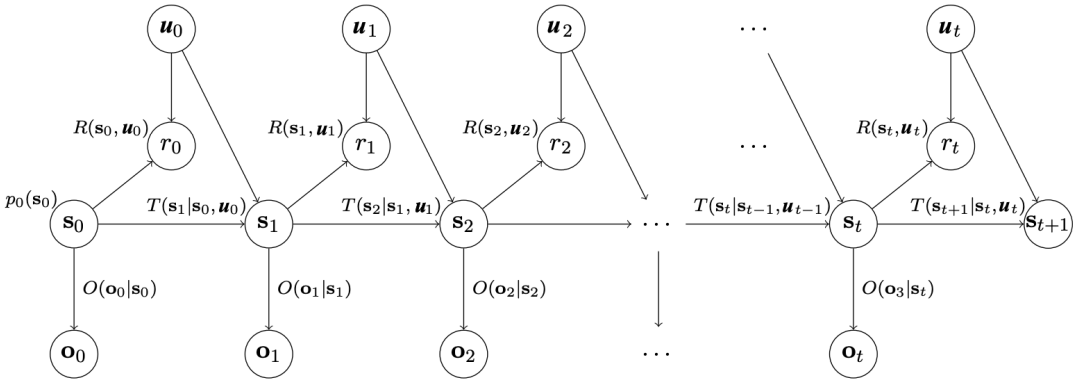

Figure 2.2: Bayesian representation of a POMDP execution (Lambrechts, 2021)

2.3 Policy Gradient Methods

Policy Gradient methods represent a class of algorithms within the RL paradigm (Sutton & Barto, 2018). In contrast to action-value based methods, which learn the values of actions and derive a policy by selecting actions based on their estimated values, PG methods directly learn a parameterized policy that select actions (Sutton & Barto, 2018). These methods learn the policy parameters through the gradient of a scalar performance measure $J(\theta)$ with respect to the policy parameters. The aim is to maximize performance by performing gradient ascent:

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla \hat{J}(\boldsymbol{\theta}_t) \tag{2.2}$$

where $\nabla \hat{J}(\boldsymbol{\theta}_t)$ is an unbiased estimator of the gradient $\nabla J(\boldsymbol{\theta}_t)$, computed from sampled trajectories. By parameterizing the agent's policy as π_θ, where $\theta \in \mathbb{R}^d$, a probability distribution over actions given states: $\pi_\theta(u|s) = P(u_t = u \mid s_t = s, \theta)$. The objective then becomes finding the optimal parameters θ^* that maximize the expected total discounted reward.

$$\theta^* = \arg\max_\theta J(\theta) = \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{T} \gamma^t r(s_t, u_t) \right] \tag{2.3}$$

where $\tau = (s_0, u_0, r_1, s_1, u_1, \dots)$ is a trajectory sampled under π_θ, $\gamma \in [0, 1]$ is the discount factor, and $r(s_t, u_t)$ is the reward at time step t.

2.3.1 The Policy Gradient Theorem

Computing the policy gradient $\nabla_\theta J(\theta)$ is challenging due to the dependence of trajectory distributions $\tau = (s_0, u_0, s_1, u_1, \dots, s_T)$ on the policy parameters θ through the environment dynamics, which are typically unknown and non-differentiable. An important result in policy gradient methods, the *Policy Gradient Theorem* (Sutton et al., 1999) resolves this by expressing the

gradient in a form that don't need to differentiate the environment's transition model. Starting from the expected return objective:

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right], \quad \text{where } R(\tau) = \sum_{t=0}^{T} \gamma^t r(s_t, u_t), \tag{2.4}$$

The log-likelihood trick is applied:

$$\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)] = \mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\nabla_\theta \log p_\theta(\tau)\right], \tag{2.5}$$

where $p_\theta(\tau)$ denotes the likelihood of the trajectory τ under policy π_θ. Assuming a Markovian policy, $\pi_\theta(u_t \mid s_t)$, which defines a probability distribution over actions conditioned only on the current state s_t and fixed environment dynamics, the trajectory likelihood factorizes as:

$$\log p_\theta(\tau) = \log \rho(s_0) + \sum_{t=0}^{T} \log \pi_\theta(u_t|s_t) + \sum_{t=0}^{T-1} \log p(s_{t+1}|s_t, u_t), \tag{2.6}$$

where $\rho(s_0)$ is the initial state distribution and $p(s_{t+1}|s_t, u_t)$ is the transition distribution. Since neither term depends on θ, their gradients vanish:

$$\nabla_\theta \log p_\theta(\tau) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t|s_t). \tag{2.7}$$

Substituting this into the gradient of $J(\theta)$ gives us:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} R(\tau)\nabla_\theta \log \pi_\theta(u_t|s_t)\right]. \tag{2.8}$$

To reduce variance, the total return $R(\tau)$ is usually replaced by the discounted cumulative reward G_t defined as:

$$G_t = \sum_{k=t}^{T} \gamma^{k-t} r(s_k, u_k), \tag{2.9}$$

This gives us the final form of the policy gradient:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t|s_t)\, G_t\right]. \tag{2.10}$$

This expression allows to estimate the gradient using sampled trajectories without the need of the environment's transition probabilities. The term $\nabla_\theta \log \pi_\theta(u_t|s_t)$ is often referred to as the *score function*, a concept from statistics that represents the sensitivity of the log-likelihood to the parameters θ. It indicates how a small change in the policy parameters affects the probability of the chosen action.

2.3.2 The REINFORCE Algorithm

The REINFORCE algorithm, also known as Monte Carlo Policy Gradient, is a direct application of the Policy Gradient Theorem (Williams, 1992). It employs Monte Carlo sampling by collecting complete trajectories to estimate the expectation defined in Equation 2.10. For each trajectory, the policy parameters are updated in the direction of the estimated gradient. The update rule for θ is given by:

$$\hat{\nabla}_\theta J(\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t|s_t)\, G_t, \tag{2.11}$$

A principal challenge associated with the REINFORCE algorithm is the high variance of its gradient estimator. In REINFORCE, the policy gradient is approximated using sampled trajectories as:

$$\theta \leftarrow \theta + \alpha \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t|s_t) G_t \tag{2.12}$$

This estimator is unbiased, but since G_t is a sum of random variables influenced by the stochasticity of both the policy and environment, its variance can be important. As a result, the gradient updates may be noisy, leading to slow and unstable convergence (Williams, 1992). To reduce this variance and maintain an unbiased estimate, a common technique is to subtract a state-dependent *baseline* $b(s_t)$ from the return G_t. Provided that $b(s_t)$ does not depend on the action u_t, this subtraction does not introduce bias, since:

$$\mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(u_t|s_t)\, b(s_t) \right] = 0. \tag{2.13}$$

The resulting gradient estimator with the baseline becomes:

$$\hat{\nabla}_\theta J(\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(u_t|s_t)\ \left(G_t - b(s_t) \right), \tag{2.14}$$

which typically exhibits lower variance and leads to more stable learning. A well-chosen baseline can significantly reduce the variance of the estimate (Williams, 1992). A natural choice for the baseline is the state-value function, $b(s_t) = V^{\pi_\theta}(s_t) = \mathbb{E}_{\pi_\theta}[G_t|s_t = s]$. The term $(G_t - V^{\pi_\theta}(s_t))$ is an estimate of the *advantage function*, $A^{\pi_\theta}(s_t, u_t) = Q^{\pi_\theta}(s_t, u_t) - V^{\pi_\theta}(s_t)$, which quantifies whether an action is better or worse than the policy's average behavior in that state.

2.3.3 Actor-Critic Methods

The concept of using a baseline leads to *Actor-Critic* methods introduced by Sutton et al. (1999). These methods explicitly learn both a policy (the Actor) and a value function (the Critic).

- **The Actor** is the parameterized policy $\pi_\theta(a|s)$ responsible for action selection. It updates its parameters θ using the policy gradient.
- **The Critic** is a parameterized value function, typically the state-value function $V_w(s) \approx V^{\pi_\theta}(s)$ with parameters w. It evaluates the actions taken by the Actor by providing a low-variance estimate of the return, often in the form of a Temporal Difference (TD) error, which serves as an estimate of the advantage function.

The Critic is trained on TD errors, while the Actor is updated using the output from the Critic. This synergy allows for more stable and sample-efficient learning compared to vanilla REINFORCE, making Actor-Critic architectures, such as Advantage Actor-Critic (A2C) and Asynchronous Advantage Actor-Critic (A3C), a dominant paradigm in modern reinforcement learning (Mnih et al., 2016). Despite the introduction of variance reduction techniques and Actor-Critic architectures, vanilla policy gradient methods still face a critical challenge: the sensitivity to the step size α. An inappropriately large step can lead to a catastrophic degradation in performance, from which the policy may not recover (Schulman et al., 2015). This occurs because a single bad update can move the policy parameters θ into a region that samples poor trajectories, preventing effective learning. This sensitivity makes the optimization process unstable and difficult to tune. The following methods aim to address this issue by restricting the magnitude of policy updates.

Trust Region Policy Optimization

Advancements in policy gradient methods are crucial for addressing the high-dimensional, continuous control problems inherent in robotics. However, standard policy gradient algorithms are often susceptible to performance collapse due to large, high-variance gradient updates. A significant step towards mitigating this instability was the development of TRPO (Schulman et al., 2015). TRPO reframes the policy update as a constrained optimization problem, ensuring that each new policy remains within a "trust region" of the previous one, thereby guaranteeing monotonic policy improvement under certain theoretical assumptions.

The core principle of TRPO is to maximize a surrogate objective function, which approximates the expected return of the new policy, subject to a constraint on the "size" of the policy update. This size is measured by the Kullback-Leibler (KL) divergence, a statistical measure of the difference between two probability distributions. At each iteration, TRPO solves the following constrained optimization problem:

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & \mathbb{E}_{s_t \sim \rho_{\theta_{\text{old}}}, u_t \sim \pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(u_t|s_t)}{\pi_{\theta_{\text{old}}}(u_t|s_t)} \hat{A}_{\theta_{\text{old}}}(s_t, u_t) \right] \\ \text{subject to} \quad & \mathbb{E}_{s_t \sim \rho_{\theta_{\text{old}}}} \left[D_{\text{KL}} \left(\pi_{\theta_{\text{old}}}(\cdot|s_t) \parallel \pi_\theta(\cdot|s_t) \right) \right] \leq \delta \end{aligned} \tag{2.15}$$

In this formulation, π_θ represents the new policy parameterized by θ, while $\pi_{\theta_{\text{old}}}$ is the old policy from which the data was sampled. The term $\hat{A}_{\theta_{\text{old}}}(s_t, u_t)$ is an estimator of the advantage function under the old policy (Schulman et al., 2015). The objective function leverages importance sampling to estimate the performance of the new policy using trajectories collected under $\pi_{\theta_{\text{old}}}$. The constraint, governed by the KL divergence D_{KL}, limits the average divergence between the old and new policies to a small value δ, which effectively defines the trust region for the update (Schulman et al., 2015).

The KL divergence between policies is a nonlinear function of the policy parameters, making it computationally expensive and mathematically complex to evaluate and optimize exactly, especially in high-dimensional parameter spaces. Directly enforcing a constraint on the exact KL divergence during policy updates would require repeatedly computing this divergence for different parameter values, which is often intractable. To address this, TRPO approximates the KL divergence using a second-order Taylor expansion around the current policy parameters. This local quadratic approximation expresses the KL divergence in terms of the Fisher Information Matrix, which captures the curvature of the divergence with respect to the policy parameters. The approximation transforms the original complex constraint into a quadratic form:

$$D_{\text{KL}}(\pi_{\theta_{\text{old}}} \| \pi_\theta) \approx \frac{1}{2} (\theta - \theta_{\text{old}})^\top \mathbf{F} (\theta - \theta_{\text{old}}). \tag{2.16}$$

The Fisher Information Matrix $\mathbf{F}$ can be interpreted as the Hessian (second derivative matrix) of the KL divergence with respect to the policy parameters. Since $\mathbf{F}$ is typically very large, explicitly computing, storing, and inverting it is computationally expensive. To overcome this, TRPO uses the conjugate gradient (CG) method to approximately solve the constrained optimization problem. The CG method is an iterative algorithm for solving large linear systems of the form $\mathbf{F}x = g$, where g is the policy gradient. Instead of requiring explicit computation or inversion of $\mathbf{F}$, CG only needs to compute matrix vector products with $\mathbf{F}$. These matrix-vector products, are equivalent to *Hessian-vector products* of the KL divergence. Hessian-vector products can be efficiently computed using automatic differentiation. However, these operations still involve computing second-order derivatives, which remain computationally intensive.

Proximal Policy Optimization

To address the implementation complexity and computational overhead of TRPO, Schulman et al. (2017) introduced PPO. This algorithm seeks to retain the benefits of TRPO's trust region updates while providing a simpler and more computationally efficient approach. Instead of enforcing a hard constraint on the KL-divergence, PPO replaces it with a novel clipped surrogate objective that implicitly limits policy updates. Let the probability ratio between the new and old policies be defined as $r_t(\theta) = \frac{\pi_\theta(u_t|s_t)}{\pi_{\theta_{\text{old}}}(u_t|s_t)}$. Then, the PPO clipped objective is given by:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{2.17}$$

where ϵ is a hyperparameter controlling the clipping range. The clipping mechanism takes the minimum of two terms. The first term, $r_t(\theta)\hat{A}_t$, corresponds to the standard policy gradient objective, which encourages increasing the probability of actions with positive advantage and decreasing it for actions with negative advantage. The second term, $\text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t$, modifies this objective by restricting the probability ratio $r_t(\theta)$ to lie within the interval $[1-\epsilon, 1+\epsilon]$. When the advantage $\hat{A}_t$ is positive, if the policy update attempts to increase the probability ratio beyond $1+\epsilon$, the clipped term limits this increase, preventing overly large updates that could destabilize training. Similarly, when the advantage is negative, if the update tries to reduce the ratio below $1-\epsilon$, the clipping restricts this decrease, avoiding excessively aggressive reductions in action probability. This mechanism prevents the policy from changing too drastically in a single update, thereby approximating the trust region constraint of TRPO while avoiding the computational complexity of second-order optimization (Schulman et al., 2017).

The full loss function for PPO also includes terms for value function approximation and entropy regularization, which encourages exploration:

$$L_t(\theta) = \mathbb{E}_t \left[L_t^{\text{CLIP}}(\theta) - c_1 L_t^{\text{VF}}(\theta) + c_2 S[\pi_\theta](s_t) \right], \tag{2.18}$$

where $L_t^{\text{VF}} = (V_\theta(s_t) - V_t^{\text{targ}})^2$ is the squared-error loss for the value function critic, $S[\pi_\theta](s_t)$ is an entropy term, and c_1 and c_2 are weighting coefficients balancing the contributions of the value function loss and the entropy regularization, respectively. Due to its favorable trade-off between sample efficiency, stability, and ease of implementation, PPO has become one of the most widely adopted RL algorithms in robotics and other domains.

CHAPTER 3

RELATED WORK

Designing control systems is a fundamental challenge in robotics. These systems are essential for real-world robotic applications such as autonomous drone control. This chapter presents related work in drone control through the two main paradigms that have emerged: model-based classical and optimal control and learning driven reinforcement learning.

3.1 Model-Based Control Strategies

The control of quadrotor drones is a complex challenge due to their inherently nonlinear and unstable dynamics. Over the years, researchers have applied a wide range of control strategies, which can be categorized into two main paradigms: *classical control* and *optimal control*.

Autonomous drone control was first explored through the field of classical control. The first studies adopted linear controllers and proved effective for achieving stable flight in hovering conditions. Bouabdallah et al. (2004) focused on achieving full attitude and altitude stabilization. They designed a control system using a structure of nested PID loops. An inner loop managed the high-frequency angular velocities (the speed of roll, pitch, and yaw) to provide rapid damping, while an outer loop corrected the slower-changing attitude angles and altitude. Their work provided a foundational proof that a standard, well-tuned PID structure was effective for achieving a stable hover. Hoffman et al. (2007) extended this to achieve stable 3D position tracking. They also used a cascaded architecture where an outer PID loop calculated the error between the drone's desired and actual position. This loop's output was not a direct motor command, but rather the desired roll and pitch angles needed to move the drone toward its target. An inner PID loop then received these desired angles and actuated the motors to achieve them, enabling the drone to accurately follow a path in space.

Another popular linear method is the Linear Quadratic Regulator (LQR), an optimal control solution for linearized systems. LQR-based trajectory tracking has already show interesting results. For example Faessler et al. (2017) designed a LQR controller to stabilize the UAS in hover conditions and to track desired attitude commands.

The main limitations of linear controllers such as LQR lie in their reliance on a linearized model of the system dynamics. This simplification becomes inadequate during aggressive maneuvers or under significant external disturbances, where the drone's behavior deviates considerably from the nominal operating point. As a result, linear controllers struggle to maintain

performance across the full range of drone dynamics, necessitating the development of more advanced techniques capable of handling nonlinearities and uncertainties inherent in the system.

A popular nonlinear approach is feedback linearization, which mathematically transforms the nonlinear system into an equivalent linear one, enabling the use of classical linear controllers. The control commands computed in the linear domain are then mapped back to the original nonlinear system. Bonna et Camino (2015) successfully employed feedback linearization to achieve stable and precise tracking of three-dimensional trajectories. Other nonlinear control methods such as backstepping control and sliding mode control have also been explored (Castillo et al., 2005 ; Bouabdallah & Siegwart, 2005). Castillo et al. (2005) developed a hierarchical backstepping-based controller for real-time stabilization and trajectory tracking, it demonstrates precise control in both hover and dynamic flight conditions while providing robustness against modeling uncertainties. Bouabdallah et Siegwart (2005) apply both backstepping and sliding mode techniques to an indoor micro quadrotor and validated them through extensive flight experiments, showing that sliding mode control provided enhanced robustness to disturbances and improved tracking performance during aggressive maneuvers and in the presence of modeling errors.

While nonlinear classical methods enhance robustness, another paradigm provides an even more powerful framework, the Optimal Control (OC) field which explicitly optimize system behavior over time. OC requires detailed mathematical models to solve optimization problems, generating control inputs that minimize a predefined cost function while respecting system constraints. Among optimal control techniques, Model Predictive Control (MPC) has emerged as a state-of-the-art strategy for high-performance drone control. MPC combines predictive modeling with online optimization, enabling it to manage both system dynamics and constraints explicitly. This makes it particularly suitable for aggressive and safety-critical maneuvers. Jain et al. (2024) implemented an onboard MPC controller that empowered a drone to perform flips and high-speed flights in real time. Their work demonstrated exceptional trajectory tracking accuracy and an impressive ability to recover from extreme initial conditions, underscoring MPC's advantages in achieving both agility and robustness. By continuously recalculating the optimal control sequence at each time step, MPC effectively addresses the challenges posed by nonlinear dynamics and operational constraints, pushing the boundaries of autonomous drone flight.

3.2 Reinforcement Learning Approaches

In recent years, RL has emerged as a interesting alternative to traditional model-based control methods, particularly in the context of highly agile and autonomous drone flight. Unlike OC approaches, which require accurate system models and often rely on solving complex optimization problems online, RL methods learn control policies directly from interaction data. This allows RL to handle highly nonlinear system dynamics and complex tasks using raw sensory inputs, such as images. Tasks that are notably difficult for classical control techniques.

A study by Song et al. (2023) made an important contribution by directly comparing an RL-based controller to a MPC controller. They compared each controller on an autonomous drone racing tasks. The researchers trained an RL policy in simulation, which mapped observations directly to motor commands with the goal of minimizing race time. The performance of this learned policy was evaluated against a the MPC controller on the same racetrack in both simulated and real-world environments. Results demonstrated that the RL controller was significantly faster and more robust, particularly at the limits of the drone's physical capabilities. While the MPC controller was precise, it tended to be conservative or unstable during aggres-

sive maneuvers, whereas the RL policy effectively executed more daring and "on-the-edge" flight behaviors. A key limitation of this work, however, was the sim-to-real transfer gap: extensive domain randomization and training in simulation were required to ensure the learned policy generalized well to physical drones.

Building upon this foundation, Scaramuzza et Kaufmann (2023) have pushed the boundaries of agile drone flight through a series of innovative studies. Their work has demonstrated that RL can accomplish tasks traditionally reserved to expert human pilots. For instance, in their research on vision-based drone racing through dynamic, moving gates, the team developed a system comprising three modules: a convolutional neural network for gate position estimation from onboard camera images, a state estimator to track the drone's position and velocity, and an RL-trained control policy that computed motor commands based on these inputs. This approach allowed the drone to autonomously navigate complex race tracks using only onboard sensing and computation, a significant advancement over prior methods relying on external motion capture systems.

Moreover they succeed in performing acrobatic maneuvers, including power loops, barrel rolls, and Matty flips (Kaufmann et al., 2020). Their innovative training pipeline employed a "teacher-student" framework: the teacher policy was trained in simulation with privileged access to perfect state information, while a student policy was trained to imitate the teacher using only noisy, realistic sensor data. This approach enabled the successful sim-to-real transfer of acrobatic flight policies without any fine-tuning. Such maneuvers pose significant challenges for traditional controllers due to the need to operate beyond the drone's stable flight envelope, where system models lose accuracy and optimal control solutions become infeasible.

An important contribution to the field is the *Swift* system, presented by Kaufmann et al. (2023), which achieved performance exceeding world-champion human pilots in drone racing competitions. To achieve this, Swift uses a combination of learning-based and traditional algorithms to map onboard sensory readings to control commands. This mapping comprises two parts: an observation policy, which distils high-dimensional visual and inertial information into a task-specific low-dimensional encoding, and a control policy that transforms the encoding into commands for the drone. After training, the RL agent was deployed on a physical drone and beats human pilots.

CHAPTER 4
SINGLE-AGENT REINFORCEMENT LEARNING

This Chapter is dedicated to the formal definition of the components of the single-agent environment through its POMDP formalism, along with a detailed description of the corresponding training procedure.

4.1 Experimental Setup

The single-agent task is inspired by Xu et al. (2023), open-source platform designed for reinforcement learning research on multi-rotor drone systems. The goal of this task is to equipped a drone with a sensor such as a lidar or a camera in an unknown environment and to train it to achieve a particular target inside the environment while avoiding the pillars.

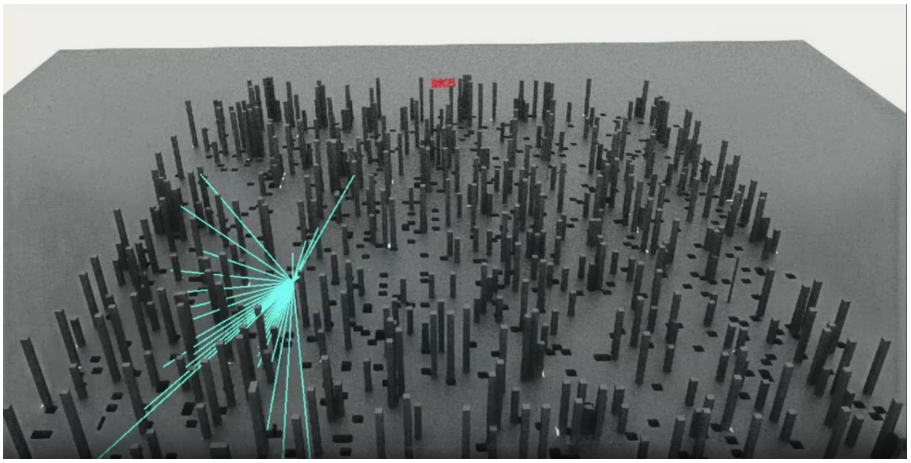

Figure 4.1: Single-agent exploration tasks

In Figure 4.1, a visual of the tasks is shown, the drone is to small to be seen but is present at the center of the blue rays, each of these ray represent the data collected by the lidar sensor. The goal of the drone is to reach the red cuboid that can be seen at the back of the environment while avoiding all the spikes.

4.1.1 State Space $\mathcal{S}$

The state space $\mathcal{S}$ consists of all possible states of the environment. In our drone combat scenario, the state space $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{\mathbf{p_d^w}, \mathbf{q_d^w}, \mathbf{p_o^w}, \mathbf{p_t^w}, \mathbf{v_d^w}, \boldsymbol{\omega}_d^w, \mathbf{g_d^b}\} \tag{4.1}$$

In which $\mathbf{p_d^w}$, $\mathbf{p_o^w}$,$\mathbf{p_t^w}$ represent respectively the position vectors of the drone, every obstacle present in the environment and the target in the world frame, $\mathbf{q_d^w}$ represents the quaternion vectors of the drone. The vectors $\mathbf{v_d^w} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_d^w$, $\boldsymbol{\omega}_a^w \in \mathbb{R}^3$ both represent the linear velocity and angular velocity vectors of the drone in the world frame and finally, $\mathbf{g_d^b} \in \mathbb{R}^3$ represents the projected gravity vector in the base frame providing orientation information.

4.1.2 Observability Space $\mathcal{O}$

The observation space $\mathcal{O}$ consists of the observation received by the agent. In our scenario, the observation space $\mathcal{O}$ is defined as:

$$\mathcal{O} = \{\mathbf{p_d^w}, \mathbf{q_d^w}, \mathbf{p_t^w}, \mathbf{v_d^w}, \boldsymbol{\omega}_d^w, \mathbf{g_d^b}, \mathbf{o_d}\} \tag{4.2}$$

Most of the components of the observation space are already defined in the state space. However for the drone the position of the pillars present are unknown, two versions of the environment were designed. One version equipped the drone with a lidar and the other with a camera. In the lidar setting $\mathbf{o_d}$ represents the lidar output observation tensor, which has shape $(1, 36, 4)$ per environment and encodes distance measurements for 36 horizontal and 4 vertical directions. In the camera setting, $\mathbf{o_d} \in \mathbb{R}^{80\times80\times3}$ represents the RGB image captured from the quadcopter's onboard camera, with a resolution of 80×80 pixels and 3 color channels.

4.1.3 Action Space

The action space of the agent, represented by Figure 4.2,consists of a vector $\mathbf{u} = (u_0, u_1, u_2, u_3)$, where:

- u_0 represents the thrust, normalized between -1 and 1.
- u_1, u_2, and u_3 represent the moments applied around the roll, pitch, and yaw axes, also normalized between -1 and 1.

This action space choice was inspired by Kaufmann et al. (2022)

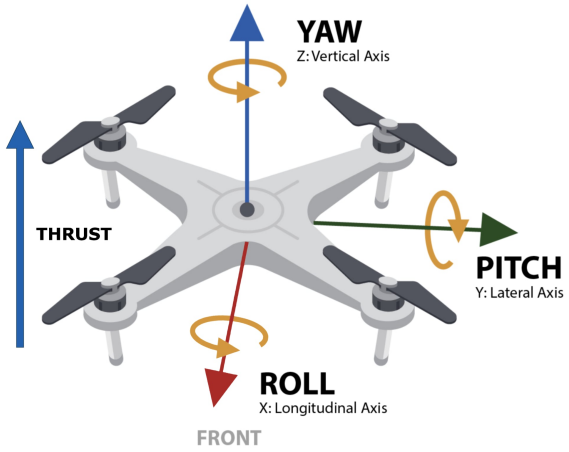

Figure 4.2: Representation of the action spaces

The thrust is scaled according to the drone's weight and a predefined thrust-to-weight ratio. The ratio for the Crazyflie model is approximately 1.9

$$\text{thrust}_i = \text{thrust\_to\_weight} \times \text{robot\_weight} \times (\frac{u_{i,0}+1}{2}) \tag{4.3}$$

This shift ensures that the thrust ranges from zero to the maximum achievable thrust. The moments are adjusted based on a moment scaling factor, moment_scale:

$$\text{moment}_i = \text{moment\_scale} \times (u_{i,1}, u_{i,2}, u_{i,3}) \tag{4.4}$$

4.1.4 Reward Function

The agent receives a reward according to a tailored reward function R. The reward r received by the drone is calculated as follows:

$$r = r_{\text{distance}} - r_{\text{lin\_vel}} - r_{\text{ang\_vel}} \tag{4.5}$$

The terms of the reward function include a distance-based component encouraging the drone to explore the environment, this term is computed as follow :

$$r_{\text{distance}} = 1 - \tanh(\frac{||p_d^w, p_t^w||}{c_1}) \tag{4.6}$$

where $||.||$ is the euclidean distance $\in \mathbb{R}^3$ between the attacker and the target, and where c_1 is a hyperparameter depending on the size of the environment. The terms $r_{\text{lin\_vel}}$ and $r_{\text{ang\_vel}}$ represent linear and angular velocity penalties, their values are quite small ranging from -0.01 to -1. They motivate the agent to learn a more human-like flight. Without them, drones start to develop very turbulent flight.

4.2 Methods

To train our agent, PPO was chosen. This algorithm already achieved great results in numerous tasks from video-games to more complex robotic systems.

Learning Architecture: The policy and value networks share a unified architecture with distinct heads. Input observations are structured as two modalities: the raycaster input is processed through a CNN composed of three layers: 32 filters with a 4×4 kernel and stride 2, followed by two 64-filter layers with 3×3 kernels and stride 2, each using ReLU activations. The resulting features are flattened and concatenated with the other modalities, the robot state before passing through a MLP with hidden layers of sizes 512, 256, and 128, activated using ELU. The policy network outputs the mean of a Gaussian distribution over actions with a learned, state-independent log standard deviation constrained between -20 and 2, initialized to 0.5 and clipped during training. The value network follows the same structure but uses a deterministic output. PPO training uses a KL-adaptive learning rate scheduler with a KL threshold of 0.016 and includes additional techniques such as advantage estimation with GAE ($\lambda = 0.95$), advantage normalization, and gradient norm clipping at 1.0. Main hyperparameters include a learning rate of 5.10^{-4}, PPO clipping ratio of 0.2, and entropy regularization with a scale of 0.0001, ensuring stable and robust policy updates across 24 rollout workers and 5 learning epochs per iteration.

4.3 Results

4.3.1 LiDAR Version

To assess learning performance, experiments were conducted using ten different random seeds. For each seed, 1000 agents were trained in parallel for 1 billion timesteps. The cumulative reward was tracked throughout the training process. Figure 4.3 presents the average episodic return across all agents.

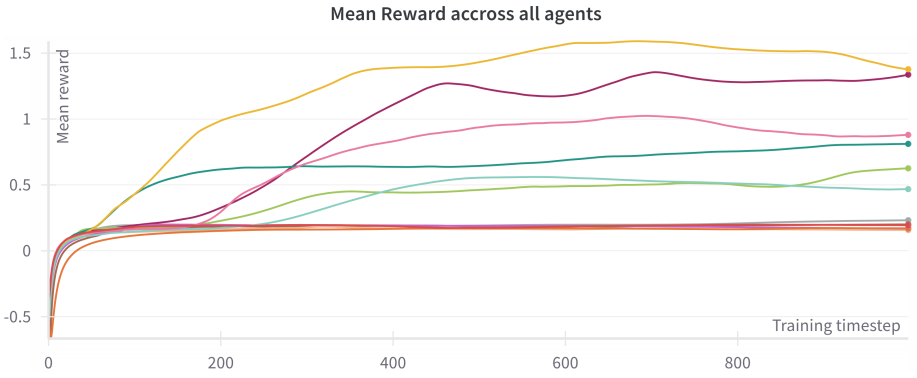

Figure 4.3: Episodic return over training iterations for the single-agent PPO policy.

As illustrated, certain seeds failed to explore the environment effectively, resulting in agents becoming stuck near the initial regions. Conversely, other seeds led to agents that demonstrated more efficient exploration, ultimately achieving higher returns and improved overall performance.

Reward Component Breakdown

To quantify the individual contribution of each reward component, their respective values were logged during training. As anticipated, the distance-based reward term exhibited the largest contribution among the three, as it primarily helps exploration within the environment.

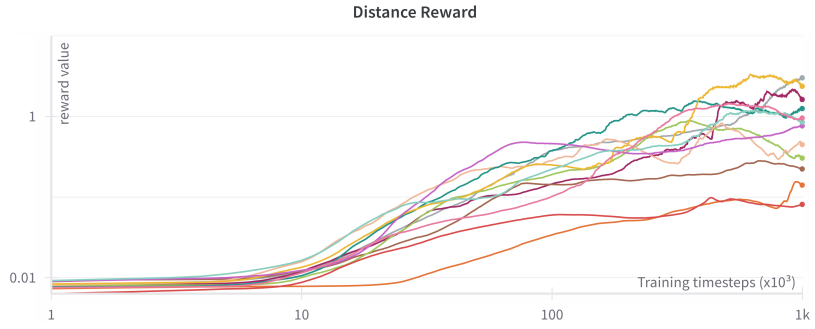

Figure 4.4: Contribution of the distance reward term to the overall return.

Although the linear and angular velocity penalty terms accounted for a smaller proportion of the mean reward, their presence significantly impacted agent behavior. Specifically, removing these penalties resulted in more aggressive flight patterns, accompanied by an increased frequency of collisions. Therefore, the $r_{\text{lin vel}}$ and $r_{\text{ang vel}}$ terms play a role in encouraging smoother and safer trajectories.

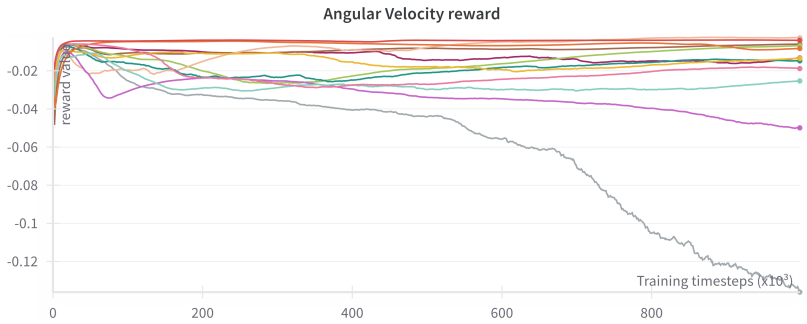

Figure 4.5: Contribution of the angular velocity penalty term.

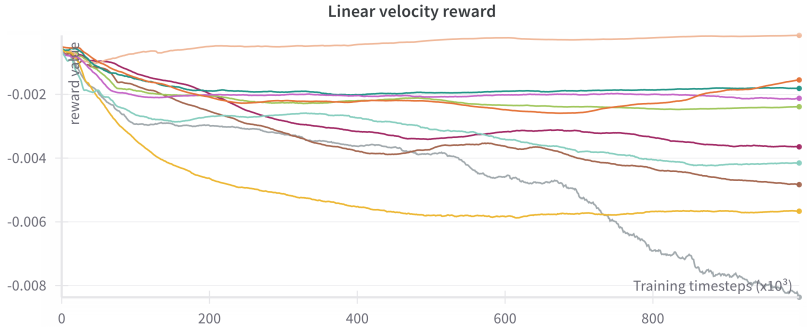

Figure 4.6: Contribution of the linear velocity penalty term.

4.3.2 Camera Version

A vision-based variant of the environment, relying on a camera-equipped agent, was also implemented. However, an issue introduced in version 2.1 of IsaacLab related the raytracing logic and lighting which led to poor camera input. As a result, the camera-based agent was unable to learn effectively under the altered visual conditions.

4.4 Conclusion

The central contribution presented in this chapter is the novel SARL environment built on Isaaclab, which features a navigation scenario where a drone attempts to reach a target in an unknown environment. The agent control a Crazyflie drone through thrust and torques. This work highlights the viability of RL-based approaches to solve complex task. Our results show that an agent, trained with PPO with minimal hyper-parameter tuning, is capable of learning how to navigate in complex environment.

This work is a first investigation of navigating in unknown environment, and we propose several future research directions. We suggest performing more testing on new environment to asses how well the agent is capable of generalizing to new terrain. Another limitation of our experiments is the absence of hyperparameter optimization. We use classical network architecture and default parameters values, we suspect that better settings could lead to better results and better behavior independently of the seed.

CHAPTER 5

MULTI AGENT REINFORCEMENT LEARNING

This final Chapter of the thesis is the paper about multi-agent reinforcement learning (MARL) scenarios. In this paper MARL approach are explored and compared in a new environment design specifically for it. This environment model a one versus one drone combat scenario where an attacker tries to reach a target while another drone, the defender tries to stop him by colliding with it. Some contents of the paper were already introduced in the thesis for clarity.

Autonomous Drone Combat: A Mutli-Agent Reinforcement Learning Approach

Hansen Julien
University of Liège
julien.hansen@student.uliege.be

Louette Arthur
University of Liège
Arthur.Louette@uliege.be

Leroy Pascal
University of Liège
pleroy@uliege.be

Ernst Damien
University of Liège
dernst@uliege.be

Abstract

Drones have become essential tool in various industries, from agriculture to surveillance and are now increasingly deployed on battlefields for detection, recognition, identification, and combat. While most systems remain controlled by human, the shift toward autonomy is intensifying, driven by breakthroughs in artificial intelligence, notably in reinforcement learning and scalable simulation techniques. This paper presents two contributions. A multi agent reinforcement learning environment for drone combat built on IsaacLab. An in-depth comparison between decentralized learning and self-play scheme in competitive settings. Our work confirmed the benefits of self-play methods for autonomous drone combat.

1 Introduction

Drones have become an important asset in various applications [1, 2]. In recent years, their numbers have drastically increased in combat zones, as they offer a low-cost solution to detect and identify strategic targets such as armed vehicles or critical infrastructures like power plants [3]. Additionally, they can easily be converted into lethal, low-cost weapons, representing a threat for these targets [4]. The conflict in Ukraine highlights the growing deployment of weaponized versions of commercial drones [5–7]. Their small size and speed impose constraints and complexity on designing countermeasures that require minimal resources and time. Multiple ways of dealing with these threats are currently considered [8, 9], a distinction in these countermeasures can be made between physical and electronic systems. Physical defenses include measures such as air defense systems, lasers, net guns, and interceptor drones [10]. Various combinations were explored, such as, integrating anti-aircraft guns with radar and laser systems [11], or deploying drones to pursue and neutralize other drones [12]. However, these approaches face key challenges: drones are highly agile, making them difficult to target, and the cost of defense systems, particularly missiles, often exceeds the value of the drones they aim to destroy [13]. In contrast, electronic defenses, exploits tactics such as radio jamming, eavesdropping and information injection [14]. In this work, a drone-based countermeasure is proposed to actively intercept and neutralize adversarial drones. Interceptor drones offer multiple advantages. They are cost-efficient, can be deploy from commercially available drones [7, 15], and do not rely on high-cost, military-grade weapons. Furthermore, the majority of their components can be fabricated through additive manufacturing techniques such as 3D printing [16], enabling fast assembly. Moreover their speed and rapid movements make them a highly adaptable countermeasure capable of operating on various terrains.

A well-established framework of artifical intelligence for control, called reinforcement learning (RL), based on the idea of agents learning to make sequences of decisions by interacting with an

environment, already shown great capabilities in robotics applications, like the ability to control first person view (FPV) drones better than the best pilots [17], or the ability to learn human actions like walking or running [18]. Multi-Agent RL (MARL) achieved human-level performance in difficult tasks such as competing with some of the best human players in StarCraft II [19], beating Chess Grandmaster [20]. It also demonstrates the capacity to solve complex drone control tasks such as flying in a formation pattern [21] or exploring an unknown environment through drone swarms [22]. There is no doubt that MARL could reshape the battlefield landscape by enabling autonomous systems to coordinate, adapt, and make complex decisions collectively. This could allow swarms of drones to respond to threats dynamically, and carry out missions with reduced human intervention. However, there is still a gap before deploying autonomous agents in a real battlefield [23]. These previous examples typically make strong hypotheses both on the drones and the environments that should be relaxed to be closer to real combat situation.

This paper analyzes and explores the inherent challenges of MARL, such as the of continual co-adaptation of multiple agents as they learn from interactions with one another named moving target problem [24, 25]. This non-stationarity induced by learning agents can lead to cyclic dynamics [26], whereby a learning agent adapts to the changing policies of other agents, which in turn adapt their policies to the learning agent's policy, and so on, creating possible infinite cycles in their strategies. Defining and achieving optimal behavior in multi-agent environments requires a shift from single-agent optimality concepts. Unlike single-agent settings where an optimal policy is defined with respect to a stationary environment, in MARL, an agent's policy is inherently linked to the policies of all other agents [27, 28]. This interdependence between agents requires additional theoretical frameworks, largely drawn from game theory. Solution concepts in multi-agents settings often revolve around achieving some form of equilibrium, such as the widely studied Nash equilibrium [29]. Other concepts like correlated equilibria or evolutionarily stable strategies also offer valuable frameworks for analyzing multi-agent learning outcomes [30, 31]. Furthermore, the dynamic and often uncertain nature of multi-agent interactions underscores the importance of developing robust policies (i.e. policies that can maintain performance despite variations in opponents' strategies)[32, 33].

To address the presented challenges, self-play methods are central in MARL by confronting an agent to its own weaknesses. Traditionally, this means an agent competes directly against copies of its current policy: in doing so, it discovers and exploits weaknesses in its play. There are several examples of games where human-level performance has been achieved with this particular scheme, such as Stratego, Go and chess [20, 34, 35]. Self-play also shown benefits for autonomous driving [36]. In our approach, an asymmetric self-play scheme is applied only the defender which periodically faces off against frozen, past checkpoints of the attacker during training.

Nowadays, high-fidelity physics simulators have a significant impact on RL research for environment creation and agent training. Training in a simulator is often divided into two steps: data collection and policy updates, where a policy refers to the agent's decision-making function that maps observations to actions. Nowadays policies are neural networks trained on GPUs, a high-fidelity simulator that also runs on GPU can significantly improve the speed of training. For this reason, the IsaacLab framework [37] and the IsaacSim simulator [38] were selected to design a new adversarial environment involving two agents. This new environment models a combat scenario where one drone attempts to reach a target, while the other tries to prevent it by colliding with it. Crazyflie drones [39] were selected for this task. Their open-source nature simplifies the acquisition of necessary data, such as their mass or thrust-to-weight ratio.

This paper is organized as follows. In Section 2, related works on autonomous drone control are reviewed. Section 3 covers the theoretical background. In Section 4, our approach to the problem is presented. Section 5 discusses the results, and finally, Section 6 concludes the paper and outlines potential future work.

2 Related Work

Game theory provides a principled framework for modeling strategic interactions among rational agents and for formally classifying different categories of games. In cooperative games, all agents optimize a shared reward function and pursue a common objective. In contrast, competitive or zero-sum games are characterized by strictly opposing interests, where the gain of one agent is exactly the loss of another. More generally, general-sum games represent settings in which the outcomes

of one player are not necessarily inversely related to those of others. Important contributions in Game theory, such as Nash's equilibrium [29], have laid the groundwork for analyzing the strategic outcomes and stability of policies learned by agents in competitive and cooperative environments. Other solution concepts exist, such as maxmin / minmax strategies [40], iterated elimination of dominated strategies [41], and correlated equilibria [42], that have further enriched our understanding of multi-agent dynamics. Game theory and RL have shaped the current field of MARL [26].

Most of the time, MARL methods fall between a fully centralized [43] or fully decentralized paradigms [44]. In the centralized paradigm, a joint policy is learned actions through global state information. Decentralized methods, on the other hand, train each agent independently based on local observations and individual rewards. While decentralized learning can handle general-sum games, it often suffers from instability due to the non-stationarity of the environment, even in simple settings [45]. An emerging compromise is the framework of centralized training and decentralized execution (CTDE) [25], in which, during training, agents exploit global information about the environment to mitigate the non stationary challenges caused by simultaneous learning, but at execution each agent selects actions based only on its own local observations. Recent works showed that CTDE methods consistently achieve state of the art results on standard multi agent reinforcement learning benchmarks [46]. There are two prominent classes of CTDE methods: actor critic and value decomposition approaches. In actor-critic approaches, each agent learns a decentralized policy using a centralized critic that has access to the joint observation history during training. This centralized critic provides better value estimates than a local critic [47], thereby improving policy learning. At execution time, the critic is no longer required; each agent acts independently based solely on its local observation history. In contrast, in value decomposition methods a centralized learner trains decentralized value functions by factorizing the joint action-value function into individual agent-specific value functions. These individual function enables decentralized execution [48].

CTDE strategies have proven highly effective in a range of cooperative drone applications requiring tight inter-agent coordination. In urban surveillance, for example, such frameworks enable drones swarms to collaboratively navigate and efficiently monitor target areas. Huang et al. [49] introduce a Multi-Agent Critic-Actor learning scheme, where a centralized critic maximize the discounted global rewards considering both safety and energy efficiency and an actor per drone to find decentralized policies to avoid collisions. In the field of mobile edge computing, MARL models trained with centralized knowledge have been applied to optimize drone trajectories and resource allocation. Park et al. [50] propose a method allowing drones to act as mobile base stations, to improve service quality in mobile access networks. Precision agriculture also benefits from these strategies, Sahu et al. [51] study a multi-agent approach to field coverage, where drones dynamically adapt to environmental changes to ensure complete monitoring while minimizing redundancy.

While competitive MARL scenarios for drone control have received less attention than cooperative settings, they present unique challenges, as agents must learn to counter adversaries with opposing objectives. To deal with these challenges, recent research combined hierarchical decompositions and self-play under a CTDE paradigm. Many approaches relies on hierarchical framework separating decision-making into macro- and micro-levels. Chai et al. [52] propose a two-tier architecture for air-to-air combat, where an outer strategic planner selects high-level combat objectives and an inner maneuver controller translates those objectives into precise flight commands. This framework relies on self-play during training. Building on this concept, Selmonaj et al. [53] design a hierarchical multi-agent system in which a high-level "commander" issues macro commands to subordinate agents. These agents, in turn, rely on self-play to refine group tactics and adapt to evolving strategies. More recently, Pang et al. [54] introduce a three-tier Leader-Follower Multi-Agent Proximal Policy Optimization scheme for drones combat. In their design, the top tier performs battlefield assessment, the middle tier determines optimal engagement angles, and the bottom tier issues flight commands.

Although most autonomous drone studies adhere to the CTDE paradigm, Recent success of Schroeder de Witt et al. [55] on the StarCraft Multi-Agent Challenge, demonstrated that decentralized agents trained with Independent proximal policy optimization (IPPO), relying only on local observation during training, can match or exceed the performance of centralized critics, which have access to global information, with surprisingly little hyperparameter tuning. Beyond StarCraft, recent work by Batra et al. [56] shows that fully decentralized RL can produce interesting swarm behaviors. They demonstrate advanced flocking behaviors, perform aggressive maneuvers in tight formations while avoiding collisions with each other, break and re-establish formations to avoid collisions with moving

obstacles, and efficiently coordinate in pursuit-evasion tasks. Our work follows this approach by investigating autonomous drone combat based on independent learning and self-play methods.

3 Theoretical Background

The stochastic game (SG), also called a Markov game [44], is at the foundation of MARL. In an SG, a set of agents interact with the environment by observing its state, choosing actions, and receiving rewards over a time sequence.

Definition: A stochastic game is defined by the tuple $(n, \mathcal{S}, \{\mathcal{U}_i\}_{i\in\mathcal{A}}, \mathcal{P}, \{R_i\}_{i\in\mathcal{A}}, \gamma, p, T)$, where $\mathcal{A} = \{1, \dots, n\}$ denotes the set of $n \geq 1$ agents, $\mathcal{S}$ denotes the state space, and $\mathcal{U}_i$ denotes the action space of agent i. Let $\mathcal{U} := \mathcal{U}_1 \times \cdots \times \mathcal{U}_n$, then $\mathcal{P} : \mathcal{S} \times \mathcal{U} \to \Delta(\mathcal{S})$ denotes the transition probability from any state $s \in \mathcal{S}$ to any state $s' \in \mathcal{S}$ for any joint action $\mathbf{u} \in \mathcal{U}$. The reward function $R_i : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to \mathbb{R}$ determines the immediate reward received by agent i. Finally, $\gamma \in [0, 1)$ is the discount factor, $p \in \Delta(\mathcal{S})$ is the initial state distribution, and $T \in \mathbb{N} \cup \{\infty\}$ is the time horizon.

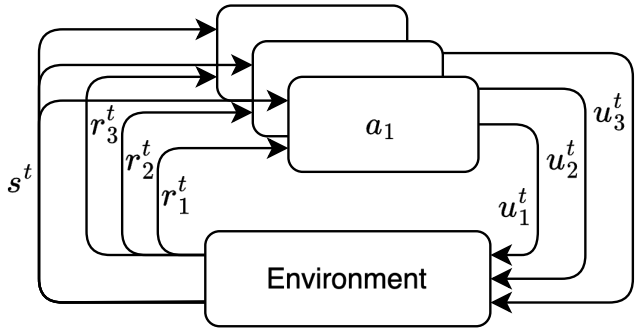

Figure 1: Interaction of three agents with the environment in a stochastic game [57]. Agents $i \in \{1, \dots, 3\}$ have access to the state s^t to select actions u_i^t. As a consequence, they each receive a reward r_i^t and the environment transitions into a new state s^{t+1}.

Sg are the standard formalism approach for fully observable environment, which is an assumptions made in this work. Within this framework, agent behavior is typically analyzed through game-theory principles. In particular, many solution concepts rely on the notion of best response, where a strategy is optimal for an agent given the fixed strategies of others, in terms of maximizing its expected cumulative reward. This notion is inherently static [58, 59], as it assumes the strategies of opponents are fixed. A Nash equilibrium is a stable point in which each agent's strategy is a best response to the strategies of the others. The existence of such a solution in any general-sum, non-repeated, normal-form game was first proven in the seminal work of Nash [29]. A formal way of defining a Nash equilibrium is through the value function noted V-function and the joint policy $\pi = (\pi_i, \pi_{-i})$ [26].

$$V_i^{\pi_i, \pi_{-i}}(s) = \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R_i(s^t, \mathbf{u}^t) \,\middle|\, s_0 \sim p(.),\, \mathbf{u}^t \sim \pi(\cdot \mid s^t),\, s^{t+1} \sim \mathcal{P}(\cdot \mid s^t, \mathbf{u}^t)\right], \tag{1}$$

where $V_i^{\pi_i, \pi_{-i}}(s)$ is the expected cumulative discounted reward for agent i, starting from state s, when it follows policy π_i and all other agents follow policies π_{-i} and where $\mathbf{u}^t = (u_1^t, ..., u_n^t)$ is the joint action at time t, drawn from the joint policy π. Following the V-function a Nash equilibrium of an SG is a joint policy $\boldsymbol{\pi}^* = (\pi_1^*, ..., \pi_n^*)$ such that:

$$V_i^{\pi_i^*, \pi_{-i}^*}(s) \geq V_i^{\pi_i, \pi_{-i}^*}(s), \quad \forall \pi_i \in \Pi \quad \forall s \in \mathcal{S} \quad \forall i \in \mathcal{A} \tag{2}$$

Intuitively, the Nash equilibrium represents a joint policy where no agent can improve its expected cumulative reward by unilaterally changing their own policy while others keep theirs fixed [29]. At equilibrium, each agent's strategy is the best response to the strategies of all other agents [60]. Even though Nash equilibria are guaranteed to exist [29], verifying their presence is hard in practice

for high-dimensional environments [61]. Learning algorithms in multi-agent settings, like policy gradient methods, especially in general-sum or partially observable environments, where agents have limited access to the full state of the environment, lack theoretical guarantees for convergence to Nash equilibria and may instead converge to suboptimal or unstable strategies [62]. This lack of guarantees has motivated the exploration of self-play paradigms in recent research [63].

Once training is complete, the performance of agents in a multi-agent setting can be assessed through several metrics. A direct and widely adopted approach, especially in single-agent settings, is to evaluate the expected cumulative reward, as formalized in the value function $V_i^{\pi_i,\pi_{-i}}(s)$ previously defined in Equation 1. This measures the expected return for agent i, assuming it follows policy π_i while the other agents follow π_{-i}. Reward-based evaluation are straightforward but often fails to capture strategic nuance in competitive settings. Indeed, high returns may simply reflect exploitation of weak opponents rather than genuinely robust play. Consequently, game-theory metrics are frequently required in adversarial domains. One such metric is the Elo system originally developed for chess [64], which assigns each player a score that reflects their relative skill level within a population. Let R_A and R_B be the Elo ratings of agents A and B, respectively. The estimated probability of winning for agent A against agent B, and vice versa, are given by:

$$E_A = \frac{1}{1 + 10^{(R_B - R_A)/400}}, \quad E_B = \frac{1}{1 + 10^{(R_A - R_B)/400}} \tag{3}$$

Note that $E_A + E_B = 1$. The constant 400 determines the steepness of the logistic function, for instance, if agent A has a rating 400 points higher than agent B, A s expected to win with a probability ten times greater than that of B. After each episode, the Elo ratings are updated according to the formula:

$$R_i' = R_i + K(S_i - E_i) \tag{4}$$

where R_i is the current rating of agent i, R_i' the updated rating, E_i the expected score, $S_i \in \{0, 0.5, 1\}$ the actual result (loss, draw, or win), and K a constant controlling the update magnitude.

Another commonly used performance metric is the win rate, which measures the proportion of matches an agent wins against others in the population. This metric offers valuable insights into an agent's relative strengths and weaknesses, particularly when evaluated against specific subgroups within the population. Unlike Elo ratings, which provide a general measure of performance, the win rate can highlight whether an agent performs especially well or poorly against certain subsets of opponents.

4 Experimental Setup

In this section, the components of the SG of our experiments are defined in addition to the training procedure.

State Space $\mathcal{S}$

The state space $\mathcal{S}$ consists of all possible states of the environment. In our drone combat scenario, the state space $\mathcal{S}$ is defined as:

$$\mathcal{S} = \{\mathbf{p}_\mathbf{d}^\mathbf{w}, \mathbf{q}_\mathbf{d}^\mathbf{w}, \mathbf{p}_\mathbf{a}^\mathbf{w}, \mathbf{q}_\mathbf{a}^\mathbf{w}, \mathbf{p}_\mathbf{t}^\mathbf{w}, \mathbf{v}_\mathbf{d}^\mathbf{w}, \boldsymbol{\omega}_d^w, \mathbf{v}_\mathbf{a}^\mathbf{w}, \boldsymbol{\omega}_a^w, \mathbf{g}_\mathbf{d}^\mathbf{b}, \mathbf{g}_\mathbf{a}^\mathbf{b}\} \in \mathbb{R}^{35} \tag{5}$$

In which $\mathbf{p}_\mathbf{d}^\mathbf{w}$, $\mathbf{p}_\mathbf{a}^\mathbf{w}$, $\mathbf{p}_\mathbf{t}^\mathbf{w}$ represent respectively the position vectors in $\mathbb{R}^3$ of the defender drone, the attacker and the target in the world frame. Vectors $\mathbf{q}_\mathbf{d}^\mathbf{w}$, $\mathbf{q}_\mathbf{a}^\mathbf{w}$ represent the quaternions in $\mathbb{R}^4$ encoding the orientation of the defender and attacker drones with respect to the world frame.The vectors $\mathbf{v}_\mathbf{d}^\mathbf{w}$, $\mathbf{v}_\mathbf{a}^\mathbf{w} \in \mathbb{R}^3$ and $\boldsymbol{\omega}_d^w$, $\boldsymbol{\omega}_a^w \in \mathbb{R}^3$ both represent the linear velocity and angular velocity vectors of each drone in their base frame. Finally, $\mathbf{g}_\mathbf{a}^\mathbf{b}$, $\mathbf{g}_\mathbf{d}^\mathbf{b} \in \mathbb{R}^3$ represents the projected gravity vector in the base frame of each drone, which provide orientation of each drone.

Joint Action Space

A joint action $\mathbf{u}$ is:

$$\mathbf{u} = (\mathbf{u_1}, \mathbf{u_2}) \in \mathcal{U}_1 \times \mathcal{U}_2 \tag{6}$$

The action space for each agent consists of a vector $\mathbf{u_i} = (u_{i,0}, u_{i,1}, u_{i,2}, u_{i,3})$, where:

- $u_{i,0}$ represents the thrust, normalized between -1 and 1.
- $u_{i,1}$, $u_{i,2}$, and $u_{i,3}$ represent the moments applied around the roll, pitch, and yaw axes, also normalized between -1 and 1.

The thrust is scaled according to the drone's weight and a predefined thrust-to-weight ratio. The ratio for the Crazyflie model is approximately 1.9 [39]. Each agent controls a drone through applied forces and torques see Figure 2.

$$\text{thrust}_i = \text{thrust\_to\_weight} \times \text{robot\_weight} \times (\frac{u_{i,0} + 1}{2}) \tag{7}$$

This shift ensures that the thrust ranges from zero to the maximum achievable thrust. The moments are adjusted based on a moment scaling factor, moment_scale:

$$\text{moment}_i = \text{moment\_scale} \times (u_{i,1}, u_{i,2}, u_{i,3}) \tag{8}$$

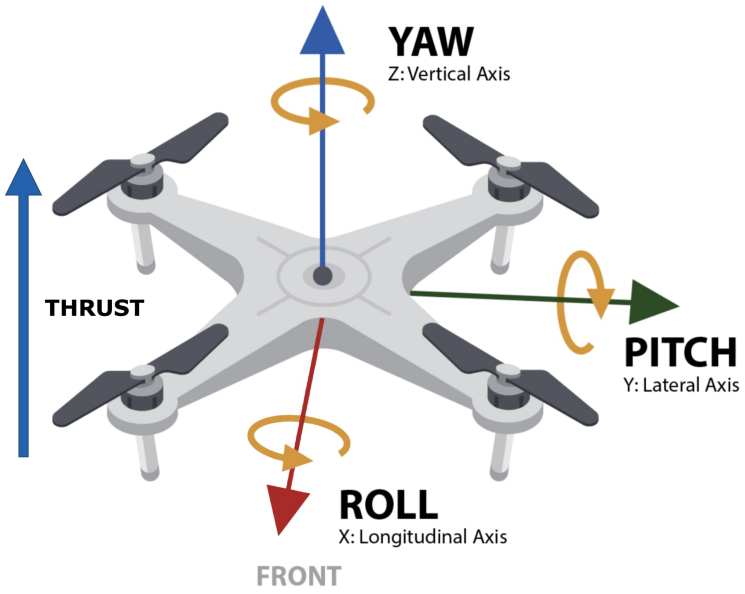

Figure 2: Representation of the action spaces

Reward Functions R_i

Each agent i receives a reward according to a tailored reward function R_i, implying that our environment is a general-sum game. The reward r_1 received by the defender drone is:

$$r_1 = r_{\text{distance}} - r_{\text{lin\_vel}} - r_{\text{ang\_vel}} + r_{\text{win}} - r_{\text{pen}} - r_{\text{alt\_pen}} \tag{9}$$

The terms of the reward function include a distance-based component encouraging the defender to get closer to the attacker, this term is computed as follow:

$$r_{\text{distance}} = 1 - \tanh(\frac{||p_d^w, p_a^w||}{c_1}) \tag{10}$$

where $||.||$ is the euclidean distance $\in \mathbb{R}^3$ between the attacker and the target, and where c_1 is a hyperparameter depending on the size of the environment.
$r_{\text{lin\_vel}}$ and $r_{\text{ang\_vel}}$ represent linear and angular velocity penalties, their values are quite small ranging from -0.01 to -1. These terms motivate the agent to learn a more human-like flight. Without these terms, drones start to develop very turbulent flight. r_{win} represents a large positive reward (+300) for successful interception of the attack drone. However, if the attacker reaches the target point, a penalty r_{pen} (-300) is assigned to the defender. Finally, an altitude penalty $r_{\text{alt\_pen}}$ (-10) is added to simulate the crash of a drone. When it happens, the other drone gains a positive reward as it can be considered a win.

Similarly, the reward r_2 received by the attacker drone according to the reward function R_2 is calculated as:

$$r_2 = r_{\text{distance}} - r_{\text{lin\_vel}} - r_{\text{ang\_vel}} + r_{\text{win}} - r_{\text{penalty}} - r_{\text{alt\_pen}} \tag{11}$$

For the attacker drone, the distance component encourages proximity to the goal. This term is computed as :

$$r_{\text{distance}} = 1 - \tanh\left(\frac{||p_d^w, p_a^w||}{c_2}\right) \tag{12}$$

where c_2 is another hyperparameter depending on the distance to the target.

5 Methods

In this paper, we compare two approaches: Independent proximal policy optimization (IPPO) and an modified version of IPPO, Self-play based IPPO (S-IPPO).

IPPO

IPPO was implemented via the SKRL library [65] following the algorithm details presented by Schroeder et al. [66]. IPPO is an extension of PPO to multi-agent settings. PPO is an algorithm derived from Trust Region Policy Optimization (TRPO) [67], which is a class of policy-gradient methods that restricts the update of a policy to within the trust region of the behavior policy by enforcing a KL divergence constraint on the policy update at each iteration. Initially, TRPO optimizes the following:

$$\max_{\theta} \quad \mathbb{E}_{s_t,u_t}\left[\frac{\pi_\theta(u_t \mid s_t)}{\pi_{\theta_{\text{old}}}(u_t \mid s_t)}\hat{A}(s_t,u_t)\right], \qquad \text{subject to } \mathbb{E}_{s_t,u_t}\left[\text{KL}\left(\pi_{\theta_{\text{old}}}, \pi_\theta\right)\right] \leq \delta \tag{13}$$

where θ_{old} are the parameters of the policy before the update and $\hat{A}(s_t, u_t)$ is an approximation of the advantage function. This formulation is computationally expensive due to the computation of multiple Hessian-vector products when approximating the KL constraint. To solve this, PPO[68] approximates the trust region constraints by policy ratio clippings, i.e. the policy loss becomes:

$$\mathcal{L}(\theta) = \mathbb{E}_{s_t,u_t}\left[\min\left(\frac{\pi_\theta(u_t \mid s_t)}{\pi_{\theta_{\text{old}}}(u_t \mid s_t)}\hat{A}(s_t,u_t), \text{clip}\left(\frac{\pi_\theta(u_t \mid s_t)}{\pi_{\theta_{\text{old}}}(u_t \mid s_t)}, 1-\epsilon, 1+\epsilon\right)\hat{A}(s_t,u_t)\right)\right] \tag{14}$$

In IPPO every agents learn a decentralized policies π^i with individual policy clipping, where each agent's independent policy updates are clipped based on the objective defined in Equation 17. A variant of the advantage function, where each agent i learns a local observation based critic $V_\phi(o_t^i)$ parameterised by ϕ is considered, using *Generalized Advantage Estimation* [69]. The network parameters θ, ϕ are not shared across critics and actors in this implementation. An entropy regularization term is also added to the final policy loss [70]. For each agent i, the advantage estimation is computed as:

$$\hat{A}_t^i = \sum_{l=0}^{\infty}(\gamma\lambda)^l\delta_{t+l}^i, \quad \text{where} \quad \delta_t^i = r_t^i + \gamma V_{\phi_{i,\text{old}}}(s_{t+1}^i) - V_{\phi_{i,\text{old}}}(s_t^i) \tag{15}$$

where δ_t^i is the temporal difference at time step t. The team reward $r_t(s_t, i_t)$ approximates $r_t(o_t^i, u_t^i)$. Following that, the final policy loss for each agent i becomes:

$$\mathcal{L}^i(\theta) = \mathbb{E}_{o_t^i, u_t^i}\left[\min\left(\frac{\pi_\theta(u_t \mid o_t^i)}{\pi_{\theta_{\text{old}}}(u_t \mid o_t^i)}\hat{A}_t^i, \text{clip}\left(\frac{\pi_\theta(u_t \mid o_t^i)}{\pi_{\theta_{\text{old}}}(u_t \mid o_t^i)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_t^i\right)\right] \tag{16}$$

Value Clipping is applied to restrict the update of the critic function as proposed by [69]:

$$\mathcal{L}^i(\theta) = \mathbb{E}_{o^i}\left[\min\left\{(V_\phi(o_t^i) - \hat{V_t^i})^2, (V_{\phi_{old}}(o_t^i)) + \text{clip}(V_\phi(o_t^i) - V_{\phi_{old}}(o_t^i), -\epsilon, +\epsilon) - \hat{V_t^i})^2\right\}\right] \tag{17}$$

where ϕ_{old} are parameters before the update and $\hat{V}_t^i = A_t^i + V_\phi(o_t^i)$. This restriction of the value function to within the trust region helps avoiding overfitting to the most recent batch of data. For each agent the overall learning loss becomes:

$$\mathcal{L}(\phi, \theta) = \sum_{i=1}^{n} \mathcal{L}^i(\theta) + \lambda_{critic}\mathcal{L}^i(\phi) + \lambda_{entropy}\mathcal{H}(\pi^i) \tag{18}$$

where $\mathcal{H}(\pi^i)$ is the entropy of policy π^i and λ_{critic} and $\lambda_{entropy}$ are hyperparameter set to 1.0 and 0.001 respectively.

Learning Architecture: We use orthogonal initialization with a gain of $\sqrt{2}$ to initialize the parameters of both the policy and value networks, a strategy known to perform well with ELU activations. The input of each network consists of a flatten state vector, which is passed through two fully connected layers with 256 and 128 hidden units respectively, each followed by ELU activations [71]. The policy network outputs the mean of a Gaussian distribution over actions, with a state-independent log standard deviation parameter. Advantage normalization is applied once before training by subtracting the mean and dividing by the standard deviation over the full rollout buffer. A KL-adaptive learning rate scheduler with a threshold of 0.008 is used. The PPO-specific hyperparameters include a clipping ratio of 0.2 and discount factor λ= 0.99.

S-IPPO

A new Self-play variant of IPPO was also implemented, this new approach aims to improve the defender robustness by modifying the training opponent selection process. The core modification involves maintaining a fixed population of M distinct and frozen attacker policies.

Training is divided in chunks. During training, the defender policy $\pi_{\theta_{\text{def}}}$ primarily collects experience by interacting with the current version of the opponent policy. However, with some probability p, at the end of a chunk, the defender instead interacts with a past version of the attacker policy sampled randomly from previously saved opponent checkpoints. The defender's policy $\pi_{\theta_{\text{def}}}$ is then updated following the standard IPPO algorithm detailed previously (Equations 15-18), exploiting the experience gathered from these interactions. The idea behind this training scheme is to help the defender develop more robust and generalized behaviors.

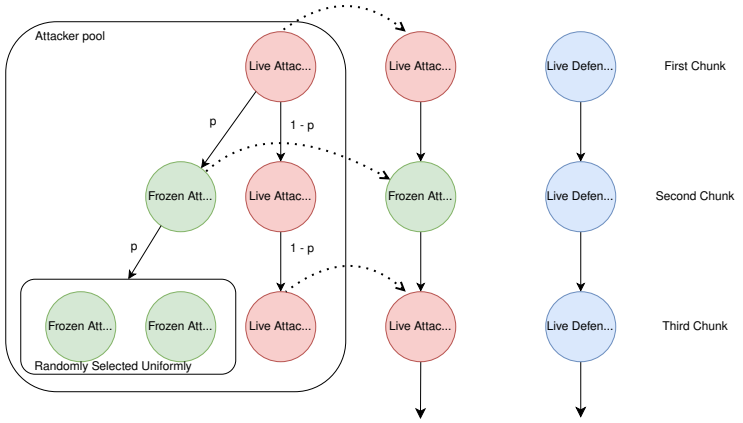

Figure 3: Overview of S-IPPO.

To test both methods we trained agents during 1 billion timesteps with 1000 environments running in parallel, checkpoints for both the attackers and the defenders were taken every chunks, the probability p of switching from the live version of the attacker to a past frozen version of it is 0,2.

6 Results

As stated in Section 5, during training, a checkpoint of each agent is saved every chunk, each chunk is composed of 10 million timesteps, resulting in a population of 100 distinct versions of attackers and defenders. To evaluate their respective Elo scores, each attacker–defender pair was matched 5 times, yielding a total of 50,000 simulated combats. The matches are randomly ordered to ensure statistical fairness and to avoid bias introduced by the ordering of evaluations. To make sure the elo score converge each pair play 100 episodes against each other. This methodology was performed eight times and the results were averaged across these runs.

Figure 4 and 5 presents the Elo score evolution of both teams trained with IPPO. The attackers achieves higher Elo ratings than the defender. This discrepancy may be partially attributed to the unfairness of the tasks: the attacker's target remains fixed, allowing it to better optimize its strategy, while the defender must adapt to the evolving and increasingly competent behavior of the attacker. The defender's Elo slighty decrease at first then remains relatively constant over time, suggesting that earlier defender might learn more general strategies than later one. The constant elo observed after suggests that more trained defender may forget early attacker and are only strong at each chunk against a subset of attacker. However elo scores obtained from S-IPPO are slightly higher, regarding the defender. The final defender saved achieves a mean elo score of $\tilde{1}450$ compared to $\tilde{1}350$ for IPPO, suggesting that more trained defender don't forget early attacker.

Figure 4: IPPO Elo Score

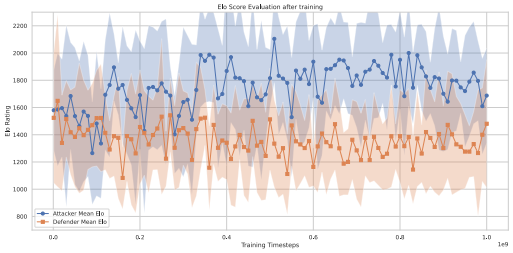

Figure 5: S-IPPO Elo Score

Our suggestion is further reinforced by the win rate analysis presented in Figures 6 and 7, which display the average win rates across all runs between ten defender and every attackers, these heatmaps were clipped between 0,25 and 0,45 to better visualize the average trend present in each methods. Under the IPPO framework, each defender exhibits marginally higher win rates against attacker

policies generated around the same stage of training. This trend indicates a form of temporal overfitting, where defenders adapt primarily to the strategies of their current attackers, but fail to generalize effectively to opponents trained significantly earlier or later. Notably, a marked decline in defender performance becomes evident beyond approximately 600 million timesteps, suggesting a degradation in the ability to counter earlier attack strategies over time.

In contrast, the S-IPPO approach demonstrates higher win rates against earlier attacker checkpoints. This outcome highlights the benefits of training defenders against a diverse set of opponent policies drawn from different stages of training, thereby improving robustness and mitigating the forgetting effect.

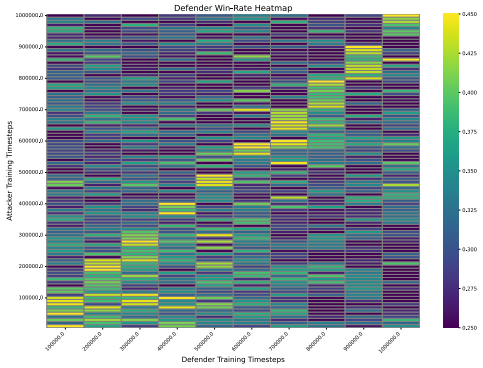

Figure 6: Win rate of IPPO.

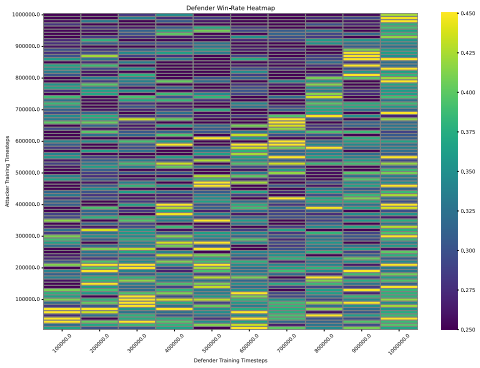

Figure 7: Win rate of S-IPPO.

Regarding the variance in performance, S-IPPO displays a higher variance in win rates compared to IPPO. This observation suggests that the outcome of training under S-IPPO is more sensitive to the rate and diversity of opponent exposure during training, potentially due to the broader behavioral spectrum introduced by replaying past attacker checkpoints. This could also come from inherent stochasticity of RL. Two runs can output really different results leading to an high variance.

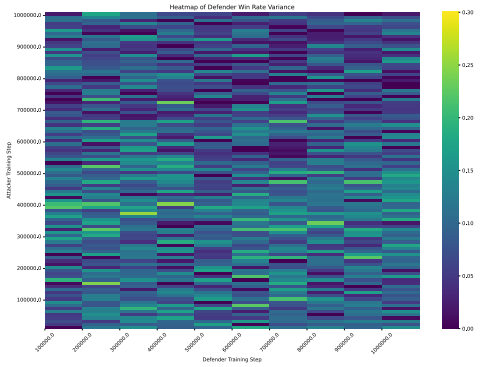

Figure 8: Variance heatmap for IPPO defenders against attacker checkpoints.

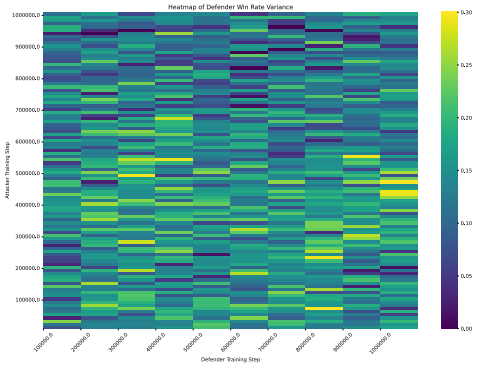

Figure 9: Variance heatmap for S-IPPO defenders against attacker checkpoints.

7 Conclusion and future works

In this paper, we develop and evaluate multi-agent reinforcement learning approaches for autonomous drone combat. We introduce a competitive MARL environment built on IsaacLab, featuring a combat scenario where an attacker drone attempts to reach a target while a defender drone aims to intercept it. Agents control Crazyflie drones through thrust and torques. Our primary investigation compare a fully decentralized learning paradigm, IPPO, against an asymmetric self-play scheme, S-IPPO, in which the defender trains against a population of attacker policies. This approach specifically addresses the moving target problem inherent in competitive multi-agent settings. We evaluate the performance

of these approaches at the end of training using the Elo rating system, and we analyze the win rates of several matchups during training to support the results provided by the Elo scores. Our results highlight the benefits of training a defender against past version of the attacker, demonstrating that S-IPPO improves the development of more robust defenders.

This work constitutes a first investigation of two-team drone competitive tasks using fully decentralized methods, and we propose several future research directions. First, we suggest performing the same experiments in more complex environments with more than two agents and under partially observable settings. Given their impressive results in cooperative tasks, a comparison between CTDE methods and fully decentralized approaches may yield interesting insights. Another research direction consists in comparing a symmetric self-play scheme to S-IPPO. We also recommend conducting behavioral and policy analyses to better understand why some teams achieve higher Elo scores and how strategy diversity emerges within a single training population.

References

[1] Gonzalo Pajares. "Unmanned Aerial Systems for Civil Applications: A Review". In *Drones* 1.1 (2017), p. 2. DOI: 10.3390/drones1010002. URL: https://www.mdpi.com/2504-446X/1/1/2.

[2] Bangkui Fan et al. "Review on the Technological Development and Application of UAV Systems". In *Chinese Journal of Electronics* 29.2 (2020), pp. 199–207. DOI: 10.1049/cje.2019.12.006. URL: https://cje.ejournal.org.cn/article/doi/10.1049/cje.2019.12.006.

[3] DSIAC. "Unmanned Aerial Systems for Intelligence, Surveillance, Reconnaissance". https://dsiac.dtic.mil/state-of-the-art-reports/unmanned-aerial-systems-for-intelligence-surveillance-reconnaissance/. Defense Systems Information Analysis Center (DSIAC). 2018.

[4] Vinay Chamola et al. "A Comprehensive Review of Unmanned Aerial Vehicle Attacks and Neutralization Techniques". In *Ad Hoc Networks* 111 (2021), p. 102324. DOI: 10.1016/j.adhoc.2020.102324. URL: https://www.sciencedirect.com/science/article/pii/S1570870520306788.

[5] Zachary Kallenborn. "Will the Drone War Come Home? Ukraine and the Weaponization of Commercial Drones". https://mwi.westpoint.edu/will-the-drone-war-come-home-ukraine-and-the-weaponization-of-commercial-drones/. Modern War Institute at West Point. 2022.

[6] Zachary Kallenborn. "Ukraine is the First "Hackers' War"". In *IEEE Spectrum* (2023). URL: https://spectrum.ieee.org/ukraine-hackers-war.

[7] Abdullah D. Al-Garni. "Drones in the Ukrainian War: Will They Be an Effective Weapon in Future Wars?" Unpublished manuscript. 2022.

[8] Mohamed Sharaf et al. "Protect Your Sky: A Survey of Counter Unmanned Aerial Vehicle Systems". In *IEEE Communications Surveys Tutorials* 22.4 (2020), pp. 2837–2884. DOI: 10.1109/COMST.2020.3016902. URL: https://ieeexplore.ieee.org/document/9194729.

[9] Vitalii Tyurin et al. "General Approach to Counter Unmanned Aerial Vehicles". In *2019 IEEE 5th International Conference Actual Problems of Unmanned Aerial Vehicles Developments (APUAVD)*. Kiev, Ukraine: IEEE, Oct. 2019. DOI: 10.1109/APUAVD47061.2019.8943859. URL: https://ieeexplore.ieee.org/abstract/document/8943859.

[10] T. R. Oliver Parken. "Chinese soldiers train to fend off fpv drones". *The Warzone*. 2024.

[11] D. Kunertova. "The war in Ukraine shows the game-changing effect of drones depends on the game". In *Bulletin of the Atomic Scientists* (2023).

[12] G. Karadeniz et al. "Drone Wars 3D: A game-based simulation platform for testing aerial defence strategies against drone swarms". In *Journal of Aeronautics and Space Technologies* (2024).

[13] D. L. D. Silva et al. “A Soft-Kill Reinforcement Learning Counter Unmanned Aerial System (C-UAS) with Accelerated Training”. In *IEEE Access* (2023).

[14] D. He et al. “Communication security of unmanned aerial vehicles”. In *IEEE Wireless Communications* (2017).

[15] Modovolo. “The Ukrainian Drone Order of Battle: Lessons for Commercial Drone Operators”. https://modovolo.com/2025/04/11/the-ukrainian-drone-order-of-battle-lessons-for-commercial-drone-operators/. 2025.

[16] N. Muralidharan et al. “Structural analysis of mini drone developed using 3D printing technique”. In *Heliyon* 7.12 (2021), e08519. DOI: 10.1016/j.heliyon.2021.e08519. URL: https://www.sciencedirect.com/science/article/pii/S2214785321029072.

[17] Y. Song et al. “Autonomous drone racing with deep reinforcement learning”. arXiv preprint arXiv:2103.08624. https://arxiv.org/abs/2103.08624. 2021.

[18] Seungmoon Song et al. “Deep reinforcement learning for modeling human locomotion control in neuromechanical simulation”. In *Journal of NeuroEngineering and Rehabilitation* 18.1 (2021).

[19] O. Vinyals et al. “StarCraft II: A new challenge for reinforcement learning”. arXiv preprint arXiv:1708.04782. https://arxiv.org/abs/1708.04782. 2017.

[20] D. Silver et al. “Mastering chess and shogi by self-play with a general reinforcement learning algorithm”. arXiv preprint arXiv:1712.01815. https://arxiv.org/abs/1712.01815. 2017.

[21] Y. Xie et al. “Multi-UAV formation control with static and dynamic obstacle avoidance via reinforcement learning”. arXiv preprint arXiv:2410.18495. https://arxiv.org/abs/2410.18495. 2024.

[22] G. Liao et al. “Multi-UAV escape target search: A multi-agent reinforcement learning method”. In *Sensors* 24.21 (2024), p. 6859. DOI: 10.3390/s24216859. URL: https://www.mdpi.com/1424-8220/24/21/6859.

[23] Arthur Louette et al. “Existing Gaps in Reinforcement Learning for Drone Warfare”. Technical Report. Accessed: 2025-06-03. University of Liège, 2025. URL: https://orbi.uliege.be/bitstream/2268/331713/1/Existing_Gaps_In_Reinforcement_Learning_For_Drone_Warfare.pdf.

[24] Ming Tan. “Multi-agent reinforcement learning: Independent vs. cooperative agents”. In *Proceedings of the tenth international conference on machine learning*. Morgan Kaufmann. 1993, pp. 330–337.

[25] Ryan Lowe et al. “Multi-agent actor-critic for mixed cooperative-competitive environments”. In *Advances in neural information processing systems (NIPS)*. Vol. 30. 2017, pp. 6379–6390. URL: https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html.

[26] Stefano V. Albrecht et al. “Multi-Agent Reinforcement Learning: Foundations and Modern Approaches”. MIT Press, 2024. ISBN: 9780262049375. URL: https://www.marl-book.com/download/marl-book.pdf.

[27] Kaiqing Zhang et al. “Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms”. In *Handbook of Reinforcement Learning and Control*. Ed. by Kyriakos G. Vamvoudakis et al. Springer, 2021, pp. 321–384. DOI: 10.1007/978-3-030-60990-0_12. URL: https://arxiv.org/abs/1911.10635.

[28] Lucian Buşoniu et al. “A Comprehensive Survey of Multi-Agent Reinforcement Learning”. In *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2 (2008), pp. 156–172. DOI: 10.1109/TSMCC.2007.913919. URL: https://www.dcsc.tudelft.nl/~bdeschutter/pub/rep/07_019.pdf.

[29] John F. Nash Jr. “Equilibrium points in n-person games”. In *Proceedings of the National Academy of Sciences of the United States of America* 36.1 (1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48. URL: https://www.pnas.org/doi/10.1073/pnas.36.1.48.

[30] Yoav Shoham and Kevin Leyton-Brown. "Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations". Cambridge University Press, 2008. ISBN: 9780521899437. URL: https://www.cambridge.org/core/books/multiagent-systems/B11B69E0CB9032D6EC0A254F59922360.

[31] Martin A. Nowak. "Evolutionary Dynamics: Exploring the Equations of Life". Harvard University Press, 2006. ISBN: 9780674023383. URL: https://www.hup.harvard.edu/books/9780674023383.

[32] Lerrel Pinto et al. "Robust Adversarial Reinforcement Learning". In *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 2817–2826. URL: https://proceedings.mlr.press/v70/pinto17a.html.

[33] Kaiqing Zhang et al. "Robust Multi-Agent Reinforcement Learning with Model Uncertainty". In *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 1–12. URL: https://proceedings.neurips.cc/paper/2020/hash/774412967f19ea61d448977ad9749078-Abstract.html.

[34] Julien Perolat et al. "Mastering the Game of Stratego with Model-Free Multiagent Reinforcement Learning". arXiv preprint arXiv:2206.15378. 2022. URL: https://arxiv.org/abs/2206.15378.

[35] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In *Nature* 529 (2016), pp. 484–489. DOI: 10.1038/nature16961. URL: https://doi.org/10.1038/nature16961.

[36] Daphne Cornelisse and Eugene Vinitsky. "Human-compatible driving partners through data-regularized self-play reinforcement learning". arXiv preprint arXiv:2403.19648. 2024. URL: https://arxiv.org/abs/2403.19648.

[37] Mayank Mittal et al. "Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments". arXiv preprint arXiv:2301.04195. 2023. URL: https://arxiv.org/abs/2301.04195.

[38] NVIDIA. "Isaac Sim: Robotics Simulation and Synthetic Data Generation". https://developer.nvidia.com/isaac/sim. Accessed: 2025-05-13. 2023.

[39] Bitcraze AB. "Crazyflie 2.1 Open Source Quadcopter Drone". https://www.bitcraze.io/products/crazyflie-2-1-plus/. Accessed: 2025-05-13. 2023.

[40] John von Neumann. "Zur Theorie der Gesellschaftsspiele". In *Mathematische Annalen* 100.1 (1928), pp. 295–320. DOI: 10.1007/BF01448847.

[41] Drew Fudenberg and Jean Tirole. "Game Theory". MIT Press, 1991. ISBN: 9780262061414.

[42] Robert J. Aumann. "Subjectivity and Correlation in Randomized Strategies". In *Journal of Mathematical Economics* 1.1 (1974), pp. 67–96. DOI: 10.1016/0304-4068(74)90037-8.

[43] Caroline Claus and Craig Boutilier. "The Dynamics of Reinforcement Learning in Cooperative Multiagent Systems". In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*. AAAI Press, 1998, pp. 746–752. URL: https://aaai.org/papers/00746-AAAI98-106-the-dynamics-of-reinforcement-learning-in-cooperative-multiagent-systems/.

[44] Michael L. Littman. "Markov Games as a Framework for Multi-Agent Reinforcement Learning". In *Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann, 1994, pp. 157–163. DOI: 10.1016/B978-1-55860-335-6.50027-1. URL: https://www.sciencedirect.com/science/article/pii/B9781558603356500271.

[45] Jakob N. Foerster et al. "Learning with Opponent-Learning Awareness". In *arXiv preprint arXiv:1709.04326* (2017). URL: https://arxiv.org/abs/1709.04326.

[46] Jakob N Foerster et al. "Counterfactual Multi-Agent Policy Gradients". In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)* 31 (2018), pp. 2859–2867. URL: https://arxiv.org/abs/1705.08926.

[47] Xueguang Lyu et al. “On Centralized Critics in Multi-Agent Reinforcement Learning”. In *Journal of Artificial Intelligence Research* 77 (2023), pp. 295–354. DOI: 10.1613/jair.1.14386. URL: https://www.jair.org/index.php/jair/article/view/14386.

[48] Tushar Rashid et al. “QMIX: Monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning”. In *Proceedings of the 35th International Conference on Machine Learning (ICML)* 80 (2018), pp. 4295–4304. URL: https://arxiv.org/abs/1803.11485.

[49] Shuangyao Huang et al. “Multi-UAV Collision Avoidance using Multi-Agent Reinforcement Learning with Counterfactual Credit Assignment”. In *arXiv preprint arXiv:2204.08594* (2022). URL: https://arxiv.org/abs/2204.08594.

[50] Chanyoung Park et al. “Cooperative Multi-Agent Deep Reinforcement Learning for Reliable and Energy-Efficient Mobile Access via Multi-UAV Control”. In *arXiv preprint arXiv:2210.00945* (2022). URL: https://arxiv.org/abs/2210.00945.

[51] Ranjita Sahu et al. “An Analysis of the Robustness of UAV Agriculture Field Coverage Using Multi-Agent Reinforcement Learning”. In *Innovations in Systems and Software Engineering* 19.1 (2023), pp. 1–12. DOI: 10.1007/s41870-023-01264-0. URL: https://link.springer.com/article/10.1007/s41870-023-01264-0.

[52] Jiajun Chai et al. “A Hierarchical Deep Reinforcement Learning Framework for 6-DOF UCAV Air-to-Air Combat”. In *arXiv preprint arXiv:2212.03830* (2022). URL: https://arxiv.org/abs/2212.03830.

[53] Ardian Selmonaj et al. “Hierarchical Multi-Agent Reinforcement Learning for Air Combat Maneuvering”. In *arXiv preprint arXiv:2309.11247* (2023). URL: https://arxiv.org/abs/2309.11247.

[54] Jinhui Pang et al. “A Hierarchical Reinforcement Learning Framework for Multi-UAV Combat Using Leader-Follower Strategy”. In *arXiv preprint arXiv:2501.13132* (2025). URL: https://arxiv.org/abs/2501.13132.

[55] Christian Schroeder de Witt et al. “Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?” In *arXiv preprint arXiv:2011.09533* (2020). DOI: 10.48550/arXiv.2011.09533. URL: https://arxiv.org/abs/2011.09533.

[56] Sumeet Batra et al. “Decentralized Control of Quadrotor Swarms with End-to-end Deep Reinforcement Learning”. In *5th Conference on Robot Learning, CoRL 2021, 8-11 November 2021, London, England, UK*. Proceedings of Machine Learning Research. PMLR, 2021. URL: https://arxiv.org/abs/2109.07735.

[57] Leroy Pascal. “Contributions to Multi-Agent Reinforcement Learning”. PhD thesis. Liege, Belgium: University of Liege, July 2024.

[58] Michael Weinberg and Jeffrey S. Rosenschein. “Best-response multiagent learning in non-stationary environments”. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*. IEEE Computer Society. 2004, pp. 506–513.

[59] Pablo Hernandez-Leal et al. “A survey of learning in multiagent environments: Dealing with non-stationarity”. In *Journal of Artificial Intelligence Research* 65 (2017), pp. 1–50.

[60] Martin J. Osborne and Ariel Rubinstein. “A Course in Game Theory”. Cambridge, MA: MIT Press, 1994. ISBN: 0-262-56021-9.

[61] Constantinos Daskalakis et al. “The Complexity of Computing a Nash Equilibrium”. In *Journal of the ACM* 56.3 (2009), Art. 14, 57 pgs. DOI: 10.1145/1536414.1536420.

[62] Eric Mazumdar et al. “Policy-Gradient Algorithms Have No Guarantees of Convergence in Linear Quadratic Games”. In *arXiv preprint arXiv:1907.03712* (2019). URL: https://arxiv.org/abs/1907.03712.

[63] Zelai Xu et al. “Learning Global Nash Equilibrium in Team Competitive Games with Generalized Fictitious Cross-Play”. In *Journal of Machine Learning Research* 26.44 (2025), pp. 1–30. URL: https://jmlr.org/papers/v26/24-1503.html.

[64] Arpad E. Elo. “The Rating of Chessplayers, Past and Present”. New York: Arco Publishing, 1978. ISBN: 0668047216. URL: https://archive.org/details/ratingofchesspla00unse.

[65] Antonio Serrano-Muñoz et al. "skrl: Modular and Flexible Library for Reinforcement Learning". In *Journal of Machine Learning Research* 24.254 (2023), pp. 1–9. URL: http://jmlr.org/papers/v24/23-0112.html.

[66] Christian Schroeder de Witt et al. "Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?" In *arXiv preprint arXiv:2011.09533* (2020). URL: https://arxiv.org/abs/2011.09533.

[67] John Schulman et al. "Trust Region Policy Optimization". In *Proceedings of the 32nd International Conference on Machine Learning*. PMLR, 2015, pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html.

[68] John Schulman et al. "Proximal Policy Optimization Algorithms". In *arXiv preprint arXiv:1707.06347* (2017). URL: https://arxiv.org/abs/1707.06347.

[69] John Schulman et al. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*. 2016. URL: https://arxiv.org/abs/1506.02438.

[70] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In *Proceedings of the 33rd International Conference on Machine Learning*. PMLR, 2016, pp. 1928–1937. URL: https://proceedings.mlr.press/v48/mniha16.html.

[71] Djork-Arné Clevert et al. "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)". In *Proceedings of the 4th International Conference on Learning Representations (ICLR)*. 2016.

CHAPTER 6

CONCLUSION

This work tackled two problems, the navigation in unknown terrain and the viability of autonomous drone combat. The performance of each agents were evaluated and analyzed. Finally, we compared a fully decentralized approach to a new self-play variant we introduced in the multi-agent scenario designed.

6.1 Contributions

The central contributions of this work are the environment designed both in single-agent and multi-agent settings as well as the new novel self-play variant of IPPO, which mitigate the moving target problem inherent to the field of MARL. This thesis also investigate the viability of RL for autonomous drone control in both single-agent and multi-agent settings, focusing on complex tasks such as navigation in partially observable environments and adversarial aerial combat scenarios. High-fidelity simulation tools such as IsaacSim and the IsaacLab framework were selected to design a set of custom environments and for the training of the agents. The studied control problem was approached through policy gradient methods like: Proximal Policy Optimization, its multi-agent variants, IPPO, and our novel variant S-IPPO.

In the single-agent context, we demonstrate that RL-based policies can effectively guide a drone equipped with minimal sensing (LiDAR or vision) through unknown terrains. The learned policies successfully explore the terrain while avoiding.

In the multi-agent setting, a novel competitive environment was designed to simulate one-on-one drone combat. Our evaluation compare the performances of agents trained following the fully decentralized method, IPPO, and our novel variant S-IPPO. Our results highlights the benefits of this new variant compared to IPPO, we also focus on inherent problem to the field of MARL such as the notions of non-stationary, moving target problem and equilibrium solutions

6.2 Future Works

As stated in the paper, CTDE methods have achieved interesting results on several popular benchmark. This same statement appears in the single-agent environment design, alternative RL algorithms such as Soft Actor-Critic (SAC), Twin Delayed Deep Deterministic Policy Gra-

dient (TD3), or model-based RL approaches could be explored to improve the results.

Another important avenue for future work is hyperparameter optimization. We believe that hyperparameter tuning is a critical aspect of RL experimentation, and the use of automated optimization tools such as Optuna could help achieve better performance. This would, however, require additional computational resources and time.

Finally, additional quality metrics could be incorporated into the evaluation of our experiments. With more careful planning and a more systematic approach to experimental design, future studies could include a broader set of metrics, leading to deeper insights and more robust comparisons across algorithms.

6.3 Perspectives

In this work, we sometimes scratch the surface of deep problems or take important assumptions that should be investigated. Here is a list of several open challenges and research directions remaining.

- A primary concern is the transferability of policies trained in simulation to real-world platforms. Bridging the sim-to-real gap will require the integration of real sensor feedback, better modeling of environmental noise and dynamics, and potentially the use of domain adaptation or fine-tuning on physical hardware. Moreover strong assumptions regarding the agent's observation were made such as the position of the drone and the target. Relaxing these assumptions should be the subject of future work.

- Another direction for future work involves exploring more complex network architecture, for example Recurrent Neural Network, especially for the multi-agent environment to enhance the performance of the defender and to decrease the unfairness of our task.

BIBLIOGRAPHY

Amari, S.-I. (1998). Natural gradient works efficiently in learning. *Neural Computation*, *10*(2), 251–276. https://doi.org/10.1162/089976698300017746

Åström, K. J., & Hägglund, T. (1995). *PID Controllers: Theory, Design, and Tuning* (2 ed.). Instrument Society of America. https://www.ucg.ac.me/skladiste/blog_2146/objava_92847/fajlovi/Astrom.pdf

Barret, E., Howley, E., Duggan, J. (2012, May). Applying reinforcement learning towards automating resource allocation and application scalability in the cloud. *Wiley preprint.* https://onlinelibrary.wiley.com/doi/10.1002/cpe.2864

Beard, R. W., & McLain, T. W. (2012, February). *Small Unmanned Aircraft: Theory and Practice.* Princeton University Press. https://www.researchgate.net/publication/286268062_Small_Unmanned_Aircraft_Theory_and_Practice

Blom, J. D. (2010, September). *Unmanned Aerial Systems: A Historical Perspective.* Fort Leavenworth, KS: Combat Studies Institute Press. https://www.armyupress.army.mil/Portals/7/combat-studies-institute/csi-books/OP37.pdf

Bonna, R., & Camino, J. F. (2015). Trajectory tracking control of a quadrotor using feedback linearization. *International Symposium on Dynamic Problems of Mechanics, 1*(9). https://sites.fem.unicamp.br/ camino/Conference/2015b-DINAME.pdf

Bouabdallah, S., Noth, A., & Siegwart, R. (2004, January). PID vs LQ Control Techniques Applied to an Indoor Micro Quadrotor. *Intelligent Robots and Systems, 3.* https://www.researchgate.net/publication/4122107_PID_vs_LQ_Control_Techniques_Applied_to_an_Indoor_Micro_Quadrotor

Bouabdallah, S., & Siegwart, R. (2005, April). Backstepping and sliding-mode techniques applied to an indoor micro quadrotor. *Proceedings of the 2005 IEEE international conference on robotics and automation,* 2247-2252. https://ieeexplore.ieee.org/document/1570447

Castillo, P., Lozano, R., & Dzul, A. E. (2005). Stabilization of a mini rotorcraft with four rotors. *IEEE Control Systems Magazine, 25*(6), 45–55. https://scispace.com/pdf/stabilization-of-a-mini-rotorcraft-having-four-rotors-56pfvj2nfl.pdf

de Witt, C. A. S., Foerster, J. N., Farquhar, G., Torr, P. H. S., Boehmer, W., & Whiteson, S. (2020). Multi-Agent Common Knowledge Reinforcement Learning. *arXiv preprint.* https://arxiv.org/abs/1810.11702

Ernst, D., Geurts, P., & Wehenkel, L. (2005). Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, *6*, 503–556. https://jmlr.org/papers/volume6/ernst05a/ernst05a.pdf

Ernst, D., Melduvin, G., & Wehenkel, L. (2008). Power Systems Stability Control: Reinforcement Learning Framework. *IEEE TRANSACTIONS ON POWER SYSTEMS*, 19. https://people.montefiore.uliege.be/ernst/uploads/news/id40/01266597-IEEE-2004.pdf

Faessler, M., Franchi, A., & Scaramuzza, D. (2017). Differential flatness of quadrotor dynamics subject to rotor drag for accurate tracking of high-speed trajectories. *IEEE Robotics and Automation Letters, 3*(2), 620-626. https://arxiv.org/abs/1712.02402

Finn, C., Abbeel, P., & Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference on Machine Learning (ICML), 70,* 1126–1135. https://arxiv.org/abs/1703.03400

Franetic. (2025, April 20). *Transforming Logistics: The Rise of Drone Delivery Services in 2025.* Franetic Digital Marketing Agency. Retrieved March 10, 2025 from https://franetic.com/drone-delivery-services-2025/

Fujimoto, S., Hoof, H., & Meger, D. (2018). Addressing Function Approximation Error in Actor-Critic Methods. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1587–1596. https://arxiv.org/abs/1802.09477

Geoawesome. (2024). *How Drone Data Helps to Monitor and Reverse Deforestation While Empowering Local Communities.* Retrieved March 10, 2025, from https://geoawesome.com/eo-hub/how-drone-data-helps-to-monitor-and-reverse-deforestation-while-empowering-local-communities/

Hoffmann, G. M., Huang, H., Waslander, S. L., & Tomlin, C. J. (2007). Quadrotor helicopter flight dynamics and control: theory and experiment. In *AIAA Guidance, Navigation and Control Conference*. https://ai.stanford.edu/ gabeh/papers/Quadrotor$_D ynamics_G NC07.pdf$

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 1861–1870. https://arxiv.org/abs/1801.01290

Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., & Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Robotics: Science and Systems 4.26 (RSS).* https://arxiv.org/abs/1901.08652

IEEE Public Safety. (2025). *How Drones Are Revolutionizing Search and Rescue.* Retrieved April 13, 2025, from https://publicsafety.ieee.org/topics/how-drones-are-revolutionizing-search-and-rescue/

Igl, M., Zintgraf, L., Le, T. A., Wood, F., & Whiteson, S. (2018). Deep Variational Reinforcement Learning for POMDPs. *Proceedings of the 35th International Conference on Machine Learning, 80*, 2117–2126. https://proceedings.mlr.press/v80/igl18a.html

Jain, S., Shethwala, Y., & Das, J. (2024). Towards Model Predictive Control for Acrobatic Quadrotor Flights. *arXiv preprint* https://arxiv.org/abs/2401.17418

Kaelbling, L. P., Littman, M. L., & Cassandra, A. R. (1998, May). Planning and acting in partially observable stochastic domains. *Artificial Intelligence, 101* (1-2), 99–134. https://doi.org/10.1016/S0004-3702(98)00023-X

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., & Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature, 620*, 982–987. https://doi.org/10.1038/s41586-023-06419-4

Kaufmann, E., Bauersfeld, L., & Scaramuzza, D. (2022). A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight. *arXiv preprint* https://arxiv.org/abs/2202.10796

Kaufmann, E., Loquercio, A., Ranftl, R., Müller, M., Koltun, V., & Scaramuzza, D. (2020). Deep drone acrobatics. *Robotics, Science, and Systems (RSS).* https://arxiv.org/abs/2006.05768

Khairuddin, I. M., Majeed, A. P., Lim, A., M. Jizat, J. A., & Jaafar, A. A. (2014). Modelling and PID Control of a Quadrotor Aerial Robot. *Advanced Materials Research, 903,* 327-331. https://www.researchgate.net/publication/263659597_Modelling_and_PID_Control_of_a_Quadrotor_Aerial_Robot

Kober, J., Bagnell, J. A., & Peters, J. (2013, August 23). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research, 32* (11), 1238–1274. https://doi.org/10.1177/0278364913495721

Konda, V. R., & Tsitsiklis, J. N. (1999). Actor-critic algorithms. *Advances in Neural Information Processing Systems, 12,* 1008–1014. https://proceedings.neurips.cc/paper/1999/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

Kwakernaak, H., & Sivan, R. (1972). *Linear Optimal Control Systems.* Wiley-Interscience. https://sina.sharif.edu/ namvar/index_files/Optimal/locs_hk_rs_cont.pdf

Lambrechts, G. (2021). *Bistable Recurrent Cells and Belief Filtering for Q-learning in Partially Observable Markov Decision Processes* [Thesis, University of Liege]. Mathéo. https://matheo.uliege.be/bitstream/2268.2/11474/11/thesis.pdf

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv.* https://arxiv.org/abs/1509.02971

LocoRobo. (2025). *How Drones Improve Infrastructure Inspection.* LocoRobo. Retrieved May 20, 2025, from https://locorobo.co/how-drones-improve-infrastructure-inspection/

Loh, M. (2025). *Russia has a new drone tactic: dive-bombing with Iranian-designed Shaheds from on high to avoid small arms fire.* Business Insider. Retrieved May 23, 2025, from https://www.businessinsider.com/russia-shahed-dive-bomb-shahed-kilometers-avoid-small-arms-fire-2025-5

Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., & Scaramuzza, D. (2021, October 6). Learning high-speed flight in the wild. *Science Robotics, 6*(59). https://www.science.org/doi/10.1126/scirobotics.abg5810

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. *Proceedings of the 33rd International Conference on Machine Learning (ICML), 48*, 1928–1937. https://arxiv.org/abs/1602.01783

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv preprint* arXiv:1312.5602. https://arxiv.org/abs/1312.5602

Moerland, T. M., Broekens, J., Plaat, A., & Jonker, C. M. (2020). Model-based reinforcement learning: A survey. *Foundations and Trends in Machine Learning, 16*(1), 1-118. https://arxiv.org/abs/2006.16712

NVIDIA. (2024). *NVIDIA IsaacLab: Modular robotics research framework.* NVIDIA DEVELOPER. Retrieved February 10, 2025, from https://developer.nvidia.com/isaac-lab

NVIDIA. (2024). *NVIDIA IsaacSim: Physically accurate robot simulation.* NVIDIA DEVELOPER. Retrieved February 10, 2025, from https://developer.nvidia.com/isaac-sim

Porter, T., & Baker, S. (2025, May 28). *Drones are the future of warfare, like gunpowder was in the 1300s, defense minister says.* Business Insider. Retrieved March 22, 2025, from https://www.businessinsider.com/drones-future-warfare-like-gunpowder-once-was-ukraine-russia-belgium-2025-5

O'Grady, S., Khudov, K., & Korolchuk, S. (2025). *Ukraine scrambles to overcome Russia's edge in fiber-optic drones.* The Washington Post. Retrieved March 10, 2025, from https://www.washingtonpost.com/world/2025/05/23/ukraine-russia-drones-fiberoptic-jamming/

Scaramuzza, D., & Kaufmann, E. (2023). Learning Agile, Vision-based Drone Flight: from Simulation to Reality. *arXiv preprint* https://arxiv.org/abs/2304.04128

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., & Abbeel, P. (2015). Trust region policy optimization. *Proceedings of the 32nd International Conference on Machine Learning (ICML), 37*, 1889–1897. http://proceedings.mlr.press/v37/schulman15.pdf

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint.* https://arxiv.org/abs/1707.06347

Schroeder de Witt, C., Gupta, T., Makoviichuk, D., Makoviychuk, V., Torr, P.H.S., Sun, M., & Whiteson, S. (2020, November 18). Is independent learning all you need in the StarCraft multi-agent challenge? *arXiv preprint.* https://arxiv.org/abs/2011.09533

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016, January 27). Mastering the game of Go with deep neural networks and

tree search. *Nature, 529*(7587), 484–489. https://www.nature.com/articles/nature16961

Song, Y., Romero, A., Müller, M., Koltun, V., & Scaramuzza, D. (2023). Reaching the Limit in Autonomous Racing: Optimal Control versus Reinforcement Learning. *arXiv preprint* https://arxiv.org/abs/2310.10943

Spaan, M. T. J. (2012). Partially observable Markov decision processes. In Wiering, M., & van Otterlo, M. (Eds.), *Reinforcement learning: State-of-the-art* (pp. 387–414). Springer. https://doi.org/10.1007/978-3-642-27645-3_12

Srivastava, A., Indu, S., & Sharma, R. (2024). Design And Flight Testing Of LQRi Attitude Control For Quadcopter UAV. *arXiv preprint.* https://arxiv.org/abs/2404.12261

Stone, P. (2025, June 7). *Lidar Drones in Climate Research: Mapping Changes in Real-Time.* LidarTechPros.com. Retrieved May 12, 2025, from https://www.lidartechpros.com/climate-research-with-lidar-drones/

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press. https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf

Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (1999). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems, 12*, 1057–1063. https://papers.nips.cc/paper/1713-policy-gradient-methods-for-reinforcement-learning-with-function-approximation.pdf

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., & Abbeel, P. (2017). Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 23–30. https://arxiv.org/abs/1703.06907

Tzeng, E., Hoffman, J., Saenko, K., & Darrell, T. (2017). Adversarial discriminative domain adaptation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 7167–7176. https://arxiv.org/abs/1702.05464

Vengerov, D. (2008, July). A gradient-based reinforcement learning approach to dynamic pricing in partially-observable environments *Future Generation Computer Systems, 24*(7), 687-693. https://www.sciencedirect.com/science/article/abs/pii/S0167739X08000307

Vinyals, O., Babuschkin, I., WCzarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., Oh, J., Horgan, D., Kroiss, M., Danihelka, I., Huang, A., Sifre, L., Cai, T., Agapiou, J. P., Jaderberg, M., ... Silver, D. (2019). Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature, 575*(7782), 350–354. https://www.nature.com/articles/s41586-019-1724-z

Viper Drones. (2025). *Aerial Photography and Videography with Drones.* Retrieved March 25, 2025, from https://viper-drones.com/drone-solutions/aerial-photography-and-videography-with-drones/

Wiering, M., & van Otterlo, M. (Eds.). (2012). *Reinforcement learning: State-of-the-art.* Springer. https://doi.org/10.1007/978-3-642-27645-3

Wikipedia contributors. (2025). *List of Unmanned Aerial Vehicle Applications.* Wikipedia. Retrieved January 12, 2025, from https://en.wikipedia.org/wiki/List_of_unmanned_aerial_vehicle_applications

Williams, R. J. (1992). Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning, 8*, 229–256. https://doi.org/10.1007/BF00992696

Xu, B., Gao, F., Yu, C., Zhang, R., Wu, Y., & Wang, Y. (2023). OmniDrones: An Efficient and Flexible Platform for Reinforcement Learning in Drone Control. *arXiv preprint.* https://arxiv.org/abs/2309.12825

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., & Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 3357–3364. https://arxiv.org/abs/1609.05143