drones

MDPI

Article

A Survey on UAV Control with Multi-Agent Reinforcement Learning

Chijioke C. Ekechi [1], Tarek Elfouly [1,*], Ali Alouani [1] and Tamer Khattab [2]

[1] Department of Electrical and Computer Engineering, College of Engineering Tennessee Technological University, 115 West 10th Street Campus Box 5004 Brown Hall 210, Cookeville, TN 38505, USA; ccekechi42@tntech.edu (C.C.E.); aalouani@tntech.edu (A.A.)

[2] Department of Electrical Enginerring, College of Engineering, Qatar University, Doha 2713, Qatar; tkhattab@qu.edu.qa

* Correspondence: telfouly@tntech.edu; Tel.: +1-931-372-3847

Abstract

Unmanned Aerial Vehicles (UAVs) have become increasingly prevalent in both governmental and civilian applications, offering significant reductions in operational costs by minimizing human involvement. There is a growing demand for autonomous, scalable, and intelligent coordination strategies in complex aerial missions involving multiple Unmanned Aerial Vehicles (UAVs). Traditional control techniques often fall short in dynamic, uncertain, or large-scale environments where decentralized decision-making and inter-agent cooperation are crucial. A potentially effective technique used for UAV fleet operation is Multi-Agent Reinforcement Learning (MARL). MARL offers a powerful framework for addressing these challenges by enabling UAVs to learn optimal behaviors through interaction with the environment and each other. Despite significant progress, the field remains fragmented, with a wide variety of algorithms, architectures, and evaluation metrics spread across domains. This survey aims to systematically review and categorize state-of-the-art MARL approaches applied to UAV control, identify prevailing trends and research gaps, and provide a structured foundation for future advancements in cooperative aerial robotics. The advantages and limitations of these techniques are discussed along with suggestions for further research to improve the effectiveness of MARL application to UAV fleet management.

Keywords: Unmanned Aerial Vehicles; multi-agent systems; Markov Decision Process; artificial intelligence; wireless communication

Academic Editors: Zhi Feng, Rui Yan, Yishi Liu, Xiaoduo Li and Qing Wang

Received: 5 June 2025
Revised: 6 July 2025
Accepted: 7 July 2025
Published: 9 July 2025

Citation: Ekechi, C.C.; Elfouly, T.; Alouani, A.; Khattab, T. A Survey on UAV Control with Multi-Agent Reinforcement Learning. *Drones* **2025**, *9*, 484. https://doi.org/10.3390/drones9070484

1. Introduction

Unmanned Aerial Vehicles (UAVs), commonly referred to as drones, have experienced a significant surge in popularity due to their versatility, cost-effectiveness, and ability to access hard-to-reach areas. In search-and-rescue (SAR) operations, UAVs provide real-time aerial surveillance, enabling fast response in challenging terrains such as forests, mountains, or disaster-stricken zones. Being equipped with thermal imaging and high-resolution cameras allows them to operate in low-visibility and obstructed environments. In the commercial sector, UAVs are revolutionizing delivery services by offering faster, contactless transportation of goods, especially in remote or urban areas where traditional delivery methods face logistical challenges. Additionally, in geographical mapping [1], UAVs enable the creation of highly accurate topographical maps and 3D models by capturing detailed aerial imagery [2] and LiDAR data, aiding in urban planning, agriculture [3],

and environmental monitoring [4]. UAVs also offer a viable alternative to conventional platforms for acquiring high-resolution remote sensing data [5]. In military operations, UAVs play a critical role in reconnaissance [6,7], surveillance [8,9], target acquisition [10], detecting mines [11] , and even precision strikes [12], offering strategic advantages while minimizing the risk to human personnel.

Traditionally, UAVs are controlled through methods such as remote piloting via radio frequency (RF) links, ground control stations (GCS) with dedicated software interfaces, and pre-programmed GPS waypoint navigation. In manual control modes, human operators rely on handheld transmitters or laptops connected to the drone to manage flight paths, altitude, and onboard sensors. The manual control mode often requires constant line of sight, stable communication channels, and skilled human operators, especially for complex maneuvers or in unpredictable environments. Traditional control methods present several challenges that can limit the effectiveness and scalability of UAV operations. RF communication is susceptible to signal interference, latency, and range limitations, particularly in urban or mountainous regions where signal obstruction is common. This can compromise the drone's responsiveness or even result in its loss. Ground control stations, while powerful, are often bulky and rely on stable network connections, making rapid deployment in disaster zones or battlefield conditions more difficult. Pre-programmed GPS navigation, though useful for repetitive tasks like mapping, lacks adaptability to real-time changes such as dynamic obstacles or weather conditions. In all cases, manual piloting requires trained personnel, which can be a bottleneck in emergency situations whenever fast deployment is critical. These challenges highlight the need for more intelligent, autonomous UAV systems that can operate reliably with minimal human intervention, adapt to changing environments, and maintain operational continuity even under adverse conditions.

The limitations of traditional UAV control methods have led researchers to integrate Machine Learning (ML) techniques to enhance drone autonomy, adaptability, and performance [13,14]. ML algorithms enable UAVs to interpret complex environmental data, make decisions in real time, and learn from previous experiences without requiring constant human input [15]. For instance, in search-and-rescue operations, ML-based image recognition can analyze live video feeds to detect human shapes or anomalies in terrain more accurately and faster than human operators [16]. In navigation, Machine Learning allows drones to dynamically adjust their flight paths in response to obstacles, changing weather, or GPS signal loss, outperforming rigid, pre-programmed routes [17]. Additionally, ML techniques used for optimal sensor fusion and predictive modeling can improve situational awareness by combining data from multiple onboard sensors (e.g., cameras, LiDAR, IMUs) to infer surroundings and anticipate future decisions, thereby reducing collision risks and enhancing stability [18–20].

The use of ML also reduces the reliance on high-bandwidth, low-latency communication channels by allowing the UAV to operate semi- or fully autonomously [21–24]. This is especially beneficial in remote areas or signal-compromised environments like dense forests, urban canyons, or disaster zones. Furthermore, Machine Learning can support fleet-level coordination or swarms [25], enabling multiple UAVs to collaborate, optimize coverage areas, and share learned experiences across missions. By embedding intelligence directly into the UAV's onboard systems, these approaches not only address the shortcomings of traditional control methods but also unlock new levels of operational efficiency, scalability, and resilience. As hardware capabilities continue to improve and ML algorithms become more sophisticated, their integration into UAV systems is becoming not just a technological advancement but a necessity for meeting the demands of increasingly complex aerial missions.

The methodology used to analyze the surveyed papers in this study was structured around a systematic and comparative framework to evaluate the effectiveness of Multi-Agent Reinforcement Learning (MARL) techniques in UAV control applications. First, a comprehensive collection of recent and relevant peer-reviewed papers was assembled from high-impact journals and conferences, focusing on those that implemented MARL for UAV coordination, navigation, exploration, target tracking, and area coverage. Each paper was then examined based on a set of predefined systematic performance metrics, including environment type (e.g., discrete, continuous, simulated, or real-world), training technique (centralized, decentralized, or hybrid), swarm size, success rate, scalability, fault tolerance, and convergence speed.

For consistency, quantitative values such as swarm size and success rate were extracted directly from experimental results or approximated when reported in graphical form. Qualitative attributes like scalability were categorized as high, moderate, or low based on the authors' claims, observed performance under varying conditions, and robustness to agent failure or environmental changes. When possible, convergence behavior was assessed through reported training curves or episode counts. This structured comparison enabled a cross-paper analysis, revealing trends, strengths, and limitations of existing approaches. The findings were tabulated and used to identify promising methods, gaps in current research, and areas requiring further investigation.

This paper is organized as follows: first, in Section 2, we present the different methods of drone control, followed by Section 3 on Reinforcement Learning (RL) and how it could be used in drone control. Section 4 presents Value-Based Multi-Agent Reinforcement Learning Techniques, followed by Policy-Based Multi-Agent Reinforcement Learning Techniques in Section 5. Multi-agent Federated Reinforcement Learning Techniques are presented in Section 6. A comparison between all three of the techniques is presented in Section 7. Open research problems and discussions are presented in Section 8 followed by the Conclusion in Section 9.

2. Methods of Drone Control

The primary purpose of Unmanned Aerial Vehicles (UAVs) is to enable commercial, cargo, and military aircraft to undertake missions without a human pilot onboard. UAVs can be operated individually or in groups and are categorized based on their level of autonomy, as shown in Figure 1: manually controlled, semi-automatic, and fully autonomous drones. Each category has distinct applications in today's world. In this discussion, we will explore various methods and examples of how drones can be controlled, highlighting the technological diversity and the range of functionalities that UAVs provide.

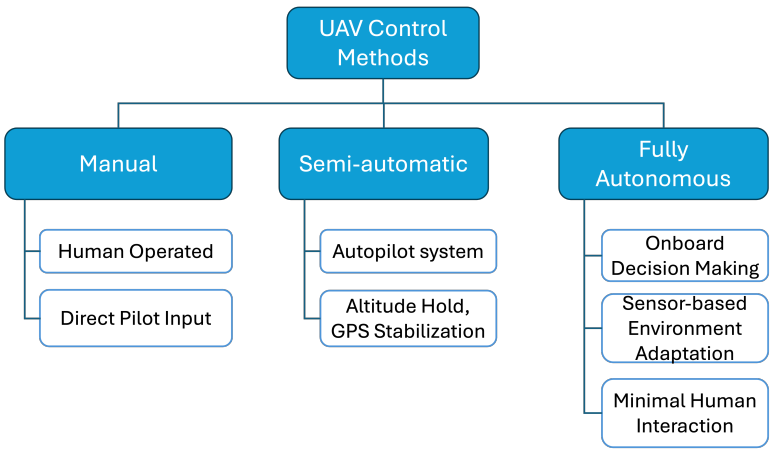

Figure 1. Conceptual diagram that summarizes the different UAV control methods.

2.1. Manually Controlled Drones

These UAVs are typically controlled entirely by human operators or require significant human intervention during operation. A common example of manual drone control involves using a programmed remote controller. Control signals are transmitted to UAVs via radio frequency antennas, Bluetooth technology, or Wi-Fi, depending on factors like distance, cost implications, and the quality and quantity of information to be exchanged between the drone and controller. While this approach grants operators full control over the UAVs, it is generally constrained by distance, and the controllers themselves become increasingly complex and challenging to design and implement depending on the required functionalities.

Manually controlled drones are well-suited for close-range tasks such as taking aerial photographs and conducting surveys. However, they are not effective when controlling long-range swarms; coordinating multiple drones simultaneously and accurately by several operators to avoid collisions is difficult. One potential solution is to control multiple UAVs from a single point using a single controller, but this approach complicates control as the number of UAVs increases and is not scalable.

These limitations prompted researchers, as discussed in [26], to develop a drone control system utilizing tactile feedback and impedance control, which responds to the operator's hand signals. This system is based on the principle of impedance matching. Impedance matching is a critical concept in radio frequency (RF) communication systems, which plays a significant role in ensuring effective long-range control of Unmanned Aerial Vehicles (UAVs). In the context of UAVs, impedance matching typically refers to aligning the impedance of the transmitter and the antenna to a common standard value—usually 50 ohms. This alignment is essential because any mismatch leads to signal reflection and loss of power, which degrades the signal quality and reduces the communication range. For UAVs operating over long distances, maintaining a high-quality RF link is crucial to ensure reliable command and control [27]. Impedance matching enables efficient and reliable RF communication [28], which is essential for UAV operators to maintain stable long-range control, especially in critical applications such as search and rescue, surveillance, or remote inspection missions. While impedance matching is essential for maximizing power transfer and communication efficiency in UAV systems, it introduces design challenges, sensitivity to environmental factors, and potential limitations in adaptability and system efficiency. It also poses practical and technical challenges in swarm configurations that require dynamic, adaptable, and interference-resilient communication systems. These significant drawbacks highlight the need for enhancements in UAV control technologies, leading to the development of semi-automatic drones. These UAVs offer several advantages over fully manual drones, particularly in terms of usability, efficiency, and reliability. These UAVs can perform certain tasks—such as takeoff, landing, hovering, and waypoint navigation—autonomously while still allowing the operator to intervene or guide key decisions. This reduces the cognitive load and training requirements for users, making drone operation more accessible and less error-prone [29]. Additionally, semi-automatic systems enhance mission consistency and accuracy, especially in repetitive or complex tasks like surveying, inspection, and search and rescue. They also improve safety by providing built-in fail-safe behaviors such as return-to-home and obstacle avoidance. Overall, semi-automatic drones strike a practical balance between full autonomy and manual control, offering more reliability and efficiency without completely removing human oversight, resulting in improved efficiency, safety, and data reliability across various applications.

2.2. Semi-Automatic Drones

These drones operate with a blend of human input and autonomous decision-making. The UAVs are equipped with built-in controllers but require occasional updates on operational parameters from the operator. An example of this approach is the L1 adaptive control model proposed in [30]. This control strategy adjusts the motion control parameters of a UAV based on relevant performance and robustness metrics predefined by the system designer. While this approach enables the UAV to adapt according to established metrics, it necessitates frequent updates of these metrics by the operator to accommodate changing environmental conditions and other external factors, introducing the potential for human error and delays in adjustments.

Semi-automatic UAVs are suitable for tasks that do not require constant modifications during operation. They are particularly effective in applications like image projection, where maintaining a stable position is crucial for generating holographic images. Additionally, they are well-suited for short-duration, repetitive tasks such as delivery services, where the complexity and frequency of operational adjustments are minimal. In [31], the use of a proportional differential (PD) controller for UAV control is detailed. A proportional-derivative (PD) compensator is used to help the drone approach the desired location. The PD gains are adjusted by the operator as needed. This method aims to ensure that the drones reach their intended positions as accurately as possible. While this system significantly narrows the steady state error, it still requires human intervention to adjust the proportional and differential gains based on the drone's position and accounting for the environmental conditions and drone dynamics. The overarching goal is to minimize human input, enhancing the autonomy of the UAVs while maintaining precise control over their operation. Reference [32] proposes a safe deployment of semi-autonomous drones in populated areas within visual line-of-sight of operators. It focuses on reducing the workload of human operators, enhancing mission efficiency, and ensuring safety through system-centric approaches. The proposed system integrates standard procedures to optimize predictability and simplicity. It includes a safety analysis to validate its reliability, and its effectiveness is demonstrated through a bridge inspection case study using simulations and scenario analysis. Its success lies in its strict predictability requirement, which enhances situational awareness and the use of the CAE block methodology to formally verify safety claims through deductive and inductive reasoning. This approach balances operational simplicity, safety, and reliability, making it both practical and effective. The research in [33] introduces a semi-autonomous drone swarm system for persistent airborne surveillance, addressing limitations of traditional methods like static cameras and helicopters. The proposed system uses a swarm mobility model optimized for maximum utility and energy efficiency, enabling drones to follow predefined trajectories without manual control. The drones stream video to a ground control station (GCS), where an object detection framework based on the YOLOv3 Real-Time Object Detection Algorithm [34] identifies objects of interest. The work of [35] describes a system and method for verifying aviation navigation signals using a semi-automated drone platform. This invention addresses the limitations of traditional flight inspection methods—which typically require manned aircraft—and proposes a cost-effective, safer, and more flexible alternative using Unmanned Aerial Vehicles (UAVs) by measuring the accuracy of signals transmitted by a localizer. It also determines the accuracy of signals transmitted by a glide slope station to perform a flight check of one or more navigational aid systems. Hutton [36] explores the use of augmented reality (AR) as a natural user interface for controlling task-oriented semi-automatic drones. The core idea is that while drone hardware has advanced rapidly, user interfaces—especially for non-expert users—remain intuitive and cognitively demanding. The research proposes leveraging AR to simplify drone control using intuitive gestures and visual overlays that reduce the

training burden and enhance usability. As the capabilities of semi-autonomous drones continue to evolve—enabling users to guide drone behavior through intuitive interfaces—the natural progression is toward full autonomy. While semi-autonomous systems reduce operator workload and increase usability, they still rely on human oversight for complex decision-making and task execution. Fully autonomous drones, on the other hand, are designed to operate independently, leveraging onboard sensors, artificial intelligence, and advanced path-planning algorithms to perform missions without real-time human input. Transitioning to full autonomy not only enables greater operational efficiency and scalability but also allows drones to function reliably in environments that are too dynamic, hazardous, or remote for continuous human supervision. This shift represents a critical step in unlocking the full potential of UAVs across applications such as logistics, surveillance, infrastructure inspection, and environmental monitoring.

2.3. Fully Autonomous Drones

Efforts to further reduce human intervention in UAV operations have led to the development of various methods aimed at enhancing machine intelligence and autonomy. Autonomous machines are designed to operate with no human input, typically requiring only the initial setup and final instructions. These systems are equipped to autonomously make decisions based on assigned tasks, guided by information provided by onboard sensors and mapping of the environment. The integration of Machine Learning (ML) techniques has significantly enhanced the intelligence of Unmanned Aerial Vehicles (UAVs), enabling them to make precise decisions during operations. Recent studies highlight the pivotal role of ML in advancing UAV capabilities across various domains. For instance, Rezwan and Choi [37] provides a comprehensive survey of artificial intelligence approaches for UAV navigation, emphasizing how ML algorithms facilitate autonomous decision-making in dynamic environments. Their work categorizes different AI methodologies, including optimization-based and learning-based approaches, that contribute to efficient UAV navigation. As a result, drones have become versatile tools, applicable to a broad range of activities including military [38], search-and-rescue missions [39], disaster recovery [40], security surveillance [41], and even unsupervised package delivery [42]. The work of [43] utilized a PID controller, a semi-automated control method for UAVs, and integrated a technique called fuzzy logic to minimize human intervention during operations. Fuzzy logic, a form of logical computation, uses degrees of truth rather than the binary true or false states typical of Boolean logic. In fuzzy logic, truths are expressed on a continuum from 0 to 1, allowing for more nuanced representations, such as depicting the likelihood of an event occurring as 0.9. This approach was employed to automate the tuning of the PID controller, dynamically adjusting the gains based on the fuzzy logic algorithm. While the method demonstrates improved disturbance reduction and robust trajectory tracking compared to conventional PID controllers, the fuzzy-logic-based control algorithm can be complex for sophisticated and complex applications, which require significant computational resources.

A more straightforward approach involves employing meta-heuristic Machine Learning algorithms, such as Swarm Intelligence and Genetic Algorithms. The authors in [44] investigated the use of the Particle Swarm Optimization Algorithm (PSO), a Swarm Intelligence technique, to manage the formation of a drone swarm. Simultaneously, they utilized a PID controller to maintain the stability of individual drones. While this method simplifies some aspects of control, it still relies on the traditional PID control technique, which works well only for linear systems. Additional human intervention is needed to accomplish system stability and accurate tracking. The researchers in [45] explored a combination of Particle Swarm Optimization (PSO) and the Genetic Algorithm (GA) for controlling both individual drones and their collective behavior in a swarm. The Genetic

Algorithm tailors individual control traits within the drones, whereas PSO governs the overall swarm behavior. Despite the sophistication of this combined approach, a significant drawback is the convergence issues of the meta-heuristic algorithms, which can affect the efficiency, stability, and reliability of the system. This challenge has prompted researchers to investigate alternative strategies for achieving full autonomy in UAV machines.

3. Reinforcement Learning

To possibly overcome the previously discussed limitations, researchers have turned to the use of Reinforcement Learning (RL), one of the three main paradigms of Machine Learning. Unlike data-training-based Machine Learning, RL uses learning through experience in which an agent learns to accomplish a goal by performing actions and receiving feedback in the form of rewards or penalties. The rewards inform the agent of the effectiveness of the action, influencing subsequent decisions [46]. The goal is to maximize the total reward over time. This approach is inspired by behavioral psychology and is commonly used in areas such as robotics, game playing, and autonomous systems, where learning from direct experience is essential. RL, particularly Deep Reinforcement Learning (DRL), has been instrumental in enabling Unmanned Aerial Vehicles (UAVs) to autonomously execute specific tasks with minimal human oversight. This advancement is evident across various applications, including navigation, obstacle avoidance, and mission planning [47,48]. However, a major ongoing challenge is managing a group of these UAVs, each tasked with individual and collective tasks, particularly under demanding conditions like high turbulence and severe storms. In [49], a DRL algorithm known as Proximal Policy Optimization (PPO) was implemented to control two drones equipped with distinct memory capabilities. Additionally, Curriculum Learning was employed to progressively train the drones to navigate environments ranging from those with few obstacles to others featuring multiple and complex obstacles. For example, in [50], the authors present a path planning method that combines goal-conditioned Reinforcement Learning (RL) with Curriculum Learning to enable Unmanned Aerial Vehicles (UAVs) to handle increasingly complex missions. The training begins with simple tasks, such as reaching a single target, and progressively advances to more challenging scenarios, including round-trip navigation involving multiple sub-goals. This structured approach allows the UAVs to build upon foundational skills, leading to improved performance in environments with varying levels of complexity. This method demonstrated remarkable improvements with a 92% success rate in relatively complex tasks and a 77% success rate in round-trip missions. However, the agent did not take the most efficient route to the goal in some cases. This efficiency limitation becomes more evident in complex missions involving multiple waypoints or round-trip scenarios.

The concept of RL is straightforward when applied to a single agent. In this setting, the agent interacts with an environment by making actions based on a policy, observes the impact of the actions, and receives rewards or penalties. The agent's goal is to learn an optimal policy that maximizes the cumulative reward over time. The environment is typically considered stationary, meaning its dynamics do not change based on the agent's actions. This allows the agent to gradually improve its decision-making through trial and error, leveraging algorithms like Q-learning [51,52], SARSA [53], or policy gradient methods [54]. The simplicity of this setup—one agent, one policy, one learning objective—makes it easier to model, analyze, and implement. It provides a controlled framework where concepts such as exploration vs. exploitation, reward shaping, and value estimation can be clearly defined and systematically improved. However, complexities arise when multiple agents are involved. Optimizing the learning process to train multiple agents efficiently, with minimal computational resources and script usage, becomes a challenge.

Additionally, for parameters that provide a continuous action space—such as drone velocity, acceleration, or angular control—selecting the most suitable algorithm to handle such dynamics is crucial. In other words, adapting Reinforcement Learning (RL) algorithms for such scenarios requires the use of specialized methods beyond traditional discrete-action algorithms like Q-learning. In such environments, policy-based and actor–critic algorithms are preferred, as they are better suited for learning continuous control policies. More advanced actor–critic approaches like Deep Deterministic Policy Gradient (DDPG) [55], Twin Delayed DDPG (TD3) [56] and Soft Actor–Critic (SAC) [57] offer improved stability and performance. These methods combine a value function (critic) with a policy function (actor), allowing for efficient learning in high-dimensional, continuous domains.

Reinforcement Learning (RL) relies on the assumption that an agent operates within a Markovian environment, leading to decision problems commonly modeled by the Markov Decision Process (MDP) [58]. The foundational concept in this model is the Markov Property, which asserts that the probability of transitioning to the next state depends solely on the current state and not on any previous states [59].

A Markov Decision Process (MDP) is a mathematical framework for modeling decision-making in environments where outcomes are partly random and partly under the control of an agent. A Markov Decision Process (MDP) is defined as a 5-tuple:

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$$

where

- $\mathcal{S}$ is a finite set of states.
- $\mathcal{A}$ is a finite set of actions.
- $\mathcal{P}(s' \mid s, a)$ is the transition probability function, giving the probability of transitioning to state s' from state s after taking action a.
- $\mathcal{R}(s, a)$ is the reward function, specifying the expected reward received after taking action a in state s.
- $\gamma \in [0, 1]$ is the discount factor, representing the importance of future rewards.

The agent's objective is to learn a policy $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes the expected cumulative discounted reward:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right]$$

The Multi-Agent Reinforcement Learning (MARL) algorithm and its variations address the complexities of controlling multiple agents, such as a swarm of UAVs. This paper explores a broad spectrum of drone control methods, focusing on how MARL facilitates the coordination of UAV swarms during specific operations. Unlike Single-Agent Reinforcement Learning (SARL), which involves one agent interacting solely with its environment Figure 2, MARL introduces multiple agents into the same environment Figure 3. A Multi-Agent Reinforcement Learning (MARL) environment can be modeled as a Markov Game (Stochastic Game), defined by the tuple

$$\mathcal{G} = (\mathcal{S}, \{\mathcal{A}^i\}_{i=1}^N, \mathcal{P}, \{\mathcal{R}^i\}_{i=1}^N, \gamma)$$

where

- $\mathcal{S}$ is the set of global states.
- $\mathcal{A}^i$ is the action space for agent i, and the joint action space is $\mathcal{A} = \mathcal{A}^1 \times \cdots \times \mathcal{A}^N$.
- $\mathcal{P}(s' \mid s, a^1, \ldots, a^N)$ is the transition function, determining the probability of transitioning to state s' given current state s and joint action $(a^1, \ldots, a^N)$.

- $\mathcal{R}^i(s, a^1, \ldots, a^N)$ is the reward function for agent i.
- $\gamma \in [0, 1]$ is the discount factor.

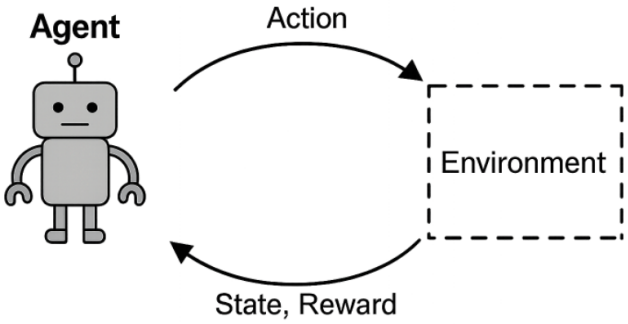

Figure 2. Pictorial representation of Single-Agent Reinforcement Learning. The figure shows the flow of interaction in a typical Single-Agent Reinforcement Learning algorithm.

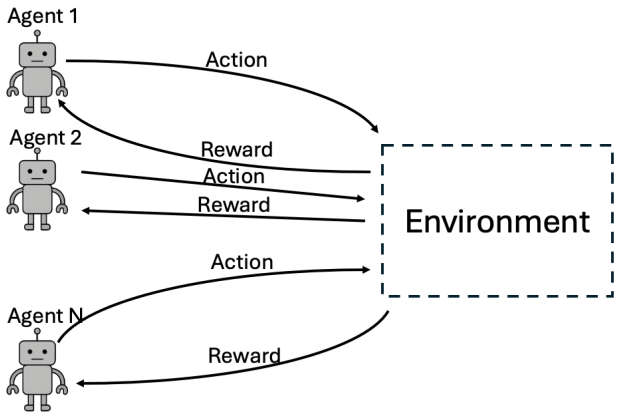

Figure 3. Pictorial representation of Multi-Agent Reinforcement Learning. The figure shows the flow of interaction in a typical Multi-Agent Reinforcement Learning Algorithm.

Each agent i seeks to learn a policy $\pi^i : \mathcal{S} \rightarrow \mathcal{A}^i$ that maximizes its own expected return:

$$\mathbb{E}_{\pi^1,\ldots,\pi^N}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}^i(s_t, a_t^1, \ldots, a_t^N)\right]$$

In MARL, these agents interact not only with the environment but also with each other [60]. While each agent individually observes the shared environment's state and decides on an action, it must also consider the actions and potential reactions of other agents. This scenario presents a unique challenge: although there is only one environment and one collective state, the presence of multiple agents necessitates individual policies and results in diverse rewards. Other major challenges arise from the non-stationarity of the environment and the scaling or dimensionality [61,62]. MARL algorithms can be broadly categorized into three types [63]: value-based MARL, which focuses on updating the value functions of state–action pairs; policy-based MARL, which directly optimizes the policy functions guiding the agents' actions; and actor–critic methods, which combine the strengths of both value-based and policy-based approaches. The "actor" updates the policy directly, while the "critic" evaluates the action by estimating value functions [64]. This combination allows for more stable and efficient learning in complex environments [65]. Actor–critic methods are generally considered a subclass of policy-based methods in the field of Reinforcement Learning [66].

As mentioned before, Reinforcement Learning (RL) algorithms can be categorized based on several criteria. Firstly, they are divided into value-based and policy-based

algorithms, depending on whether the updates are applied to value functions or policies. Secondly, they can be classified according to the nature of their state and action spaces into discrete- or continuous-time states and action RL algorithms. Lastly, RL algorithms are differentiated based on the number of agents they manage, categorized as Single-Agent and Multi-Agent Reinforcement Learning algorithms. In subsequent sections, these classifications are explored in greater detail.

4. Value-Based Multi-Agent Reinforcement Learning Techniques

The primary goal of value-based Reinforcement Learning (RL) methods is to accurately estimate the optimal state–action or state–value function, commonly referred to as the optimal Q-function (Q_*^π). The optimal policy can be attained by utilizing a greedy action of the estimated Q-value [67]. Value-based RL algorithms aim to achieve convergence by consistently updating the Q-function values to reflect the expected rewards. In the context of UAVs, which are often modeled in grid world environments where the operational space is represented as a discrete grid of cells, the actions can be discretized, with the primary actions being movements from one grid cell to another. Value-based Multi-Agent Reinforcement Learning (MARL), which operates in discrete action spaces, is well-suited for managing such tasks.

4.1. QMIX

QMIX is a deep Multi-Agent Reinforcement Learning (MARL) technique designed to learn decentralized policies within a centralized training framework, effectively utilizing additional state information [68]. This approach enables the learning of an optimal joint action–value function that can be decomposed into individual action–value functions for each agent, simplifying control and coordination. In [43], MARL was applied to the strategy game StarCraft, where multiple agents are controlled to defeat opponents, simulating the coordination of unmanned vehicle movements. Two MARL techniques, QMIX and Value Decomposition Network (VDN), were used, both employing a centralized reward system for the agents. It was observed that QMIX outperformed VDN in terms of efficacy. However, the discrete nature of the actions in this scenario made them particularly well-suited for value-based methods.

The QMIX learning technique high level structure is presented in Figure 4, and the algorithm is presented in Algorithm 1. The key idea of the algorithm is to allow centralized training with access to the global state s and decentralized execution, where each agent selects actions based only on its local observation o_i. Each agent learns an individual action–value function $Q_i(o_i, a_i)$, conditioned only on its local observation o_i. During training, these individual Q-values are combined through a mixing network into a global action–value pair:

$$Q_{\text{tot}}(s, a) = f_\phi(Q_1, Q_2, \ldots, Q_N; s)$$

Here, f_ϕ is a mixing network parameterized by ϕ that incorporates the global state s to guide the learning process. To ensure that decentralized execution is feasible, the mixing network is constrained to be monotonic in each agent's Q-value:

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \quad \forall i$$

Algorithm 1 QMIX Algorithm

Parameters θ of agent networks $\{Q_i\}$ for $i = 1$ to N
Parameters ϕ of the mixing network Q_{tot}
Target networks $\{Q_i'\}$ and Q_{tot}' initialized with θ and ϕ
Replay buffer $\mathcal{D}$
for episode $= 1$ to $MaxEpisodes$ **do**
  Initialize environment, get global state s and local observations $\{o_1, \ldots, o_N\}$
  for $t = 1$ to T (episode length) **do**
    for each agent $i = 1$ to N **do**
      Select action a_i using ϵ-greedy policy from $Q_i(o_i)$
    end for
    Execute joint action $a = (a_1, \ldots, a_N)$
    Observe reward r, next state s', next observations $\{o_1', \ldots, o_N'\}$
    Store $(s, \{o_i\}, a, r, s', \{o_i'\})$ in replay buffer $\mathcal{D}$
  end for
  for training_step = 1 to NumUpdates **do**
    Sample mini-batch from $\mathcal{D}$ each transition $(s, \{o_i\}, a, r, s', \{o_i'\})$
    for each agent $i = 1$ to N **do**
      Compute $a_i' = \arg\max_{a'} Q_i'(o_i', a'; \theta_i')$
      Compute $Q_i^{\text{next}} = Q_i'(o_i', a_i'; \theta_i')$
    end for
    Compute $Q_{\text{tot}}^{\text{next}} = \text{MixingNetwork}(Q_1^{\text{next}}, \ldots, Q_N^{\text{next}} \mid s'; \phi')$
    $y = r + \gamma Q_{\text{tot}}^{\text{next}}$
    for each agent $i = 1$ to N **do**
      Compute $Q_i^{\text{current}} = Q_i(o_i, a_i; \theta_i)$
    end for
    Compute $Q_{\text{tot}}^{\text{current}} = \text{MixingNetwork}(Q_1^{\text{current}}, \ldots, Q_N^{\text{current}} \mid s; \phi)$
    Compute loss $\mathcal{L} = (y - Q_{\text{tot}}^{\text{current}})^2$
  end for
  Update θ and ϕ using gradient descent on L
  Every C steps: update target networks: $\theta' \leftarrow \theta, \phi' \leftarrow \phi$
end for

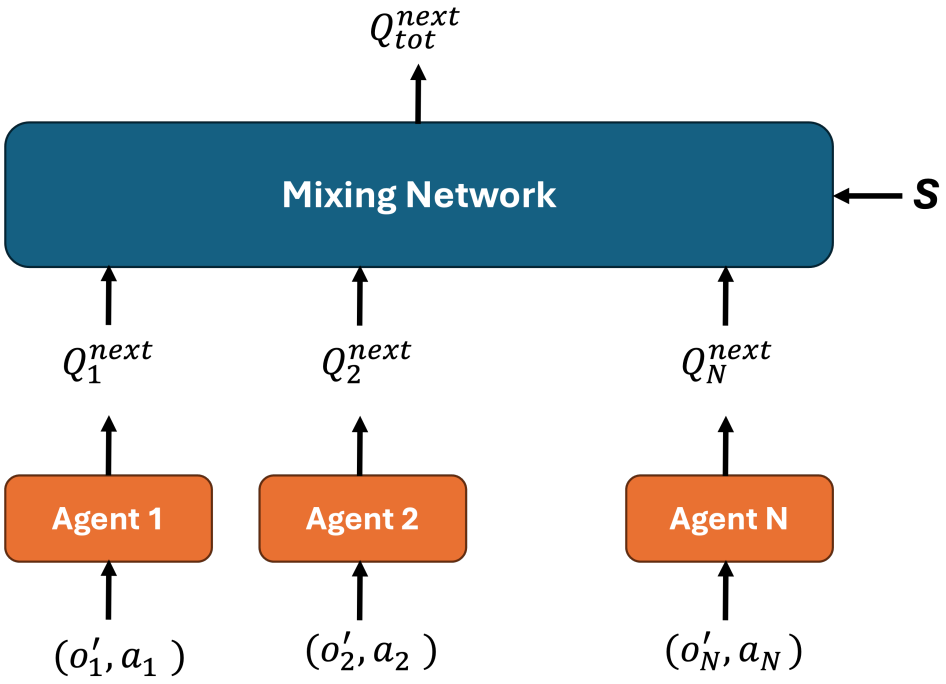

Figure 4. QMIX algorithm structure.

This constraint guarantees that maximizing $Q_i(o_i, a_i)$ for each agent leads to an increase in the joint Q_{tot}, making decentralized greedy action selection effective. During training, transitions are stored in a replay buffer in the form

$$(s, \{o_i\}, \mathbf{a}, r, s', \{o_i'\})$$

The update rule uses the target

$$y = r + \gamma Q'_{\text{tot}}(s', \mathbf{a}')$$

where each agent i selects $a_i' = \arg\max_a Q_i'(o_i', a)$ using the target networks. The loss is computed as:

$$\mathcal{L} = (y - Q_{\text{tot}}(s, \mathbf{a}))^2$$

Gradient descent is then applied to minimize this loss and update both agent networks and the mixing network.

There are many systems where the the QMIX algorithm was employed to control multiple agents such as in the StarCraft Multi-Agent Competition and in a multi-player soccer game, as described in [69]. In the soccer simulation, the environment featured 22 players, each with a specific objective: either attacking or defending by moving towards the ball. QMIX was effectively implemented in this context, enabling the simulation of straightforward decision-making and actions. In contrast, the StarCraft game presents a more complex scenario. As a strategy game requiring numerous decisions, the actions in StarCraft are more intricate than those in the soccer game, though they remain discrete. The game's rapidly changing states and environmental complexity add further challenges. Despite these complexities, QMIX was successfully applied to StarCraft, demonstrating its adaptability to environments with discrete actions and varying levels of decision-making complexity.

4.2. Deep Q Network (DQN)

The core concept of Q-learning involves updating the values of $Q(.)$ iteratively for all state–action pairs in a structure known as the Q-table [70]. This approach can become computationally intensive in scenarios involving large action spaces [71,72]. In such cases, function approximation is employed to estimate Q values [71]. For instance, a function with parameter θ to compute Q values can be denoted as $Q(s, a; \theta)$. Function approximation is typically nonlinear, which can lead to instability in Q-learning [73,74]. To address the computational challenges and improve stability, the Deep Q-Network (DQN) was developed. It enhances the training process of Q-learning by incorporating experience replay and periodically updating the Q-values toward specified target values. It combines Q-learning with deep neural networks to handle environments with high-dimensional state spaces. It was introduced by DeepMind in 2013 and gained attention after successfully playing Atari 2600 games directly from pixel inputs, as detailed in [75].

The Multi-Agent Deep Q-Network (MADQN) algorithm extends the classical Deep Q-Network (DQN) to environments involving multiple agents. Each agent independently learns its own Q-function, as shown in Figure 5, using local observations and rewards. The algorithm shown in Algorithm 2 is fully decentralized during both training and execution. During the initialization phase, each agent initializes its Q-networks Q_i and target networks Q_i^{target} with weights θ_i and θ_i', respectively, and creates a replay buffer $\mathcal{D}_i$ for each agent $i \in \{1, \ldots, N\}$. During each episode, the environment resets and obtains initial states $\{s_1, s_2, \ldots, s_N\}$. Each agent i selects an action a_i and then executes a joint action $\{a_1, \ldots, a_N\}$ in the environment while observing next states $\{s_1', \ldots, s_N'\}$ and rewards $\{r_1, \ldots, r_N\}$. Each

agent stores its experience (s_i, a_i, r_i, s_i') in its replay buffer $\mathcal{D}_i$. After sampling, a minibatch of transitions from $\mathcal{D}_i$ the target is computed:

$$y_i = r_i + \gamma \max_{a'} Q_i^{\text{target}}(s_i', a'; \theta_i')$$

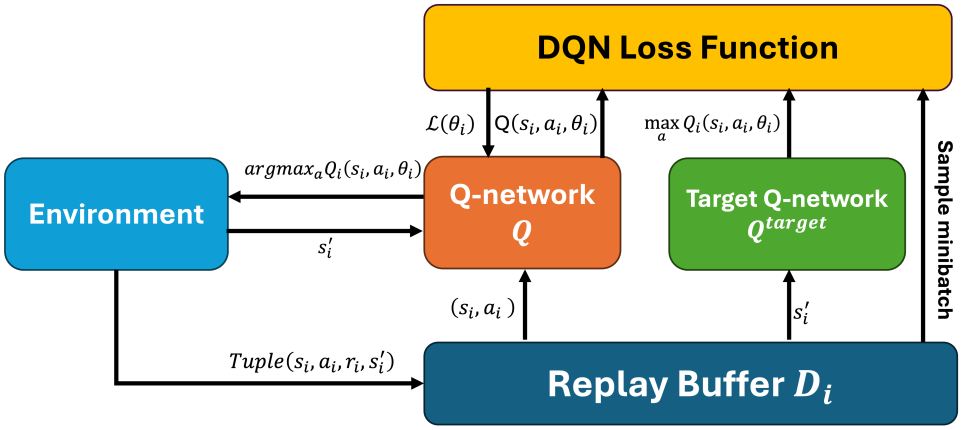

Figure 5. Deep Q-Network for Agent (i).

Algorithm 2 Multi-Agent Deep Q-Network (MADQN) Algorithm

Initialize replay buffer $\mathcal{D}_i$ for each agent $i \in \{1, \ldots, N\}$
Initialize Q-network $Q_i(s_i, a_i; \theta_i)$ with random weights θ_i for each agent
Initialize target Q-network $Q_i^{\text{target}}(s_i, a_i; \theta_i')$ with $\theta_i^{'\leftarrow\theta_i}$
for episode $= 1$ to $MaxEpisodes$ **do**
  Initialize environment and get initial states $\{s_1, s_2, \ldots, s_N\}$
  for $t = 1$ to T **do**
    for each agent $i = 1$ to N **do**
      Select action a_i:

- with probability ϵ, select random a_i
- otherwise, select $a_i = \arg\max_a Q_i(s_i, a; \theta_i)$

    end for
    Execute joint action $\{a_1, \ldots, a_N\}$ in environment
    Observe next states $\{s_1', \ldots, s_N'\}$ and rewards $\{r_1, \ldots, r_N\}$
    for each agent $i = 1$ to N **do**
      Store transition (s_i, a_i, r_i, s_i') in buffer $\mathcal{D}_i$
    end for
    for each agent $i = 1$ to N **do**
      Sample minibatch of transitions (s_i, a_i, r_i, s_i') from $\mathcal{D}_i$
      Compute target:

$$y_i = r_i + \gamma \max_{a'} Q_i^{\text{target}}(s_i', a'; \theta_i')$$

      Update θ_i by minimizing:

$$\mathcal{L}(\theta_i) = (y_i - Q_i(s_i, a_i; \theta_i))^2$$

    end for
    if every C steps **then**
      Update target networks: $\theta_i' \leftarrow \theta_i$ for all agents i
    end if
  end for
end for

Parameters θ_i are then updated by minimizing the loss:

$$\mathcal{L}(\theta_i) = (y_i - Q_i(s_i, a_i; \theta_i))^2$$

In [76], DQN was implemented in swarms of two and three drones tasked with surveilling four targets on a simulated 2D grid plane. Transfer Learning was integrated into the DQN algorithm to facilitate adapting the learning model to new drones added to the network. The drones operated within discrete grid boxes, allowing the DQN algorithm to effectively manage the swarm. Initially, a swarm of two drones was created by transferring the model from the primary UAV to the additional drone, which was effective in the absence of an established UAV network. However, when a third drone was added to the established network, the Transfer Learning process became less efficient, slowing the learning rate, but eventual convergence was achieved. The main issues with the presented results are scalability, the lack of a realistic scenario, and limited evaluation against benchmarks such as QMIX, MADDPG, and MAPPO. In [77], combat drones were simultaneously trained using both the Multi-Agent Deep Q-Network (MADQN) and Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithms for strike force and mobility tasks. The authors explored three training strategies for MADQN. The first strategy, scenario transfer, involved transferring learned models from simple combat tasks to other drones. The second strategy, self-play training, allowed agents to continually learn by iterating the training algorithm. The third strategy, rule-coupled training, incorporated rules from aerial combat operations into the training algorithm. Combining these three strategies significantly enhanced both the training effectiveness and post-training functionality of the UAVs. In [78], SERT-DQN, a novel Reinforcement Learning algorithm was presented. It was designed to optimize multi-agent coverage path planning (CPP) for UAV-UGV systems, particularly in uncertain and fault-prone environments like Mars. The framework combines decentralized DQN-based primary networks for individual UAVs with a centralized intermediate Q-network located on the UGV, allowing coordination through a hybrid local-global policy structure. The SERT-DQN algorithm significantly improves upon both regular DQN and QMIX in the context of multi-agent coverage path planning (CPP) but also introduces new trade-offs. For example, SERT-DQN introduces greater computational complexity and dependency on centralized updates, making it more demanding to implement in real-world scenarios

The effectiveness of decomposition in Value-Based Multi-Agent Reinforcement Learning (MARL) lies in the ability to approximate a global action–value function $Q_{\text{tot}}(\mathbf{s}, \mathbf{a})$ using individual agent value functions $Q_i(s_i, a_i)$, thereby enabling decentralized execution while retaining the advantages of centralized training. In simple settings such as Value Decomposition Networks (VDNs), the total value function is linearly decomposed as $Q_{\text{tot}} \approx \sum_i Q_i$, which is theoretically sound under the assumptions of additive rewards and independent agent transitions. However, this linear formulation cannot capture complex inter-agent dependencies. To address this limitation, QMIX introduces a monotonic mixing network that guarantees $\partial Q_{\text{tot}}/\partial Q_i \geq 0$, ensuring that individual greedy actions with respect to Q_i still correspond to the maximization of the global action–value function. This allows QMIX to be more expressive than VDN while supporting decentralized policies. Nonetheless, QMIX cannot model non-monotonic interactions or highly coordinated joint behaviors. Advanced methods such as QTRAN [79,80] relax the monotonicity constraint via constrained optimization, achieving theoretically complete representational power, albeit with increased computational cost. Further extensions like MAVEN [81] and ROMA incorporate latent variables or role-based decompositions to capture diverse coordination patterns among agents. Overall, while decomposition facilitates scalable learning and decision-making, it imposes structural biases that may limit the algorithm's ability to learn optimal joint policies in environments with complex agent interactions.

Table 1 presents a brief comparison between the two most common Value-Based Multi-Agent Reinforcement Learning Techniques, QMIX and MADQN, in Multi-Agent Reinforcement Learning. On the other hand, Table 2 compares four recent value-based Multi-Agent Reinforcement Learning (MARL) frameworks applied to UAV and multi-agent coordination tasks. The algorithms span a range of domains, including UAV swarm surveillance [76], combat scenarios [77], hybrid UAV–UGV coverage missions [78], and multi-task micromanagement in UAVs [69]. Key metrics such as scalability, training technique, fault tolerance, and convergence speed are considered. Notably, [78] stands out for its explicit fault-tolerant design using SERT-DQN, while [76] demonstrates strong scalability and effective Transfer Learning, whereas [77] achieves improved training efficiency through scenario-transfer and self-play.

Table 1. Comparison between QMIX and MADQN in Multi-Agent Reinforcement Learning.

Feature	QMIX	MADQN
Mode of Operation	Utilizes a mixing network to derive decentralized policies from centralized action–state value functions.	Integrates Q-learning with deep neural networks to enhance the approximation capability of Q-functions.
Performance in Partially Observable Environments	Highly effective due to its decentralized policy design. Ideal for defense or combat scenarios where agents must act on limited information, such as hidden enemy locations.	Less suitable for partially observable settings. Limited ability to disambiguate ally vs. enemy behavior makes it inadequate for military rescue missions.
Scalability	Scalability is limited; the addition of agents after training may degrade performance due to reliance on a fixed network structure.	Highly scalable; supports dynamic swarm expansion without significant loss in performance, learning stability, or accuracy.
Computational Complexity	Requires frequent value function updates, increasing computational demands. This can affect drone payload constraints and swarm design.	Performs periodic updates, reducing overall computational load. Enables compact hardware deployment and energy-efficient swarm operations.

Table 2. Comparison between articles on Value-Based Multi-Agent Reinforcement Techniques

Paper ID	Environment Type	Training Technique	Swarm Size	S.R.[4]	Scal.[5]	F.T.[7]	T.L.[3]	Conv.[6]
[76]	Grid-based UAV surveillance	Distributed DQN,	3 UAVs	60–80%	High	N/A [1]	Yes	40,000
[77]	Multi-UAV combat (RTS-style)	MADQN, MADDPG	Two teams, ∼5 UAVs	70–90%	Mod. [2]	Partial	Yes	15×10^6
[78]	Hybrid UAV–UGV coverage mission	SERT-DQN	5 UAVs	90%	Mod. (UGV-limited)	Yes	N/A	N/A
[69]	Multi-task StarCraft micromanagement	REFIL (QMIX)	Variable (6–20 agents)	60–80%	High	N/A	Yes	10^7

[1] N/A stands for Not available. [2] Mod. stands for Moderate. [3] T.L. stands for Transfer Learning. [4] S.R. stands for Success Rate. [5] Scal.. stands for Scalability. [6] Conv. stands for Convergence (#of Steps). [7] F.T. stands for Fault Tolerance.

5. Policy-Based Multi-Agent Reinforcement Learning Techniques

Policy-based methods, also known as policy gradient methods, aim to directly model and optimize the policy. The parameterized policy, represented as $\pi_\theta(a|s)$, is defined as a function of θ [82]. The objective function's value depends on this policy, and various algorithms are employed to optimize the parameter θ to maximize the reward, as outlined in [82]. Policy-Based Multi-Agent Reinforcement Learning (MARL) techniques are a pivotal framework in the field of Reinforcement Learning, especially when dealing with environments involving multiple interacting agents [67,83]. These methods synergize policy-based (actor) and value-based (critic) approaches, enabling agents to learn optimal behaviors in complex, dynamic, and often partially observable settings [84,85]. At the core of policy-based methods lies the direct optimization of policies, which can be particularly advantageous in high-dimensional or continuous action spaces [86,87]. Unlike value-based methods that derive policies indirectly through value functions, policy-based approaches adjust the policy parameters directly to maximize expected returns. This direct optimization can lead to more stable and efficient learning, especially in scenarios where value estimation is challenging [88]. In multi-agent scenarios, policy-based methods must account for the non-stationarity introduced by concurrently learning agents, making the design of effective algorithms particularly challenging [83,89]. Techniques such as Centralized Training with Decentralized Execution (CTDE) have been proposed to address these challenges, allowing agents to learn coordinated behaviors while operating independently during execution [84,90]. Policy-based MARL techniques have been applied across various domains, including autonomous driving, robotic coordination, and resource management [91,92]. For instance, in network load balancing, these methods have been utilized to dynamically distribute traffic, optimizing performance and reducing congestion [92]. In robotic swarms, policy-based algorithms facilitate coordinated behaviors without centralized control, enhancing adaptability and robustness [93,94]. One can conclude that these methods are particularly well-suited for applications requiring continuous action control, such as robotic manipulations and drone speed adjustments. Despite their successes, policy-based MARL methods face challenges such as scalability to large agent populations, efficient communication protocols, and robustness to partial observability [62,67]. Future research is directed toward developing more scalable architectures, incorporating meta-learning for adaptability, and designing algorithms that can operate effectively with limited communication [67,68]. In the following subsections, we will explore three Policy-Gradient Multi-Agent Reinforcement Learning (MARL) algorithms and their applications in UAV control.

5.1. Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) is a model-free, off-policy actor–critic algorithm that employs a deterministic policy to address the challenge of discrete action spaces inherent in Deep Q-Networks (DQNs) [95]. In contrast, DQNs stabilize the Q-function learning process through experience replay and a frozen target network. Multi-agent DDPG (MADDPG) extends the DDPG framework to environments with multiple agents operating synchronously, each using locally observed information [84]. In MADDPG, each agent perceives the environment as dynamic due to the continuous policy updates from other agents, whose policies are unknown. It was designed to address the challenges posed by decentralized decision-making and non-stationarity in multi-agent systems. It was introduced by [84] as an extension of the Deep Deterministic Policy Gradient (DDPG) algorithm [55] to cooperative and competitive environments involving multiple agents. In Single-Agent Reinforcement Learning, DDPG has been shown to perform well in continuous action spaces using a deterministic actor–critic architecture. However, applying

DDPG independently in multi-agent environments leads to non-stationary learning dynamics, as the environment's state distribution changes with the evolving policies of other agents [83,89]. This non-stationarity can degrade learning performance or even lead to divergence. To address this, MADDPG introduces a centralized training and decentralized execution (CTDE) paradigm. During training, each agent has access to a centralized critic that takes into account the global state and actions of all agents. This allows the critic to estimate more accurate gradients despite the non-stationarity caused by other learning agents. However, the actor (policy network) for each agent uses only local observations during execution, enabling scalability and applicability in real-world distributed systems [84].

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm structure is shown in Figure 6, while the algorithm is shown in Algorithm 3. It extends the Deep Deterministic Policy Gradient (DDPG) method to multi-agent environments by adopting the Centralized Training with Decentralized Execution (CTDE) paradigm. In MADDPG, each agent $i \in \{1, \ldots, N\}$ learns a deterministic policy (actor) $\mu_i(s_i; \theta^{\mu_i})$, where s_i is the local observation of agent i and θ^{μ_i} represents the parameters of the actor network. The corresponding action is given by $a_i = \mu_i(s_i)$.

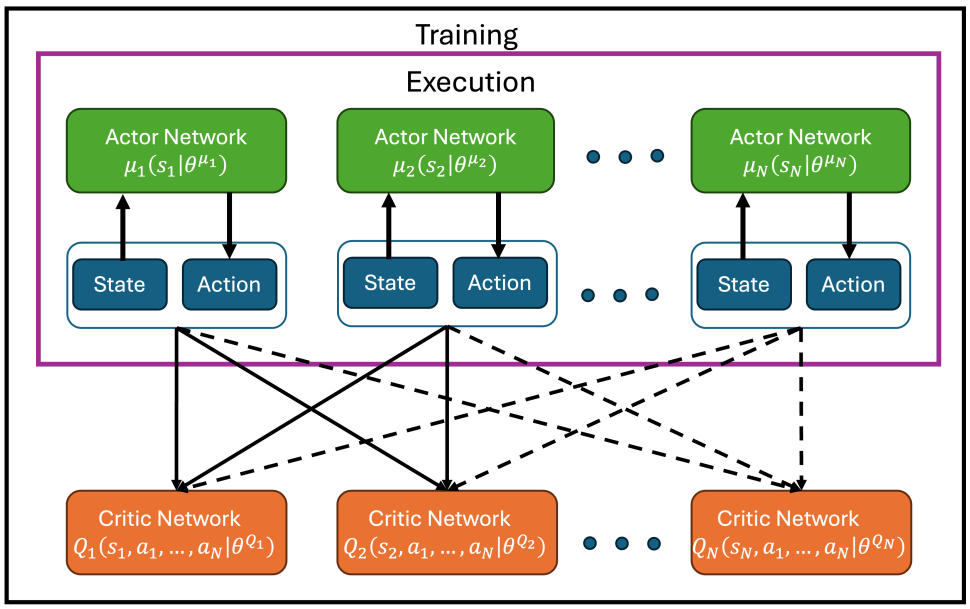

Figure 6. Multi-Agent Deep Deterministic Policy Gradient structure.

During training, each agent also maintains a centralized critic network $Q_i(s_i, a_1, \ldots, a_N; \theta^{Q_i})$, where x is the global state (which may include all agents' observations), $a_1, \ldots, a_N$ are the actions of all agents, and θ^{Q_i} are the critic's parameters. The target value for critic training is computed as

$$y_i = r_i + \gamma Q_i'(s_i', a_1', \ldots, a_N'; \theta^{Q_i'}), \tag{1}$$

where r_i is the reward received by agent i, $\gamma \in [0, 1]$ is the discount factor, Q_i' is the target critic network, and $a_j' = \mu_j'(s_j'; \theta^{\mu_j'})$ are the actions predicted by the target actor networks at the next state s_j' for all agents j.

The critic network is updated by minimizing the following loss function:

$$\mathcal{L}(\theta^{Q_i}) = \frac{1}{K} \sum_{k=1}^{K} \left(y_i^{(k)} - Q_i(x^{(k)}, a_1^{(k)}, \ldots, a_N^{(k)}; \theta^{Q_i}) \right)^2, \tag{2}$$

where K is the minibatch size and $(x^{(k)}, a_1^{(k)}, \ldots, a_N^{(k)}, y_i^{(k)})$ are sampled from the replay buffer.

Algorithm 3 Multi-Agent Deep Deterministic Policy Gradient (MADDPG)

Initialize environment with N agents
for each agent $i \in \{1, ..., N\}$ **do**
  Initialize actor network $\mu_i(s_i|\theta^{\mu_i})$ and critic network $Q_i(s_i, a_1, ..., a_N|\theta^{Q_i})$
  Initialize target networks: $\theta^{\mu_i'} \leftarrow \theta^{\mu_i}$, $\theta^{Q_i'} \leftarrow \theta^{Q_i}$
  Initialize replay buffer $\mathcal{D}_i$
end for
for episode = 1 to M **do**
  Initialize random process $\mathcal{N}_i$ for exploration
  Receive initial State $s = (s_1, ..., s_N)$
  for time step $t = 1$ to T **do**
    for each agent i **do**
      Select action $a_i = \mu_i(s_i|\theta^{\mu_i}) + \mathcal{N}_i$
    end for
    Execute joint action $(a_1, ..., a_N)$ and observe reward r_i and next State s_i' for each agent
    for each agent i **do**
      Store (s_i, a_t, r_t, s_i') in $\mathcal{D}_i$
      Sample a minibatch of K transitions from $\mathcal{D}_i$
      Set target:

$$y_i = r_i + \gamma Q_i'(s_i', a_1', ..., a_N'|\theta^{Q_i'})$$

      where $a_j' = \mu_j'(s_j'|\theta^{\mu_j'})$ for all agents j
      Update critic by minimizing:

$$\mathcal{L}(\theta^{Q_i}) = \frac{1}{K}\sum(y_i - Q_i(s, a_1, ..., a_N|\theta^{Q_i}))^2$$

      Update actor using the policy gradient:

$$\nabla_{\theta^{\mu_i}} J \approx \frac{1}{K}\sum \nabla_{a_i} Q_i(s, a_1, ..., a_i, ..., a_N) \nabla_{\theta^{\mu_i}} \mu_i(s_i)$$

      Update target networks:

$$\theta^{Q_i'} \leftarrow \tau\theta^{Q_i} + (1-\tau)\theta^{Q_i'}$$

$$\theta^{\mu_i'} \leftarrow \tau\theta^{\mu_i} + (1-\tau)\theta^{\mu_i'}$$

    end for
  end for
end for

The actor is updated using the deterministic policy gradient, given by

$$\nabla_{\theta^{\mu_i}} J \approx \frac{1}{K}\sum_{k=1}^{K} \nabla_{a_i} Q_i(x, a_1, \ldots, a_N)\big|_{a_i=\mu_i(s_i)} \cdot \nabla_{\theta^{\mu_i}} \mu_i(s_i). \tag{3}$$

Target networks for both the actor and critic are updated using soft updates as follows:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \tag{4}$$

where $\tau \ll 1$ is the soft update rate.

MADDPG has demonstrated significant success in cooperative navigation tasks, autonomous vehicle coordination, and multi-robot systems. In the Multi-Agent Particle Environment (MPE) benchmark, MADDPG outperformed independent DDPG and Q-learning baselines by enabling agents to learn coordinated strategies [84]. It has also been adopted in domains such as autonomous drone swarms [67] and smart grid energy

management [92], showcasing its flexibility. Despite its success, MADDPG is not without limitations. The use of separate critics for each agent limits scalability to large agent populations. Furthermore, it assumes that agents can access the actions and observations of others during training, which may not always be feasible in partially observable settings. These limitations have inspired variants and improvements, such as MAAC and MAPPO, that aim to improve coordination, scalability, and robustness [90,93]. In [77], MADDPG was compared with the Multi-agent Deep Q-Network (MADQN) algorithm, incorporating three learning strategies to enhance the speed and accuracy of UAV training for combat tasks. MADDPG demonstrated superior performance, achieving better convergence and providing more decision-making options for UAVs due to its operation in the continuous domain and the use of a continuous action space. Article [96] compares the implementation of the DDPG, TRPO, and PPO algorithms in drone control against each other and a well-tuned PID controller. Specifically, DDPG was used to train drones for flight and stability maintenance. Although DDPG required a lengthy training period, it achieved convergence more quickly than TRPO. MADDPG works well for drones as it supports continuous action spaces, suitable for the control of thrust, pitch, and yaw, and it allows decentralized execution, which is necessary for independent UAV operation. It also enables the scalable training of multiple drones via centralized critics and finally, it is adaptable to partial observability and communication-constrained environments. In [67], MADDPG was used to train quadrotors to maintain formations while avoiding collisions, using only partial observations in simulated environments like AirSim or Gazebo. In [92] MADDPG allows drones to navigate between buildings and through air lanes by learning optimal control under flight constraints in urban air mobility simulations. In [97], an artificial intelligence approach called Simultaneous Target Assignment and Path Planning (STAPP) is used to solve the multi-UAV target assignment and path planning (MUTAPP problem), which leverages the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm to model and train agents to handle both target assignment and path planning concurrently, guided by a specially designed reward structure.

5.2. *Multi-Agent Trust Region Policy Optimization (MATRPO)*

Trust Region Policy Optimization (TRPO) aims to limit excessive changes in policy updates by incorporating a KL divergence constraint, thereby reducing the number of parameter updates per step [98]. However, TRPO has not fully resolved the issue of minimizing parameter updates, making it less reliable for controlling hardware agents. This limitation led to the development of the PPO algorithm.

Multi-Agent Trust Region Policy Optimization (MA-TRPO) extends the principles of Trust Region Policy Optimization (TRPO) to Multi-Agent Reinforcement Learning (MARL) scenarios. While TRPO has demonstrated stability and performance in single-agent settings by ensuring monotonic policy improvements through constrained updates, its direct application to multi-agent environments introduces challenges due to the dynamic interactions among multiple learning agents [83]. In MARL, each agent's learning process can affect and be affected by the learning processes of other agents, leading to a non-stationary environment from any individual agent's perspective. This interdependence complicates the application of TRPO, which assumes a stationary environment for its theoretical guarantees to hold. To address this, researchers have developed adaptations of TRPO tailored for multi-agent settings. One such adaptation is the Multi-Agent Trust Region Policy Optimization (MATRPO) algorithm proposed by [99], which reformulates the TRPO update as a distributed consensus optimization problem. This approach allows agents to optimize their policies based on local observations and private rewards without requiring access to other agents' information, promoting a decentralized and privacy-preserving learning process.

Empirical evaluations have demonstrated MATRPO's robust performance in cooperative multi-agent tasks.

The *Multi-Agent Trust Region Policy Optimization* (MATRPO) structure is shown in Figure 7, while the algorithm is shown in Algorithm 4 [100,101]. In this framework, a set of agents $\{1, 2, \ldots, N\}$ interact with a shared environment modeled as a Markov Game. Each agent i has its own policy $\pi_{\theta_i}(a_i \mid s_i)$ and value function $V_{\phi_i}(s_i)$, where θ_i and ϕ_i denote the respective parameter vectors. Agents observe local states s_i, take actions a_i, and receive rewards r_i, aiming to maximize the expected return $\mathbb{E}[R_i]$. Training is conducted under the Centralized Training with Decentralized Execution (CTDE) paradigm.

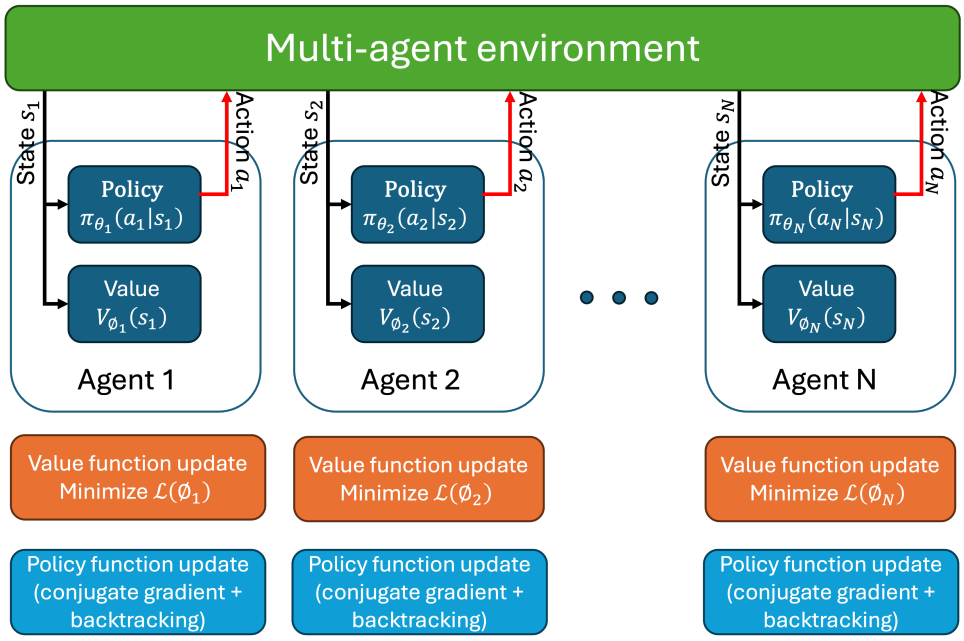

Figure 7. Trust region policy optimization (TRPO).

During each iteration, each agent collects trajectories τ_i by executing its current policy in the environment. The advantage function $A_i(s, a_i)$, often estimated using Generalized Advantage Estimation (GAE), is computed to evaluate the benefit of taking action a_i in state s. The policy gradient for agent i is computed as

$$g_i = \nabla_{\theta_i} \mathbb{E}_{\tau_i} \left[\log \pi_{\theta_i}(a_i \mid s_i) \cdot A_i(s, a_i) \right].$$

To ensure stable updates, MATRPO imposes a trust region constraint using the Kullback–Leibler (KL) divergence between the old and new policies. This leads to the following constrained optimization problem:

$$\max_{\Delta\theta_i} \quad g_i^\top \Delta\theta_i \quad \text{subject to} \quad \Delta\theta_i^\top H_i \Delta\theta_i \leq \delta,$$

where H_i is the Hessian of the sample average KL-divergence, approximated as

$$H_i \approx \mathbb{E}_{\tau_i} \left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid s_i) \cdot \nabla_{\theta_i} \log \pi_{\theta_i}(a_i \mid s_i)^\top \right],$$

and δ is a predefined KL-divergence bound. The optimal update direction $\Delta\theta_i$ is computed using the conjugate gradient method, followed by a backtracking line search to enforce

the KL constraint. Finally, each agent updates its value function by minimizing the mean squared error between predicted and actual returns:

$$\mathcal{L}(\phi_i) = \mathbb{E}_{\tau_i}\left[\left(V_{\phi_i}(s_i) - R_i\right)^2\right].$$

MATRPO allows each agent to learn a stable and effective decentralized policy while leveraging centralized training to improve coordination and performance in multi-agent environments.

Algorithm 4 Multi-Agent Trust Region Policy Optimization (MATRPO)

Initialize policy parameters $\{\theta_i\}_{i=1}^N$ and value function parameters $\{\phi_i\}_{i=1}^N$
Set KL-divergence constraint $\delta > 0$
for each iteration **do**
  for each agent $i \in \{1, \ldots, N\}$ **do**
    Collect trajectories τ_i using policy π_{θ_i} in the environment
    Estimate advantage function $A_i(s, a_i)$ using value function $V_{\phi_i}(s)$
    Compute policy gradient:

$$g_i \leftarrow \nabla_{\theta_i}\mathbb{E}_{\tau_i}\left[\log \pi_{\theta_i}(a_i|s_i) \cdot A_i(s, a_i)\right]$$

    Compute the Hessian of the sample average KL-divergence:

$$H_i \approx \mathbb{E}_{\tau_i}\left[\nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s_i) \cdot \nabla_{\theta_i} \log \pi_{\theta_i}(a_i|s_i)^T\right]$$

    Solve constrained optimization:

$$\begin{aligned} \underset{\Delta\theta_i}{\text{maximize}} \quad & g_i^T \Delta\theta_i \\ \text{subject to} \quad & \Delta\theta_i^T H_i \Delta\theta_i \leq \delta \end{aligned}$$

    Use conjugate gradient to compute step direction $\Delta\theta_i$
    Perform backtracking line search to ensure KL-divergence $\leq \delta$
    Update policy: $\theta_i \leftarrow \theta_i + \Delta\theta_i$
    Update value function parameters ϕ_i by minimizing:

$$\mathcal{L}(\phi_i) = \mathbb{E}_{\tau_i}\left[\left(V_{\phi_i}(s) - R_i\right)^2\right]$$

  end for
end for

Another significant development is the Heterogeneous-Agent Trust Region Policy Optimization (HATRPO) algorithm [102], which introduces a sequential policy update scheme and a multi-agent advantage decomposition lemma. HATRPO enables agents with different policy and value function structures to learn effectively without sharing parameters or relying on restrictive assumptions about the decomposability of the joint value function. This method has shown superior performance in various benchmarks, including Multi-Agent MuJoCo and StarCraft II tasks. In parallel, Wen et al. [103] proposed a game-theoretic formulation of trust-region optimization in multi-agent systems, resulting in the Multi-Agent Trust-Region Learning (MATRL) method. MATRL uses Nash equilibrium concepts to guide policy updates and ensure convergence in competitive and cooperative environments, further strengthening the theoretical underpinnings of trust-region approaches in multi-agent learning.

Multi-Agent Trust-Region Policy Optimization (MATRPO) has shown significant promise in the field of autonomous drones (UAVs), particularly in scenarios that require

decentralized coordination, robustness, and policy stability in dynamic environments. MATRPO extends the Trust Region Policy Optimization (TRPO) framework into multi-agent systems by reformulating the policy optimization as a distributed consensus optimization problem [99]. This formulation is particularly suitable for UAV applications where each drone must act autonomously using local observations while collectively achieving a global task. In many UAV missions, such as aerial surveying, disaster relief, and military surveillance, individual drones have access to only partial state information due to limited sensing and communication constraints. MATRPO enables each agent to update its policy independently without needing access to the policies or observations of other agents, a critical advantage over fully centralized algorithms [83,99]. This makes MATRPO ideal for distributed drone swarms operating in GPS-denied or bandwidth-constrained environments, where decentralized execution is not just efficient but also necessary [104]. UAVs are often deployed in non-stationary environments, such as regions with changing wind patterns, moving obstacles, or adversarial interference. MATRPO's trust region mechanism, which restricts the KL divergence between consecutive policies, ensures that policy updates are stable and conservative, avoiding performance collapses due to overly aggressive updates [86,99]. This is particularly important for aerial applications, where erratic behavior can lead to mission failure or hardware loss. Applications of MATRPO spans areas for coverage and exploration [105], search and rescue (SAR) [106], and emergency communication and network access [107]. Compared to methods like MADDPG or MAPPO, which often require centralized critics or synchronized updates, MATRPO offers a fully decentralized and privacy-preserving framework with the stability benefits of trust region optimization. This makes it highly suitable for UAVs operating in real-world scenarios where central coordination is infeasible or unsafe [99]. The major drawback is that TRPO exhibits the longest training times compared to MADDPG and MAPPO, explaining its infrequent use in UAV control. Despite this drawback, TRPO achieves precise convergence during training, making it beneficial for controlling ground vehicles and robotics.

5.3. Proximal Policy Optimization (PPO)

PPO performs optimization using a batch of navigation examples and minibatch stochastic gradient descent to maximize the objective [87], simplifying the algorithm by eliminating the KL penalty and the need for versatile updates. It balances updates between the current policy π and the previous policy π_{old} that created the batch, preventing excessive updates beyond the region where the sample provides a reliable estimate. PPO also integrates with the actor–critic framework, where each agent uses one actor and multiple critic networks for faster and distributed learning. Multi-Agent Proximal Policy Optimization (MAPPO) is an extension of the Proximal Policy Optimization (PPO) algorithm tailored for Multi-Agent Reinforcement Learning (MARL) environments. It employs a Centralized Training with Decentralized Execution (CTDE) framework, where each agent learns its policy while a centralized critic evaluates the joint actions, facilitating stable learning in cooperative settings.

The Multi-Agent Proximal Policy Optimization (MAPPO)structure is shown in Figure 8, while the algorithm is shown in Algorithm 5 [108]. In MAPPO, each agent i learns a decentralized policy $\pi_{\theta_i}(a_i|s_i)$ based on its own local observation s_i, while benefiting from a centralized critic $V_\phi(s)$ that has access to the global state s during training.

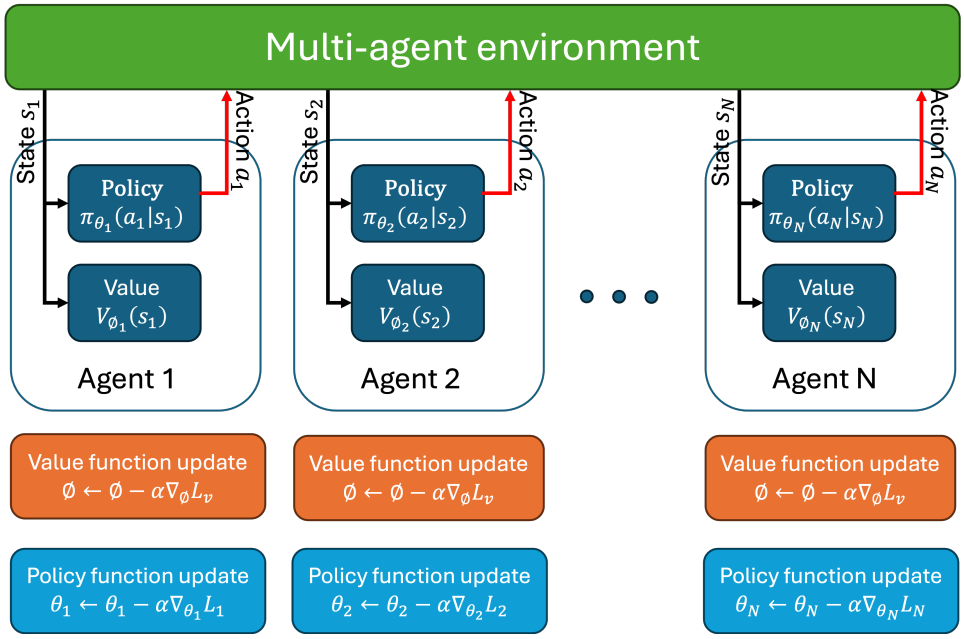

Figure 8. Multi-Agent Proximal Policy Optimization (MAPPO).

Algorithm 5 Multi-Agent Proximal Policy Optimization (MAPPO)

Initialize policy parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_N\}$ for n agents
Initialize centralized value function parameters ϕ
for each iteration **do**
  for each environment rollout **do**
    for $t = 1$ to T **do**
      for each agent $i = 1$ to N **do**
        Observe local state s_i
        Sample action $a_i \sim \pi_{\theta_i}(a_i|s_i)$
      end for
      Execute joint action $a = (a_1, \ldots, a_n)$
      Observe next state s and joint reward $r = (r_1, \ldots, r_N)$
      Store (s, a, r, s') in buffer
    end for
  end for
  for each agent $i = 1$ to N **do**
    Compute advantages $\hat{A}_i$ using Generalized Advantage Estimation (GAE):

$$\delta_i = r_i + \gamma V_\phi(s') - V_\phi(s)$$

$$\hat{A}_i = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_i^{t+l}$$

    Compute the PPO objective:

$$L_i = \min\left(\frac{\pi_{\theta_i}(a_i|s_i)}{\pi_{\theta_i}^{\text{old}}(a_i|s_i)}\hat{A}_t^i, \text{clip}\left(\frac{\pi_{\theta_i}(a_i|s_i)}{\pi_{\theta_i}^{\text{old}}(a_i|s_i)}, 1-\epsilon, 1+\epsilon\right)\hat{A}_i\right)$$

    Update policy parameters:

$$\theta_i \leftarrow \theta_i + \alpha\nabla_{\theta_i} L_i$$

  end for
  Update value function parameters by minimizing:

$$L_v = \sum_t \left(V_\phi(s) - R\right)^2$$

$$\phi \leftarrow \phi - \alpha\nabla_\phi L_v$$

end for

The learning objective of each agent is adapted from PPO's clipped surrogate loss, which stabilizes policy updates by preventing large deviations between the new and old policy. The surrogate loss function for agent i is defined as

$$L_i(\theta_i) = \mathbb{E}\big[\min\big(r_i(\theta_i)\hat{A}_i,\ \text{clip}(r_i(\theta_i), 1-\epsilon, 1+\epsilon)\hat{A}_i\big)\big],$$

where the probability ratio $r_i(\theta_i)$ is given by

$$r_i(\theta_i) = \frac{\pi_{\theta_i}(a_i|s_i)}{\pi_{\theta_i}^{\text{old}}(a_i|s_i)},$$

and $\hat{A}_i$ denotes the advantage function for agent i, which is estimated using Generalized Advantage Estimation (GAE):

$$\delta_i = r_i + \gamma V_\phi(s') - V_\phi(s),$$

$$\hat{A}_i = \sum_{l=0}^{T-t} (\gamma\lambda)^l \delta_i^{t+l}.$$

Here, γ is the discount factor and λ controls the bias-variance tradeoff in advantage estimation.

The centralized value function $V_\phi(s)$ is updated by minimizing the mean squared error between the predicted value and the actual return R:

$$L_v = \sum_t \big(V_\phi(s) - R\big)^2$$

MAPPO has demonstrated strong empirical performance in several benchmark environments, such as the StarCraft Multi-Agent Challenge (SMAC) and Multi-Agent MuJoCo, by leveraging stable and efficient policy optimization for cooperative multi-agent systems. In [49], PPO was used to train two UAVs individually and as a pair for navigation recommendations, while the drone control was manually handled by a remote pilot. This semi-autonomous control approach is suitable for military operations in war zones, where distinguishing between allies and foes is challenging, providing precise navigation paths for manual operation by a remote pilot. In [109], PPO was employed to train 50 UAVs for cooperative persistent surveillance in an unknown environment. A neural network using PPO enabled the swarm to navigate and avoid collisions effectively in a complex environment with numerous obstacles. PPO was also used to train UAVs for stability in multi-agent environments in [110], simulated using the PyBullet gym in OpenAI. The dynamic environment created various scenarios and obstacles, and PPO was found to be well-balanced in training time and convergence accuracy compared to other algorithms. In [77], a complex UAV-to-UAV combat situation was simulated, requiring UAVs to gather information from teammates and enemies. A decentralized PPO approach with decentralized actors and a centralized critic was implemented in a four-UAV swarm combating a two-UAV swarm. This approach outperformed a centralized PPO, which struggled with convergence as the number of UAVs increased. PPO remains the most versatile training algorithm in robotics [7]. In [111], the authors introduce a multi-UAV escape target search algorithm based on MAPPO to address the escape target search difficulty problem. The authors of [90] demonstrated the effectiveness of MAPPO across various MARL benchmarks, including the StarCraft Multi-Agent Challenge (SMAC) and Google Research Football, highlighting its superior performance over other algorithms like MADDPG and COMA. Further advancements, such as the MAPPO-PIS algorithm introduced in [112], incorporates intent-sharing mechanisms to enhance cooperative decision-making among connected and autonomous vehicles (CAVs), showcasing improved safety and efficiency in

complex traffic scenarios. Liu et al. [113] presents a path planning approach utilizing an action-mask-based Multi-Agent Proximal Policy Optimization (AM-MAPPO) algorithm to coordinate multiple UAVs in the search for moving targets within three-dimensional (3D) environments. Specifically, it introduces a multi-UAV high–low altitude collaborative search architecture that combines the wide detection range offered by high-altitude UAVs with the enhanced detection accuracy provided by low-altitude UAVs. Multi-Agent Proximal Policy Optimization (MAPPO) has demonstrated significant potential in enhancing the coordination and control of multi-UAV systems, particularly in complex, dynamic, and partially observable environments. By leveraging the Centralized Training with Decentralized Execution (CTDE) paradigm, MAPPO allows for scalable and robust policy learning across fleets of autonomous drones [90]. Despite this decentralized execution, the training process embeds cooperative behavior learned centrally, ensuring coordinated action across the fleet. This separation of training and execution allows MAPPO to scale to large numbers of agents, as the centralized training phase can efficiently exploit inter-agent dependencies while keeping the execution decentralized and lightweight, which enhances convergence speed. Applications of MAPPO in drone systems span a wide array of domains, including target tracking, surveillance, collaborative exploration, and search-and-rescue missions. These techniques not only improve search efficiency but also enhance mission safety and redundancy, which are critical in real-world deployments. Moreover, MAPPO's ability to maintain stable policy updates through clipped objectives while effectively handling the non-stationarity inherent in multi-agent systems, makes it an ideal candidate for UAV Swarm Intelligence and autonomous aerial systems. As research continues, integrating MAPPO with communication-aware strategies, intent prediction, and sensor fusion is expected to further extend its capabilities in urban air mobility, precision agriculture, and disaster response operations.

The comparative analysis across Policy-Based Multi-Agent Reinforcement Learning (MARL) techniques reveals distinct performance trade-offs tied to each algorithm's environment and coordination strategy. Table 3 presents a brief comparison between the most common Policy Gradient-Based Algorithms for UAV Swarm Control in Multi-Agent Reinforcement Learning. Table 4 presents a comparison between many of the research articles cited in this section. For example, [113], using AM-MAPPO, demonstrates strong scalability and moderate convergence for multi-UAV target tracking by leveraging redundant coverage, while [112] applies PIS-MAPPO in traffic merging scenarios, achieving enhanced safety but at the cost of slower convergence (>10,000 steps). In contrast, Liao et al. [111] introduces ETS-MAPPO with STEE for obstacle-rich, moving-target environments, offering a 30% success rate improvement and moderate convergence (<1600 steps) due to dynamic agent reallocation. Classical methods such as Q-learning [67] and DRL with reward shaping [49] show moderate success (85%) but suffer from poor scalability and slower convergence due to their discrete and environment-specific designs. Transformer-based MARL in [105] offers the highest coverage (97%) with fast convergence and high scalability, while centralized MADDPG in [106] delivers 95% target coverage with rapid convergence (<100 steps) but limited fault tolerance. Chen et al. [107] and Qie et al. [97] explore urban and threat-aware UAV scenarios, respectively, where both demonstrate moderate convergence and fault resilience, with [97] notably requiring over 150,000 steps due to its complex STAPP framework. Overall, advanced MARL frameworks such as AM-MAPPO, Transformer-based MARL, and ETS-MAPPO stand out in handling larger swarms and dynamic environments and achieving higher success rates with relatively stable training dynamics.

Table 3. Comparison of Policy Gradient-Based Algorithms for UAV Swarm Control.

Criteria	MADDPG	TRPO	PPO
Mode of Operation	Uses a deterministic policy gradient with actor–critic networks to enable continuous action spaces.	Minimizes policy updates by enforcing a KL-divergence constraint during optimization.	Uses a clipped objective to limit policy deviation, enabling more stable updates via stochastic gradient descent.
Action Space	Continuous	Continuous (extensions exist for discrete)	Discrete and Continuous
Stochastic / Deterministic	Deterministic policy	Stochastic policy	Stochastic policy
Stability	Stable in moderate-scale applications	High theoretical stability but sensitive in practice	Very stable; robust across diverse tasks
Scalability	Moderate; more suitable for small-to-medium swarms	Poor; not ideal for large-scale applications	High; excellent for large multi-agent systems
Use in Multi-Agent RL	Widely used (e.g., MADDPG) with decentralized actors	Less common due to complexity; experimental in MARL	Very common; MAPPO is widely used in cooperative MARL
Computational Requirements	Requires moderate computational resources—higher than PPO but lower than TRPO. Suitable if drone memory capacity is sufficient and updates are infrequent.	Most computationally demanding due to second-order derivatives. Best suited for high-resource platforms or offline learning.	Most efficient computationally. Suitable for large drone swarms with limited onboard processing capabilities.
Convergence Rate	Exhibits the slowest convergence among the three, requiring more training time for UAV swarms.	Faster than MADDPG but slower than PPO. Adequate for swarm applications with infrequent decision-making.	Fastest convergence. Ideal for real-time UAV applications requiring rapid responses to dynamic environments.
Convergence Accuracy	Offers stable and reliable convergence but slightly less accurate than PPO. Effective in tasks where precision is secondary, such as search and rescue.	Comparable to MADDPG in accuracy but suffers from instability, making it less suitable for real-world deployment.	Delivers the most accurate and stable convergence. Suitable for precision-critical tasks like UAV-based targeting or surveillance.
Implementation Complexity	Moderately complex. Easier than TRPO but more involved than PPO due to its reliance on multi-agent architecture.	Complex implementation due to second-order KL-divergence constraints. Best used in controlled research environments.	Simplest among the three. Its clipped objective makes it straightforward and practical for fast deployment in drone swarms.

Table 4. Comparison between articles on Policy-Based Multi-Agent Reinforcement techniques.

Paper ID	Environment Type	Training Technique	Swarm Size	S.R.[2]	Scal.[3]	F.T.[5]	Conv.[4]
[49]	UAV Target Search	DRL with reward shaping	1	∼85%		Limited Basic avoidance	Mod. > 2000
[67]	Dynamic 3D obstacle environment	Q-learning	4	∼85%	Limited	Poor Basic collision avoidance	Mod. > 2000
[90]	2D Static Environment	MAPPO	5 agents	∼80%	Low (static setup)	Limited	Mod [1]. > 800
[97]	Dynamic threat environment	MADDPG in STAPP framework	5 UAVs	90–95%	Mod.	Yes (not tested)	slow > 150,000.
[105]	Variable-size UAV network	Transformer-based MARL	8 UAVs	∼97% coverage	High	Robust	Fast
[106]	UAV path planning	Centralized MADDPG	20 UAVs	∼95% target coverage	Mod.	Limited	Fast <100
[107]	Urban simulated environment for resource sharing	DDPG	10 UAVs	∼90%	Mod.	Mod. (Backup UAV)	Mod. < 1000
[109]	Urban Intersection (CAVs)	MAPPO	3–5 (50)	∼88%	Limited to CAV scope	Safety constraints	Mod. >2000
[111]	Moving targets and obstacle	ETS–MAPPO + STEE	29	>30% vs. baselines	Mod.	STEEN facilitates reallocation	Mod <1600
[112]	Traffic Merging (CAVs)	MAPPO-PIS	6–10	Better than MAPPO	High	Safety Enhanced Module	Slow >10,000
[113]	multi-UAV target tracking	AM-MAPPO	3–8	Better in target search only	High	redundant coverage	Mod. < 2000

[1] Mod. stands for Moderate. [2] S.R. stands for Success Rate. [3] Scal. stands for Scalability. [4] Conv. stands for Convergence (#of Steps). [5] F.T. stands for Fault Tolerance.

6. Multi-Agent Federated Reinforcement Learning Techniques

Federated Learning (FL) has emerged as a powerful framework for enabling decentralized intelligence, particularly in scenarios where data privacy and communication constraints are critical. When combined with Multi-Agent Reinforcement Learning (MARL), FL facilitates the training of distributed agents across multiple edge devices or environments without requiring direct access to local data. This synergy allows agents to collaboratively learn shared policies or value functions while preserving data locality and reducing bandwidth usage. In federated MARL, each agent or a group of agents independently interacts with its environment to collect experience and update local models, which are then periodically aggregated using techniques such as Federated Averaging (FedAvg) on a central server or coordinator. These methods help address key challenges in MARL, including scalability, heterogeneity of agent experiences, and security concerns. As such,

federated MARL is increasingly being explored in domains such as autonomous vehicle coordination, distributed robotics, and smart IoT systems, where collaborative learning must occur without centralized data collection.

Multi-Agent Federated Reinforcement Learning (MAFRL) techniques have become a powerful paradigm for establishing stable [114] and intelligent communication networks, particularly in scenarios involving distributed and mobile agents such as UAVs. By combining the decentralized decision-making capabilities of Multi-Agent Reinforcement Learning with the privacy-preserving and communication-efficient nature of federated learning, MAFRL enables each agent to learn locally from its environment while benefiting from global knowledge through periodic model aggregation. This architecture is particularly advantageous in dynamic wireless networks, where maintaining stability in terms of connectivity, data throughput, and fairness is critical. For instance, UAVs acting as aerial relays can collaboratively learn optimal flight paths, power allocations, and scheduling policies without sharing raw data, thereby reducing latency and enhancing scalability. Moreover, the federated approach ensures robustness against network failures or mobility-induced disruptions by enabling asynchronous updates and distributed training. As demonstrated in recent studies, MAFRL achieves improved convergence [115], reduced interference, and enhanced fairness in access to limited communication resources [116], making it a highly effective solution for the future of resilient and adaptive communication networks. It was also used to improve spectrum utilization [117]. MAFRL was applied to optimize the quality of service in various LoRa network slices.

Recent research in Multi-Agent Federated Reinforcement Learning (MAFRL) for UAV applications has yielded a diverse range of frameworks aimed at optimizing communication efficiency, trajectory planning, and resource allocation in dynamic environments. Gao et al. [118] introduced a federated Deep Reinforcement Learning framework for UAV trajectory design, enabling privacy-preserving collaboration across multiple UAVs serving mobile users. Their focus on trajectory learning demonstrated strong performance under mobile user scenarios. However, the study focused on a limited number of UAVs (only five), which raises questions about the scalability. It lacks real-life applications, as it was only validated by simulation. Nie et al. [119] addressed semi-distributed resource management in UAV-aided mobile edge computing (MEC) systems by proposing an MAFRL algorithm that balances computation offloading and power minimization while preserving user privacy using Gaussian noise. But this framework has scalability concerns as it was validated by simulations on a limited number of UAVs, as well as its dependence on a central server. In contrast, Wang et al. [120] applied a distributed federated Deep Reinforcement Learning (DFDRL) approach to optimize UAV trajectories in emergency communication networks, with an emphasis on adaptability and reduced latency in air–ground cooperation. Tarekegn et al. [121] proposed a federated multi-agent DRL system for trajectory control and fair communication (TCFC), which uniquely emphasized fairness in user data rates and applied real channel measurements for link estimation—setting it apart in terms of practical realism. Finally, Han et al. [122] proposed a collaborative learning model for UAV deployment and resource allocation using environment-centric learning, focusing on system-wide throughput and fair access. This framework poses potential privacy concerns as raw observations may still be shared among agents. While all papers demonstrate the scalability and privacy-preserving benefits of federated learning, they differ in primary focus: [118–120] target performance and latency, [121] centers on fairness and real-world deployment, and [122] integrates collaborative policy learning for UAV deployment. Table 5 presents a comparison between these different research articles.

Table 5. Comparison between the MAFRL techniques referenced.

Article	Swarm Size	(Avg. Time of Episode)/#Episodes	Accum. Reward	Application	Pros	Cons
[118]	5	8.093 s/420	2000	Trajectory Design for UAV-assisted networks	Incorporates dueling DDQN and multi-step bootstrapping	It did not converge in some cases, and in others it took >2000 s, scalability and static Server UAV
[119]	10	0.5664 s/20,000	0.4	Resource Management in UAV	use of PPO provides stable convergence	Dependence on Central Server, Scalability
[121]	3	0.738 s/1000	80	Trajectory Control for Multi-UAV Networks	Joint Optimization of Trajectory and Fairness	Assumes stable and reliable communication between UAVs and the server; the use of MADDPG increases training complexity
[122]	6	1.492 s/1200	200	UAV Deployment and Resource Allocation	Decentralized scalability, optimizes UAV positioning and communication resource allocation	No Federated Privacy Mechanism, Energy Efficiency Ignored
[120]	6	-/4000	1.5	Optimizing UAV trajectories	Distributed Privacy-Preserving Design, Trajectory Optimization for Emergency Response	Central Aggregator Dependency, Simplified Channel Models

7. Comparison Between MARL Learning Techniques

Table 6 presents a brief comparison between Multi-Agent Reinforcement Learning (MARL) techniques. This comparison summarizes the key points relevant to UAV control, depending on factors such as the environment, application, and performance requirements. The comparison of Value-Based, Policy-Based, and Federated Multi-Agent Reinforcement Learning (MARL) techniques reveals distinct strengths and limitations across key performance dimensions. Value-based methods, such as QMIX and QTRAN, focus on learning joint or decomposed value functions and are generally efficient in simpler or discrete environments, but they can suffer from scalability issues and instability in partially observable or dynamic settings. In contrast, policy-based approaches like MAPPO and MADDPG directly optimize decentralized or centralized policies, offering better adaptability to continuous control tasks such as UAV formation flying and cooperative maneuvering. These methods typically require more training episodes but yield greater robustness and flexibility in complex multi-agent scenarios. Federated MARL introduces an additional layer of scalability and privacy by enabling agents to train locally while sharing model parameters rather than raw data. Although convergence may be slower due to asynchronous updates and communication delays, federated approaches excel in settings where data locality, communication efficiency, or privacy is a concern—making them highly suitable for distributed UAV swarms or edge AI systems. Ultimately, the choice among these techniques depends on the application's control complexity, environmental dynamics, system scale, and communication constraints.

Table 6. Expanded comparison of MARL learning techniques.

Criterion	Value-Based MARL	Policy-Based MARL	Federated MARL
Core Idea	Learns joint or factorized value functions (e.g., Q-values)	Learns decentralized or centralized policies directly	Distributes learning across agents while preserving data locality
Representative Algorithms	QMIX, VDN, QTRAN, WQMIX	MAPPO, MADDPG, TRPO, HAPPO	FedRL, FedMARL, FedAvg+MARL
Action Space	Discrete	Continuous and discrete	Both (depends on local model)
Performance	High in structured/cooperative tasks	High in continuous, dynamic tasks	Moderate to high; depends on aggregation frequency and bandwidth
Environment	Best in fully observable or low-noise discrete settings	Works well in dynamic, partially observable, or continuous environments	Well-suited for distributed, privacy-sensitive, and bandwidth-limited environments
Scalability	Moderate (limited by value function complexity)	High (decentralized execution is effective)	High (supports large-scale distributed systems)
Communication Overhead	Low to moderate	Moderate (depends on policy sharing)	Low (only model parameters shared)
Training Stability	Sensitive to agent number and observability	More stable with proper exploration (e.g., GAE)	Depends on sync method and local training accuracy
Convergence Speed	Fast in simple settings	Moderate; requires more samples but is stable	Slower due to delayed aggregation and heterogeneity
Adaptability to Dynamic Environments	Low to moderate	High	High (supports model personalization)
Privacy Preservation	Not supported inherently	Not supported inherently	Strong (data remains local)
Suitability for UAV Applications	Suitable for cooperative discrete tasks (e.g., coverage)	Ideal for continuous control (e.g., formation, tracking)	Ideal for distributed UAV systems with privacy and bandwidth constraints

When comparing some of the most common MARL techniques used for applications related to UAVs, as presented in Table 7, one should note that the training time of Multi-Agent Reinforcement Learning (MARL) techniques is influenced by several critical factors. These include the number of agents involved, as more agents increase the dimensionality of the joint action space and coordination complexity. The algorithm type also plays a role; for example, value-based methods like QMIX may train faster but struggle with scalability, whereas policy-based methods like MAPPO can offer stability at the cost of longer convergence. Communication overhead, especially in centralized or hybrid systems, can slow training when agents frequently exchange state or policy information. Additionally, environmental complexity, such as dynamic obstacles or partial observability, and the reward structure (sparse vs. dense) significantly affect the sample efficiency and number of episodes required to reach optimal performance. This comparison also highlights a diverse landscape of algorithmic strengths tailored to different operational demands. SERT-QDN [78] stands out with high scalability, robust fault tolerance, and high success rates,

making it ideal for fault-tolerant swarm operations in harsh environments, although it demands moderate computational resources. QMIX [68] and QTRAN [79] present trade-offs; QMIX offers high success with moderate scalability but limited fault tolerance and lower training complexity, whereas QTRAN, while constrained in scalability and convergence, demands high computational and energy resources due to its constraint-based exploration. MAVEN [81] and VDAC [123] both feature moderate capabilities but diverge in exploration strategy, with MAVEN leveraging latent variables and VDAC using decentralized advantages suitable for entropy-regularized applications. ROMA [124] and MAPPO [90] deliver high performance in scalability and convergence, with ROMA employing role assignment for structured swarm behavior and MAPPO suited for formation control with general advantage estimation. HAPPO [102], a hierarchical variant, matches MAPPO's strengths and adds precision for coordinated missions. Lastly, FACMAC [125,126] and WQMIX [127] focus on bandwidth-awareness and energy efficiency, respectively, with both showing strong success rates and scalability. Overall, the selection of a method depends on mission-specific requirements such as fault tolerance, deployment readiness, and computational constraints.

Table 7. Comparative analysis of MARL techniques for UAV applications.

Technique	Scalability	Fault Tolerance	Success Rate	Exploration Strategy	Training Time/ Convergence	Special Features/ Use Case
SERT-QDN [78]	High	Strong	High	Similarity-based	Moderate	Fault-tolerant swarms in harsh environments
QMIX [68]	Moderate	Limited	High	ε-greedy	Fast	Cooperative exploration, low training complexity
QTRAN [79]	Low	None	Moderate	Constraint-based	Slow	Computationally heavy, resource-intensive coordination
MAVEN [81]	Moderate	None	High	Latent variable	Moderate	Supports diverse behavior patterns
VDAC [123]	Moderate	None	Moderate	Decentralized advantage	Fast	Entropy-regularized learning
ROMA [124]	High	None	High	Role assignment	Fast	Structured swarm roles for robust coordination
MAPPO [90]	High	Limited	High	GAE	Fast	Stable formation control, scalable and robust
HAPPO [102]	High	Limited	High	GAE	Moderate	Precision missions, hierarchical advantage policy
FACMAC [125,126]	Moderate	None	High	Entropy maximization	Moderate	Bandwidth-aware policies, decentralized training
WQMIX [127]	Moderate	None	High	Weighted mixing	Moderate	Energy-efficient swarm learning

8. Open Research Problems

Despite the significant advancements and promising results achieved in UAV control [128] using Multi-Agent Reinforcement Learning (MARL), several open research problems remain. Addressing these issues is crucial for the further development and deployment of MARL in UAV systems [129]. This section outlines key areas that require further investigation.

1. Scalability to Large-Scale Systems: The growing demand for scalable, efficient, and real-time aerial operations has underscored the need for deploying large numbers of UAVs across various domains. In search-and-rescue missions, fleets of UAVs can rapidly survey expansive or hazardous areas; in agriculture, they enable large-scale crop monitoring and precision farming. Military and defense sectors rely on multiple UAVs for surveillance, reconnaissance, and coordinated strikes. Similarly, environmental monitoring, logistics, and infrastructure inspection benefit from UAV swarms that provide wide coverage, improve redundancy, and enhance operational efficiency. These applications collectively justify the push toward large-scale UAV coordination and automation. However, while MARL has shown success in controlling small to medium-sized UAV swarms, scaling these techniques to larger networks remains a challenge. Research is needed to develop algorithms that can efficiently manage a large number of UAVs without compromising performance or increasing computational complexity. In [130] , a novel multi-layer topology network and consensus control approach for a large-scale UAV swarm moving under a stable configuration was presented. The proposed topology can make the swarm remain robust in spite of the large number of UAVs. Although this control approach is a solid contribution to UAV swarm coordination, it lacks provisions for handling agent failures, communication breakdowns, or adversarial conditions, which are critical for resilient swarm coordination. The simulation environment also lacks dynamic complexity such as moving obstacles and wind turbulence. In [131], the authors introduced an agent masking technique that allows a standard Deep Q-Network (DQN) algorithm to scale effectively to large swarms, even when training is performed on comparatively smaller swarm sizes. The paper does not analyze convergence behavior, learning curves, or policy variance across training runs. It is unclear how stable or sample-efficient the proposed approach is. Also, no analysis is provided on how performance degrades or scales with increasing swarm size or communication complexity.
2. Real-Time Learning and Adaptation: In dynamic and unpredictable environments, UAVs must be able to learn and adapt in real-time. Developing MARL algorithms that can quickly respond to changes and learn from new experiences without extensive retraining is a critical area for future research. For example, [131] presents a simulation of a large swarm size, but the simulation environment is relatively simplistic, lacking dynamic elements such as mobile obstacles, environmental noise, or sensor failures. Real-world UAV operations often involve uncertainties (e.g., GPS noise and wind) and constrained communication bandwidth, which are not modeled or discussed. While [132] presents some promising results, it lacks scaling to large-scale swarms.
3. Control of a Swarm of UAVs in areas of high turbulence: Controlling a swarm of UAVs in areas of high turbulence involves significant challenges that necessitate advanced strategies and robust algorithms due to the unpredictable nature of turbulent airflows. Key issues include maintaining stability amid rapidly changing conditions, managing increased computational complexity, ensuring robustness and stability, and dealing with communication disruptions. While [133] presents a framework for optimizing UAV performance in turbulent environments using a cascaded model predictive control algorithm and Pixhawk hardware, it lacks scalability as it focuses only on the

control of one UAV. It also lacks the generalization of the convergence and resource management, which is expected to be significant in such environments. Research on the control of a swarm of UAVs in areas of high turbulence is scarce.

4. Energy Efficiency and Resource Management: UAVs typically have limited energy resources. Efficient energy management is crucial for prolonged operation. Future research should focus on integrating energy-aware strategies into MARL frameworks to optimize the use of power and computational resources. Jung et al. [134] presents a framework for enabling cooperative UAV teams to track multiple dynamic ground targets. While the proposed method reduces communication delay by 64% and energy consumption by 78% compared with traditional MESH networks for 10 UAVs, several limitations and weaknesses reduce the generalizability and robustness of the work as it lacks scalability and convergence studies. Ma et al. [135] presents an improved DRL-based energy-efficient UAV control for maximum lifecycle which is based on DDPG. It lacks the study of Transfer Learning as well as convergence in case of large scale swarms.
5. Transfer Learning and Generalization: Developing MARL algorithms that can generalize across different tasks and environments is a significant challenge. Transfer Learning techniques, which allow knowledge gained in one context to be applied in another, could significantly enhance the flexibility and applicability of MARL for UAV control. The learned policies in [129,131,132,134] are not tested on new environments, larger team sizes, or with altered dynamics (e.g., target speed or number). This absence of generalization testing makes it hard to trust the method's robustness. Reference [136] presents Real-Time Policy Optimization for UAV Swarms based on evolution strategies but lacks scalability and does not guarantee convergence.
6. Convergence: UAVs must make decisions in real time under stringent safety constraints and uncertain environmental dynamics (e.g., wind disturbances, obstacle movement), which means the RL agent must generalize well from limited, high-risk data. This often results in sparse and delayed reward signals, further slowing convergence. Additionally, simulation-to-real-world transfer is another major hurdle: policies trained in a simulation may not converge effectively in the real world due to discrepancies between simulated and physical dynamics (the "reality gap"). In multi-agent UAV systems, convergence is even more complex. The environment becomes non-stationary from each agent's perspective because other agents are simultaneously learning and changing their policies. This non-stationarity breaks the standard assumptions of RL algorithms, degrading learning performance and making stable convergence much harder to achieve. Techniques like Centralized Training with Decentralized Execution (CTDE) and frameworks like MAPPO and QMIX have been proposed to address this, but convergence still depends heavily on careful hyperparameter tuning, reward shaping, and training stability mechanisms. Wang et al. [130], Blais and Akhloufi [131] provide control approaches for large-scale systems but did not present any studies on the convergence. Jung et al. [134] present an energy efficient approach but lacks the study of convergence.

9. Conclusions

This paper provided a comprehensive survey of the control of drone swarms using Multi-Agent Reinforcement Learning (MARL). The integration of UAVs with MARL has emerged as a promising solution for tackling complex control and coordination challenges in dynamic and uncertain environments. The limitations of MARL in providing a robust and flexible framework for managing multiple UAVs in intricate scenarios have been

pointed out. Further research is needed to enhance the efficiency, autonomy, and reliability of UAV systems to unlock the full potential of UAVs across diverse sectors.

Author Contributions: Conceptualization, C.C.E., T.E., A.A. and T.K.; methodology, C.C.E., T.E. and T.K.; investigation, C.C.E.; writing—original draft preparation, C.C.E., writing—modified draft, T.E.; writing—review and editing, C.C.E., T.E., A.A. and T.K.; project administration, T.E. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Colomina, I.; Molina, P. Unmanned aerial systems for photogrammetry and remote sensing: A review. *ISPRS J. Photogramm. Remote Sens.* **2014**, *92*, 79–97. [CrossRef]
2. Nex, F.; Remondino, F. UAV for 3D mapping applications: A review. *Appl. Geomat.* **2014**, *6*, 1–15. [CrossRef]
3. Zhang, C.; Kovacs, J.M. The application of small unmanned aerial systems for precision agriculture: A review. *Precis. Agric.* **2012**, *13*, 693–712. [CrossRef]
4. Floreano, D.; Wood, R. Science, technology and the future of small autonomous drones. *Nature* **2015**, *521*, 460–466. [CrossRef] [PubMed]
5. Hugenholtz, C.H.; Moorman, B.J.; Riddell, K.; Whitehead, K. Small unmanned aircraft systems for remote sensing and Earth science research. *Eos Trans. Am. Geophys. Union* **2012**, *93*, 236–236. [CrossRef]
6. Gertler, J. *U.S. Unmanned Aerial Systems*; Technical Report. 2012. Available online: https://sgp.fas.org/crs/natsec/R42136.pdf (accessed on 25 May 2025)
7. Kim, D. PPO: Efficient, Stable and Scalable Policy Optimization. 2021. Available online: https://medium.com/@kdk199604/ppo-efficient-stable-and-scalable-policy-optimization-15b5b9c74a88 (accessed on 28 May 2025).
8. Zegart, A.B. *Spies, Lies, and Algorithms: The History and Future of American Intelligence*; Princeton Press: Princeton, NJ, USA, 2022.
9. Paucar, C.; Morales, L.; Pinto, K.; Sánchez, M.; Rodríguez, R.; Gutierrez, M.; Palacios, L. Use of Drones for Surveillance and Reconnaissance of Military Areas. In *Developments and Advances in Defense and Security*; Rocha, Á., Guarda, T., Eds.; Springer: Cham, Switzerland, 2018; pp. 119–132.
10. Singer, P. *Wired for War: The Robotics Revolution and Conflict in the 21st Century*; Penguin Books: London, UK, 2009.
11. Yoo, L.S.; Lee, J.H.; Lee, Y.K.; Jung, S.K.; Choi, Y. Application of a Drone Magnetometer System to Military Mine Detection in the Demilitarized Zone. *Sensors* **2021**, *21*, 3175. [CrossRef]
12. Gusterson, H. *Drone: Remote Control Warfare*; MIT Press: Cambridge, MA, USA, 2016.
13. Alghamdi, S.; Alahmari, S.; Yonbawi, S.; Alsaleem, K.; Ateeq, F.; Almushir, F. Autonomous Navigation Systems in GPS-Denied Environments: A Review of Techniques and Applications. In Proceedings of the 2025 11th International Conference on Automation, Robotics, and Applications (ICARA), Zagreb, Croatia, 12–14 February 2025; pp. 290–299. [CrossRef]
14. Shaik, N.; Chitralingappa, P.; Harichandana, B. Machine Learning in Drones for Enhancing Autonomous Flight and Decision-Making. In Proceedings of the 3rd International Conference on Optimization Techniques in the Field of Engineering (ICOFE-2024), Online 15 November 2024. Available online: https://ssrn.com/abstract=5088972 (accessed on 25 May 2025).
15. Boroujeni, S.P.H.; Razi, A.; Khoshdel, S.; Afghah, F.; Coen, J.L.; O'Neill, L.; Fule, P.; Watts, A.; Kokolakis, N.M.T.; Vamvoudakis, K.G. A comprehensive survey of research towards AI-enabled unmanned aerial systems in pre-, active-, and post-wildfire management. *Inf. Fusion* **2024**, *108*, 102369. [CrossRef]
16. Şengönül, E.; Samet, R.; Abu Al-Haija, Q.; Alqahtani, A.; Alturki, B.; Alsulami, A.A. An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey. *Appl. Sci.* **2023**, *13*, 4956. [CrossRef]
17. Tai, L.; Paolo, G.; Liu, M. Virtual-to-real Deep Reinforcement Learning: Continuous control of mobile robots for mapless navigation. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 31–36. [CrossRef]
18. Wen, J.; Tu, Q.; Wu, H. Dynamic Environment Adaptive Fusion. In Proceedings of the International Conference on Image, Vision and Intelligent Systems 2024 (ICIVIS 2024): Volume II, Xining, China, 16–17 June 2024; Springer Nature: Singapore, 2025; Volume 1362, p. 403.
19. Katkuri, A.V.R.; Madan, H.; Khatri, N.; Abdul-Qawy, A.S.H.; Patnaik, K.S. Autonomous UAV navigation using deep learning-based computer vision frameworks: A systematic literature review. *Array* **2024**, *23*, 100361. [CrossRef]

20. Tiwari, R.; Srinivaas, A.; Velamati, R.K. Adaptive Navigation in Collaborative Robots: A Reinforcement Learning and Sensor Fusion Approach. *Appl. Syst. Innov.* **2025**, *8*, 9. [CrossRef]
21. Teixeira, K.; Miguel, G.; Silva, H.S.; Madeiro, F. A Survey on Applications of Unmanned Aerial Vehicles Using Machine Learning. *IEEE Access* **2023**, *11*, 117582–117621. [CrossRef]
22. Albanese, A.; Nardello, M.; Brunelli, D. Low-power deep learning edge computing platform for resource constrained lightweight compact UAVs. *Sustain. Comput. Inform. Syst.* **2022**, *34*, 100725. [CrossRef]
23. Kabas, B. Autonomous UAV Navigation via Deep Reinforcement Learning Using PPO. In Proceedings of the 2022 30th Signal Processing and Communications Applications Conference (SIU), Safranbolu, Turkey, 15–18 May 2022; pp. 1–4. [CrossRef]
24. Alshaibani, W.T.; Shayea, I.; Caglar, R.; Din, J.; Daradkeh, Y.I. Mobility Management of Unmanned Aerial Vehicles in Ultra–Dense Heterogeneous Networks. *Sensors* **2022**, *22*, 6013. [CrossRef] [PubMed]
25. Chung, S.J.; Paranjape, A.A.; Dames, P.; Shen, S.; Kumar, V. A survey on aerial swarm robotics. *IEEE Trans. Robot.* **2018**, *34*, 837–855. [CrossRef]
26. Tsykunov, E.; Agishev, R.; Ibrahimov, R.; Labazanova, L.; Tleugazy, A.; Tsetserukou, D. SwarmTouch: Guiding a Swarm of Micro-Quadrotors with Impedance Control using a Wearable Tactile Interface. *IEEE Trans. Haptics* **2019**, *12*, 363–374. [CrossRef] [PubMed]
27. Ahmed, Z.; Xiong, X. Adaptive Impedance Control of Multirotor UAV for Accurate and Robust Path Following. *Machines* **2024**, *12*, 868. [CrossRef]
28. Nguyen, M.T.; Nguyen, C.V.; Truong, L.H.; Le, A.M.; Quyen, T.V.; Masaracchia, A.; Teague, K.A. Electromagnetic Field Based WPT Technologies for UAVs: A Comprehensive Survey. *Electronics* **2020**, *9*, 461. [CrossRef]
29. Turk, Y.; Aydın, A.; Eker, R. Comparison of Autonomous and Manual UAV Flights in Determining Forest Road Surface Deformations. *Eur. J. For. Eng.* **2022**, *8*, 77–84. [CrossRef]
30. Thu, K.M.; Gavrilov, A.I. Designing and Modeling of Quadcopter Control System Using L1 Adaptive Control. *Procedia Comput. Sci.* **2016**, *103*, 528–535.
31. Chen, L.; Wang, G. Attitude Stabilization for a Quadcopter Helicopter using a PD Controller. In Proceedings of the 3rd IFAC International Conference on Intelligent Control and Automation Science, Chengdu, China, 2–4 September 2013.
32. Goudarzi, H.; Richards, A. Semi-autonomous drone control with safety analysis. *Drone Syst. Appl.* **2023**, *11*, 1–13. [CrossRef]
33. Bandarupalli, A.; Swarup, D.; Weston, N.; Chaterji, S. Persistent Airborne Surveillance using Semi-Autonomous Drone Swarms. In Proceedings of the 7th Workshop on Micro Aerial Vehicle Networks, Systems, and Applications, Virtual, 24 June 2021; pp. 19–24. [CrossRef]
34. Klingler, N. YOLOv3: Real-Time Object Detection Algorithm (Guide). 2022. Available online: https://viso.ai/deep-learning/yolov3-overview/ (accessed on 5 May 2025).
35. Eluganti, A.R.; Mezins, A. Semi-Automated Drone for Avionics Navigation Signal Verification and Methods of Operation and Use Thereof. U.S. Patent US20180308298A1, 25 October 2018.
36. Hutton, C. Augmented Reality Interfaces for Semi-Autonomous Drones. In Proceedings of the IEEE Conference on Virtual Reality and 3D User Interfaces, Osaka, Japan, 23–27 March 2019, pp. 1361–1362. [CrossRef]
37. Rezwan, S.; Choi, W. Artificial Intelligence Approaches for UAV Navigation: Recent Advances and Future Challenges. *IEEE Access* **2022**, *10*, 26320–26339. [CrossRef]
38. Elistair. 5 Ways Drones Enhance Military Security. *Elistair* 2023. Available online: https://elistair.com/resources/military-drones/5-ways-drones-enhance-military-security/ (accessed on 23 April 2025).
39. Elistair. Search and Rescue Drones: Supporting Rescue Teams in Critical Situations. *Elistair* 2023. Available online: https://elistair.com/search-and-rescue-drones/ (accessed on 23 April 2025).
40. DSLRPros. Disaster Response Drones and Payloads. *DSLRPros* 2023. Available online: https://www.dslrpros.com/pages/uses/disaster-response-drones (accessed on 23 April 2025).
41. Innovations, D. Long-Range Surveillance Drone. *Draganfly* 2023. Available online: https://draganfly.com/long-range-surveillance-drone/ (accessed on 23 April 2025).
42. Contributors, W. Zipline (Drone Delivery Company). 2024. Available online: https://en.wikipedia.org/wiki/Zipline_(drone_delivery_company) (accessed on 23 April 2025).
43. Hall, G. Behavior Shaping and Analysis in Multi-Agent Reinforcement Learning: Experiments in Combat Tactics and Survivability. Master's Thesis, The University of Texas at San Antonio, San Antonio, TX, USA, 2022.
44. Noordin, A.; Basri, M.A.M.; Mohammed, Z.; Abidin, A.F.Z. Modelling and PSO Fine-tuned PID Control of Quadrotor UAV. *Int. J. Adv. Sci. Eng. Inf. Eng.* **2017**, *7*, 1367–1373.
45. Abd-El-Wahed, W.F.; Mousa, A.A.; El-Shorbagy, M.A. Integrating Particle Swarm Optimization with Genetic Algorithms for Solving Nonlinear Optimization Problems. *Elsevier J. Comput. Appl. Math.* **2009**, *235*, 1446–1453. [CrossRef]

46. Knopp, M.; Aykin, C.; Feldmaier, J.; Shen, H. Formation Control using GQ(λ) Reinforcement Learning. In Proceedings of the 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Lisbon, Portugal, 28 August–1 September 2017; pp. 1043–1048. [CrossRef]
47. AlMahamid, F.; Grolinger, K. Autonomous Unmanned Aerial Vehicle navigation using Reinforcement Learning: A systematic review. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105321. [CrossRef]
48. Sheng, Y.; Liu, H.; Li, J.; Han, Q. UAV Autonomous Navigation Based on Deep Reinforcement Learning in Highly Dynamic and High-Density Environments. *Drones* **2024**, *8*, 516. [CrossRef]
49. Hodge, V.J.; Hawkings, R.; Alexander, R. Deep Reinforcement Learning for drone navigation using sensor data. *Neural Comput. Appl.* **2020**, *33*, 2015–2033. [CrossRef]
50. Kim, H.; Choi, J.; Do, H.; Lee, G.T. A Fully Controllable UAV Using Curriculum Learning and Goal-Conditioned Reinforcement Learning: From Straight Forward to Round Trip Missions. *Drones* **2025**, *9*, 26. [CrossRef]
51. Watkins, C.J.C.H. Learning from Delayed Rewards. Ph.D. Thesis, King's College, Cambridge, UK, 1989.
52. Rummery, G.A.; Niranjan, M. *On-Line Q-Learning Using Connectionist Systems*; University of Cambridge, Department of Engineering: Cambridge, UK, 1994; Volume 37.
53. Sutton, R.S.; Barto, A.G. Temporal-difference learning. In *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 1998; pp. 133–160.
54. Sutton, R.S.; McAllester, D.; Singh, S.; Mansour, Y. Policy gradient methods for Reinforcement Learning with function approximation. *Adv. Neural Inf. Process. Syst.* **2000**, *12*, 1057–1063.
55. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with Deep Reinforcement Learning. *arXiv* **2019**, arXiv:cs.LG/1509.02971.
56. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor–critic Methods. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; PMLR, Volume 80, pp. 1587–1596.
57. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor–Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; Dy, J., Krause, A., Eds.; PMLR, Volume 80, pp. 1861–1870.
58. Chen, Y.J.; Chang, D.K.; Zhang, C. Autonomous Tracking using a Swarm of UAVs: A Constrained Multi-Agent Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* **2020**, *69*, 13702–13717. [CrossRef]
59. Jagtap, R. Understanding Markov Decision Process (MDP). 2024. Available online: https://builtin.com/machine-learning/markov-decision-process (accessed on 1 April 2025).
60. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Nefian, A. Cooperative and Distributed Reinforcement Learning of Drones for Field Coverage. *arXiv* **2020**, arXiv:1803.07250v2 [cs.RO].
61. Wong, A.; Bäck, T.; Kononova, A.V.; Plaat A. Deep multiagent reinforcement learning: Challenges and directions. *Artif. Intell. Rev.* **2023**. *56*, 5023–5056. [CrossRef]
62. Canese, L.; Cardarilli, G.C.; Di Nunzio, L.; Fazzolari, R.; Giardino, D.; Re, M.; Spanò, S. Multi-Agent Reinforcement Learning: A Review of Challenges and Applications. *Appl. Sci.* **2021**, *11*, 4948. [CrossRef]
63. Liang, J.; Miao, H.; Li, K.; Tan, J.; Wang, X.; Luo, R.; Jiang, Y. A Review of Multi-Agent Reinforcement Learning Algorithms. *Electronics* **2025**, *14*, 820. [CrossRef]
64. Konda, V.; Tsitsiklis, J. Actor–critic Algorithms. In *Advances in Neural Information Processing Systems*; Solla, S., Leen, T., Müller, K., Eds.; MIT Press: Cambridge, MA, USA, 1999; Volume 12.
65. Lai, X.; Yang, Z.; Xie, J.; Liu, Y. Reinforcement Learning in transportation research: Frontiers and future directions. *Multimodal Transp.* **2024**, *3*, 100164. [CrossRef]
66. Bhatnagar, S.; Sutton, R.S.; Ghavamzadeh, M.; Lee, M. Natural actor–critic algorithms. *Automatica* **2009**, *45*, 2471–2482. [CrossRef]
67. Zhang, K.; Yang, Z.; Başar, T. Multi-Agent Reinforcement Learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Cham, Switzerland, 2021; pp. 321–384. [CrossRef]
68. Rashid, T.; Samvelyan, M.; de Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51.
69. Iqbal, S.; De Witt, C.A.S.; Peng, B.; Boehmer, W.; Whiteson, S.; Sha, F. Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning. In Proceedings of the 38th International Conference on Machine Learning, Virtual, 18–24 July 2021; Meila, M., Zhang, T., Eds.; PMLR, Volume 139, pp. 4596–4606.
70. Pham, H.X.; La, H.M. Performance Comparison of Function Approximation-based Q-Learning Algorithms for UAV Navigation. In Proceedings of the 2018 IEEE International Conference on Real-time Computing and Robotics (RCAR), Kandima, Maldives, 1–5 August 2018; pp. 1–6. [CrossRef]
71. Fourati, F.; Aggarwal, V.; Alouini, M.S. Stochastic Q-learning for Large Discrete Action Spaces. *arXiv* **2024**, arXiv:cs.LG/2405.10310.

72. Tessler, C.; Zahavy, T.; Cohen, D.; Mankowitz, D.J.; Mannor, S. Action Assembly: Sparse Imitation Learning for Text Based Games with Combinatorial Action Spaces. *arXiv* **2020**, arXiv:cs.LG/1905.09700.
73. Duryea, E.; Ganger, M.; Hu, W. Exploring Deep Reinforcement Learning with Multi Q-Learning. *Intell. Control Autom.* **2016**, *7*, 129–144. [CrossRef]
74. Song, S.; Zhao, M.; Gong, D.; Zhu, M. Convergence and stability analysis of value iteration Q-learning under non-discounted cost for discrete-time optimal control. *Neurocomputing* **2024**, *606*, 128370. [CrossRef]
75. Minh, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602v1 [cs.LG].
76. Venturini, F.; Mason, F.; Pase, F.; Chiariotti, F.; Testolin, A.; Zanella, A.; Zorzi, M. Distributed Reinforcement Learning for Flexible UAV Swarm Control with Transfer Learning Capabilities. In Proceedings of the DroNet'20, Toronto, ON, Canada, 19 June 2020; pp. 1–6. [CrossRef]
77. Zhang, G.; Li, Y.; Xu, X.; Dai, H. Efficient Training Techniques for Multi-Agent Reinforcement Learning in Combat Tasks. *IEEE Access* **2019**, *7*, 109301–109310. [CrossRef]
78. Ramezani, M.; Amiri Atashgah, M.A.; Rezaee, A. A Fault-Tolerant Multi-Agent Reinforcement Learning Framework for Unmanned Aerial Vehicles–Unmanned Ground Vehicle Coverage Path Planning. *Drones* **2024**, *8*, 537. [CrossRef]
79. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. Qtran: Learning to factorize with transformation for cooperative Multi-Agent Reinforcement Learning. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.
80. Zhu, Y.; Chen, M.; Wang, S.; Hu, Y.; Liu, Y.; Yin, C. Collaborative Reinforcement Learning Based Unmanned Aerial Vehicle (UAV) Trajectory Design for 3D UAV Tracking. *IEEE Trans. Mob. Comput.* **2024**, *23*, 10787–10802. [CrossRef]
81. Mahajan, A.; Rashid, T.; Samvelyan, M.; Whiteson, S. Maven: Multi-agent variational exploration. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 7613–7624 Available online:https://dl.acm.org/doi/10.5555/3454287.3454971 (accessed on 4 March 2025).
82. Karunakaran, D. REINFORCE: A Policy-Gradient Based Reinforcement Learning Algorithm. 2020. Available online: https://medium.com/intro-to-artificial-intelligence/reinforce-a-policy-gradient-based-reinforcement-learning-algorithm-84bde440c816 (accessed on 4 March 2025).
83. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent Deep Reinforcement Learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
84. Lowe, R.; WU, Y.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-Agent Actor–critic for Mixed Cooperative-Competitive Environments. In *Proceedings of the Advances in Neural Information Processing Systems*; Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
85. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An introduction*; MIT Press: Cambridge, MA, USA, 2018.
86. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 7–9 July 2015; Bach, F., Blei, D., Eds.; Volume 37, pp. 1889–1897.
87. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:cs.LG/1707.06347.
88. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 20–22 June 2016; Balcan, M.F., Weinberger, K.Q., Eds.; Volume 48, pp. 1928–1937.
89. Foerster, J.; Assael, Y.M.; de Freitas, N.; Whiteson, S. Learning to communicate with deep Multi-Agent Reinforcement Learning. In *Proceedings of the Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29, pp. 2137–2145.
90. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; WU, Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.
91. Christianos, F.; Schäfer, L.; Albrecht, S. Shared Experience Actor–critic for Multi-Agent Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10707–10717.
92. Khani, M.; Sadr, M.M.; Jamali, S. Deep Reinforcement Learning-based resource allocation in multi-access edge computing. *Concurr. Comput. Pract. Exp.* **2024**, *36*, e7995,
93. Iqbal, S.; Sha, F. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In Proceedings of the 36th International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; Chaudhuri, K., Salakhutdinov, R., Eds.; PMLR, Volume 97, pp. 2961–2970.
94. Jiang, J.; Lu, Z. Learning Attentional Communication for Multi-Agent Cooperation. In *Proceedings of the Advances in Neural Information Processing Systems*; Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31, pp. 7254–7264.
95. Huang, Y. Deep Q-Networks. In *Deep Reinforcement Learning: Fundamentals, Research and Applications*; Springer: Singapore, 2020; pp. 135–160. [CrossRef]

96. Koch, W.; Mancuso, R.; West, R.; Bestavros, A. Reinforcement Learning for UAV attitude control. *ACM Trans. Cyber-Phys. Syst.* **2019**, *3*, 1–21.
97. Qie, H.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; Wang, L. Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning. *IEEE Access* **2019**, *7*, 146264–146272. [CrossRef]
98. Schulman, J.; Levine, S.; Moritz, P.; Jordan, M.; Abbeel, P. Trust Region Policy Optimization. *arXiv* **2017**, arXiv:1502.05477v5 [cs.LG].
99. Li, H.; He, H. Multiagent Trust Region Policy Optimization. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *35*, 12873–12887. [CrossRef]
100. OpenAI Spinning Up. Trust Region Policy Optimization (TRPO). 2025. Available online: https://spinningup.openai.com/en/latest/algorithms/trpo.html (accessed on 25 June 2025).
101. Replicable-MARL. Trust Region Policy Optimization Family. 2023. Available online: https://marllib.readthedocs.io/en/latest/algorithm/trpo_family.html (accessed on 25 June 2025).
102. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning. In Proceedings of the International Conference on Learning Representations, Virtual, 25–29 April 2022.
103. Wen, Y.; Chen, H.; Yang, Y.; Tian, Z.; Li, M.; Chen, X.; Wang, J. A Game-Theoretic Approach to Multi-Agent Trust Region Optimization. In *Distributed Artificial Intelligence*; Springer Nature: Cham, Switzerland, 2023; pp. 74–87. [CrossRef]
104. Alharbi, A.M.; Alshehri, G.; Elhag, S. Reinforcement Learning of Emerging Swarm Technologies: A Literature Review. In *Proceedings of the Future Technologies Conference* ; Springer Nature: Cham, Switzerland, 2024; pp. 478–494.
105. Chen, D.; Qi, Q.; Fu, Q.; Wang, J.; Liao, J.; Han, Z. Transformer-Based Reinforcement Learning for Scalable Multi-UAV Area Coverage. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 10062–10077. [CrossRef]
106. Su, K.; Qian, F. Multi-UAV Cooperative Searching and Tracking for Moving Targets Based on Multi-Agent Reinforcement Learning. *Appl. Sci.* **2023**, *13*, 1905. [CrossRef]
107. Chen, D.; Qi, Q.; Zhuang, Z.; Wang, J.; Liao, J.; Han, Z. Mean Field Deep Reinforcement Learning for Fair and Efficient UAV Control. *IEEE Internet Things J.* **2021**, *8*, 813–828. [CrossRef]
108. Jiang, J.; Contributors, M. MARLlib Documentation: PPO Family of Algorithms. 2023. Available online: https://marllib.readthedocs.io/en/latest/algorithm/ppo_family.html (accessed on 23 June 2025).
109. Liu, Y.; Liu, H.; Tian, Y.; Sun, C. Reinforcement Learning based on Two-level Control Framework of UAV Swarm for Cooperative Persistent Surveillance in an Unknown Urban Area. *Aerosp. Sci. Technol.* **2020**, *98*, 105671. [CrossRef]
110. Panerati, J.; Zheng, H.; Zhou, S.; Xu, J.; Prorok, A.; Schoellig, A.P. Learning to Fly–A Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control. *arXiv* 2021, arXiv:cs.RO/2103.02142].
111. Liao, G.; Wang, J.; Yang, D.; Yang, J. Multi-UAV Escape Target Search: A Multi-Agent Reinforcement Learning Method. *Sensors* **2024**, *24*, 6859. [CrossRef]
112. Guo, Y.; Liu, J.; Yu, R.; Hang, P.; Sun, J. MAPPO-PIS: A Multi-Agent Proximal Policy Optimization Method with Prior Intent Sharing for CAVs' Cooperative Decision-Making. In *Computer Vision–ECCV 2024 Workshops*; Springer: Cham, Switzerland, 2025; pp. 244–263. [CrossRef]
113. Liu, Y.; Li, X.; Wang, J.; Wei, F.; Yang, J. Reinforcement-Learning-Based Multi-UAV Cooperative Search for Moving Targets in 3D Scenarios. *Drones* **2024**, *8*, 378. [CrossRef]
114. Shiri, H.; Park, J.; Bennis, M. Communication-Efficient Massive UAV Online Path Control: Federated Learning Meets Mean-Field Game Theory. *IEEE Trans. Commun.* **2020**, *68*, 6840–6857. [CrossRef]
115. Krouka, M.; Elgabli, A.; Issaid, C.B.; Bennis, M. Communication-Efficient and Federated Multi-Agent Reinforcement Learning. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 311–320. [CrossRef]
116. Seid, A.M.; Erbad, A.; Abishu, H.N.; Albaseer, A.; Abdallah, M.; Guizani, M. Multiagent Federated Reinforcement Learning for Resource Allocation in UAV-Enabled Internet of Medical Things Networks. *IEEE Internet Things J.* **2023**, *10*, 19695–19711. [CrossRef]
117. Chang, H.H.; Song, Y.; Doan, T.T.; Liu, L. Federated Multi-Agent Deep Reinforcement Learning (Fed-MADRL) for Dynamic Spectrum Access. *IEEE Trans. Wirel. Commun.* **2023**, *22*, 5337–5348. [CrossRef]
118. Gao, Y.; Liu, M.; Yuan, X.; et al. Federated Deep Reinforcement Learning based trajectory design for UAV-assisted networks with mobile ground devices. *Sci. Rep.* **2024**, *14*, 22753. [CrossRef]
119. Nie, Y.; Zhao, J.; Gao, F.; Yu, F.R. Semi-Distributed Resource Management in UAV-Aided MEC Systems: A Multi-Agent Federated Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* **2021**, *70*, 13162–13173. [CrossRef]
120. Wu, S.; Xu, W.; Wang, F.; Li, G.; Pan, M. Distributed Federated Deep Reinforcement Learning Based Trajectory Optimization for Air-Ground Cooperative Emergency Networks. *IEEE Trans. Veh. Technol.* **2022**, *71*, 9107–9112. [CrossRef]
121. Tarekegn, G.B.; Tesfaw, B.A.; Juang, R.T.; Saha, D.; Tarekegn, R.B.; Lin, H.P.; Tai, L.C. Trajectory Control and Fair Communications for Multi-UAV Networks: A Federated Multi-Agent Deep Reinforcement Learning Approach. *IEEE Trans. Wirel. Commun.* **2025**, 1. [CrossRef]

122. Dai, Z.; Zhang, Y.; Zhang, W.; Luo, X.; He, Z. A Multi-Agent Collaborative Environment Learning Method for UAV Deployment and Resource Allocation. *IEEE Trans. Signal Inf. Process. Over Netw.* **2022**, *8*, 120–130. [CrossRef]
123. Su, J.; Adams, S.; Beling, P. Value-Decomposition Multi-Agent Actor–critics. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 11352–11360. [CrossRef]
124. Wang, T.; Zhang, H.D.; Yang, J.; Zheng, W.; Wang, H.; Zhang, C. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In Proceedings of the International Conference on Machine Learning (ICML), Online, 18–24 July 2021; pp. 10893–10902.
125. Peng, B.; Rashid, T.; Schroeder de Witt, C.; Kamienny, P.A.; Torr, P.; Boehmer, W.; Whiteson, S. FACMAC: Factored Multi-Agent Centralised Policy Gradients. In *Proceedings of the Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 12208–12221.
126. Shek, C.L.; Bedi, A.S.; Basak, A.; Novoseller, E.; Waytowich, N.; Narayanan, P.; Manocha, D.; Tokekar, P. Learning Multi-Robot Coordination through Locality-Based Factorized Multi-Agent Actor–critic Algorithm. *arXiv* **2025**, arXiv:cs.RO/2503.18816.
127. Rashid, T.; Farquhar, G.; Peng, B.; Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep Multi-Agent Reinforcement Learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10199–10210.
128. Alqudsi, Y.; Makaraci, M. UAV swarms: Research, challenges, and future directions. *J. Eng. Appl. Sci.* **2025**, *72*, 12.
129. Bu, Y.; Yan, Y.; Yang, Y. Advancement Challenges in UAV Swarm Formation Control: A Comprehensive Review. *Drones* **2024**, *8*, 320. [CrossRef]
130. Wang, T.; Zhao, S.; Xia, Y.; Pan, Z.; Tian, H. Consensus Control of Large-Scale UAV Swarm Based on Multi-Layer Graph. *Drones* **2022**, *6*, 402. [CrossRef]
131. Blais, M.A.; Akhloufi, M.A. Scalable and cohesive swarm control based on Reinforcement Learning. *Cogn. Robot.* **2024**, *4*, 88–103. [CrossRef]
132. Stodola, P.; Nohel, J.; Horák, L. Dynamic reconnaissance operations with UAV swarms: Adapting to environmental changes. *Sci. Rep.* **2025**, *15*, 15092.
133. Sadi, M.A.; Jamali, A.; Abang Kamaruddin, A.M.N. Optimizing UAV performance in turbulent environments using cascaded model predictive control algorithm and Pixhawk hardware. *J. Braz. Soc. Mech. Sci. Eng.* **2025**, *47*, 1–26.
134. Jung, W.; Park, C.; Lee, S.; Kim, H. Enhancing UAV Swarm Tactics with Edge AI: Adaptive Decision Making in Changing Environments. *Drones* **2024**, *8*, 582. [CrossRef]
135. Ma, H.; Yang, G.; Sun, X.; Qu, D.; Chen, G.; Jin, X.; Zhou, N.; Liu, X. Improved DRL-based energy-efficient UAV control for maximum lifecycle. *J. Frankl. Inst.* **2024**, *361*, 106718. [CrossRef]
136. Chen, Z.; Liu, H.; Liu, G. Real-Time Policy Optimization for UAV Swarms Based on Evolution Strategies. *Drones* **2024**, *8*, 619. [CrossRef]