Existing Gaps In Reinforcement Learning For Drone Warfare

Arthur Louette[1*†], Pascal Leroy[1†], Yanis Geurts[1], Damien Ernst[1]

[1]Montefiore Institute, ULiège, Liège, Belgium.

*Corresponding author(s). E-mail(s): arthur.louette@uliege.be;
†These authors contributed equally to this work.

Abstract

Drones have changed warfare and are deployed daily on the battlefield for surveillance or as offensive and defensive weapons. While humans continue to control drones and weapon systems, the transition to autonomous control, which removes the human decision, is imminent. Indeed, advances in artificial intelligence (AI) are extremely rapid and AI-driven drones seem to represent the future of warfare. This motivates the need to improve systems to face autonomous drones and build better ones. Reinforcement learning (RL) is a paradigm of AI focusing on the resolution of sequential decision-making problems. Its deployment in robotics shows its potential to address complex real-world challenges. After presenting RL foundations with a practical battlefield example, we propose a framework to deploy RL in robotics. We identify five axes of complexity to deploy RL on robots for any real-world problem. These axes allow us to analyze the state-of-the-art and identify gaps required by the future of drone warfare. We conclude the paper with a roadmap to bridge these gaps and ethical considerations.

Keywords: reinforcement learning, drones

1 Introduction

Unmanned aerial systems (UAS) have long played an important role in modern warfare. It started with large drones designed for precision strikes and reconnaissance missions [1]. The conflict in Ukraine showcases a groundbreaking shift toward smaller drones, most of the time by weaponizing commercial drones. Such drones provide strategic advantages, disrupt enemy operations, and gather critical data, marking a significant advancement in drone warfare [2–4]. They allow the control of contested

battlefield areas, offering a cost-effective way to conduct strikes and gather intelligence without risking human lives. Nowadays, military personnel analyze data and control drones. For example, identifying a target by analyzing data from a reconnaissance drone, and then controlling a weaponized drone to strike it [5].

AI has already been identified as the perfect candidate to perform some of the drone tasks on the battlefield. This is the case for vision-related tasks, such as identifying potential targets [6]. This enables the analysis of vast amounts of data faster than human operators, improving the speed of decisions on the battlefield. Such integration of AI increases the efficiency and effectiveness of drone operations. However, even if AI facilitates pilots' work, drone control remains mostly handled by humans.

In the Ukrainian conflict, we especially observe that first-person view (FPV) pilots have become a precious resource for the war [1, 3, 5, 7]. The FPV drones suffer from several challenges, such as their deployment efficiency, the number of pilots available to pilot them, the exposure of the latest, and the communication channel between the drone and its operator that can be jammed. Although drones have become an essential weapon on the modern battlefield, countering them has emerged as an additional challenge, particularly with small, agile drones weighing less than 10 kg. These highlight the need to improve AI systems to control UAS and counter UAS (CUAS). This is why we focus in this paper on AI for control, especially in the context of small drones and the associated countermeasures.

A well-established framework of AI for control is called reinforcement learning (RL). In RL, an agent learns to make decisions by trial and error. Over the last years, it has shown great capabilities for sequential decision-making problems, and achieved superhuman performance in complex games such as StarCraft II [8] or Stratego [9]. Recently, RL has demonstrated the capacity to control FPV drones better than the best pilots in FPV races [10] and has more generally made big advancements in many domains, such as autonomous navigation [11]. There is no doubt that it has the potential to improve control algorithms on battlefields. However, there is still a gap in deploying autonomous controllers in warfare. These previous examples typically make strong hypotheses on the drone conditions that should be relaxed to be closer to battlefields. This paper analyses and classifies these gaps before suggesting a roadmap for remaining efforts to control small UAS drones and defend against them.

The paper is organized as follows. Section 2 details the current battlefield landscape, oriented towards small drones and associated countermeasures. We then formally describe reinforcement learning in Section 3 and illustrate the framework with a practical example. We propose a framework for the deployment of RL to solve such problems in Section 4. Afterward, we propose in Section 5 five axes of complexity to deploy RL. For each of these axes, we identify the state-of-the-art performance algorithms applied to robotics, especially drones. In Section 6, five scenarios are presented, each representing an innovation milestone that progressively advances drone warfare toward autonomous UAS and CUAS.

2 Drone Warfare

Currently, commercial drones are the most common type on the battlefield [3]. Brands like DJI and Autel dominate the market with low-cost drones with advanced features.

Affordable drones can also be built by connecting a flight controller to a 3D-printed drone frame. These commercial drones have several advantages. Anyone can buy one. They do not require large infrastructure to launch and are difficult to track [7]. Small FPV drones equipped with explosive warheads cost several hundred dollars. This allows anyone to produce them in large quantities [5]. For example, due to this affordable price, Ukraine and Russia produced millions in 2024 [12]. These FPV drones mainly carry out two missions: surveillance and combat. During surveillance missions, drones determine or monitor targets [12, 13]. In combat missions, drones can act like bombers, kamikazes, and even be equipped with guns. The drones can be equipped with various munitions, such as armor-piercing, cluster, or thermobaric ones. Drones are known to be more precise than artillery, especially for moving targets [5]. Based on the success of drones on the Ukrainian battlefield, countries like the USA, Canada, and China have already started developing their own FPV drone systems and counters [13].

There exist countermeasures to these drones, distinguished between physical and electronic systems. On one side, physical defenses include air defense systems, lasers, net guns, or even drones [14]. Multiple options have been investigated, such as combining anti-aircraft guns with radar and laser systems [3], or drones tracking other drones [15]. One of the main issues is that drones are agile and can dodge defense ammunition. Another is their cost. Indeed, most air defense missiles are more expensive than their target [16]. On the other side, electronic defense, called electronic warfare (EW), consists of radio jammers or spoofers but also eavesdropping, denial-of-service, and information injection [17]. The EW systems can be mounted on vehicles, drones, or even as backpacks. Radio jamming and spoofing systems aim to find the right communication frequency of the targeted drone and block or replace the signal. Some electronic reconnaissance systems even try to trace the drone's signals back to the enemy pilots' location [5]. EW countermeasures also have some vulnerabilities. Firstly, pilots can deploy signal repeaters on the ground or other drones to avoid being detected. Secondly, once they detect interference, some drone pilots quickly switch to another operational channel or trigger a specific reaction, like coming back home or switching the communication frequency [17–19]. Thirdly, and arguably the most promising, AI-controlled drones don't need to be connected to an operator and, therefore, cannot be jammed [6]. A common vulnerability for both physical and electronic countermeasures is saturation attacks. These attacks are composed of multiple drones to saturate the CUAS and challenge the system to target efficiently to avoid saturation. Finally, if drone control strategies improve using AI, the efficiency of such anti-drone weapons would probably decrease and, therefore, also require AI integration. For instance, one might design an AI system to control an anti-aircraft gun with superhuman precision or pilot a drone capable of neutralizing enemy drones. This highlights the need to include AI in countermeasure systems, which could lead to an ensemble of AI-driven systems.

In parallel with the emergence of FPV drones, AI has been increasingly integrated into drone operations. AI-enhanced systems currently improve target detection [20]. Machine learning algorithms, particularly computer vision techniques, allow drones to autonomously track and identify targets with high precision from the real-time

image stream captured by drones' cameras [6]. Even if AI has started to be standardized for some vision tasks, the use of AI for autonomous drone control is still in its early stages. This limitation emphasizes the value of FPV pilots. Since their mission requires high skill, the FPV pilots controlling these agile drones become a priority target [5]. Additionally, due to the short-range nature of FPV drone operations, pilots are often exposed to enemy forces. As the demand for drones grows, production is scaling up to meet this need. Following this, the increasing demand for drone operations may outpace the availability of skilled pilots. This again highlights the need for AI-controlled drones, that could soon outperform human pilots. Indeed, it has already been demonstrated that AI could beat some of the best aircraft pilots in combat aircraft simulations [21]. Automating control would also enable the deployment of coordinated drone swarms, amplifying their combat effectiveness and operational reach. This transformation could redefine modern warfare, where machines continuously adapt and learn.

3 Reinforcement Learning

Reinforcement learning has emerged as a promising method for autonomous control. In RL, the autonomous control is handled by an agent. An agent is defined as anything that can act based on the information it perceives from its environment [22]. Different classes of agents exist, but in RL, agents learn to act rationally to achieve a predefined goal. This goal is defined by numerical rewards the agent obtains after making any decision, allowing it to learn. Specifically, as illustrated in Figure 1, an agent learns to maximize its expected sum of rewards by sequentially interacting with its environment, taking an action based on its observation, and receiving a reward. Designing the reward function is thus one of the crucial aspects leading to defining the task of the agent.

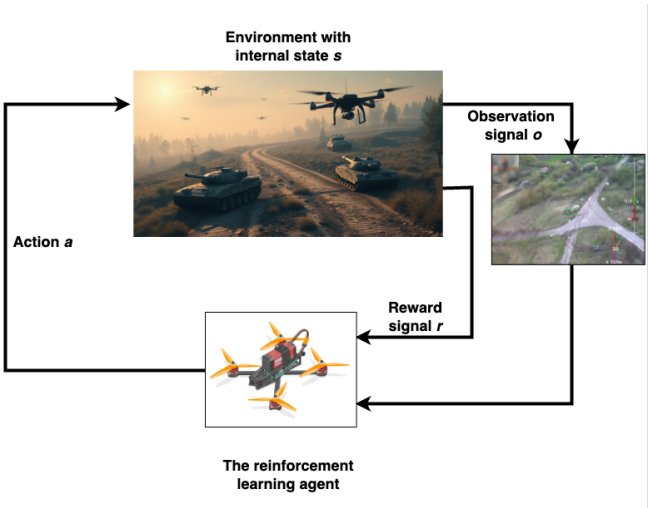

Fig. 1: Interaction of an agent with its environment in RL.

RL problems are commonly formulated as Markov decision processes (MDPs), which offer a rigorous mathematical framework for modeling the interaction between an agent and its environment. An MDP is formally defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, H, p_0)$. The state of the environment is $s \in \mathcal{S}$, where $\mathcal{S}$ is the set of all possible states. In a fully observable MDP, the agent maps a state to an action $a \in \mathcal{A}$ based on its policy $\pi(a|s) : \mathcal{S} \to \Delta(\mathcal{A})$. Based on a state-action pair, the environment transitions to a new state s' with a probability defined by the transition probability distribution $\mathcal{P}(s'|s, a)$. It also receives a reward $r = R(s, a, s')$ where R is the reward function $R : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$. The quality of an agent is quantified by the expected sum of discounted rewards it receives during an episode called the return $G(\pi) = \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r_t | s_0 \sim p_0, \pi]$ which depends on its policy π and where p_0 is the initial state distribution, H is the horizon and γ is the discount factor. The horizon can be finite or infinite. The discount factor allows for prioritizing short-term rewards. With RL, the objective is to find an optimal policy $\pi^* \in \text{argmax}_\pi G(\pi)$.

In real life, especially when dealing with robots, the agent cannot fully perceive the state of its environment. It has only access to an observation $o \in \mathcal{O}$ typically defined by its sensors, e.g., an image for an FPV drone, $\mathcal{O}$ being the set of observations. The probability of receiving a particular observation o is defined by the observation model $O(o|s) : \mathcal{S} \to \Delta(\mathcal{O})$. In this case, the problem is formalized as a partially observable MDP, or POMDP in short [23]. In such a framework, the action is generally selected based on the history of past actions and observations to act optimally.

The MDP and POMDP are single-agent problem formulations that allow one to simulate the battlefield to learn to control a drone with respectively full and partial observability. However, the (PO)MDP settings are often used by implicitly considering that other agents of the battlefield are not changing strategies over time. Their policies are stationary. When multiple agents with non-stationary policies are considered, one common approach is to use multi-agent reinforcement learning (MARL), a mix of RL and game theory. The general framework is the partially observable stochastic game (POSG) [24]. Formally, a POSG is defined by a tuple $(n, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, O, \mathcal{R}, \gamma, H, p_0)$ where n agents interact in the same environment. After all agents take their action $\mathbf{a} \in \mathcal{A}$, the state transitions in the new state s' with a probability given by $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ and $\mathcal{A} = \{\mathcal{A}^1 \times .. \times \mathcal{A}^n\}$ is the set of n action spaces. The probability of all agent observations $(o^1, .., o^n)$ is defined by the observation function $O(o^1, .., o^n|s) : S \to \Delta(\mathcal{O})$ where $\mathcal{O} = \{\mathcal{O}^1 \times ... \times \mathcal{O}^n\}$ is the set of n observation spaces. Based on all actions, each agent receives a reward $r^i = R^i(s, \mathbf{a}, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ and $\mathcal{R} = \{R^1, .., R^n\}$ is thus the set of reward functions. The time horizon H and initial state distribution p_0 serve the same purposes as the previous definitions. The return of agent i is $G^i(\boldsymbol{\pi}) = \mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r_t^i | s_0 \sim p_0, \boldsymbol{\pi}]$, where $\boldsymbol{\pi} = (\pi^1, .., \pi^n)$ is the joint policy. Its objective is to find its policy $\pi^i(a^i|o^i) : \mathcal{O}^i \to \Delta(\mathcal{A}^i)$ that maximizes its return $max_{\pi^i} G^i(\boldsymbol{\pi})$. POSG can be designed for cooperation, e.g., agents receive the same reward to achieve a common objective, or for competition, e.g., with zero-sum rewards, i.e., any benefit to one agent results in an equivalent loss for the other. The general case is when there is no clear relation between the rewards of agents. In a battlefield, the environment is a mixed cooperative-competitive POSG since some agents cooperate to achieve a goal at the detriment of other agents.

Since the return maximized by each agent is a function of all agents' policies, changing one agent's reward function also changes the POSG's solution. Specifically, at a given time, a battlefield is a POSG with a given set of agents $\{\{\mathcal{A}^i, \mathcal{O}^i, R^i\}_{i\in\{1,n\}}\}$, each defined by a 3-tuple defining their capabilities and tasks. We consider the simplification that the definitions of $\mathcal{S}$, $\mathcal{P}$ and O depend on the set of agents. When considering a battlefield over its total duration, the number of agents, their capabilities, and their tasks change over time. A battlefield can thus be defined as a non-stationary POSG with a dynamic set of agents. However, a non-stationary POSG can always be defined as stationary with additional information, such as the time. Therefore, this paper does not make the distinction.

To illustrate the formulations with a drone warfare example, we hereafter define each component in a scenario where a drone has to identify and neutralize enemy targets in a combat zone protected by an anti-drone system, pointing out the complexity associated with each element of a POSG.

- For a given set of agents, the state space $\mathcal{S}$ represents all possible battlefield configurations, including, among others, the drone's current location, altitude, speed, battery level, but also other agents' locations, weather conditions, and obstacles such as buildings. There is an infinite number of possible values only for the drone-related states, already reflecting the complexity induced by the state space of a battlefield.
- The sensors shipped with the drone define its observation space, the same for the anti-drone system. For example, the drone might only detect enemies within a certain range or receive noisy signals about enemy positions. The number of available sensors for a drone illustrates the number of available observation spaces when considering a single agent with non-stationary observation spaces.
- A drone can be controlled with high-level actions, such as moving to a new location, adjusting its altitude, engaging a target, or returning to base for recharge. But it can also be controlled by deciding on thrust, yaw, pitch, and roll, or even directly by controlling the volts in each motor. This generalizes to any controllable system. Defining the action space $\mathcal{A}^i$ inherently affects the complexity of learning a policy.
- The transition probability $\mathcal{P}(s'|s, \mathbf{a})$ models the uncertainty in how the environment changes after an action is taken. For instance, if a drone fires a missile, the target will be destroyed, damaged, or missed, each with a probability depending on factors like distance, wind, and enemy defense systems. Having a good model of reality is still a complex task nowadays.
- The reward function of an agent, R^i, defines its behavior once trained and thus its task on the battlefield. For example, the drone neutralizing the anti-drone system will provide the drone with a positive reward while a negative one for the anti-drone system. The rewards in such situations are most likely built with different factors, encompassing task success while minimizing risks and collateral damage.

The last component, the reward function, defines the task solved by the agent. Since the success of a policy is evaluated by the reward function, it is thus crucial to design it accordingly. For example, sparse rewards, such as $+1$ if the task is solved and 0 otherwise, often lead to inefficient learning. This is because agents receive minimal feedback, requiring extensive exploration before they can learn useful behaviors [25, 26]. Indeed,

dense rewards guiding the agents smoothly toward successful policies are preferred but often necessitate advanced knowledge to be designed [27]. In practice, multiple objectives compose the reward, e.g., navigating with a drone as fast as possible while minimizing battery usage. The reward encodes the trade-off between these objectives, balancing factors such as increased energy consumption for faster navigation versus reduced energy consumption at the cost of slower progress. Finding a policy for any trade-off has been addressed in the literature on multi-objective reinforcement learning (MORL) [28]. Moreover, large language models (LLMs) outperform human experts at designing reward functions, allowing training some RL agents for robotic tasks for the first time [29–32]. These LLMs allow humans to define tasks or objectives for agents through language or speech instead of designing a numerical reward function. The reward function defines a single task. However, the multi-task setting exists, where a policy must be successful in several tasks. Commonly in this setting, the reward function and the policy are conditioned on the task, such that there is still a single optimal policy. This implies that the policy dynamically adapts its behavior based on the conditioning of the task.

RL algorithms are one efficient way to obtain policies for problems formulated in these frameworks. For MDP, popular methods include PPO [33], SAC [34], or DQN [35], depending on the applications and the types of action and state spaces. For POMDP, the same methods are often applied with models whose architecture can deal with a history of observations and actions [36–38]. In these two single-agent frameworks, aside from the complexity of finding them, selecting the best policy among others is straightforward when knowing their expected returns. However, an agent maximizing its return will be to the detriment of the return of some others in a mixed cooperative-competitive POSG, making the choice of an optimal joint policy non-trivial. If you fix the policies of all agents except i, denoted $\boldsymbol{\pi}^{-i}$, you can find the best response policy in the set of $\text{argmax}_{\pi^i} G^i(\pi^i, \boldsymbol{\pi}^{-i})$ with single-agent methods. It is then possible to iteratively find the best response to the best response of others. However, agents can all adapt to others' strategies, leading to an infinite cycle of adaptation. Another type of solution is a Nash equilibrium, achieved when no agents benefit from changing its policy: $\forall i$, any $\pi^{i'}$ is such that $G^i(\pi^{i'}, \boldsymbol{\pi}^{-\boldsymbol{i}}) \leq G^i(\boldsymbol{\pi})$ [39]. Several Nash equilibria may exist in a POSG and may not provide the highest return. Approaches to solving POSG are described in [40] and can be summarized grossly as training a population of agents and thus learning against various strategies. Since agents of the population will have different strategies, there is a need for a selection strategy to deploy one in the real world. The Elo score [41] is an example of a metric to rank agents from a population. The Elo system assigns each individual a rating computed to evaluate one individual's probability of winning against another. Examples of such population-based training include Quake III Arena in Capture the Flag mode [42], StarCraft [8] or the competitive StarCraft multi-agent challenge [43]. A popular but complex method is policy space response oracles (PSRO) [44, 45].

RL is not the single existing approach to solve sequential decision-making problems. We can, for example, also mention model predictive control techniques (MPC), originating from the control community [46]. These techniques use a model of the problem,

both of its transitions and reward functions, to compute a policy using mathematical programming approaches. Contrary to RL techniques, they are often constrained to optimal sequential decision-making problems whose dynamics and reward function exhibit specific strong properties, such as linearity or convexity. This is necessary for the mathematical programming techniques to extract a successful policy from the models, which can be given a priori or learned. One drawback of these techniques is the higher computational cost since the algorithm solves an optimization problem at the execution phase. In contrast, in RL, this computation is moved to the learning phase. However, these mathematical programming techniques provide better actions for a short-term horizon if the models are accurate, while RL usually performs better in a long-term horizon [47]. Finally, it is possible to combine MPC and RL in different ways [46, 47].

Another approach for solving sequential decision-making problems revolves around imitation learning, where a dataset of states and actions is provided to learn a policy. A modern example is foundation models that learn to solve several tasks from internet-scale datasets [48]. When applied to decision-making, they are often called action foundation models. These models are made of large networks trained to reproduce actions taken in large datasets of demonstrations. These models are usually trained to perform multiple tasks and are conditioned on the task. Hence, when conditioned on the task targeted by an RL problem, they can be used to play the role of an RL policy [49].

4 Reinforcement Learning In Practice

The training process for RL agents involves several phases to deploy successful policies in real-world applications. In these applications, RL agents are most commonly employed as individual robot controllers. In this paper, higher-level agents that coordinate or assign missions to others are not considered, i.e., hierarchical agents. A critical challenge in the training process is ensuring that policies, whether pre-trained in simulation or with synthetic or real data, can perform well when deployed in the real world. This section highlights the steps in the training process and addresses the challenges posed by such deployment.

Pre-Training

Directly learning in the real world from scratch is possible, but challenging. Today, it is accomplished for tasks with a low probability of damaging the hardware and involving low human interaction during the learning phase, e.g., learning to walk for a quadruped [50]. In addition, it has to be data-efficient to be trained in a reasonable time. For many tasks, speeding up training and avoiding catastrophic states and actions that could break the hardware, usually explored when training from scratch, are mandatory. Therefore, policies are usually trained in a simulator or with a dataset before being deployed in reality. This is called pre-training [51–53]. Pre-training the agents aims to provide successful policies for the desired tasks in confined conditions. We distinguish three possible approaches for this phase:

1. *Training in a simulator*: The first training approach relies on a simulator to replicate the robot's real environment. RL agents are thus trained directly by interacting with the simulator. A simulator enables gathering experience in a controlled environment at scale [54–56] and without the risk of damaging hardware [53]. In real conditions, it is difficult to accumulate the large amounts of training data that we can with simulated environments executed in parallel. Moreover, evaluating the reward is sometimes impossible in real life, but it is always possible in the simulator because of the perfect information. The main challenge is bridging the gap between the optimal policy obtained in simulation and reality. Indeed, the goal is not necessarily to simulate reality perfectly but to ensure that if we train the agent to act optimally in the simulator, then it will also act optimally in reality.
2. *Training with a dataset*: This second approach involves building a dataset of real transitions. These transitions are typically tuples of state-action pairs, sometimes with additional information such as a reward signal and the next state achieved. These transitions are often collected from expert demonstrations, from synthetic or real data. Gigantic datasets of robotic demonstrations already exist to train any models [57]. This approach can avoid running a simulator, which also requires significant effort to build. We distinguish two approaches for training with such datasets. The first is to train a model to reproduce the behavior from the data is referred to as imitation learning or behavior cloning. However, the quality of the pre-trained model will depend on the quality and quantity of the training demonstrations and cannot achieve better performance than these expert demonstrations [58]. The second approach uses the dataset with rewards and next state for reinforcement learning, and it is referred to as offline RL. Such methods can achieve better performance than the expert ones [59].
3. *Robotic foundation model*: Another possibility is to take a model off the shelf already trained for general purposes, such as foundation models [60, 61]. These are large pre-trained models with an architecture designed to generalize across a wide range of robotic tasks. They are typically trained on massive and diverse datasets of real-world transitions, which provide the ability to perform well in various environments. However, the performance of a foundation model is limited by the quality and diversity of the training data to the desired task.

A mix of these approaches is the most promising for agile drones. This mixed pre-training could include starting from a policy pretrained with expert demonstrations and improved by training with RL in simulation. This pre-training can reduce the risk of breaking the drone when deployed because it has hopefully learned how to avoid catastrophic failure and can explore the environment safely. Indeed, drones require challenging pre-training or adding a safety layer to avoid breaking too many of them [62].

Evaluation

Evaluating agents is a mandatory step before deploying policies in the real world and validating them in real conditions. In contrast, the evaluation after deployment validates the performance in the desired task, identifies performance gaps, and guides fine-tuning to address these shortcomings. Usually, the evaluations at the pre-training

and the deployment phase differ. Indeed, the pre-training evaluation can leverage perfect information knowledge in a simulator to compute the reward function and task-specific metrics, in contrast to the evaluation after deployment, where the information is imperfect. In addition, when evaluating multiple agents, computing an Elo score, defined in Section 3, typically requires comparing one agent's policy with many others, which is not always possible at the deployment phase. In addition, all the information needed to evaluate a reward function or task-specific metrics in the real world is not always known. After deployment, a first evaluation without any fine-tuning of the model is often necessary to evaluate the performance. This is called zero-shot evaluation. All components of the reward may not be observable in the real world, and only those available can be considered to evaluate the agent.

Deployment

Once a high-quality pre-trained policy is obtained, it will be deployed in reality. The performance of the pre-trained policy often degrades when applied outside the confined conditions of the simulator or dataset. Indeed, when a policy is deployed, it suffers from distribution shifts. These shifts can happen in all components around the policy, such as the observation space, the action space, the transition, and reward functions. This justifies again the zero-shot evaluation. Usually, these shifts degrade the policy's performance, and a fine-tuning process is thus required.

Fine-tuning

Fine-tuning follows evaluation and involves refining the policy based on identified gaps. This evaluation and fine-tuning process is often repeated multiple times. This iterative process may include adapting the simulator, incorporating new real-world data, or modifying the reward and transition functions to better align with deployment scenarios. Through this cycle of evaluation and fine-tuning, RL agents progressively improve their real-world performance. Kaufmann et al. [10], for instance, achieves state-of-the-art performance in FPV drone racing by training an RL agent in simulation and fine-tuning the simulator iteratively from real-world performance evaluation, ultimately surpassing human expert FPV pilots.

To conclude, while pre-training provides an important starting point, the deployment phase introduces significant challenges, requiring careful iterative evaluation and fine-tuning to achieve successful real-world performance. The next section covers these challenges and the state-of-the-art methods to address them.

5 Challenges to deploy Reinforcement Learning for Robots for any POSG

As described in Section 3, the complexity of solving a POSG is inherently related to the definition of its components $(n, \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{O}, O, \mathcal{R}, \gamma, H, p_0)$. Successful autonomous robots in any POSG, meaning in any environment, for any task, interacting cooperatively or competitively with any number of agents, for any observation space, and any action space, are still a utopia. Based on this, we consider five axes of complexity when defining any real-world RL problem. These five axes are the diversity of the

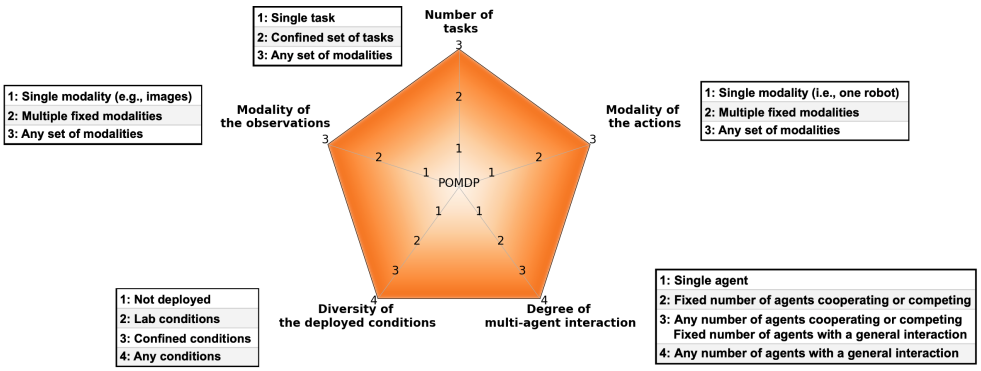

Fig. 2: Axes of complexity for any POSG. The axes have three or four levels described in the corresponding legends. The simplest setting is a POMDP, represented by level 1 for each axis.

deployed conditions of the agent, the number of tasks where the agent needs to be successful, its degree of multi-agent interaction, the modalities of its action space, and the modalities of its observation space. Each axis intrinsically affects the definition of the POSG tuple and, therefore, the complexity of finding a successful policy. To identify the gaps and the roadmap toward learning a successful policy for any POSG, we identified different levels in each axis.

The origin of all axes, the first level, is the standard RL setting, a POMDP. Indeed, it comprises a stationary transition function, a stationary reward function, a single agent, and stationary observation and action spaces. Recent works successfully deployed robots in such settings [63, 64]. Others study the deployment by increasing the level of one or two axes, but not all five [49, 60, 65]. Dealing with higher levels of complexity in all five axes is required to train a single successful policy for any POSG.

In the following, we discuss each axis independently and then their correlation. These axes are represented by a radar chart in Figure 2, where reaching the edge of all axes represents the case where the environment can be any POSG. Each axis, its state-of-the-art, and its levels are presented hereafter in dedicated sections.

5.1 Descriptions of the axes

Diversity of the deployed conditions.

The first axis represents the diversity of the deployed conditions for one task. The components involved are $\mathcal{P}, \mathcal{O}$ and O. Simulators allow RL agents to train in safe, controlled, and scalable environments. This is also the case when pre-training from data. However, as described in Section 4, deploying the learned policies to the real world often results in significant performance degradation due to shifts in dynamics, sensory inputs, and unmodeled noises.

Moreover, adversarial attacks, i.e., techniques to deceive machine learning models by introducing subtle perturbations to inputs, could become a standard method to counter autonomous drone swarms, exploiting these vulnerabilities by targeting

sensory inputs. For example, adversarial perturbations in visual data might lead an AI-driven drone to misidentify objects or targets, compromising its mission. Countermeasures for autonomous drones must include defensive AI systems that are resilient to adversarial attacks, while also developing offensive techniques to exploit these vulnerabilities.

Different techniques exist to bridge the deployment gap and ensure resilience against adversarial attacks:

1. *Domain randomization* [66] exposes agents to a wide range of simulated conditions by varying environmental parameters (e.g., lighting, textures, or dynamics). By training policies on diverse simulated experiences, agents learn to generalize better in real-world scenarios.
2. *Simulation fidelity improvements* increase the fidelity of simulators by incorporating better models of real-world dynamics. This can involve using advanced physics engines or integrating real-world measurements to refine simulation parameters.
3. *Robust policy learning*, or robust RL algorithms, focus on learning policies that can handle uncertainty and variability. Methods such as adversarial training have shown promise in mitigating deployment discrepancies [67]. Adversarial training also allows for mitigating adversarial attacks, enhancing the robustness under adversarial conditions [68, 69].

These techniques improve the transferability of RL agents from confined simulated conditions to real and more diverse ones. Successful robot controllers have bridged the deployment gap in various challenging real environments [70]. This axis is composed of four levels. The first level is when the policy is not deployed in the real world. The second level consists of policies deployed in conditions strictly confined to those encountered during pretraining, e.g., replicating the simulated environment in a lab. The third level is achieved when the policy generalizes to unseen real-world environments close to the training conditions. Finally, a policy achieves the fourth level when it succeeds in any real-world conditions.

To deploy AI-controlled drones on the battlefield, the 4th level needs to be reached. Indeed, the drone control policy should generalize to the diversity encountered in the combat zone. Nowadays, state-of-the-art FPV drone controllers are reaching level 3 [10], i.e., they are limited to a confined distribution of real-world environments close to the training conditions. The next step for reinforcement learning applied to robots is to train agents in environments with more diverse conditions, such as uncertain or hostile airspace domains.

Number of tasks.

The second axis represents the number of tasks learned by a single policy. The component concerned is mostly R, which can be conditioned on the task to be learned by simply adding to the state a description of the task. As introduced in Section 3, the task learned by the agent is defined by the reward function. To obtain a successful policy for any POSG, it is thus necessary to be successful in any task. Advances in RL have unlocked new capabilities for robots to solve many complex tasks [10, 29, 71–73].

Recent works show a shift from using a set of small agents, each solving a particular task, toward foundation models able to handle a set of tasks [48, 49, 60, 74]. This paradigm is inspired by the success of LLMs, which can generalize across multiple tasks and adapt to new ones through fine-tuning or prompting. Similarly, robotic foundation models can be fine-tuned to a particular set of tasks to improve performance [61, 75]. Further research is needed to deploy robots that are successful at zero-shot evaluation. This means they can directly execute any behavior, i.e., define their reward function and execute the policy that will maximize it to solve any task specified by the user using speech, image, or language. The main challenge in this axis remains the availability of the data. Despite the progress made in the architecture of these models, the biggest breakthrough comes from the internet-scaled high-quality labeled datasets training them. In this direction, labeling with machine learning models unlabeled internet data seems to be a promising approach [49, 76]. Reinforcement learning has also been used to generate data to train such models [58]. It remains to be seen whether robotic foundation models can scale to solve any task and achieve the level of performance observed for LLMs.

This axis has three levels. The first level represents policies that can only solve one task. In the second level, the policies are successful in a set of tasks encountered during training. The third and final level is when the agent can solve any task.

Degree of multi-agent interaction.

The third axis represents the degree of multi-agent interaction. The components concerned are n, R and thus $\{\{\mathcal{A}^i, \mathcal{O}^i, R^i\}_{i\in\{1,n\}}\}$. In the real world, several robots interact together. In most of the works presented in this paper, deployed agents are trained without considering the potential adaptations of other agents to their policy. Considering that a single agent is learning in a multi-agent environment is equivalent to considering that other agents have a stationary policy. This hypothesis is too restrictive and illustrates the need for MARL, the extension of single-agent reinforcement learning when several agents learn in the environment and therefore also adapt to others. When looking at today's works, such as in recent surveys [77, 78], multi-agent approaches are common in robotics, especially for cooperative scenarios [79, 80]. However, it seems that the community is only at the premise of considering adaptations of other agents, particularly in environments with opposing goals. Especially, deployed agents rarely consider the adaptations of others.

An additional parameter in multi-agent systems is the number of agents, and how this number changes over time. In MARL, some environments are composed of a fixed number of agents [8, 81], and others with a number of agents that can change over time [82, 83]. The number of agents is a critical parameter for MARL, and, as mentioned by Albrecht et al. [40], only between 2 and 10 agents are often considered, and one MARL goal is to scale this number.

This third axis is made up of four levels. The first level is when there is no multi-agent adaptation, meaning agents are trained agnostically of others. This is therefore the single-agent paradigm. The second level consists of environments with a fixed number of agents and a single fixed type of multi-agent interaction, meaning it is either cooperation or competition. The third level is a mix. Either it is a cooperation

or competition, but with a variable number of agents, or it is a general interaction, typically a mixed cooperative-competitive interaction, but with a fixed number of agents. The fourth and last level is any interaction with any variable number of agents.

Modality of the observation space.

The fourth axis represents the modalities of the observation space. The components concerned are $\mathcal{O}$ and O. The future robot controllers will process inputs of different modalities, meaning texts, vocals, and images, etc. Indeed, robots acquire data through their sensors, and their controller should account for all data sources to act. Nowadays, RL algorithms fuse many sensors to act [10, 73]. Recent research has focused on the flexibility of the observation space to switch from one set of modes to another [60]. However, the research has not reached the performance level to operate robots with any set of observation modes without retraining. For instance, future robots should adapt to scenarios where sensors are unavailable or broken, or when new modalities are introduced. Developing adaptive controllers capable of processing any combination of modalities on the fly will be crucial for deploying robots in dynamic environments. Foundations models trained with various data, from any sensors embedded in a common feature space, and fine-tuned with RL, show a promising avenue to bridge the remaining gap. More research remains to be done to determine whether these models can generalize and produce successful actions for unseen regions of these large spaces.

For this fourth axis, we also define three levels. The first level is when the observation space contains only one modality. The second level considers policies with an observation space of multiple modalities. Finally, the final level is when the observation space can switch from one set of modes to another.

Modality of the action space.

The fifth axis represents the modalities of the action space. The components concerned are $\mathcal{A}$. Finally, robot policies must handle diverse action spaces to accommodate multiple robots with different capabilities, e.g., when a CUAS can be equipped with different effectors. Traditional methods often train models for one robot, which limits their applicability to others. Cross-embodiment learning focuses on the ability of a single policy to perform across different robots. Nowadays, it is usually done by determining a high-level action space common to all robots and using an abstraction layer. Afterwards, it might be, for example, a control-theory-based controller that converts this high-level action to low-level commands for the actuators. Recent work has demonstrated that foundation models can benefit from data collected in various embodiments [84]. However, using high-level actions can limit the robot's capabilities because it does not directly control the robot's actuators [65]. Extending learning frameworks to handle heterogeneous action spaces and directly control actuators without such abstractions presents an exciting avenue for future research [85].

Based on this observation, we distinguish three levels for this last axis. The first level is when the agent has only one modality, i.e. it can only control one specific robot. The second represents agents that operate different robots with a unique action space. The third is composed of policies that can switch the action space for any low-level robot controller.

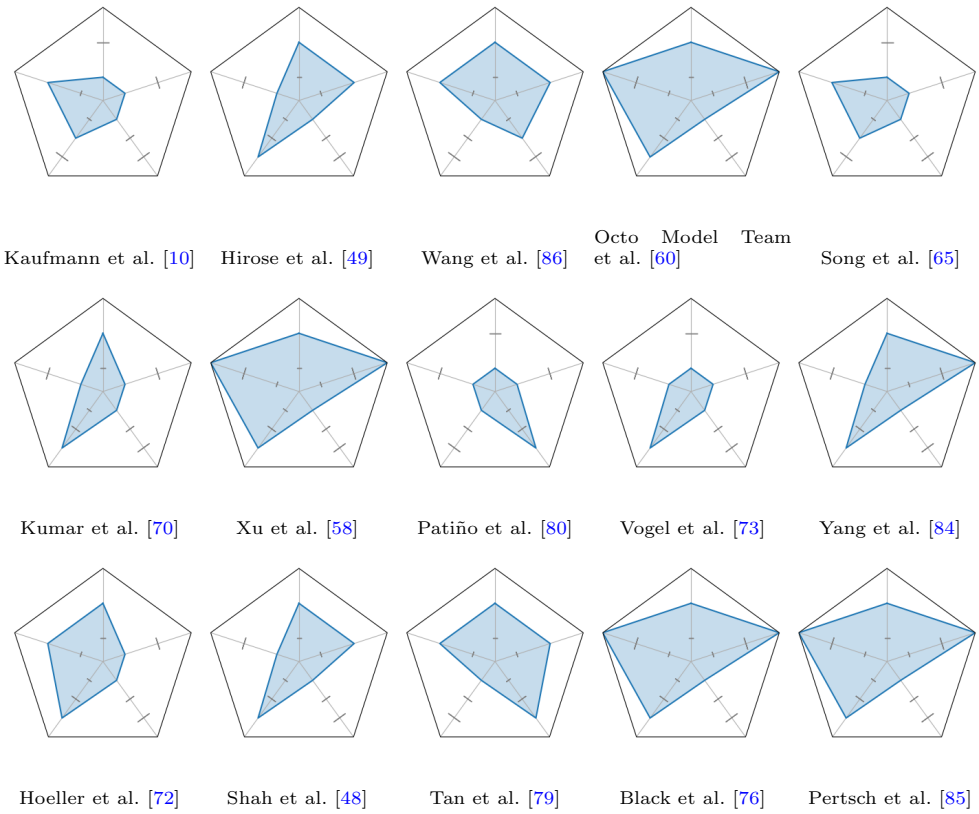

Fig. 3: An overview of the levels in each axis of multiple state-of-the-art articles in robotics. The axes are arranged as follows: number of tasks at the top, modality of the observation space at the top-left, diversity of deployed conditions at the bottom-left, degree of multi-agent interaction at the bottom-right, modality of the action space at the top-right.

5.2 Relations between axes

While significant progress has been made in advancing individual axes of complexity, achieving high levels across all five simultaneously remains challenging. Figure 3 represents recent papers to which we referred earlier in this section on radar charts to visualize the correlation between the different axes.

This highlights global trends in the robotics field. Robots capable of handling multiple tasks, diverse observation modalities, and heterogeneous action spaces have gained attention. These developments represent simultaneous advancements in three out of the five axes, facilitated primarily by the emergence of foundation models. Foundation models demonstrate promising results in zero-shot and fine-tuning evaluations, though their performance is highly dependent on the quantity and quality of

the training data. In fact, the agents replicate expert demonstrations since these models are trained primarily with imitation learning. A key future direction is to improve their capabilities by supplying more diverse data, and RL offers significant potential in this regard. Indeed, RL can improve the performance of expert demonstrations using a reward function. RL has already been employed to enhance foundation models in language processing, addressing the challenges posed by the lack of data [87]. In addition, it remains to be seen whether foundation models could be applied in multi-agent environments. The computational and memory constraints of embedded devices likely require the optimization of these big models to ensure real-time operation without compromising performance.

An additional observation is that when the levels of the diversity of the deployed conditions increase, it reduces the advancement in the other axes. Successful deployed policies in many conditions often struggle on the other axes but are still gradually emerging. For instance, Black et al. [76] has deployed robots that handle a set of tasks using multiple modes as inputs for different robots.

While single-agent tasks have seen advancements simultaneously across multiple axes, extending these models to environments with multiple interacting agents, whether cooperative, competitive, or mixed, is largely unexplored.

6 Roadmap toward the future of drone warfare

This section presents several scenarios of increasing complexity that define a roadmap for the future of drone warfare. These scenarios are analyzed through the previously introduced radar charts, allowing us to identify their respective level of complexity for each axis. This represents our vision to lead the innovation towards augmenting intelligence in warfare.

Scenario 1: Autonomous UAS and CUAS.

In the future, drones will be autonomous and might be controlled by RL controllers [65]. In this case, since drones will not communicate with a pilot, jamming and spoofing could become obsolete. Hence, EW countermeasures that disrupt such communications may become less relevant in future CUAS developments. On the physical countermeasures side, AI-driven drones take full advantage of the drone's capabilities and, therefore, decrease the performance and cost-efficiency of laser, defense drones, or anti-aircraft guns, which already struggle to handle drones piloted by humans. One solution is to also develop RL controllers to achieve superhuman performance in controlling these CUAS.

In this first scenario, the agents have to succeed in a particular task with a fixed observation and action space. Therefore, it involves solving a POMDP and deploying the policy in any conditions, such as night or windy conditions, which means reaching the edge of the first axis. We believe that if the axis related to the diversity of deployed conditions is not mastered, a drone could be useless on the battlefield. This axis is mandatory to ensure the robustness and reliability of autonomous systems, either UAS or CUAS, and is crucial for real-world deployment. The levels of complexity of this scenario are represented in Figure 4.

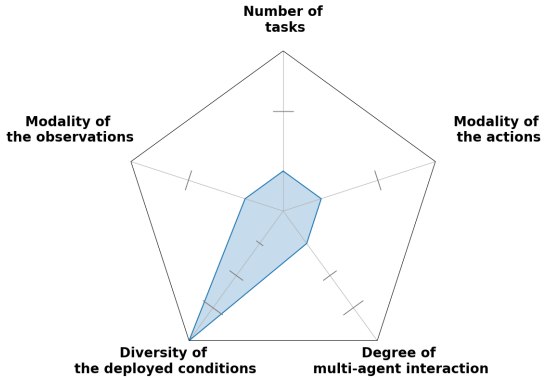

Fig. 4: Levels of Scenario 1.

Scenario 2: Autonomous competitive or cooperative UAS and CUAS swarms.

As drone autonomy increases, human pilots will transition from controlling individual FPV drones to managing swarms of autonomous systems. In such a setting, both UAS and CUAS systems must operate in environments where the presence of other autonomous intelligent agents cannot be ignored. Traditional CUAS training approaches often assume a stationary opponent and do not account for the evolving strategies of adversaries. To address this non-stationarity, it becomes necessary to train effectors in a multi-agent setting, where competition drives adaptation and robustness.

Furthermore, these autonomous systems might need to collaborate in real time to defend against coordinated drone swarms. This requires not only effective policy learning for cooperation and competition, but also mechanisms for decentralized coordination. While current defense systems rely heavily on hierarchical command and control to assign tasks to effectors, such centralized processes can introduce computational delays. In future scenarios, decentralized cooperation may become essential, not just for scalability but also for real-time effectiveness against swarms. Deploying agents that can cooperate or compete effectively is already a big challenge and deserves a specific step on the roadmap. The levels of complexity of this scenario are represented in Figure 5.

Scenario 3: Autonomous mixed competitive-cooperative UAS and CUAS swarms.

Building on the multi-agent interactions introduced in Scenario 2, this scenario generalizes the problem to environments where agents must simultaneously cooperate with allies and compete against adversaries. A key challenge lies in the variable composition of these teams. The emergence and stability of cooperative or adversarial strategies can be significantly affected by fluctuations in the number of agents, allies, or enemies. Algorithms must therefore be robust not only to adversarial adaptation but also to

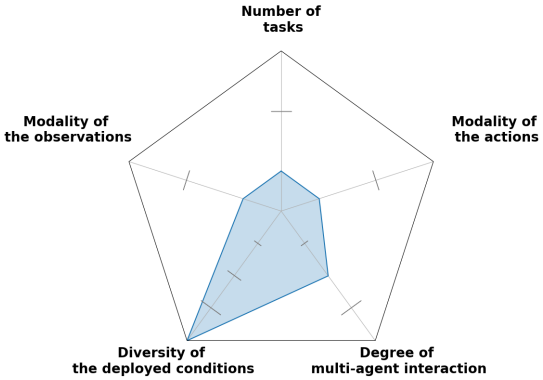

Fig. 5: Levels of Scenario 2.

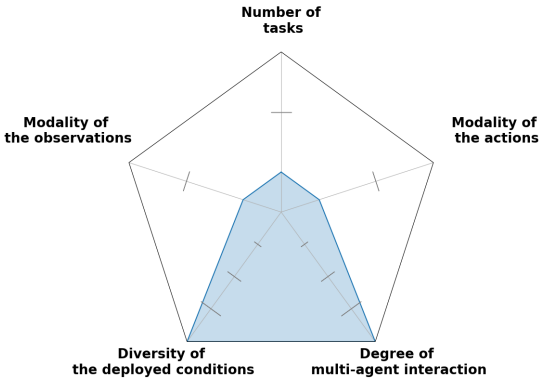

Fig. 6: Levels of Scenario 3.

changes in team scales. Compared to Scenario 2, the deployment of effective policies in this setting is more complex. It requires learning across a broader strategy space. The increasing level of complexity of this scenario is illustrated in Figure 6.

Scenario 4: Multi-sensors multi-effectors autonomous UAS and CUAS swarms.

UAS and CUAS can be equipped with different sensors to collect different types of information or improve their capabilities, e.g., a drone with an infrared camera to navigate at night instead of an RGB camera during daylight. On the effector side, drones are increasingly weaponized to tackle more missions, and must therefore adapt their strategy based on the effector equipped. It is indeed possible to train a single policy for each pair of sensors/effectors, but future controllers must be able to adapt to the available sensors and make decisions depending on the system's capabilities.

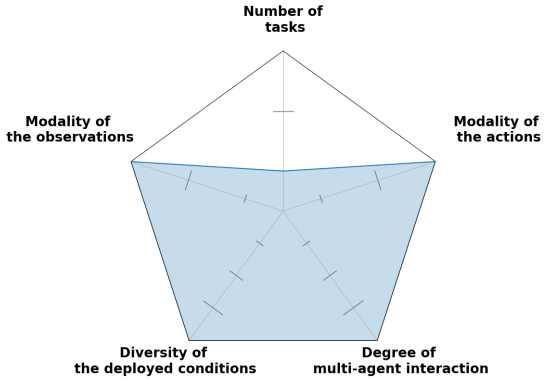

Fig. 7: Levels of Scenario 4.

Consequently, the selection of the effector could be learned to handle the threat because the agents have learned which agent should act with which effector in a particular scenario. The levels of complexity of this scenario are represented in Figure 7.

Scenario 5: Multi-tasks multi-sensors multi-effectors autonomous UAS and CUAS swarms.

In practice, it would be convenient to change the task of UAS and CUAS without retraining agents. Indeed, previous scenarios involve training agents for a specific task. In the future, a drone swarm commander will decide the task, and this task will condition the drone controllers. The difficulties lie in handling any task given by the commander. Additionally, this task can be defined in natural language or voice command, which facilitates the interaction of the commander with their swarm, facilitating the human-machine interface. This scenario would be practical for future warfare. Future CUAS and UAS could reach the last level for each axis of complexity of the radar chart by leveraging a multi-agent configuration that can act effectively in any condition for any threat using any sensor and any end-effector. The levels of complexity of this scenario are represented in Figure 8.

7 Conclusion

In conclusion, this article presents an overview of drone warfare and reinforcement learning in robotics. It enumerates the gaps between the state-of-the-art in RL and desired solutions for future drone warfare. It finally proposes a roadmap, through different scenarios, for transforming drone warfare through AI innovation, ensuring these systems are prepared to meet the evolving demands of modern combat.

8 Ethical considerations

Although the transition from helping humans make decisions to fully autonomous control of UAS promises to reduce human risk and accelerate decision-making on the

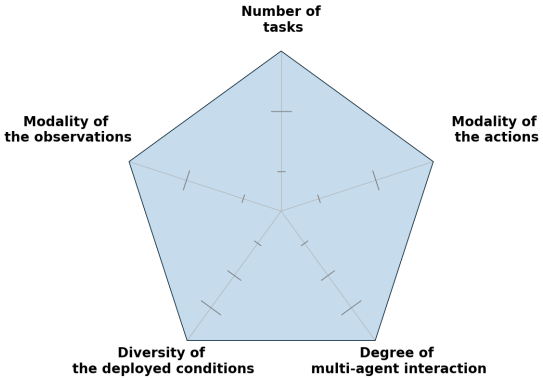

Fig. 8: Levels of Scenario 5.

battlefield, it simultaneously poses several ethical challenges. Automating the control of weaponized systems prone to being misled by adversarial perturbations in sensor inputs or jamming strategies can lead to unintended outcomes and raise questions of accountability. When an AI-driven drone makes a mistake, it is unclear whether the manufacturer, programmer, or commanding officer bears responsibility. To safeguard against these dangers, deploying such algorithms requires rigorous explainability studies, human in the loop mechanisms, and harmonized legal frameworks that embed laws directly into verifiable code. In their absence, the technologies designed to protect may instead exacerbate instability and undermine the moral foundations of armed conflict.

References

[1] Kunertova, D.: Drones have boots: Learning from russia's war in ukraine. Contemporary Security Policy (2023)

[2] Konaev, M.: Tomorrow's technology in today's war: The use of AI and autonomous technologies in the war in ukraine and implications for strategic stability. Technical report, CNA, Arlington, VA (2023)

[3] Kunertova, D.: The war in ukraine shows the game-changing effect of drones depends on the game. Bulletin of the Atomic Scientists (2023)

[4] Koukoudakis, G.: Drones' contribution to the transformation of contemporary warfare. Journal of Military Studies (2024)

[5] Mariano Zafra, A.R. Max Hunder, Kiyada, S.: How drone combat in ukraine is changing warfare. Reuters (2024)

[6] Petrovski, A., Radovanović, M., Behlić, A.: Application of drones with artificial intelligence for military purposes. In: 10th International Scientific Conference on

Defensive Technologies (OTEH) (2022)

[7] Bender, H., Kanderske, M.: Consumer drone warfare: Practices, aesthetics and discourses of consumer drones in the Russo-Ukrainian War. Springer, Cham (2024)

[8] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. nature (2019)

[9] Perolat, J., De Vylder, B., Hennes, D., Tarassov, E., Strub, F., Boer, V., Muller, P., Connor, J.T., Burch, N., Anthony, T., et al.: Mastering the game of stratego with model-free multiagent reinforcement learning. Science (2022)

[10] Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., Scaramuzza, D.: Champion-level drone racing using deep reinforcement learning. Nature (2023)

[11] Jang, K., Lichtlé, N., Vinitsky, E., Shah, A., Bunting, M., Nice, M., Piccoli, B., Seibold, B., Work, D.B., Monache, M.L.D., et al.: Reinforcement learning based oscillation dampening: Scaling up single-agent rl algorithms to a 100 av highway field operational test. arXiv:2402.17050 (2024)

[12] Ben, A.: Long read: Ukraine is losing the drone war. Intellinews (2024)

[13] Strategic Studies, T.E.C., Research: First person view drones. ECSSR (2024)

[14] Oliver Parken, T.R.: Chinese soldiers train to fend off fpv drones. The Warzone (2024)

[15] Karadeniz, G., Özcan, A., Bayram, M., İnce, G.: Drone wars 3d: A game-based simulation platform for testing aerial defence strategies against drone swarms. Journal of Aeronautics and Space Technologies (2024)

[16] Silva, D.L.D., Machado, R., Coutinho, O.L., Antreich, F.: A soft-kill reinforcement learning counter unmanned aerial system (c-uas) with accelerated training. IEEE Access (2023)

[17] He, D., Chan, S., Guizani, M.: Communication security of unmanned aerial vehicles. IEEE Wireless Communications (2017)

[18] Dhomane, P., Mathew, R.: Counter-measures to spoofing and jamming of drone signals. Available at SSRN 3774955 (2020)

[19] Holding, R.C.: Red cat successfully passes field testing with doodle labs against electronic warfare technology in ukraine. Indicate Media (2024)

[20] Kumari, N., Lee, K., Barca, J.C., Ranaweera, C.: Towards reliable identification

and tracking of drones within a swarm. Journal of Intelligent & Robotic Systems (2024)

[21] Pope, A.P., Ide, J.S., Micovic, D., Diaz, H., Rosenbluth, D., Ritholtz, L., Twedt, J.C., Walker, T.T., Alcedo, K., Javorsek, D.: Hierarchical Reinforcement Learning for Air-to-Air Combat (2021)

[22] Russell, S.J., Norvig, P.: Artificial Intelligence: a Modern Approach. Pearson, London, England (2016)

[23] Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence (1998)

[24] Shapley, L.S.: Stochastic games. Proceedings of the national academy of sciences (1953)

[25] Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT press, Cambridge (2018)

[26] Rengarajan, D., Vaidya, G., Sarvesh, A., Kalathil, D., Shakkottai, S.: Reinforcement learning with sparse rewards using guidance from offline demonstration. arXiv:2202.04628 (2022)

[27] Trott, A., Zheng, S., Xiong, C., Socher, R.: Keeping your distance: Solving sparse reward tasks using self-balancing shaped rewards. Advances in Neural Information Processing Systems (2019)

[28] Felten, F., Alegre, L.N., Nowe, A., Bazzan, A., Talbi, E.G., Danoy, G., Silva, B.: A toolkit for reliable benchmarking and research in multi-objective reinforcement learning. Advances in Neural Information Processing Systems (2023)

[29] Ma, Y.J., Liang, W., Wang, G., Huang, D.-A., Bastani, O., Jayaraman, D., Zhu, Y., Fan, L., Anandkumar, A.: Eureka: Human-level reward design via coding large language models. In: The Twelfth International Conference on Learning Representations (2024)

[30] Yu, W., Gileadi, N., Fu, C., Kirmani, S., Lee, K.-H., Gonzalez Arenas, M., Lewis Chiang, H.-T., Erez, T., Hasenclever, L., Humplik, J., Ichter, B., Xiao, T., Xu, P., Zeng, A., Zhang, T., Heess, N., Sadigh, D., Tan, J., Tassa, Y., Xia, F.: Language to rewards for robotic skill synthesis. Conference of Robot Learning 2023 (2023)

[31] Kwon, M., Xie, S.M., Bullard, K., Sadigh, D.: Reward design with language models. In: The Eleventh International Conference on Learning Representations (2023)

[32] Yu, C., Tan, Q., Lu, H., Gao, J., Yang, X., Wang, Y., Wu, Y., Vinitsky, E.: Icpl:

Few-shot in-context preference learning via llms. arXiv:2410.17233 (2025)

[33] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv:1707.06347 (2017)

[34] Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: International Conference on Machine Learning, pp. 1861–1870 (2018)

[35] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature (2015)

[36] Wierstra, D., Förster, A., Peters, J., Schmidhuber, J.: Recurrent policy gradients. Logic Journal of IGPL (2010)

[37] Zhu, P., Li, X., Poupart, P., Miao, G.: On improving deep reinforcement learning for pomdps. arXiv:1704.07978 (2017)

[38] Zhang, M., McCarthy, Z., Finn, C., Levine, S., Abbeel, P.: Learning deep neural network policies with continuous memory states. In: IEEE International Conference on Robotics and Automation (ICRA) (2016)

[39] Nash, J.F.: Non-cooperative games. In: The Foundations of Price Theory Vol 4, pp. 329–340. Routledge, London, England (2024)

[40] Albrecht, S.V., Christianos, F., Schäfer, L.: Multi-Agent Reinforcement Learning: Foundations and Modern Approaches. MIT Press, Cambridge (2024)

[41] Elo, A.E.: The Rating of Chessplayers, Past and Present. Ishi press international, New York (1978)

[42] Jaderberg, M., Czarnecki, W.M., Dunning, I., Marris, L., Lever, G., Castaneda, A.G., Beattie, C., Rabinowitz, N.C., Morcos, A.S., Ruderman, A., et al.: Human-level performance in 3d multiplayer games with population-based reinforcement learning. Science (2019)

[43] Leroy, P., Pisane, J., Ernst, D.: Value-based CTDE methods in symmetric two-team markov game: from cooperation to team competition. In: Deep Reinforcement Learning Workshop NeurIPS (2022)

[44] Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Perolat, J., Silver, D., Graepel, T.: A unified game-theoretic approach to multiagent reinforcement learning. In: Advances in Neural Information Processing Systems (2017)

[45] Muller, P., Omidshafiei, S., Rowland, M., Tuyls, K., Perolat, J., Liu, S., Hennes, D., Marris, L., Lanctot, M., Hughes, E., Wang, Z., Lever, G., Heess, N., Graepel, T., Munos, R.: A generalized training approach for multiagent learning. In: International Conference on Learning Representations (2020)

[46] Reiter, R., Hoffmann, J., Reinhardt, D., Messerer, F., BaumgÃĪrtner, K., Sawant, S., Boedecker, J., Diehl, M., Gros, S.: Synthesis of model predictive control and reinforcement learning: Survey and classification. arXiv:2502.02133 (2025)

[47] Romero, A., Aljalbout, E., Song, Y., Scaramuzza, D.: Actor-critic model predictive control: Differentiable optimization meets reinforcement learning. arXiv:2306.09852 (2025)

[48] Shah, D., Sridhar, A., Dashora, N., Stachowicz, K., Black, K., Hirose, N., Levine, S.: ViNT: A foundation model for visual navigation. In: 7th Annual Conference on Robot Learning (2023)

[49] Hirose, N., Glossop, C., Sridhar, A., Shah, D., Mees, O., Levine, S.: Lelan: Learning a language-conditioned navigation policy from in-the-wild video. In: Conference on Robot Learning (2024)

[50] Smith, L., Kostrikov, I., Levine, S.: A walk in the park: Learning to walk in 20 minutes with model-free reinforcement learning. arXiv:2208.07860 (2022)

[51] Walke, H.R., Yang, J.H., Yu, A., Kumar, A., Orbik, J., Singh, A., Levine, S.: Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning. In: Liu, K., Kulic, D., Ichnowski, J. (eds.) Proceedings of The 6th Conference on Robot Learning. Proceedings of Machine Learning Research. PMLR, New York (2023)

[52] Chebotar, Y., Vuong, Q., Hausman, K., Xia, F., Lu, Y., Irpan, A., Kumar, A., Yu, T., Herzog, A., Pertsch, K., Gopalakrishnan, K., Ibarz, J., Nachum, O., Sontakke, S.A., Salazar, G., Tran, H.T., Peralta, J., Tan, C., Manjunath, D., Singh, J., Zitkovich, B., Jackson, T., Rao, K., Finn, C., Levine, S.: Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In: Tan, J., Toussaint, M., Darvish, K. (eds.) Proceedings of The 7th Conference on Robot Learning. Proceedings of Machine Learning Research. PMLR, New York (2023)

[53] Sadeghi, F., Levine, S.: Cad2rl: Real single-image flight without a single real image. In: Proceedings of Robotics: Science and Systems, Cambridge, Massachusetts (2017)

[54] Mittal, M., Yu, C., Yu, Q., Liu, J., Rudin, N., Hoeller, D., Yuan, J.L., Singh, R., Guo, Y., Mazhar, H., Mandlekar, A., Babich, B., State, G., Hutter, M., Garg, A.: Orbit: A unified simulation framework for interactive robot learning environments. IEEE Robotics and Automation Letters (2023)

[55] Coumans, E., Bai, Y.: PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016–2021)

[56] Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2012). IEEE

[57] Collaboration, E., O'Neill, A., Rehman, A., Gupta, A., Maddukuri, A., Gupta, A., Padalkar, A., Lee, A., Pooley, A., Gupta, A., Mandlekar, A., Jain, A., Tung, A., Bewley, A., Herzog, A., Irpan, A., Khazatsky, A., Rai, A., Gupta, A., Wang, A., Kolobov, A., Singh, A., Garg, A., Kembhavi, A., Xie, A., Brohan, A., Raffin, A., Sharma, A., Yavary, A., Jain, A., Balakrishna, A., Wahid, A., Burgess-Limerick, B., Kim, B., Schölkopf, B., Wulfe, B., Ichter, B., Lu, C., Xu, C., Le, C., Finn, C., Wang, C., Xu, C., Chi, C., Huang, C., Chan, C., Agia, C., Pan, C., Fu, C., Devin, C., Xu, D., Morton, D., Driess, D., Chen, D., Pathak, D., Shah, D., Büchler, D., Jayaraman, D., Kalashnikov, D., Sadigh, D., Johns, E., Foster, E., Liu, F., Ceola, F., Xia, F., Zhao, F., Frujeri, F.V., Stulp, F., Zhou, G., Sukhatme, G.S., Salhotra, G., Yan, G., Feng, G., Schiavi, G., Berseth, G., Kahn, G., Yang, G., Wang, G., Su, H., Fang, H.-S., Shi, H., Bao, H., Amor, H.B., Christensen, H.I., Furuta, H., Bharadhwaj, H., Walke, H., Fang, H., Ha, H., Mordatch, I., Radosavovic, I., Leal, I., Liang, J., Abou-Chakra, J., Kim, J., Drake, J., Peters, J., Schneider, J., Hsu, J., Vakil, J., Bohg, J., Bingham, J., Wu, J., Gao, J., Hu, J., Wu, J., Wu, J., Sun, J., Luo, J., Gu, J., Tan, J., Oh, J., Wu, J., Lu, J., Yang, J., Malik, J., Silvério, J., Hejna, J., Booher, J., Tompson, J., Yang, J., Salvador, J., Lim, J.J., Han, J., Wang, K., Rao, K., Pertsch, K., Hausman, K., Go, K., Gopalakrishnan, K., Goldberg, K., Byrne, K., Oslund, K., Kawaharazuka, K., Black, K., Lin, K., Zhang, K., Ehsani, K., Lekkala, K., Ellis, K., Rana, K., Srinivasan, K., Fang, K., Singh, K.P., Zeng, K.-H., Hatch, K., Hsu, K., Itti, L., Chen, L.Y., Pinto, L., Fei-Fei, L., Tan, L., Fan, L.J., Ott, L., Lee, L., Weihs, L., Chen, M., Lepert, M., Memmel, M., Tomizuka, M., Itkina, M., Castro, M.G., Spero, M., Du, M., Ahn, M., Yip, M.C., Zhang, M., Ding, M., Heo, M., Srirama, M.K., Sharma, M., Kim, M.J., Kanazawa, N., Hansen, N., Heess, N., Joshi, N.J., Suenderhauf, N., Liu, N., Palo, N.D., Shafiullah, N.M.M., Mees, O., Kroemer, O., Bastani, O., Sanketi, P.R., Miller, P.T., Yin, P., Wohlhart, P., Xu, P., Fagan, P.D., Mitrano, P., Sermanet, P., Abbeel, P., Sundaresan, P., Chen, Q., Vuong, Q., Rafailov, R., Tian, R., Doshi, R., Mart'in-Mart'in, R., Baijal, R., Scalise, R., Hendrix, R., Lin, R., Qian, R., Zhang, R., Mendonca, R., Shah, R., Hoque, R., Julian, R., Bustamante, S., Kirmani, S., Levine, S., Lin, S., Moore, S., Bahl, S., Dass, S., Sonawani, S., Tulsiani, S., Song, S., Xu, S., Haldar, S., Karamcheti, S., Adebola, S., Guist, S., Nasiriany, S., Schaal, S., Welker, S., Tian, S., Ramamoorthy, S., Dasari, S., Belkhale, S., Park, S., Nair, S., Mirchandani, S., Osa, T., Gupta, T., Harada, T., Matsushima, T., Xiao, T., Kollar, T., Yu, T., Ding, T., Davchev, T., Zhao, T.Z., Armstrong, T., Darrell, T., Chung, T., Jain, V., Kumar, V., Vanhoucke, V., Zhan, W., Zhou, W., Burgard, W., Chen, X., Chen, X., Wang, X., Zhu, X., Geng, X., Liu, X., Liangwei, X., Li, X., Pang, Y., Lu, Y., Ma, Y.J., Kim, Y., Chebotar, Y., Zhou, Y., Zhu, Y., Wu, Y.,

Xu, Y., Wang, Y., Bisk, Y., Dou, Y., Cho, Y., Lee, Y., Cui, Y., Cao, Y., Wu, Y.-H., Tang, Y., Zhu, Y., Zhang, Y., Jiang, Y., Li, Y., Li, Y., Iwasawa, Y., Matsuo, Y., Ma, Z., Xu, Z., Cui, Z.J., Zhang, Z., Fu, Z., Lin, Z.: Open x-embodiment: Robotic learning datasets and rt-x models. arXiv:2310.08864 (2024)

[58] Xu, C., Li, Q., Luo, J., Levine, S.: Rldg: Robotic generalist policy distillation via reinforcement learning. arXiv:2412.09858 (2024)

[59] Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv:2005.01643 (2020)

[60] Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Xu, C., Luo, J., Kreiman, T., Tan, Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., Levine, S.: Octo: An open-source generalist robot policy. In: Proceedings of Robotics: Science and Systems, Delft, Netherlands (2024)

[61] Yang, S., Nachum, O., Du, Y., Wei, J., Abbeel, P., Schuurmans, D.: Foundation models for decision making: Problems, methods, and opportunities. arXiv:2303.04129 (2023)

[62] Tang, C., Abbatematteo, B., Hu, J., Chandra, R., Martín-Martín, R., Stone, P.: Deep reinforcement learning for robotics: A survey of real-world successes. In: Proceedings of the AAAI Conference on Artificial Intelligence (2025)

[63] Johannink, T., Bahl, S., Nair, A., Luo, J., Kumar, A., Loskyll, M., Ojea, J.A., Solowjow, E., Levine, S.: Residual reinforcement learning for robot control. In: 2019 International Conference on Robotics and Automation (ICRA) (2019)

[64] Wang, D., Gao, N., Liu, D., Li, J., Lewis, F.L.: Recent progress in reinforcement learning and adaptive dynamic programming for advanced control applications. IEEE/CAA Journal of Automatica Sinica (2024)

[65] Song, Y., Romero, A., Müller, M., Koltun, V., Scaramuzza, D.: Reaching the limit in autonomous racing: Optimal control versus reinforcement learning. Science Robotics (2023)

[66] Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2017)

[67] Schott, L., Delas, J., Hajri, H., Gherbi, E., Yaich, R., Boulahia-Cuppens, N., Cuppens, F., Lamprier, S.: Robust deep reinforcement learning through adversarial attacks and training: A survey. arXiv:2403.00420 (2024)

[68] Gleave, A., Dennis, M., Wild, C., Kant, N., Levine, S., Russell, S.: Adversarial policies: Attacking deep reinforcement learning. arXiv:1905.10615 (2019)

[69] Abouelyazid, M.: Adversarial deep reinforcement learning to mitigate sensor and communication attacks for secure swarm robotics. Journal of Intelligent Connectivity and Emerging Technologies (2023)

[70] Kumar, A., Fu, Z., Pathak, D., Malik, J.: Rma: Rapid motor adaptation for legged robots. arXiv:2107.04034 (2021)

[71] Han, D., Mulyana, B., Stankovic, V., Cheng, S.: A survey on deep reinforcement learning algorithms for robotic manipulation. Sensors (2023)

[72] Hoeller, D., Rudin, N., Sako, D., Hutter, M.: Anymal parkour: Learning agile navigation for quadrupedal robots. Science Robotics (2024)

[73] Vogel, D., Baines, R., Church, J., Lotzer, J., Werner, K., Hutter, M.: Robust ladder climbing with a quadrupedal robot. arXiv:2409.17731 (2024)

[74] Quartey, B., Rosen, E., Tellex, S., Konidaris, G.: Verifiably following complex robot instructions with foundation models. arXiv:2402.11498 (2024)

[75] Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., Zhu, Y., Song, S., Kapoor, A., Hausman, K., Ichter, B., Driess, D., Wu, J., Lu, C., Schwager, M.: Foundation models in robotics: Applications, challenges, and the future. arXiv:2312.07843 (2023)

[76] Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: π_0: A vision-language-action flow model for general robot control. arXiv:2410.24164 (2024)

[77] Orr, J., Dutta, A.: Multi-agent deep reinforcement learning for multi-robot applications: A survey. Sensors (2023)

[78] Wang, Y., Damani, M., Wang, P., Cao, Y., Sartoretti, G.: Distributed reinforcement learning for robot teams: A review. Current Robotics Reports (2022)

[79] Tan, A.H., Bejarano, F.P., Zhu, Y., Ren, R., Nejat, G.: Deep reinforcement learning for decentralized multi-robot exploration with macro actions. IEEE Robotics and Automation Letters (2022)

[80] Patiño, D., Mayya, S., Calderon, J., Daniilidis, K., Saldaña, D.: Learning to navigate in turbulent flows with aerial robot swarms: A cooperative deep reinforcement learning approach. IEEE Robotics and Automation Letters (2023)

[81] Samvelyan, M., Rashid, T., Witt, C., Farquhar, G., Nardelli, N., Rudner, T.G.J., Hung, C.-M., Torr, P.H.S., Foerster, J., Whiteson, S.: The starcraft multi-agent challenge. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (2019)

[82] Baker, B., Kanitscheider, I., Markov, T., Wu, Y., Powell, G., McGrew, B., Mordatch, I.: Emergent tool use from multi-agent autocurricula. In: International Conference on Learning Representations (2019)

[83] Rahman, A., Carlucho, I., Höpner, N., Albrecht, S.V.: A general learning framework for open ad hoc teamwork using graph-based policy learning. Journal of Machine Learning Research (2023)

[84] Yang, J., Glossop, C., Bhorkar, A., Shah, D., Vuong, Q., Finn, C., Sadigh, D., Levine, S.: Pushing the limits of cross-embodiment learning for manipulation and navigation. arXiv:2402.19432 (2024)

[85] Pertsch, K., Stachowicz, K., Ichter, B., Driess, D., Nair, S., Vuong, Q., Mees, O., Finn, C., Levine, S.: Fast: Efficient action tokenization for vision-language-action models. arXiv:2501.09747 (2025)

[86] Wang, R., Lyu, M., Zhang, J.: A multi-robot collaborative exploration method based on deep reinforcement learning and knowledge distillation. Mathematics (2025)

[87] OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H.W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S.P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S.S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Kaiser, Kamali, A., Kanitscheider, I., Keskar, N.S., Khan, T., Kilpatrick, L., Kim, J.W., Kim, C., Kim, Y., Kirchner, J.H., Kiros, J., Knight, M., Kokotajlo, D., Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C.M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S.M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L.,

O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., Avila Belbute Peres, F., Petrov, M., Oliveira Pinto, H.P., Michael, Pokorny, Pokrass, M., Pong, V.H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F.P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M.B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J.F.C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J.J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., Zoph, B.: Gpt-4 technical report. arXiv:2303.08774 (2024)