



Kent Academic Repository

Mardzuki, Imran, Ibrahim, Azhar Mohd, De Wilde, Philippe and Baqalaql, Odai Saeed (2026) *Towards Optimization of Deep Reinforcement Learning for Autonomous Exploration in Single and Multi-Robot Environments*. ACM Computing Surveys, 58 (13). pp. 1-27. ISSN 0360-0300.

Downloaded from

<https://kar.kent.ac.uk/116061/> The University of Kent's Academic Repository KAR

The version of record is available from

<https://doi.org/10.1145/3818604>

This document version

Publisher pdf

DOI for this version

Licence for this version

CC BY (Attribution)

Additional information

Versions of research works

Versions of Record

If this version is the version of record, it is the same as the published version available on the publisher's web site. Cite as the published version.

Author Accepted Manuscripts

If this document is identified as the Author Accepted Manuscript it is the version after peer review but before type setting, copy editing or publisher branding. Cite as Surname, Initial. (Year) 'Title of article'. To be published in **Title of Journal**, Volume and issue numbers [peer-reviewed accepted version]. Available at: DOI or URL (Accessed: date).

Enquiries

If you have questions about this document contact ResearchSupport@kent.ac.uk. Please include the URL of the record in KAR. If you believe that your, or a third party's rights have been compromised through this document please see our [Take Down policy](https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies) (available from <https://www.kent.ac.uk/guides/kar-the-kent-academic-repository#policies>).

Towards Optimization of Deep Reinforcement Learning for Autonomous Exploration in Single and Multi-Robot Environments

IMRAN MARDZUKI, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia

AZHAR MOHD IBRAHIM, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia

PHILIPPE DE WILDE, University of Kent, Canterbury, United Kingdom of Great Britain and Northern Ireland

ODAI SAEED BAQALAQL, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia

Autonomous exploration of unknown environments stands as a critical challenge in contemporary robotics applications. The application of Deep Reinforcement Learning (DRL) in robotics exploration has been developed to design efficient exploration strategies suitable for complex real world scenarios. In the past, there have been reviews about the application of DRL in exploration, but the optimization of DRL has not been extensively reviewed. This survey focuses on exploration for both single robot and multi-robot applications, specifically addressing path planning aspects such as navigation, coverage, and trajectory. The survey emphasizes key elements of DRL derived from previous work, encompassing state and observation, action, and reward. Finally, the main contribution of this survey is to review how past work has optimized the DRL implementation to overcome inherent limitations in the approach, including challenges related to inefficient exploration in the training process, which collectively limit the adaptability and robustness of these algorithms in truly unknown environments due to their resource-extensiveness. From our survey, current research is lacking on exploration with reasoning entities for the training, relying instead on prior environmental knowledge. This underscores the need for exploration strategies with built-in reasoning and less dependence on pre-existing data.

CCS Concepts: • **Computing methodologies** → **Motion path planning**;

Additional Key Words and Phrases: Deep reinforcement learning, multi-agent reinforcement learning, path planning, survey

This research is supported by the FRGS 2023 Grant: FRGS/1/2023/TK02/UIAM/02/1 awarded by the Ministry of Education Malaysia and also by International Islamic University Malaysia under the KOE Engineering Merit Scholarships 2023 (KOEiems2023).

Authors' Contact Information: Imran Mardzuki, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia; e-mail: imranmardzuki881@gmail.com; Azhar Mohd Ibrahim, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia; e-mail: azhar_ibrahim@iiu.edu.my; Philippe De Wilde (corresponding author), University of Kent, Canterbury, United Kingdom of Great Britain and Northern Ireland; e-mail: p.dewilde@kent.ac.uk; Oday Saeed Baqalaql, Department Of Mechatronics Engineering, International Islamic University Malaysia, Kuala Lumpur, Malaysia; e-mail: oday3456783@gmail.com.



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

© 2026 Copyright held by the owner/author(s).

ACM 0360-0300/2026/06-ART336

<https://doi.org/10.1145/3818604>

ACM Reference Format:

Imran Mardzuki, Azhar Mohd Ibrahim, Philippe De Wilde, and Odai Saeed Baqalaql. 2026. Towards Optimization of Deep Reinforcement Learning for Autonomous Exploration in Single and Multi-Robot Environments. *ACM Comput. Surv.* 58, 13, Article 336 (June 2026), 27 pages. <https://doi.org/10.1145/3818604>

1 Introduction

In recent past, the rapid progress in robotics technology has ignited a growing fascination in academia and real-world applications of mobile robots in different fields. In terms of exploration, in scenarios where the map or model of the environment is unknown beforehand, the deployment of mobile robots is highly dependent on a robust and effective exploration strategy. The ultimate objective of mobile robot's exploration of the unknown area is to create a map of the explored area, taking into account the coverage of the area, the accuracy of the mapping and the speed of completion of the exploration [47]. Autonomous mobile robots offer distinct advantages in improving the safety and feasibility of exploration, even in environments with significant risks to human well-being. Autonomous mobile robots should have the capability to navigate the environment, gather information about the environment, construct a suitable map, and determine its location on that map simultaneously [75]. Integrated exploration for autonomous mobile robots encompasses field research in localization, mapping, and motion control [47].

Localization involves determining the robot's pose based on its surroundings when the map is known but the robot's current position is unknown, while mapping entails estimating the environment's model using sensor data when the robot's position is known but the surroundings are not [35]. Localization and mapping are interdependent processes, leading to the development of **Simultaneous Localization and Mapping (SLAM)**. SLAM has been extensively utilized in static environments, but its effectiveness in dynamic environments is still a work in progress [8]. In terms of path planning, it comprises the global planner, which relies on pre-existing environmental data to generate an optimal path, and the local planner, responsible for adjusting the original plan dynamically to evade potential obstacles [50]. The traditional path planner exhibits limitations due to its reliance on prior knowledge of the environment, which leads to inflexibility and unreliability when encountering changes or inaccuracies in the map [50]. To overcome this, reinforcement Learning emerged as a fitting solution for path planning in unknown environments, as it offers a suitable method to create control strategies for navigating mobile robots where it involves learning agents that navigate in unfamiliar environments to maximize reward for evaluation of states and actions [20]. The integration of **Deep Learning (DL)** into RL, known as **Deep Reinforcement Learning (DRL)**, has significantly advanced the field, as deep learning equips RL to handle complex decision-making scenarios that were previously unsolvable, particularly those involving high-dimensional state and action spaces [3]. In DRL, agents must spend a considerable amount of time interacting with their environment to gain insight into the outcomes of different actions. However, collecting such interaction data in the real world can be costly, time-consuming and sometimes unsafe, so optimization of the algorithm is needed [13]. Moreover, to achieve high-quality autonomous navigation, the limitation lies in the constraints in human experience and data acquisition, the inefficiencies in exploration strategies and collision risk assessment methods, and underscore the need for continued research and development in this field [99].

Optimizing the exploration objective for a single mobile robot presents challenges due to constraints associated with the capabilities of a single robot, such as efficiently covering expansive areas. Additionally, if the robot experiences a failure, exploration cannot continue. Multi-robot systems provide a means to mitigate these limitations, but they introduce the complexity of managing a system that requires coordination and collaboration among the robots [66]. The cooperative behavior among

the multi-robot is the most important thing to consider in making the exploration process more effective. Cooperation reduces redundant efforts among the robots in exploring areas already explored by other robots and minimizes overlapping exploration, thereby enhancing efficiency [29]. Using DRL in a multi-robot system has introduced **Multi-Agent Reinforcement Learning (MADRL)**, expected to address the unique challenges of controlling multi-agent systems in dynamic environments. However, the presence of other agents introduces an incomplete model of an agent in the system, leading to a nonstationary environment, and the joint action and state spaces grow exponentially with the number of agents, raising concerns about space and computational complexity [39].

DRL has emerged as a valuable tool for tackling exploration-related application in multi-robot contexts. This study specifically concentrates on exploring the recent developments in multi-robot exploration through DRL, recognizing the presence of several survey articles, such as by Y. Li [41], which provides an extensive overview of advancements in deep reinforcement learning, covering core elements, important mechanisms, and various applications spanning fields such as gaming, robotics, natural language processing, computer vision, and more, along with additional resources and discussions [41]. In term of **multiagent systems (MAS)**, Rizk et al. [65] explore into the advancements in cooperative MAS decision-making models, encompassing methods such as MARL, while also delving into their practical applications across different domains and addressing critical challenges [65]. Hernandez-Leal et al. [30], Nguyen et al. [56], Orr and Dutta [59] and K. Zhang et al. [94] offer an overview of the landscape in MADRL highlighting the adaptation of key components from single-agent reinforcement learning to multiagent scenarios and focuses on the challenges and various approaches within MADRL [30][56][59][94]. Lastly, Garaffa et al. [23] explores the utilization of RL in designing exploration strategies for both single and multi-robot autonomous systems, summarizing the RL algorithms employed, their incorporation into mobile robot exploration strategies, and how these solutions tackle RL challenges [23].

However, from these previous surveys, there are no articles explaining how the DRL has been optimized for robot exploration. In this survey, the focus is on exploration for both single robot and multi-robot applications, specifically in terms of path planning aspects like navigation, coverage, and trajectory. The survey highlights key elements of DRL drawn from the application of previous work, including state and observation, action, and reward. Lastly, the main contribution of this survey is examining how previous methods optimize the implementation of DRL to overcome limitations inherent in DRL.

This survey adopts a structured and algorithm-centric literature review approach to analyze recent advancements in DRL for autonomous exploration. Relevant studies were collected from major academic databases, including IEEE Xplore, Scopus, and SpringerLink, using keyword combinations such as deep reinforcement learning, multi-agent reinforcement learning, and autonomous navigation. The selection process focuses on peer-reviewed journal articles and conference papers published primarily between 2016 and 2024, reflecting the rapid development of DRL techniques, while also incorporating foundational works for theoretical grounding. To ensure relevance and consistency, the survey emphasizes studies that apply DRL to exploration or navigation tasks, address single-robot and multi-robot systems, and propose optimization strategies.

The analysis is organized in a structured manner by first selecting representative DRL algorithms, including value-based methods and policy-based or actor-critic methods, as the foundation for comparison. For each algorithm, application-driven studies are identified and included in the review. These works are analyzed to examine how the algorithm is implemented in exploration and navigation tasks, with particular focus on key components and optimization strategies. Multi-agent approaches are further categorized based on their training and execution strategies, including **centralized training with decentralized execution (CTDE)**, fully decentralized learning, and centralized approaches.

In Section 2, we provide an overview of the foundational concepts behind single-agent exploration and its utilization in DRL. Moving on to Section 3, we delve into the background of multi-agent exploration and its integration into DRL applications. In Section 4, we outline the key elements employed in DRL. Section 5 discusses practical implications and experimental validations derived from the reviewed studies. Section 6 presents the optimization of DRL algorithms. To conclude, our findings and insights are presented in Section 7. The taxonomy of exploration is presented in Figure 1.

2 Single Robot Exploration

2.1 Traditional Method

One of the traditional methods for exploration is by using SLAM. This involves creating and continuously updating a map of its environment while determining its own position within that map. Initially employed in unknown environments to make them known, SLAM's accuracy improves when the environment becomes familiar. The robot estimates the positions of landmarks and its own position relative to these landmarks. If the robot deviates from its path, it corrects itself using landmark positions, but sensor inaccuracies can hinder this process. Uncertainties in sensor data accumulate over time, necessitating the use of probabilistic methods to accurately model these uncertainties [35]. The limitations of SLAM include the laborious process of constructing and updating obstacle maps, along with a significant reliance on precise dense laser sensors for mapping prediction [81].

In global path planning, the objective is to establish a collision-free route for a robot from its initial position to the goal position by providing a detailed description of the goal and static obstacle space, allowing the robot to reference and follow the generated global path for guidance [7]. Global path planning for path search algorithms can be categorized into two types: heuristics methods and artificial intelligence methods or **Evolutionary Computation (EC)** method [93]. An example of a heuristic method are using graph search-based algorithms such as Dijkstra and A*. The Dijkstra Algorithm is the shortest path algorithm that navigates from a starting point to an endpoint in a directed graph by minimizing the sum of edge weights. The process involves updating the shortest path length from vertices to other vertices if a shorter path is found [84]. The A* Algorithm builds upon Dijkstra's algorithm, employing an evaluation function:

$$f(n) = g(n) + h(n), \quad (1)$$

where $g(n)$ is the actual cost and $h(n)$ is the estimated optimal path cost, enabling efficient path searches with consideration for the target node's position [93].

For the EC method, examples of algorithms include **Particle Swarm Optimization (PSO)** and **Ant Colony Optimization (ACO)**. ACO simulates ant behavior to discover the optimal path between a food source and their nest, utilizing pheromone communication among ants to reinforce attractive paths, adapt to environmental changes, and establish a loop of positive feedback [9]. PSO on the other hand, simulates swarm behaviors inspired by birds flocking or fish schooling, with each particle in the swarm representing a potential solution in a D-dimensional search space, updating its position based on velocity vectors and historical best positions, denoted as local best and global best, using specific updating models involving inertial weight, cognitive and social parameters, and random numbers [77]. However, the drawback of a global path planner lies in its dependence on known static maps, making it challenging to navigate dynamic environments [104].

Local path planning involves guiding a robot in unknown or dynamic environments, responding to obstacles and environmental changes in real-time through sensory-based information and continually updating crucial information for accurate destination reaching [6]. One of the algorithms for local path planning, the **Artificial Potential Field (APF)** method, employs attractive and

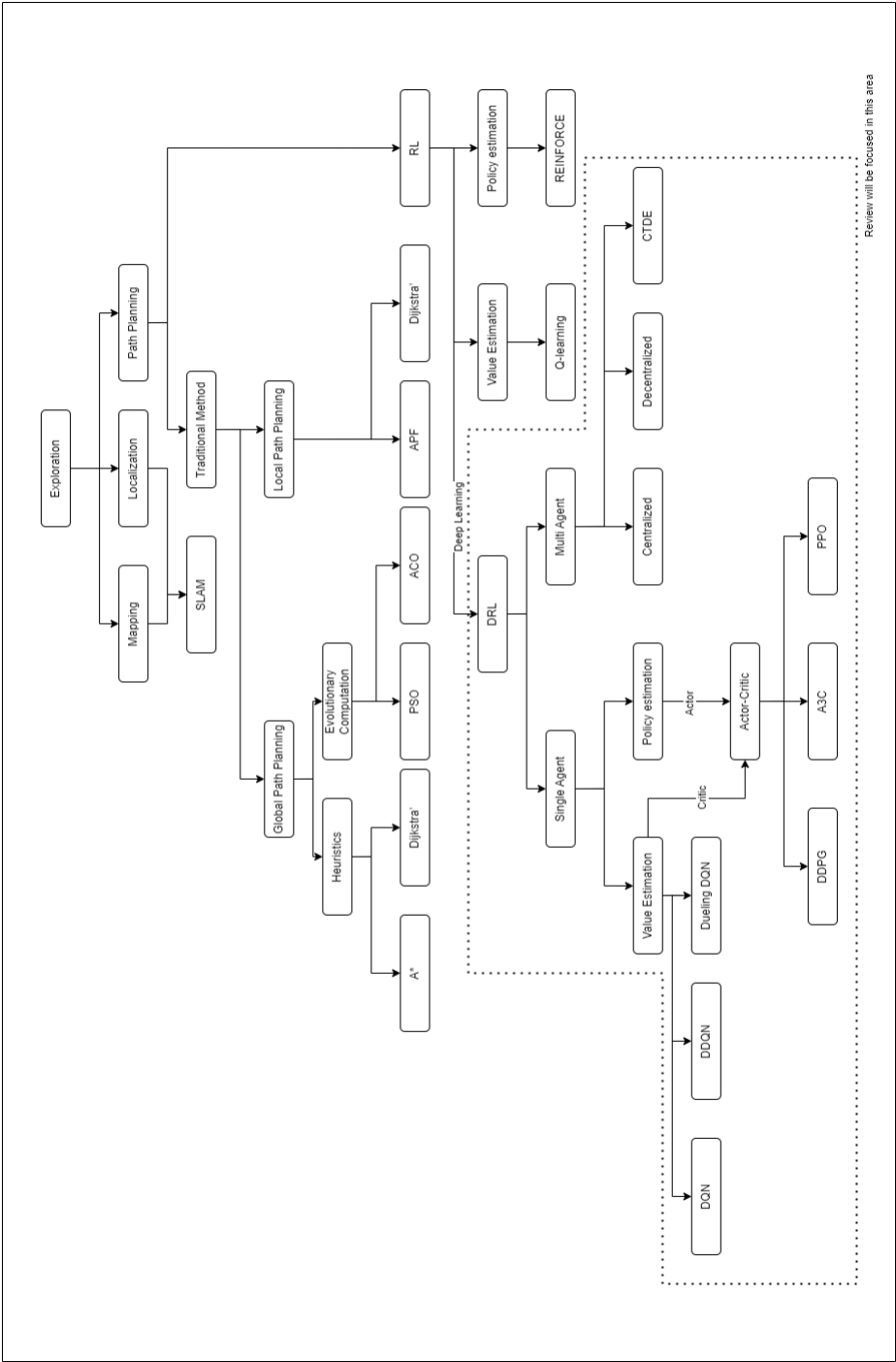


Fig. 1. The taxonomy of the exploration (PSO = particle swarm optimization, ACO = ant colony optimization, APF = artificial potential field, DQN = deep Q-network, DDQN = double deep Q-network, DDPG = deep deterministic policy gradient, A3C = asynchronous advantage actor-critic, PPO = proximal policy optimization, CTDE = centralized training with decentralized execution).

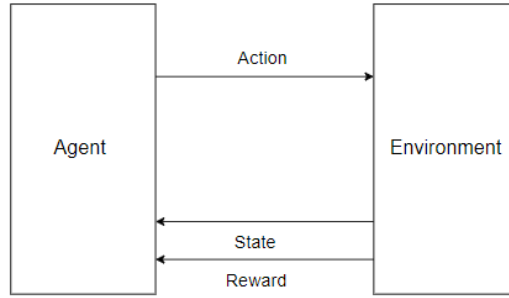


Fig. 2. RL process.

repulsive forces to guide a robot through an environment, resulting in high-quality paths but is prone to local minimum points and difficulty reaching the target when obstacles are nearby [1]. In contrast, the **Rapidly Exploring Randomized Trees (RRT)** method utilizes a tree-like structure with randomly sampled points to explore the planning space, offering probabilistic completeness but suffering from randomness and low efficiency [1]. The constraint arises from the need for intensive calculations during real-time replanning of the navigation path in a dynamic environment, which somewhat curtails its responsiveness. The limitation stems from traditional exploration algorithms, prompting the utilization of DRL.

2.2 Reinforcement Learning

RL is a process where an agent interacts with an environment to learn how to make the best possible actions in different states by making sequential decisions in an environment, aiming at maximizing cumulative rewards. The agent takes an action, a_t where agent is in the current state, s_t which changes the state of the environment and results in a reward. The process can be illustrated in Figure 2. The agent uses these rewards to improve its decision-making over time and come out with policy, π . The **Markov Decision Process (MDP)** is the foundation of RL. It assumes the environment is in a particular state, transitioning probabilistically to the next state upon taking an action. Rewards are also associated with state-action pairs and depend on the action taken. MDPs aim at finding a policy that maximizes the expected return, connecting future rewards to current actions [63]. An optimal policy for an MDP is one that maximizes the value of every state or state-action pair, thus solving the RL problem. MDP can be defined as a tuple (S, A, P, R, γ) , where S , represents a set of states ($s \in S$), A , represents a set of actions ($a \in A$), P is the probability associated with the states, R is the reward function and γ , discounting factor where $\gamma \in (0, 1]$. A discounting factor is introduced in RL to adjust the value of future rewards to their present-day equivalent, considering that the future reward is uncertain rewards which refers to the unpredictability and probabilistic nature of the outcomes [73].

Bellman equation is the backbone of the RL. The state value function, represented as $v[s_t]$, calculates the expected return an agent can obtain on average when starting from a specific state and following a given policy, π . It reflects how good a state is under that policy and the highest for states where it is likely to yield substantial rewards in the long run. On the other hand, the state-action value function, denoted as $q[s_t, a_t]$, estimates the expected return from taking a particular action in each state and then continuing with the specified policy. The state action value approach is a more direct means of achieving the goal of finding the best action, as it directly estimates the value of taking specific actions, while state value estimation primarily focuses on estimating the value of states and may require an additional step to determine the best action in those states. The

equation of the $v[s_t]$ and $q[s_t, a_t]$ can be described in a recursive equation as

$$v[s_t] = \sum_{a_t} \pi[a_t|s_t](r[s_t, a_t] + \gamma \sum_{s_{t+1}} P[s_{t+1}|s_t, a_t]v[s_{t+1}]), \quad (2)$$

$$q[s_t, a_t] = r[s_t, a_t] + \gamma \sum_{s_{t+1}} P[s_{t+1}|s_t, a_t] \left(\sum_{a_t} \pi[a_t|s_t]q[s_{t+1}, a_{t+1}] \right). \quad (3)$$

It helps RL algorithms associate future rewards with current actions, addressing the challenge of temporal credit assignment. The optimal policy of the action is derived from the optimal state action value function where the actions are giving the highest state action value function. The equation for updating policy:

$$\pi[a_t|s_t] = \operatorname{argmax}_a q[s_t, a_t]. \quad (4)$$

RL can be categorized into two approaches, which are model-based and model-free [73]. Model-based methods rely on a deep understanding of the MDP model, involving the knowledge of state-transition probabilities and significant data for modeling. These approaches typically require dynamic programming for solving. In contrast to many RL methods, model-based approaches do not learn interactions from data but instead rely on initialized state values and policies. The process involves iterative policy evaluation and improvement, updating state values using the Bellman equation and selecting actions that maximize state values, effectively finding optimal solutions in MDP. On the other hand, model-free approaches do not necessitate a complete MDP model and can operate in real-time without prior knowledge of the model's details. Model-free methods can be further categorized into two main classes which are value estimation approaches and policy estimation approaches. RL can be further categorized into on-policy and off-policy methods, where on-policy methods directly optimize the decision-making policy and off-policy methods learn from data generated by another policy. Off-policy methods can store past experiences in a buffer for reuse during learning, which not only reduces sample complexity but also enhances the stability of reinforcement learning algorithms [17].

Value estimation approaches focus on estimating the optimal state-action value function and subsequently derive the policy based on the highest values in each state. It introduces bias and approximation errors when using function approximators, leading to convergence and hyperparameter sensitivity and computational intensity [73]. A well-known value estimation approaches in RL is Q-learning. Q-learning is an off-policy algorithm that initializes the Q-function with zeros and updates it to maximize future rewards by considering all possible Q-values for a state, making it suitable for problems where exploration is crucial for learning an optimal policy. The Framework of Q-learning is shown in Figure 3. The equation for updating Q-values in this algorithm:

$$q[s_t, a_t] = q[s_t, a_t] + \alpha(r[s_t, a_t] + \gamma \max_{a_{t+1}}(q[s_{t+1}, a]) - q[s_t, a_t]). \quad (5)$$

In contrast, policy estimation approaches directly estimate the optimal policy through techniques like gradient descent, bypassing the need for an intermediate model or value estimations. Additionally, policy estimation methods can naturally handle stochastic policies, which can be more appropriate in situations with multiple equally good actions or in continuous action spaces. However, the policy estimation approaches may converge to local optima rather than global optima during training and evaluating a policy can be computationally inefficient and high in variance [73]. REINFORCE algorithm is a policy estimation RL method that simplifies the policy gradient estimation for training [63].

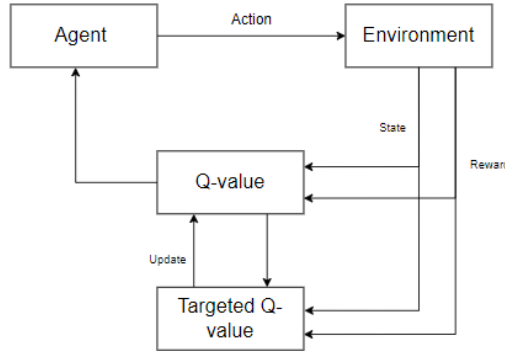


Fig. 3. Q-learning framework.

2.3 Deep Reinforcement Learning

DRL merges conventional RL with DL to address the challenges posed by intricate environments with extensive state spaces and high computational requirements. DRL leverages Neural Networks to approximate value, policy, or models, with these approximations obtained by storing state-action pairs, a concept inherited from traditional RL [41].

2.3.1 Value Estimation. In the value-based paradigm, **Deep Q-Network (DQN)** extends Q-learning by employing neural networks to approximate the action-value function [53]. A key innovation of DQN lies in the use of experience replay and target networks, which improve data efficiency and stabilize training. The model is trained by minimizing the temporal-difference loss between predicted and target Q-values derived from the Bellman equation. This process enhances data efficiency by reusing samples and reduces variance by lowering correlation among the samples used in updates. The update of the network is done by minimizing the loss function which is

$$L[\theta] = (y - q[s, a; \theta])^2, \quad (6)$$

The y in the loss function represents the target Q-value, denoted as

$$y = r[s_t, a_t] + \gamma \max_{a_{t+1}} (q[s_{t+1}, a; \theta]). \quad (7)$$

Existing works primarily apply DQN to exploration and navigation by varying input representations and reward design. For instance, Tai and Liu [80] utilize depth images as input for corridor exploration, whereas Zhou et al. [102] employ RGB-based environmental representations for global path planning. These approaches highlight that the performance is highly dependent on state representation. With richer sensory inputs, it will improve navigation capability but also increases computational complexity. In contrast, simpler environments such as grid-based settings reduce complexity but limit real-world applicability [19]. To address sparse reward issues and inefficient exploration, several studies focus on reward engineering and exploration strategies. Guan et al. [25], introduce heuristic reward shaping and adaptive exploration to balance exploration and exploitation. While such methods improve convergence speed, they often introduce task-specific bias, reducing generalizability across environments.

Despite its success, DQN suffers from overestimation bias due to the use of a single network for both action selection and evaluation. **Double Deep Q-Network (DDQN)** mitigates this issue by decoupling these two processes, resulting in more stable learning with minimal computational overhead [83]. Empirical applications demonstrate that DDQN improves robustness in complex

navigation tasks, such as path tracking in high-curvature environments [97] and autonomous driving in unknown terrains [34]. Extensions such as EDDQN further modify the value update mechanism and incorporate multi-modal inputs, improving performance in aerial exploration tasks [46].

Another architectural improvement is the Dueling DQN, which decomposes the Q-value into state-value and advantage components. This separation enables the model to learn which states are valuable independently of the chosen action, leading to improved stability and faster convergence [86]. The advantage function can be defined as a measure relative to the state that determines the importance of each action which is denoted as

$$A[s_t, a_t] = q[s_t, a_t] - v[s_t]. \quad (8)$$

This innovative design enhances optimization stability and performance in RL tasks, all integrated into the network itself, eliminating the need for separate algorithmic steps [86]. Compared to standard DQN and DDQN, dueling architectures are particularly effective in environments where many actions yield similar outcomes. For example, Cao et al. [10] demonstrate that Dueling DQN achieves faster convergence and better performance in path planning tasks with complex state spaces. Overall, value-based methods such as DQN and its variants improve stability and learning efficiency through architectural enhancements, but their performance remains highly dependent on state representation and reward design. Furthermore, their reliance on discrete action spaces limits applicability in continuous control tasks, motivating the use of policy-based approaches.

2.3.2 Policy Estimation. Policy-based methods directly optimize the policy using gradient-based updates, making them well suited for continuous and high-dimensional action spaces. Most modern approaches adopt an actor-critic architecture, where the actor learns the policy while the critic estimates value functions to guide updates [37]. Among these methods, **Deep Deterministic Policy Gradient (DDPG)** extends deterministic policy gradients with deep neural networks and off-policy learning [42]. DDPG maintains actor and critic functions and used replay buffers to handle diverse tasks with complex movements and high-dimensional sensory inputs. DDPG used an actor network, which determines the agent's actions given states, denoted as

$$a = \mu[s, a|\theta^\mu], \quad (9)$$

where θ^μ , is the parameter for the weight of the actor network. By modifying the θ^μ through updates, it generates action depending on the present state and interacts with the environment to generate the succeeding state and subsequent reward. The Actor target network adjusts parameters within the Critic network, establishing the subsequent optimal action. The critic network, $Q(s, a|\theta^Q)$, is to estimates the Q-value to evaluate the actor networks where θ^Q , is the parameter for the weight of the critic network. The critic is updated by minimizing the loss function, L , where it is defined as

$$L = \frac{1}{N} \sum_i (r(s_i, a_i) + \gamma Q'(s_{i+1}, \mu[s_{i+1}|\theta^\mu]|\theta^{Q'}) - Q(s_i, a_i|\theta^Q))^2. \quad (10)$$

While the actor is updated by using sampled policy gradient:

$$\nabla_{\theta^\mu} J = \frac{1}{N} \sum_i \nabla_a Q(s_i, a|\theta^Q)|_{a=\mu[s_i, a|\theta^\mu]} \nabla_{\theta^\mu} \mu[s_i|\theta^\mu]. \quad (11)$$

In robot navigation, several works extend DDPG by integrating it with complementary strategies to improve training efficiency and path planning performance. Dong and Zou [18] incorporate limited prior knowledge to reduce trial-and-error during learning, while Xiangrui et al. [89] enhance the traditional Seeker Optimization Algorithm through a hybrid DDPGSOA framework to address its inherent limitations. Similarly, Y. Chen and Liang [15] combine Sequential Linear Path Planning with DDPG to introduce structured sub-goal generation, improving stability and efficiency. Overall,

these approaches demonstrate a common trend of augmenting DDPG with heuristic or model-based components to accelerate convergence and enhance navigation robustness.

In contrast, Asynchronous **Advantage Actor–Critic (A3C)** focuses on improving training efficiency through parallelism [52]. By employing multiple actors that interact with the environment simultaneously, A3C reduces sample correlation and stabilizes learning without requiring replay buffers. This algorithm maintains policy as $\pi(a_t|s_t; \theta)$, where θ is the weight for the policy and value function as $V(s_t; \theta_v)$, where θ_v is the weight for value function. Both are operating in the forward view and employing a mix of n-step returns to concurrently update both networks. Updates take place after a specified number of actions or upon reaching a terminal state. The algorithm's update process involves adjusting the policy and value function based on an estimate of the advantage function. The update performed by the algorithm for policy and value function can be represented as below, respectively:

$$\theta = \theta + \nabla_{\theta} \log \pi(a_t|s_t; \theta)(R - V(s_t; \theta_v)), \quad (12)$$

$$\theta_v = \theta_v + \nabla_{\theta_v} (R - V(s_t; \theta_v))^2. \quad (13)$$

Tongloy et al. [82] adapted Asynchronous A3C for mobile robot navigation by incorporating supervised auxiliary tasks, including depth prediction and position estimation. Similarly, Mirowski et al. [51] leveraged auxiliary tasks such as depth prediction and loop closure classification with multi-modal sensory inputs to formulate navigation as a reinforcement learning problem. Other works focus on enhancing motion planning, for example, Sasaki et al. [68] applied A3C to address path planning challenges in dynamic, crowded environments, while C. Zhou et al. [101] modified the advantage actor-critic framework to generate optimal robot speeds and directions. Compared to DDPG, A3C offers better stability and scalability but may sacrifice sample efficiency due to its on-policy nature.

Proximal Policy Optimization (PPO) is the other policy estimation method and is an improved version of **Trust Region Policy Optimization (TRPO)** designed to balance data efficiency and reliability in training. In TRPO, the algorithm maximizes the surrogate objective, defined as

$$L(\theta) = r_t(\theta)A_t, \quad (14)$$

$$r_t = \frac{\pi_{\theta}(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}. \quad (15)$$

PPO uses a novel objective with clipped probability ratios to ensure stable and efficient policy updates and alternates between collecting data and optimizing the policy in multiple epochs [70]. In each iteration, multiple parallel actors interact with the environment, running the existing policy for a fixed number of timesteps. Advantage estimates are computed to gauge the effectiveness of chosen actions. Subsequently, the algorithm optimizes a surrogate loss function with respect to the policy parameters, striking a balance between policy updates and stability. The clipped surrogate loss can be defined as

$$L^{clip} = \min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t). \quad (16)$$

Where ϵ is a hyperparameter. Optimization occurs over several epochs, using a minibatch of data collected from the actors. The policy parameters are then updated, and this iterative process continues, with the algorithm refining its policy through successive iterations. The approach emphasizes parallel actor interaction, advantage estimation, and a carefully managed optimization process to enhance policy learning in sequential decision-making tasks. This mechanism prevents excessively large policy updates and ensures stable learning [70]. Unlike DDPG and A3C, PPO achieves strong performance with relatively simple implementation and fewer hyperparameter sensitivities. As a result, PPO has become a widely adopted baseline for navigation and control tasks.

S. Zhang and Wang [96] address mobile robot trajectory tracking using PPO in a distributed framework to accelerate sample collection and reduce transition correlation during updates. Gromniak and Stenzel [24] apply PPO to robot navigation in dynamic, changing environments. J. Li et al. [38] employ a behavior-based approach, mapping raw sensor inputs and goal information directly to control commands, with PPO training obstacle avoidance in continuous action spaces for both single- and multi-agent scenarios. Together, these works illustrate PPO's versatility in improving learning efficiency and robustness in complex navigation tasks.

Across these methods, DDPG is effective for continuous control but often requires additional stabilization mechanisms, A3C improves exploration through parallelism but may introduce variance, and PPO provides a more stable and practical alternative with simpler tuning requirements. These tradeoffs highlight the balance between sample efficiency, stability, and implementation complexity in policy-based DRL methods.

2.3.3 Summary for Single Agent DRL Applications. Value-based methods, such as DQN, estimate action-values with neural networks and stabilize learning via experience replay and target networks. Variants like DDQN and Dueling DQN reduce overestimation bias and improve convergence by decoupling action evaluation and separating state-value from advantage functions. These methods perform well in discrete action spaces, though results are sensitive to state representation and reward design. Policy-based methods directly optimize policies via gradient updates, suitable for continuous and high-dimensional actions. Actor-critic architectures like DDPG use separate actor and critic networks with replay buffers to improve navigation, while A3C leverages parallel actors for stable exploration. PPO balances stability and data efficiency with clipped surrogate objectives. Overall, DDPG excels in continuous control, A3C in exploration, and PPO offers robust, practical policy optimization. Table 1 summarizes the single-agent DRL applications reviewed. Further discussion of DRL elements appears in Section 4.

3 Multi-Robot Exploration

3.1 Swarm Robotics

Swarm robotics, a field in robotics where multiple robots collaborate in a decentralized manner to achieve collective goals, categorizes algorithms into insect-based and animal-based groups [12]. The concept of swarm systems involves multiple identical agents collaborating to achieve shared objectives, such as those seen in swarm robotics or wireless sensor networks. Such systems offer redundancy against individual agent loss and can mimic natural swarm behaviors like foraging or formation control [33]. The examples algorithm that can be used for swarm intelligence are ACO and PSO. Within swarm robotics, these robot swarms operate without a leader, relying on interactions and local communication facilitated by sensors or shared information among neighbouring robots [22]. Although some principled manual methods have shown promise, a generally applicable solution is lacking. The challenge in swarm robotics lies in designing control software for these swarms where current methods primarily rely on manual approaches guided by intuition and experience, lacking a comprehensive solution applicable to all scenarios [5]. In addition to the challenges mentioned earlier, another significant drawback of manually derived controllers in swarm robotics is their inability to adapt to environmental changes.

3.2 Communication Constraints

Effective cooperation in multi-agent systems depends strongly on the reliability of inter-agent communication, which is often far more limited in real-world deployments than assumed in simulation. Recent empirical studies in **Flying Ad Hoc Networks (FANETs)** highlight that communication performance can vary substantially depending on the selected routing protocol. Real-world

Table 1. Summary of Single-agent DRL Applications

Authors	DRL Used	State or Observation	Action	Reward	Sensor	Training and Execution
Tai and Liu, 2016	DQN	Continuous: CNN features from depth image (Kinect)	Discrete: turn left, go straight, turn right	Dense	RGB-D camera (Kinect)	Simulation in Gazebo
Zhou et al., 2018	DQN	Discrete: binary map image	Discrete: turn right, go forward	Sparse	N/A	Simulation
Erkan and Arserim, 2022	DQN	Discrete: 8x8 MiniGrid map (top-view camera)	Discrete: left, right, forward	Dense	Camera (top-down)	Simulation and real-world adaptation
Guan et al., 2021	DQN	Discrete: grid map	Discrete: up, down, left, right	Dense	N/A	Simulation
Mirowski et al., 2016	A3C	RGB visual input (vision + proprioception)	Discrete: 8 actions (rotations, accelerations)	Sparse	Camera	Simulation
Issa et al., 2020	DDQN	Grayscale camera image	Discrete: forward, left, right, stop, decelerate	Sparse	Camera	Simulation
Maciel-Pearson et al., 2019	EDDQN	Image + top-down visited-area map	Discrete: 4 directional movements	Dense	Camera	Simulation in AirSim
W. Zhang et al., 2019	DDQN	Continuous: speed + path-tracking errors	Continuous: steering curvature	Dense	N/A	Simulation + real-world adaptation
Cao et al., 2021	Dueling DQN	Discrete: grid map	Discrete	Sparse	N/A	Simulation
Dong and Zou, 2020	DDPG	Continuous state	Continuous action	Dense	N/A	Simulation + real robot (ROS/Gazebo)
Xiangrui et al., 2022	DDPGSOA	Continuous: 6-dimensional state (fitness history + iteration progress)	Continuous: 4 outputs (SOA parameters)	Dense	N/A	Simulation in MATLAB
Y. Chen and Liang, 2023	DDPGSLP	Continuous LiDAR-based state	Discrete: left, front-left, forward, front-right, right	Dense	LiDAR	Simulation in Gazebo
Tongloy et al., 2017	A3C	Discrete: coordinate grids (agent + target)	Discrete: fixed angles	N/A	N/A	Simulation in Gazebo
Sasaki et al., 2019	A3C	Discrete: local occupancy map with channels	Discrete: fixed-angle movements	Dense	LiDAR	Simulation
C. Zhou et al., 2021	A2C-MP	Continuous state	Continuous action	Sparse	N/A	Simulation
S. Zhang and Wang, 2019	PPO	Continuous: position, orientation, velocities, rotation speed	Continuous	Dense	N/A	Simulation
Gromniak and Stenzel, 2019	PPO	Discrete: multi-channel grid image	Discrete	Sparse	N/A	Simulation

evaluations of Babel, BATMAN-ADV, and OLSR show large differences in throughput and packet loss, with Babel consistently outperforming the others under dynamic UAV mobility, while BATMAN-ADV performs poorly due to its slow routing-table update interval [28]. Beyond routing-layer variability, hardware-level factors also play a critical role. Controlled radiation-pattern measurements of WiFi devices embedded in UAV platforms demonstrate that the UAV chassis can induce up to a 10 dB drop in transmitted power depending on antenna orientation, contradicting the common assumption of isotropic emissions and revealing significant degradation in link stability [26]. These communication limitations lead to delayed, partial, or inconsistent observations among agents, amplifying environmental non-stationarity and reducing the effectiveness of centralized or rule-based coordination methods. Consequently, there is a growing need for MADRL frameworks capable of learning robust cooperative behaviors that tolerate imperfect, asymmetric, or unstable communication, while maintaining scalability as team size and mission complexity increase.

3.3 Multi-Agent Deep Reinforcement Learning

In a multi-agent environment, the traditional assumptions of MDP no longer hold, making MDP invalid for this context. In multi-agent settings, agents interact simultaneously with the environment and with each other thus leading to non-stationarity, where the future dynamics, transitions, and rewards depend not only on the current state but also on the actions of other agents. In MDP, the Markov property assumes that the future is solely dependent on the current state, which becomes invalid in multi-agent domains. As a result, each agent's policy is influenced by the actions and strategies of other agents, and this interdependence renders the MDP framework inadequate [30]. Markov games, also known as stochastic games, represent a fundamental framework for addressing **multi-agent reinforcement learning (MARL)** problems [94].

In a Markov game, there are multiple agents interacting with a common environment, and each agent's action influences the rewards and actions of others, making it a challenging setting for MADRL. Markov game can be defined as a tuple (N, S, A, P, R, γ) , N represents the set of more than one agent, S denotes the shared state space, A_i is the action space of agent i , A represents the joint action space, P signifies the probability from one state to another for any joint action, R_i represents agent i 's reward function determining immediate rewards for state-action transitions, and γ is the discount factor. In this cooperative context, where all agents share a common goal and reward function, their value functions are identical, with the goal of achieving the Nash equilibrium as the optimal solution and aiming at learning coordinated joint policies [59]. The MADRL can be categorized by their types of learning and execution. There are three types which are centralized learning and execution, decentralized learning and execution, and **centralized learning and decentralized execution (CTDE)**.

3.3.1 Centralized Learning and Execution. Centralized learning and execution in multi-agent systems rely on a single model that aggregates global information from all agents to produce joint actions [11]. In this paradigm, the system is treated as a unified entity during both training and execution, enabling optimal coordination since all observations and actions are jointly considered. Applications such as vehicle routing have demonstrated the effectiveness of this approach under controlled environments with limited variability [11].

Despite its advantages in coordination, this paradigm suffers from several fundamental limitations. First, scalability becomes a critical issue due to the exponential growth of the joint action and observation spaces as the number of agents increases [71]. Second, centralized methods incur high computational complexity, as a single model must process global information and optimize over joint policies [61]. Third, these approaches depend heavily on reliable communication infrastructure, making them impractical in environments with communication constraints [54]. Additionally,

shared policy learning may lead to the “lazy agent” problem, where agents fail to sufficiently explore diverse behaviors, reducing overall system performance [48]. These challenges limit the applicability of fully centralized approaches in a large world scenarios.

3.3.2 Decentralized Learning and Execution. To address scalability limitations, decentralized learning and execution distribute decision-making across agents, where each agent learns and acts based solely on local observations [43]. This paradigm significantly reduces computational and communication overhead, making it more suitable for large-scale and partially observable environments. However, it introduces challenges such as non-stationarity, credit assignment, and limited coordination among agents [60].

Decentralized approaches can be broadly categorized into several groups. The first method is independent learning methods where, each agent learns its own policy while treating other agents as part of the environment. Algorithms such as **Independent DQN (IDQN)** follow this approach [31, 87]. While simple and scalable, these methods suffer from severe non-stationarity due to concurrently evolving policies. The second method is parameter sharing and parallel learning. To improve sample efficiency, some methods employ shared network parameters across agents while maintaining decentralized execution. Examples include Parallel DDPG and parallel DQN-based approaches [14, 78]. Although this reduces training cost, it may limit behavioral diversity and lead to suboptimal exploration. The third method is task decomposition and hierarchical methods. These approaches divide complex tasks into smaller sub-problems, often combining global planning with local decision-making. Methods such as MAPPER and hierarchical planning frameworks improve scalability and performance in large environments [43]. However, coordination remains implicit and may not be optimal. Lastly, the extensions of single-agent DRL algorithms. Several works adapt algorithms such as DQN, PPO, and A2C for decentralized multi-agent settings [61, 92, 95, 98]. While effective in certain applications, these methods typically lack mechanisms for explicit coordination.

Beyond these methods, several works focus on enhancing learning performance through hybrid control integration and improved reward design. For instance, Xue et al. [91] combined **Model Predictive Control (MPC)** with DDPG to incorporate predictive trajectory planning for dynamic obstacles, while leveraging learning-based policies for decision-making. Similarly, Jin et al. [36] proposed an interlaced DQN framework that jointly learns target selection and collision avoidance policies, with a reward function derived directly from the optimization objective rather than heuristic design. These approaches highlight a broader trend in decentralized MARL toward integrating domain knowledge and structured learning objectives to improve performance. However, despite these enhancements, decentralized methods remain fundamentally limited by non-stationarity, challenges in credit assignment, and the lack of explicit coordination mechanisms [60].

3.3.3 Centralized Learning and Decentralized Execution. Centralized learning suffers from scalability limitations, while fully decentralized learning introduces non-stationarity due to concurrently evolving agent policies. To address these challenges, the paradigm of CTDE has emerged as a dominant framework in MADRL. Under CTDE, agents are trained using additional global information, such as joint observations or actions, but execute policies independently based solely on local observations. This design improves learning stability while preserving scalability during deployment, effectively positioning CTDE as a tradeoff between coordination and efficiency.

Early applications in path planning and exploration reflect varying degrees of centralization and highlight a shift toward structured coordination mechanisms. For instance, Pathfinding via reinforcement and imitation multi-agent learning (Primal) integrates reinforcement and imitation learning to learn decentralized policies with implicit coordination [67], while **Multi-Agent Graph DQN (MAG-DQN)** introduces graph-based representations to explicitly capture spatial relationships between agents and environment regions [45]. Similarly, centralized CNN-based

approaches employ shared representations with agent-specific outputs to enhance scalability under homogeneous assumptions [44]. Although not always explicitly framed within CTDE, these methods collectively indicate a broader trend toward partial centralization through structured representations, particularly graph-based formulations. Within this context, CTDE methods can be broadly categorized into two primary classes which are centralized critic approaches and value decomposition methods.

Centralized critic methods extend actor-critic algorithms by incorporating global information during training while maintaining decentralized execution. Representative algorithms such as **Multi-agent DDPG (MADDPG)**, Multi-agent A3C (MAA3C), and **Multi-agent PPO (MAPPO)** enable improved credit assignment and policy stability [32]. Across applications, these methods consistently enhance coordination but they exhibit tradeoffs in terms of sample efficiency and scalability due to their reliance on joint information. Variants of MADDPG illustrate how different extensions target specific limitations: experience replay optimization and prioritized sampling improve convergence speed and sample efficiency [40], whereas decentralized UAV coordination frameworks emphasize cooperation through implicit information sharing [4], and safety-aware extensions incorporate constraints to prevent collisions in confined environments [76]. These developments highlight that improvements within centralized critic methods are often application-driven, focusing on either efficiency or robustness rather than fundamentally altering the coordination mechanism.

Recent works further extend centralized critic methods by introducing structural mechanisms to address scalability and interaction complexity. Mean-field approximations, as introduced in improved MADDPG, reduce computational complexity by modeling interactions through population-level distributions instead of explicit joint actions [100]. Hybrid planning frameworks, such as MAPPOHR, integrate learning-based policies with heuristic global planners to enhance real-time navigation performance [62]. Similarly, CNN-based MAPPO variants and application-driven frameworks demonstrate adaptability to complex environments, including exploration and construction robotics [16, 90]. Attention-based extensions refine coordination by selectively focusing on relevant agent interactions, improving performance in structured and densely populated environments [74]. Collectively, these approaches can be interpreted as attempts to balance coordination fidelity with scalability by either approximating interactions or selectively filtering global information.

A notable centralized critic approach is **Counterfactual multi-agent (COMA)**, which introduces a counterfactual baseline to address the multi-agent credit assignment problem [21]. By isolating each agent's contribution relative to a global baseline, COMA enables more precise policy updates and has been successfully applied in cooperative path planning scenarios [88]. Despite these advances, centralized critic methods remain constrained by their dependence on global information, which can limit scalability in large-scale systems.

Within the CTDE paradigm, **Value Decomposition Networks (VDNs)** provide a more scalable alternative by learning a team-value function and decomposing it into individual agent-wise value functions [79, 85]. This additive factorization stabilizes centralized training while enabling decentralized execution, allowing agents to effectively learn in joint reward settings. Compared to centralized critic methods, VDN simplifies coordination by restricting interactions to a linear decomposition, thereby improving scalability and training stability. However, this assumption limits representational capacity, as complex inter-agent dependencies cannot be fully captured [105]. Furthermore, VDN does not leverage additional global state information during training, which reduces its ability to exploit the full benefits of centralization. Nevertheless, it consistently outperforms fully independent or purely centralized approaches in cooperative multi-agent tasks.

QMIX extends VDN to address these representational limitations by introducing a non-linear mixing network that enforces a monotonic relationship between individual agent values and the global team value function [64, 105]. This design increases expressiveness while maintaining tractable decentralized policy extraction. Compared to VDN, QMIX enables more complex coordination

patterns by allowing flexible aggregation of individual contributions but the monotonicity constraint still restricts certain forms of interaction, potentially limiting optimality in highly coupled tasks. Subsequent extensions, such as masked highway connections and Global Duelling Q-learning, aim at furthering enhance joint action optimization and incorporate global state information for improved value estimation [49, 105]. These developments position QMIX as an intermediate solution that balances scalability and expressiveness, bridging the gap between simple factorization and fully centralized approaches.

Recent applications of MADRL in cooperative autonomous systems extend beyond navigation to large-scale traffic coordination, further illustrating the generality of CTDE-based approaches. For instance, **Learning-from-Oracle Demonstrations (LfODs)** has been used to accelerate TD3-based control in Autonomous Intersection Management systems, achieving faster learning and significantly reducing vehicle waiting times [27]. Similarly, an end-to-end MADRL approach trained via curriculum self-play improves intersection performance in connected autonomous vehicle networks by reducing travel time [2]. These studies demonstrate that CTDE-based methods can generalize across domains requiring coordinated decision-making under shared constraints.

In summary, centralized critic methods emphasize coordination and learning stability through explicit use of global information but face scalability challenges, whereas value decomposition approaches prioritize scalability and efficiency through structured factorization at the cost of representational flexibility. Recent extensions increasingly aim at bridging these paradigms by introducing mechanisms that selectively incorporate global information or approximate agent interactions, enabling more efficient and scalable coordination in complex multi-agent environments.

3.4 Summary for Multi-Agent Applications

MADRL addresses multi-agent environments where traditional MDP assumptions break down due to non-stationarity and interdependent agent policies, making Markov games a more suitable framework. MADRL approaches are broadly categorized into centralized, decentralized, and CTDE paradigms. Centralized methods enable strong coordination but suffer from scalability, computational, and communication limitations, while decentralized methods improve scalability but face challenges such as non-stationarity, weak coordination, and credit assignment issues despite various enhancements. CTDE has emerged as the dominant paradigm by leveraging global information during training while maintaining decentralized execution, balancing coordination and scalability. Within CTDE, centralized critic methods improve coordination and learning stability, whereas value decomposition methods enhance scalability through structured value factorization, with recent advances incorporating attention mechanisms, mean-field approximations, and hybrid planning. Table 2 and Table 3 summarizes the multi-agent DRL applications reviewed. Further discussion of DRL elements appears in Section 4.

4 Elements of DRL

In DRL, the key elements are state, action, and reward. The state serves as the input, encapsulating relevant information for decision-making, while actions represent decisions and serve as the output of the algorithm. Meanwhile, rewards provide feedback for shaping the learning process. Reviewing these elements ensures that the algorithm receives meaningful data for effective exploration and outputs a diverse set of choices suitable for adaptive behavior. Additionally, optimizing reward functions aligns incentives with task objectives.

4.1 State and Observation

In the context of exploration and planning paths the utilization of continuous states and observations is instrumental. These different categories of continuous data bring diverse but essential information

Table 2. Summary of Multi-agent DRL Applications (Part 1)

Authors	DRL Used	State or Observation	Action	Reward	Sensor	Training and Execution
(Y. Huang et al., 2020)	MADDPG	N/A	N/A	Dense	N/A	CTDE (simulation)
(W. Chen et al., 2019)	PDDPG	Continuous: distance and angle to target	Continuous: linear and angular velocity	Dense	2D LiDAR	Decentralized (Gazebo simulation)
(P. Li et al., 2021)	MADDPG	Continuous: linear/angular velocity + target info	Continuous: linear and angular velocity	Dense	2D LiDAR	CTDE (Gazebo simulation)
(Peng, 2023)	MAPPOHR	Continuous: distances to goal, agents, obstacles	Discrete: move, wait, back, replan	Dense	N/A	CTDE (Python simulation)
(Shen et al., 2021)	MAA3C	N/A	Discrete: up, down, left, right, stay	Dense	N/A	CTDE (simulation)
(Çetinkaya, 2021)	DQN	Discrete: grid of agents and landmarks	Discrete: left, right, up, down	Dense	N/A	Centralized (simulation)
(Marchesini and Farinelli, 2021)	GDQN	Continuous: distance to target and agent	Continuous: linear and angular velocity	Dense	LiDAR	CTDE (Unity simulation)
(Aydemir and Cetin, 2023)	DDMDAC (MADDPG)	Discrete: grid of agents and target	Discrete: left, right, up, down, stay	Dense	N/A	CTDE (simulation)
(Sartoretti et al., 2019)	A3C	Discrete: grids of obstacles, goals, and agent goals	Discrete: left, right, up, down, stay	Dense	N/A	CTDE (simulation + real-world trial)
(Z. Chen et al., 2019)	CMAPPO	Discrete: multi-channel image map	Discrete: left, right, up, down	Dense	Laser proximity	CTDE (simulation)
(Singh and Jha, 2021)	Safe-MADDPG	Continuous: coordinates of agent, target, others	Continuous: pitch, yaw	Dense	N/A	CTDE (simulation)
(Xu and de Soto, n.d.)	MAPPO	Continuous: distances to goal/agents/obstacles + idle times	Continuous: linear velocity	Dense	Laser scanner	CTDE (Gazebo simulation)
(Zou and Tan, 2023)	QMIX	N/A	N/A	Dense	N/A	CTDE (simulation)

Table 3. Summary of Multi-agent DRL Applications (Part 2)

Authors	DRL Used	State or Observation	Action	Reward	Sensor	Training and Execution
(Luo et al., 2019)	MAG-DQN	Discrete: agent grid position	Discrete: move to neighbor node	Dense	LiDAR	CTDE (simulation)
(Wei and Zheng, 2021)	DQN/IDQN	Discrete: target grid position	Discrete: adjacent vertices	Dense	N/A	Decentralized (Python simulation)
(W. Zhang et al., 2020)	cDQN	Discrete: agent grid position	Discrete: forward, backward, right, left, ascend, descend, hover	Sparse	N/A	Decentralized (simulation)
(J. Huang et al., 2023)	IDQN	Continuous: distances to target and nearest agent	Continuous: velocity commands	Dense	Camera	Decentralized (AirSim)
(Liu et al., 2020)	A2C	Discrete: multi-channel map	Discrete: 8 directions + stay	Sparse	N/A	Decentralized (simulation)
(Park et al., n.d.)	Synchronized PPO	Continuous: previous positions, distances, trajectories	Discrete: 8 directions + stay	Dense	N/A	Decentralized (simulation)
(Westheider et al., 2023)	COMA	Discrete: positions + mission budget	Discrete: north, south, east, west, up, down	Dense	Thermal sensor	CTDE (simulation)
(Luis et al., 2021)	Dueling DQL	Discrete: multi-channel map	Discrete: 8 directions + stay	Dense	LiDAR	CTDE (sim + real-world)
(L. Zhang et al., 2020)	DQN	Continuous: velocity vector of robot + rod	Discrete: forward-left, forward-right, backward-left, backward-right	Dense	N/A	Decentralized (Gazebo simulation)
(Yang et al., 2020)	DQN	Discrete: agent grid position	Discrete: up, down, left, right, stay	Dense	N/A	Decentralized (Python simulation)
(Na et al., 2023)	DDPG	Continuous: distance to target	Continuous: linear and angular velocity	Dense	N/A	Federated RL (simulation)
(Zheng and Liu, 2019)	MADDPG	Continuous: distances between leader/agent, obstacles, target	Discrete: N, S, E, W, NE, NW, SE, SW	Dense	N/A	CTDE (MATLAB simulation)
(Sui et al., 2018)	Dueling DDQN	N/A	Discrete: forward, forward-left, forward-right, stay	Dense	N/A	Decentralized (simulation)
(Xue et al., 2021)	DDPG	Continuous: distances to agents and target	Continuous: x- and y-displacements	Dense	N/A	Decentralized (Unity3D simulation)
(Jin et al., 2019)	DQN	Continuous: distances to agents and target	Continuous: heading angle	Dense	N/A	Decentralized (simulation)

crucial for making informed decisions during navigation. One of these key categories involves positional information, offering details about distances, angles, and velocities between the agent and various elements like obstacles, other agents, and the target position. This data not only helps in gauging proximity but also predicts potential collisions and anticipates movements within the environment. Another vital category is historical information, encompassing the history of agent actions and trajectories of other agents. It's like a navigation timeline, providing insights into past actions and movement patterns, which in turn aids in predicting future movements. Lastly, budgetary information is crucial as it keeps tabs on available resources, ensuring the agent's continuous movement within the environment.

On the other hand, discrete representations simplify the complexity of environments by using structured grids or image-based formats. These discrete states and observations come in two primary forms of positional information. The grid-based representation involves coordinates within a structured grid map, each coordinate being labelled to signify whether it's explored, unexplored, an obstacle, a target, or occupied by other agents. This helps in creating a simplified yet informative view of the surroundings. Meanwhile, the image-based approach uses different channels within the image to represent specific environmental features like obstacles, other agents, and targets. These channels act as distinct indicators, offering valuable navigational insights. Altogether, these discrete representations effectively reduce computational complexity while retaining crucial navigational data, tailored to fit diverse navigation needs and computational capabilities.

4.2 Action

Discrete actions are countable and represented using one-hot vectors, whereas continuous actions have an infinite range and are defined within specific parameters [103]. In the domain of continuous actions, especially within a two-dimensional space, the emphasis typically centers on managing both speed and direction. Linear velocity serves as the prime influencer of movement, determining the way the agent traverses space, while angular velocity steers the direction the agent faces or moves toward. In specific applications, certain actions focus solely on controlling the movement angle while maintaining a consistent velocity. Moreover, fine-tuning movement along the x-axis and y-axis provides additional control mechanisms, allowing subtle adjustments across various directions. In more intricate scenarios, such as **Unmanned Aerial Vehicle (UAV)** operations, actions in a three-dimensional space extend beyond basic linear and angular movements. Here, actions like pitch and yaw take precedence, facilitating control and navigation through three-dimensional environments, allowing adjustments in both orientation and altitude.

Conversely, discrete actions typically operate within a limited set of movements. Often categorized into four primary actions which are forward, left, right, and backward. These actions delineate the permissible directions an agent can take. Expanding this set to eight actions introduces diagonal movements, covering directions such as left forward, right forward, left backward, and right backward. Furthermore, an action to "stay" aids in dynamic responses, enabling actions like replanning or waiting for obstructions or other agents to clear the path. Replanning commonly occurs when an agent is equipped with a global planner, enabling the recalibration of the course of action. Another method of discretizing actions involves segmenting the movement angle into a specific number of discrete actions, presenting an alternate approach to define and govern the agent's directional choices.

4.3 Reward

The sparse type of reward typically applies rewards or penalties based on specific events within the environment. Rewards commonly come when reaching a target or completing exploration, while penalties are incurred during collisions with other agents or obstacles. To transform rewards into a dense form, intermediate steps must carry reward values. One approach involves penalizing each

step taken during exploration or navigation, motivating the agent to seek the shortest path, and accelerating the training convergence. Another method involves assigning rewards and penalties based on distance from the target or obstacles. Using distance metrics, rewards can be allocated by comparing the distance between previous and current states; if reduced, a reward is granted, otherwise, a penalty is incurred. In terms of exploration, rewards can hinge on the coverage rate, granting rewards for exploring new areas and penalizing re-entry into previously explored territory. In multi-robot scenarios, rewards may be given for maintaining formation, extending concepts initially applied in single-agent DRL to multi-robot applications. Rewards can be constant, follow a specific function, or utilize heuristics. However, overly generalizing rewards might impede the training process.

5 Practical Implications and Experimental Validations

To strengthen the connection between the taxonomy and the real-world applicability of DRL-based exploration, we expand this section by discussing practical implications grounded directly in the studies reviewed throughout this survey.

5.1 Practical Implications of DRL-Based Exploration Frameworks

The categorization of state and observation types in Section 4 highlights several practical considerations relevant to real deployments. For instance, works relying on depth or LiDAR data such as Tai and Liu (2016) and Chen et al. (2019) demonstrate that richer sensory modalities improve environmental understanding but also increase computational load [14, 80]. In contrast, approaches that use compact numerical states or grid representations, such as those employed in PRIMAL, exhibit lower computational overhead at the cost of reduced perception fidelity [67]. These distinctions inform the selection of sensing strategies when balancing hardware constraints and navigation reliability.

Similarly, the review of discrete versus continuous action formulations shows clear practical differences. Continuous-action algorithms such as DDPG and its multi-agent variants enable smoother and more precise velocity control, which is essential for stable robot motion in real environments [40]. Meanwhile, discrete-action algorithms like DQN, DDQN, and CMAPPO are easier to train and more sample-efficient but often require an additional motion-smoothing layer or low-level controller before being deployed on physical robots [16]. These observations highlight how action-space design significantly affects controller feasibility and robustness.

The reviewed literature also provides insight into the behavior of DRL algorithms under realistic conditions. Tai and Liu (2016) showed that DQN can successfully handle simple corridor exploration tasks using depth images, but their results also reveal sensitivity to sparse rewards and exploration inefficiency [80]. More advanced methods such as PRIMAL and MAPPO-based approaches demonstrate that coordination improves substantially when centralized training is combined with decentralized execution [62, 67]. These results align with conclusions from MADDPG-based multi-robot exploration studies, which consistently report lower collision rates and faster area coverage [32].

5.2 Synthesis of Reported Experimental Outcomes

Across single-agent and multi-agent studies, several experimental patterns emerge. First, algorithms incorporating shaped or dense rewards tend to achieve faster convergence and superior path-quality metrics [46]. Second, works employing domain randomization or diverse training scenarios show greater robustness when transitioning from simulation to deployment [16]. Third, studies that align the action representation closely with physical robot motion models consistently demonstrate improved real-world stability [96].

In multi-agent exploration, CTDE-based approaches form the most reliable family of methods for real deployment. MADDPG, MAPPO, and value-decomposition methods demonstrate consistent improvements in coordination quality, area coverage efficiency, and collision avoidance when compared to purely decentralized or centralized learning alternatives. Experimental evidence across studies indicates that these architectures scale more effectively, maintain performance under partial observability, and generalize better across unseen environments. Collectively, these findings reinforce the practical relevance of the proposed taxonomy by illustrating how each component influences deployability, robustness, and overall exploration efficiency in real-world robotic systems.

6 Optimization of DRL

Using DRL for multi-robot cooperation poses challenges, notably regarding resource intensiveness. The exploration in the training process is inefficient because teaching robots the optimal policy via trial and error can be computationally costly and time-intensive. Alongside that, aligning reward functions in DRL with real-world practicalities can be problematic. Defining accurate reward functions in complex, unknown environments may result in suboptimal learning outcomes. Due to these challenges, optimization of the DRL becomes necessary.

6.1 Optimizing Sample Experience

Experience replay is essential in off-policy DRL, but earlier approaches rely on random sampling, which is inefficient as not all experiences contribute equally [103]. To address this, prioritized sampling methods such as **Prioritized Experience Replay (PER)** assign higher sampling probabilities to more informative transitions based on **temporal-difference (TD)** error, enabling the agent to focus on experiences with greater learning potential while using importance sampling to reduce bias, thereby improving training stability and convergence [32, 69]. Further enhancements include partitioning experiences into positive and negative samples to reduce training imbalance and employing efficient data structures such as SumTree for scalable sampling [40]. Overall, these methods shift from uniform random sampling to selective experience utilization, improving sample efficiency and learning effectiveness [40].

6.2 Optimization of the Training

DRL training is inherently slow due to its reliance on trial-and-error exploration, motivating methods that introduce structured guidance to accelerate convergence. **Curriculum Learning (CL)** addresses this by progressively increasing task complexity and shaping the training distribution, often combined with intrinsic rewards derived from prediction errors to encourage effective exploration [16]. Auxiliary tasks further enhance learning by providing additional supervision signals, such as depth prediction and localization, which improve representation learning and training efficiency [82]. Beyond this, guided exploration strategies reduce inefficient random search. These include switchable controllers that initially enforce safe navigation behaviors before gradually transferring control to learned policies [14], as well as heuristic-based planners that decompose tasks into intermediate sub-goals [43]. Hybrid approaches combining reinforcement learning with imitation learning or demonstration-based guidance further improve policy quality by leveraging expert knowledge [27, 67]. In parallel, adaptive exploration strategies, such as dynamically adjusted ϵ -greedy methods, balance exploration and exploitation based on feedback from the environment [25]. Additionally, evolutionary mechanisms refine training by iteratively replacing weaker policies with stronger ones, promoting continuous policy improvement across agents [43].

6.3 Optimization of Reward Function

Reward design plays a critical role in guiding DRL. Structured reward shaping methods improve learning by embedding domain knowledge, such as artificial potential fields to balance goal attraction and obstacle avoidance [91]. Alternative approaches derive reward functions directly from optimization objectives to ensure alignment with task goals and improve convergence [36]. Heuristic-based rewards provide dense feedback to mitigate sparse reward issues and encourage efficient exploration [25], while advanced methods incorporate fuzzy logic to enhance adaptability in complex environments [55].

6.4 Optimization using Human Prior Knowledge

Incorporating human prior knowledge reduces exploration requirements and improves sample efficiency. This includes embedding domain knowledge into reward functions to guide early learning [72], initializing networks with structured representations to accelerate convergence [57], and leveraging transferable priors to improve learning in complex environments [58].

6.5 Optimization Techniques for Physical Environments

Real-world deployment introduces challenges such as noise, safety constraints, and sim-to-real gaps. Safety-aware methods integrate constraints and action correction to prevent collisions [76], while mean-field approaches improve scalability and stability in dense environments [100]. Domain randomization enhances robustness and transferability to real-world conditions [16], and aligning action spaces with physical robot dynamics improves execution stability [96]. Collectively, these techniques highlight a shift toward DRL optimization strategies tailored for practical robotic applications.

7 Conclusion

In this survey, we reviewed the landscape of DRL for autonomous exploration in both single-robot and multi-robot environments, with an emphasis on how existing studies have optimized DRL to overcome the limitations of conventional trial-and-error learning. By examining state and observation design, action representations, reward formulations, and architectural choices, we provided a unified taxonomy that clarifies how these components influence learning efficiency, robustness, and deployability. A key contribution of this survey is the systematic analysis of optimization strategies tailored for exploration tasks. These include prioritized and structured experience sampling, reward shaping and curriculum learning, action-space alignment with real robot dynamics, domain and dynamics randomization, safety-aware learning, and CTDE-based coordination enhancements. Our review shows that such optimizations are essential not only for accelerating convergence but also for improving stability, reducing sample complexity, and narrowing the sim-to-real gap challenges that have historically constrained the practical deployment of DRL. Furthermore, the survey identifies persistent constraints in current approaches, particularly regarding inefficient exploration. Although past studies have integrated DRL with various supporting algorithms, the continued reliance on prior environmental knowledge limits the ability of agents to train effectively in truly unknown environments. A notable research gap is the absence of reasoning-capable entities during the exploration phase, where agents typically follow reactive or heuristic-driven behaviors rather than grounded, generalizable reasoning. This limitation underscores the need for exploration strategies with embedded reasoning capabilities and more advanced optimization techniques, ultimately reducing dependence on pre-existing, global-planner-derived knowledge.

References

- [1] Hongrui An, Jiwei Hu, and Ping Lou. 2021. Obstacle avoidance path planning based on improved APF and RRT. In *Proceedings of the 2021 4th International Conference on Advanced Electronic Materials, Computers and Software Engineering*. IEEE, 1028–1032.
- [2] Guillen-Perez Antonio and Cano Maria-Dolores. 2022. Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections. *IEEE Transactions on Vehicular Technology* 71, 7 (2022), 7033–7043. DOI: <https://doi.org/10.1109/TVT.2022.3169907>
- [3] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. 2017. A brief survey of deep reinforcement learning. arXiv:1708.05866. Retrieved from <https://arxiv.org/abs/1708.05866>
- [4] Fatih Aydemir and Aydin Cetin. 2023. Multi-agent dynamic area coverage based on reinforcement learning with connected agents. *Computer Systems Science and Engineering* 45, 1 (2023).
- [5] Mauro Birattari, Antoine Ligot, Darko Bozhinoski, Manuele Brambilla, Gianpiero Francesca, Lorenzo Garattoni, David Garzón Ramos, Ken Hasselmann, Miquel Kegeleirs, Jonas Kuckling, et al. 2019. Automatic off-line design of robot swarms: A manifesto. *Frontiers in Robotics and AI* 6 (2019), 59.
- [6] Norlida Buniyamin, W Wan Ngah, Nohaidda Sariff, Zainuddin Mohamad, et al. 2011. A simple local path planning algorithm for autonomous mobile robots. *International Journal of Systems Applications, Engineering and Development* 5, 2 (2011), 151–159.
- [7] N. Buniyamin, N. Sariff, WAJ Wan Ngah, and Z. Mohamad. 2011. Robot global path planning overview and a variation of ant colony system algorithm. *International Journal of Mathematics and Computers in Simulation* 5, 1 (2011), 9–16.
- [8] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. 2016. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics* 32, 6 (2016), 1309–1332.
- [9] Jingang Cao. 2016. Robot global path planning based on an improved ant colony algorithm. *Journal of Computer and Communications* 4, 2 (2016), 11–19.
- [10] Jingwei Cao, Shancheng Zhao, Caiqin Guo, Youtao Hu, Qianxu Gao, Fengjun Shen, Yujie Ye, and Yong Zhang. 2021. Study on the path planning algorithm based on dueling deep Q network. In *Proceedings of the Journal of Physics: Conference Series*. IOP Publishing, 012084.
- [11] Mert Çetinkaya. 2021. Multi-agent path planning using deep reinforcement learning. arXiv:2110.01460. Retrieved from <https://arxiv.org/abs/2110.01460>
- [12] Amrita Chakraborty and Arpan Kumar Kar. 2017. Swarm intelligence: A review of algorithms. *Nature-inspired Computing and Optimization: Theory and Applications* (2017), 475–494.
- [13] Guangda Chen, Lifan Pan, Pei Xu, Zhiqiang Wang, Peichen Wu, Jianmin Ji, Xiaoping Chen, et al. 2020. Robot navigation with map-based deep reinforcement learning. In *Proceedings of the 2020 IEEE International Conference on Networking, Sensing and Control*. IEEE, 1–6.
- [14] Wenzhou Chen, Shizheng Zhou, Zaisheng Pan, Huixian Zheng, and Yong Liu. 2019. Mapless collaborative navigation for a multi-robot system based on the deep reinforcement learning. *Applied Sciences* 9, 20 (2019), 4198.
- [15] Yinliang Chen and Liang Liang. 2023. SLP-Improved DDPG path-planning algorithm for mobile robot in large-scale dynamic environment. *Sensors* 23, 7 (2023), 3521.
- [16] Zichen Chen, Budhitama Subagdja, and Ah-Hwee Tan. 2019. End-to-end deep reinforcement learning for multi-agent collaborative exploration. In *Proceedings of the 2019 IEEE International Conference on Agents*. IEEE, 99–102.
- [17] Tim De Bruin, Jens Kober, Karl Tuyls, and Robert Babuška. 2018. Experience selection in deep reinforcement learning for control. *Journal of Machine Learning Research* 19, 9 (2018), 1–56.
- [18] Yuansheng Dong and Xingjie Zou. 2020. Mobile robot path planning based on improved DDPG reinforcement learning algorithm. In *Proceedings of the 2020 IEEE 11th International Conference on Software Engineering and Service Science*. IEEE, 52–56.
- [19] Emre Erkan and Muhammet Ali Arserim. 2022. Mobile robot application with hierarchical start position DQN. *Computational Intelligence and Neuroscience* 2022 (2022).
- [20] Shumin Feng, Bijo Sebastian, and Pinhas Ben-Tzvi. 2021. A collision avoidance method based on deep reinforcement learning. *Robotics* 10, 2 (2021), 73.
- [21] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [22] Gianpiero Francesca and Mauro Birattari. 2016. Automatic design of robot swarms: Achievements and challenges. *Frontiers in Robotics and AI* 3 (2016), 29.
- [23] Luiza Caetano Garaffa, Maik Basso, Andréa Aparecida Konzen, and Edison Pignatton de Freitas. 2021. Reinforcement learning for mobile robotics exploration: A survey. *IEEE Transactions on Neural Networks and Learning Systems* (2021).
- [24] Martin Gromniak and Jonas Stenzel. 2019. Deep reinforcement learning for mobile robot navigation. In *Proceedings of the 2019 4th Asia-Pacific Conference on Intelligent Robot Systems*. IEEE, 68–73.

- [25] Meng Guan, Fu Xing Yang, Ji Chao Jiao, and Xin Ping Chen. 2021. Research on path planning of mobile robot based on improved deep Q network. In *Proceedings of the Journal of Physics: Conference Series*. IOP Publishing, 012024.
- [26] Antonio Guillen-Perez and Maria-Dolores Cano. 2018. Flying ad hoc networks: A new domain for network communications. *Sensors* 18, 10 (2018). DOI: <https://doi.org/10.3390/s18103571>
- [27] Antonio Guillen-Perez and Maria-Dolores Cano. 2022. Learning from oracle demonstrations-a new approach to develop autonomous intersection management control algorithms based on multiagent deep reinforcement learning. *IEEE Access* 10 (2022), 53601–53613. DOI: <https://doi.org/10.1109/ACCESS.2022.3175493>
- [28] Antonio Guillen-Perez, Ana-Maria Montoya, Juan-Carlos Sanchez-Aarnoutse, and Maria-Dolores Cano. 2021. A comparative performance evaluation of routing protocols for flying Ad-Hoc networks in real conditions. *Applied Sciences* 11, 10 (2021). DOI: <https://doi.org/10.3390/app11104363>
- [29] Dingjie He, Dawei Feng, Hongda Jia, and Hui Liu. 2020. Decentralized exploration of a structured environment based on multi-agent deep reinforcement learning. In *Proceedings of the 2020 IEEE 26th International Conference on Parallel and Distributed Systems*. IEEE, 172–179.
- [30] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E. Taylor. 2019. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems* 33, 6 (2019), 750–797.
- [31] Jingyi Huang, Bo Li, Chao Song, and Neretin Evgeny. 2023. Multi UAV cooperative reconnaissance based on dynamic programming IDQN algorithm. In *Proceedings of the 2023 International Conference on Cyber-Physical Social Intelligence*. IEEE, 227–232.
- [32] Yixin Huang, Shufan Wu, Zhongcheng Mu, Xiangyu Long, Sunhao Chu, and Guohong Zhao. 2020. A multi-agent reinforcement learning method for swarm robots in space collaborative exploration. In *Proceedings of the 2020 6th International Conference on Control, Automation and Robotics*. IEEE, 139–144.
- [33] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. 2019. Deep reinforcement learning for swarm systems. *Journal of Machine Learning Research* 20, 54 (2019), 1–31.
- [34] Razin Bin Issa, Md. Saferi Rahman, Modhumonty Das, Monika Barua, and Md. Golam Rabiul Alam. 2020. Reinforcement learning based autonomous vehicle for exploration and exploitation of undiscovered track. In *Proceedings of the 2020 International Conference on Information Networking*. IEEE, 276–281.
- [35] Saksham Jain, Urvashi Agrawal, Amit Kumar, Anand Agrawal, and Gyan Singh Yadav. 2021. Simultaneous localization and mapping for autonomous robot navigation. In *Proceedings of the 2021 International Conference on Communication, Control and Information Sciences*. IEEE, 1–5.
- [36] Yue Jin, Yaodong Zhang, Jian Yuan, and Xudong Zhang. 2019. Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning. In *Proceedings of the ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2897–2901.
- [37] Siddique Latif, Heriberto Cuayáhuitl, Farrukh Pervez, Fahad Shamshad, Hafiz Shehbaz Ali, and Erik Cambria. 2023. A survey on deep reinforcement learning for audio-based applications. *Artificial Intelligence Review* 56, 3 (2023), 2193–2240.
- [38] Juncheng Li, Maopeng Ran, Han Wang, and Lihua Xie. 2021. A behavior-based mobile robot navigation method with deep reinforcement learning. *Unmanned Systems* 9, 03 (2021), 201–209.
- [39] Jingchen Li, Haobin Shi, and Kao-Shing Hwang. 2022. Using fuzzy logic to learn abstract policies in large-scale multiagent reinforcement learning. *IEEE Transactions on Fuzzy Systems* 30, 12 (2022), 5211–5224.
- [40] Peng Li, Siyu Jia, and Zhengyang Cai. 2021. Research on multi-robot path planning method based on improved MADDPG algorithm. In *Proceedings of the 2021 China Automation Congress*. IEEE, 3567–3572.
- [41] Yuxi Li. 2017. Deep reinforcement learning: An overview. arXiv:1701.07274. Retrieved from <https://arxiv.org/abs/1701.07274>
- [42] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. arXiv:1509.02971. Retrieved from <https://arxiv.org/abs/1509.02971>
- [43] Zuxin Liu, Baiming Chen, Hongyi Zhou, Guru Koushik, Martial Hebert, and Ding Zhao. 2020. Mapper: Multi-agent path planning with evolutionary reinforcement learning in mixed dynamic environments. In *Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 11748–11754.
- [44] Samuel Yanes Luis, Daniel Gutiérrez Reina, and Sergio I Toral Marín. 2021. A multiagent deep reinforcement learning approach for path planning in autonomous surface vehicles: The Ypacarai lake patrolling case. *IEEE Access* 9 (2021), 17084–17099.
- [45] Tianze Luo, Budhitama Subagdja, Di Wang, and Ah-Hwee Tan. 2019. Multi-agent collaborative exploration through graph-based deep reinforcement learning. In *Proceedings of the 2019 IEEE International Conference on Agents*. IEEE, 2–7.
- [46] Bruna G. Maciel-Pearson, Letizia Marchegiani, Samet Akcay, Amir Atapour-Abarghouei, James Garforth, and Toby P. Breckon. 2019. Online deep reinforcement learning for autonomous UAV navigation and exploration of outdoor environments. arXiv:1912.05684. Retrieved from <https://arxiv.org/abs/1912.05684>

- [47] Alexei A. Makarenko, Stefan B. Williams, Frederic Bourgaunt, and Hugh F. Durrant-Whyte. 2002. An experiment in integrated exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 534–539.
- [48] Enrico Marchesini and Alessandro Farinelli. 2021. Centralizing state-values in dueling networks for multi-robot reinforcement learning mapless navigation. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4583–4588.
- [49] Enrico Marchesini and Alessandro Farinelli. 2021. Centralizing state-values in dueling networks for multi-robot reinforcement learning mapless navigation. In *Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4583–4588.
- [50] Pablo Marin-Plaza, Ahmed Hussein, David Martin, and Arturo de la Escalera. 2018. Global and local path planning study in a ROS-based research platform for autonomous vehicles. *Journal of Advanced Transportation* 2018 (2018), 1–10.
- [51] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. 2016. Learning to navigate in complex environments. arXiv:1611.03673. Retrieved from <https://arxiv.org/abs/1611.03673>
- [52] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1928–1937.
- [53] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fiedelnd, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
- [54] Seongin Na, Tomáš Rouček, Jiří Ulrich, Jan Pikman, Tom s Krajník, Barry Lennox, and Farshad Arvin. 2023. Federated reinforcement learning for collective navigation of robotic swarms. *IEEE Transactions on Cognitive and Developmental Systems* (2023).
- [55] Muhammad Faqihuddin Nasary, Azhar Mohd Ibrahim, Suaib Al Mahmud, Amir Akramin Shafie, and Muhammad Imran Mardzuki. 2024. Optimizing mobile robot navigation through neuro-symbolic fusion of deep deterministic policy gradient (DDPG) and fuzzy logic. In *Proceedings of the Robotics, Computer Vision and Intelligent Systems*, Joaquim Filipe and Juha Röning (Eds.). Springer Nature Switzerland, Cham, 278–292.
- [56] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. 2020. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics* 50, 9 (2020), 3826–3839.
- [57] Kana Ogawa and Pitoyo Hartono. 2022. Infusing common-sensical prior knowledge into topological representations of learning robots. *Artificial Life and Robotics* 27 (2022). DOI: <https://doi.org/10.1007/s10015-022-00776-5>
- [58] Minako Oriyama, Pitoyo Hartono, and Hideyuki Sawada. 2024. Human-guided transfer learning for autonomous robot. In *Proceedings of the Neural Information Processing*, Biao Luo, Long Cheng, Zheng-Guang Wu, Hongyi Li, and Chaojie Li (Eds.). Springer Nature Singapore, Singapore, 186–198.
- [59] James Orr and Ayan Dutta. 2023. Multi-agent deep reinforcement learning for multi-robot applications: A survey. *Sensors* 23, 7 (2023), 3625.
- [60] Georgios Papoudakis, Filippos Christianos, Arrayas Rahman, and Stefano V. Albrecht. 2019. Dealing with non-stationarity in multi-agent deep reinforcement learning. arXiv:1906.04737. Retrieved from <https://arxiv.org/abs/1906.04737>
- [61] Jinwoo Park, Jeonghwan Kim, and Tae Kyung Park. [n. d.]. Implementation of PPO for multi-agent path finding with dynamic obstacles. ([n. d.]).
- [62] Shaoming Peng. 2023. Multi-robot path planning combining heuristics and multi-agent reinforcement learning. arXiv:2306.01270. Retrieved from <https://arxiv.org/abs/2306.01270>
- [63] Simon JD Prince. 2023. *Understanding Deep Learning*. MIT Press.
- [64] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4295–4304. Retrieved from <https://proceedings.mlr.press/v80/rashid18a.html>
- [65] Yara Rizk, Mariette Awad, and Edward W. Tunstel. 2018. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems* 10, 3 (2018), 514–529.
- [66] Sajad Saedi, Michael Trentini, Mae Seto, and Howard Li. 2016. Multiple-robot simultaneous localization and mapping: A review. *Journal of Field Robotics* 33, 1 (2016), 3–46.
- [67] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, T.K. Satish Kumar, Sven Koenig, and Howie Choset. 2019. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters* 4, 3 (2019), 2378–2385.

- [68] Yoko Sasaki, Syusuke Matsuo, Asako Kanezaki, and Hiroshi Takemura. 2019. A3C based motion learning for an autonomous mobile robot in crowds. In *Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 1036–1042.
- [69] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. 2016. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.). Retrieved from <http://arxiv.org/abs/1511.05952>
- [70] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. arXiv:1707.06347. Retrieved from <https://arxiv.org/abs/1707.06347>
- [71] Samaneh Hosseini Semnani, Hugh Liu, Michael Everett, Anton De Ruiter, and Jonathan P How. 2020. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3221–3226.
- [72] Siti Sendari, Muladi Muladi, Firman Ardiyansyah, Samsul Setumin, Norrima Mokhtar, Hsien-I Lin, and Pitoyo Hartono. 2024. Common-sensical incentive reward in deep actor-critic reinforcement learning for mobile robot navigation. *International Journal of Innovative Computing, Information and Control: IJICIC* 20 (2024), 373–389. DOI: <https://doi.org/10.24507/ijicic.20.02.373>
- [73] Mohit Sewak. 2019. *Deep Reinforcement Learning*. Springer.
- [74] Guicheng Shen, Ran Ma, Zhong Tang, and Liangliang Chang. 2021. A deep reinforcement learning algorithm for warehousing multi-agv path planning. In *Proceedings of the 2021 International Conference on Networking, Communications and Information Technology*. IEEE, 421–429.
- [75] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. 2011. *Introduction to Autonomous Mobile Robots*. MIT Press.
- [76] Arshdeep Singh and Shashi Shekhar Jha. 2021. Learning safe cooperative policies in autonomous multi-uav navigation. In *Proceedings of the 2021 IEEE 18th India Council International Conference*. IEEE, 1–6.
- [77] Baoye Song, Zidong Wang, and Lei Zou. 2017. On global smooth path planning for mobile robots using a novel multimodal delayed PSO algorithm. *Cognitive Computation* 9, 1 (2017), 5–17.
- [78] Zezhi Sui, Zhiqiang Pu, Jianqiang Yi, and Xiangmin Tan. 2018. Path planning of multiagent constrained formation through deep reinforcement learning. In *Proceedings of the 2018 International Joint Conference on Neural Networks*. IEEE, 1–8.
- [79] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2085–2087.
- [80] Lei Tai and Ming Liu. 2016. Mobile robots exploration through cnn-based reinforcement learning. *Robotics and biomimetics* 3 (2016), 1–8.
- [81] Lei Tai, Giuseppe Paolo, and Ming Liu. 2017. 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*. IEEE, 31–36.
- [82] T. Tongloy, S. Chuwongin, K. Jaksukam, C. Chousangsuntorn, and S. Boonsang. 2017. Asynchronous deep reinforcement learning for the mobile robot navigation with supervised auxiliary tasks. In *Proceedings of the 2017 2nd International Conference on Robotics and Automation Engineering*. IEEE, 68–72.
- [83] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*.
- [84] Huijuan Wang, Yuan Yu, and Quanbo Yuan. 2011. Application of dijkstra algorithm in robot path-planning. In *Proceedings of the 2011 2nd International Conference on Mechanic Automation and Control Engineering*. IEEE, 1067–1069.
- [85] Yutong Wang, Mehul Damani, Pamela Wang, Yuhong Cao, and Guillaume Sartoretti. 2022. Distributed reinforcement learning for robot teams: A review. *Current Robotics Reports* 3, 4 (2022), 239–257.
- [86] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. 2016. Dueling network architectures for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*. PMLR, 1995–2003.
- [87] Yongyong Wei and Rong Zheng. 2021. Multi-robot path planning for mobile sensing through deep reinforcement learning. In *Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.
- [88] Jonas Westheider, Julius Rückin, and Marija Popović. 2023. Multi-UAV adaptive path planning using deep reinforcement learning. arXiv:2303.01150. Retrieved from <https://arxiv.org/abs/2303.01150>
- [89] Xing Xiangrui, Guo Jia, Liang Zhuguan, Liu Qianlin, and Ding Hongwei. 2022. DDPG-based improved seeker optimization algorithm for robot path planning. In *Proceedings of the 2022 2nd International Conference on Computer, Control and Robotics*. IEEE, 27–31.

- [90] Xinghui Xu and Borja García de Soto. [n. d.]. Deep reinforcement learning-based task assignment and path planning for multi-agent construction robots. ([n. d.]).
- [91] Junxiao Xue, Xiangyan Kong, Bowei Dong, and Mingliang Xu. 2021. Multi-agent path planning based on MPC and DDPG. arXiv:2102.13283. Retrieved from <https://arxiv.org/abs/2102.13283>
- [92] Yang Yang, Li Juntao, and Peng Lingling. 2020. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. *CAAI Transactions on Intelligence Technology* 5, 3 (2020), 177–183.
- [93] Han-ye Zhang, Wei-ming Lin, and Ai-xia Chen. 2018. Path planning for the mobile robot: A review. *Symmetry* 10, 10 (2018), 450.
- [94] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. 2021. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control* (2021), 321–384.
- [95] Lin Zhang, Yufeng Sun, Andrew Barth, and Ou Ma. 2020. Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning. *IEEE Access* 8 (2020), 184109–184119.
- [96] Shansi Zhang and Weiming Wang. 2019. Tracking control for mobile robot based on deep reinforcement learning. In *Proceedings of the 2019 2nd International Conference on Intelligent Autonomous Systems*. IEEE, 155–160.
- [97] Wenyu Zhang, Jingyao Gai, Zhigang Zhang, Lie Tang, Qingxi Liao, and Youchun Ding. 2019. Double-DQN based path smoothing and tracking control method for robotic vehicle navigation. *Computers and Electronics in Agriculture* 166 (2019), 104985.
- [98] Wenqi Zhang, Qiang Wang, Xiao Liu, Yuanwei Liu, and Yue Chen. 2020. Three-dimension trajectory design for multi-UAV wireless network with deep reinforcement learning. *IEEE Transactions on Vehicular Technology* 70, 1 (2020), 600–612.
- [99] Yujiao Zhao, Yong Ma, Guibing Zhu, Songlin Hu, and Xinpin Yan. 2024. Stable training via elastic adaptive deep reinforcement learning for autonomous navigation of intelligent vehicles. *Communications Engineering* 3, 1 (2024), 37.
- [100] Shangfei Zheng and Hong Liu. 2019. Improved multi-agent deep deterministic policy gradient for path planning-based crowd simulation. *IEEE Access* 7 (2019), 147755–147770.
- [101] Chengmin Zhou, Bingding Huang, and Pasi Fränti. 2021. An advantage actor-critic algorithm for robotic motion planning in dense and dynamic scenarios. arXiv:2102.03138. Retrieved from <https://arxiv.org/abs/2102.03138>
- [102] Siyu Zhou, Xin Liu, Yingfu Xu, and Jifeng Guo. 2018. A deep Q-network (DQN) based path planning method for mobile robots. In *Proceedings of the 2018 IEEE International Conference on Information and Automation*. IEEE, 366–371.
- [103] Jie Zhu, Fengge Wu, and Junsuo Zhao. 2021. An overview of the action space for deep reinforcement learning. In *Proceedings of the 2021 4th International Conference on Algorithms, Computing and Artificial Intelligence*. 1–10.
- [104] Kai Zhu and Tao Zhang. 2021. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology* 26, 5 (2021), 674–691.
- [105] Liangji Zou and Yihua Tan. 2023. Collaborative search planning of UAV swarms based on deep reinforcement learning. In *Proceedings of the 2023 5th International Conference on Intelligent Control, Measurement and Signal Processing*. IEEE, 1184–1187.

Received 29 April 2024; revised 3 April 2026; accepted 19 May 2026