Review

Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey

James Orr and Ayan Dutta *

School of Computing, University of North Florida, Jacksonville, FL 32224, USA
* Correspondence: a.dutta@unf.edu

Abstract: Deep reinforcement learning has produced many success stories in recent years. Some example fields in which these successes have taken place include mathematics, games, health care, and robotics. In this paper, we are especially interested in multi-agent deep reinforcement learning, where multiple agents present in the environment not only learn from their own experiences but also from each other and its applications in multi-robot systems. In many real-world scenarios, one robot might not be enough to complete the given task on its own, and, therefore, we might need to deploy multiple robots who work together towards a common global objective of finishing the task. Although multi-agent deep reinforcement learning and its applications in multi-robot systems are of tremendous significance from theoretical and applied standpoints, the latest survey in this domain dates to 2004 albeit for traditional learning applications as deep reinforcement learning was not invented. We classify the reviewed papers in our survey primarily based on their multi-robot applications. Our survey also discusses a few challenges that the current research in this domain faces and provides a potential list of future applications involving multi-robot systems that can benefit from advances in multi-agent deep reinforcement learning.

Keywords: deep reinforcement learning; multi-robot systems; multi-agent learning; survey

Citation: Orr, J.; Dutta, A. Multi-Agent Deep Reinforcement Learning for Multi-Robot Applications: A Survey. *Sensors* **2023**, *23*, 3625. https://doi.org/10.3390/s23073625

Academic Editors: David Cheneler and Stephen Monk

Received: 23 January 2023
Revised: 22 March 2023
Accepted: 28 March 2023
Published: 30 March 2023

1. Introduction

In a multi-robot application, several robots are usually deployed in the same environment [1–3]. Over time, they interact with each other via radio communication, for example, and coordinate to complete a task. Application areas include precision agriculture, space exploration, and ocean monitoring, among others. However, in all such real-world applications, many situations might arise that have not been thought of before deployment, and, therefore, the robots must need to plan online based on their past experiences. Reinforcement learning (RL) is one computing principle that we can use to tackle such dynamic and non-deterministic scenarios. Its primary foundation is trial and error—in a single-agent setting, the agent takes an action in a particular state of the environment, receives a corresponding reward, and transitions to a new state [4]. Over time, the agent learns which state–action pairs are worth re-experiencing based on the received rewards and which ones are not [5]. However, the number of state–action pairs becomes intractable, even for smallish computational problems. This has led to the technique known as deep reinforcement learning (DRL), where the expected utilities of the state–action pairs are approximated using deep neural networks [6]. Such deep networks can have hundreds of hidden layers [7]. Deep reinforcement learning has recently been used in finding a faster matrix multiplication solution [8], for drug discovery [9], to beat humans in Go [10], play Atari [6], and for routing in communication networks [11], among others. Robotics is no different—DRL has been used in applications ranging from path planning [12] and coverage [13] to locomotion learning [14] and manipulation [15].

Going one step further, if we introduce multiple agents to the environment, this increases the complexity [16]. Now, the agents not only need to learn from their own observations in the environment but also be mindful of other agents' transitions. This essentially means that one agent's reward may now be influenced by the actions of other agents, and this might lead to a non-stationary system. Although inherently more difficult, the use of multiple robots and, consequently, a multi-agent reinforcement learning framework for the robots is significant [17]. Such learning multi-robot systems (MRS) may be used for precision agriculture [18], underwater exploration [19], search and rescue [20], and space missions [21]. Robots' onboard sensors play a significant role in such applications. For example, the state space of the robots might include the current discovered map of the environment, which could be created by the robots' laser scanners [22]. The state might also include locations and velocities, for which the robot might need sensory information from GPS or an overhead camera [23]. Furthermore, vision systems, such as regular or multi-spectral cameras, can be used by the robots to observe the current state of the environment, and data collected by such cameras can be used for robot-to-robot coordination [24]. Therefore, designing deep reinforcement learning algorithms, potentially lightweight and sample-efficient, that will properly utilize such sensory information, is not only of interest to the artificial intelligence research community but to robotics as well. However, the last survey that reviewed the relevant multi-robot system application papers that use multi-agent reinforcement learning techniques was conducted by Yang and Gu in 2004 [17]. Note that the entire sub-field of *DRL was not invented until 2015* [6].

In this paper, we fill this significant void by reviewing and documenting relevant MRS papers that specifically use multi-agent deep reinforcement learning (MADRL). Since today's robotic applications can have a large state space and, potentially, large action spaces, we believe that reviewing only the DRL-based approaches, and not the classic RL frameworks, is of interest to the relevant communities. The primary contribution of this article is that, to the best of our knowledge, this is the *only* study that surveys multi-robot applications via multi-agent deep reinforcement learning technologies. This survey provides a foundation for future researchers to build upon in order to develop state-of-the-art multi-robot solutions, for applications ranging from task allocation and swarm behavior modeling to path planning and object transportation. An illustration of this is shown in Figure 1.

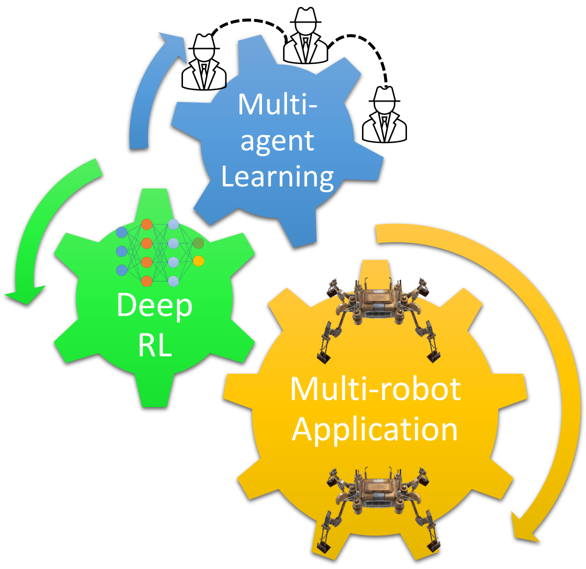

Figure 1. The main contribution of this article is that we have reviewed the latest multi-robot application papers that use multi-agent learning techniques via deep reinforcement learning. Readers will be able to find out how these three concepts are used together in the discussed studies and this survey will provide them with insight into possible future developments in this field, which, in turn, will advance the state-of-the-art.

We first provide a brief technical background and introduce terminologies necessary to understand some of the concepts and algorithms described in the reviewed papers

(Section 2). In Section 3, we categorize the multi-robot applications into (1) coverage, (2) path planning, (3) swarm behavior, (4) task allocation, (5) information collection, (6) pursuit–evasion, (7) object transportation, and (8) construction. We identify and discuss a list of crucial challenges that, in our opinion, the current studies in the literature face in Section 4, and then, finally, we conclude.

2. Background

In this section, we provide technical backgrounds on relevant computing principles.

2.1. MDP and Q-Learning

Let S and A denote the set of all states and actions available to an agent. Let R: $S \times A \to \mathbb{R}$ denote a reward function that gives the agent a virtual reward for taking action $a \in A$ in state $s \in S$. Let T denote the transition function. In a deterministic world, T: $S \times A \to S$, i.e., the actions of the agent is deterministic, whereas in a stochastic world, these actions might be probabilistic—T: $S \times A \to prob(S)$. We can use a Markov Decision Process (MDP) to model such a stochastic environment, which is defined as a tuple $\langle S, A, T, R \rangle$. The objective is to find a (optimal) policy π: $S \to A$ that maximizes the expected cumulative reward. To give higher preference to the immediate rewards than to the future ones, we discount the future reward values. The sum of the discounted rewards is called value. Therefore, to solve an MDP, we will maximize the expected value (V) over all possible sequences of states. Thus, the expected utility in a state $s \in S$ can be recursively defined as follows:

$$V(s) = R(s,a) + \gamma \max_{a' \in A} \sum_{s'} P(s'|s,a) V(s') \tag{1}$$

The above is called the Bellman equation, where $P(s'|s,a)$ is the probability of transitioning into s' from s by taking an action a. We can use value or policy iteration algorithms to solve an MDP.

However, in a situation where the R and T functions are unknown, the agent will have to try out different actions in every state to know which states are good and what action it should take in a particular state to maximize its utility. This leads to the idea of reinforcement learning (RL) where the agent will execute a in state s of the environment, and will receive a reward signal R from the environment as a result. Over time, the agent will learn the optimal policy based on this interaction between the agent and the environment [25]. An illustration is shown in Figure 2. In a *model-based* RL, the agent learns an empirical MDP by using estimated transition and reward functions. Note that these functions are approximated by interacting with the environment as mentioned earlier. Next, similar to an MDP, value or policy iteration algorithm can be employed to solve this empirical MDP model. In a *model-free* RL, the agent does not have access to T and R. This is true for numerous robotic applications in the real world. Therefore, most of the robotics papers we review in this survey use model-free RL techniques. This is also true for RL algorithms in general.

The goal of RL is to find a policy that maximizes the expected reward of the agent. Temporal difference learning is one of the most popular approach in model-free RL to learn learn the optimal utility values of each state. Q-learning is one such model-free RL technique, where the Q-value of a state–action pair (s,a) indicate the expected usefulness of that pair, which is updated as follows.

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(R(s,a) + \gamma \max_{a' \in A} Q(s',a')) \tag{2}$$

α is the learning rate that weighs the new observations against the old. It is off-policy learning and converges to an optimal policy π^* following

$$\pi^*(s) = \arg\max_{a \in A} Q(s,a) \tag{3}$$

An excellent overview of classic RL applications in robotics can be found in [26]. Keeping track of Q-values for all possible state–action pairs in such an RL setting becomes infeasible with, for example, a million such combinations. In recent years, artificial neural networks have been used to approximate the optimal Q-values instead of storing the values in a table. This has given birth to the domain of *deep* reinforcement learning.

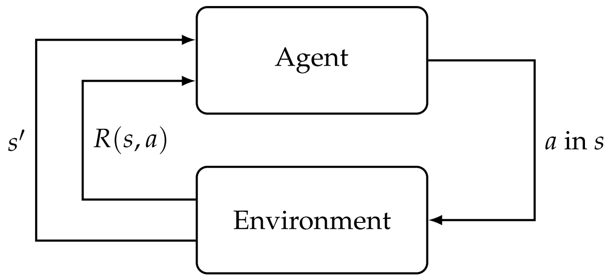

Figure 2. Illustration of Reinforcement Learning.

2.2. Multi-Agent Q-Learning

Assuming that the state space S is shared among n agents N and that there exists a common transition function T, an MDP for N is represented by the following tuple $\langle N, S, \mathbf{A}, \mathbf{O}, T, \mathbf{R} \rangle$, where the joint action space is denoted by $\mathbf{A} \leftarrow A_1 \times A_2 \cdots \times A_n$; the joint reward is denoted by $\mathbf{R} \leftarrow R_1 \times R_2 \cdots \times R_n$; and $\mathbf{O}$ denotes the joint observation of the agents, $\mathbf{O} \leftarrow O_1 \times O_2 \cdots \times O_n$. As there is more than one agent present, the action of one agent can potentially affect the reward and the consequent actions of the other agents. Therefore, the goal is to find a joint policy $\boldsymbol{\pi}^*$. However, due to the non-stationary environment and, consequently, the removal of the Markov property, convergence cannot be guaranteed unlike the single-agent setting [27]. One of the earliest approaches to learning a joint policy for two competitive agents is due to Littman [28]. It was modeled as a zero-sum two-player stochastic game (SG). It is also known as Markov Game in game theory. In SG, the goal is to find the Nash equilibrium, assuming the R and T functions are known. In a Nash equilibrium, the agents (or the players) will not have any incentive to change their adopted strategies. We slightly abuse the notation here and denote the strategy of agent N_i with π_i. Therefore, in a Nash equilibrium, the following is true

$$V_i^{\pi_i^*, \pi_{-i}^*}(s) \geq V_i^{\pi_i, \pi_{-i}^*}(s), \forall \pi_i \tag{4}$$

where $V(s)$ denotes the value of state $s \in S$ to the i-th agent and π_{-i} is the strategy of the other players. Here, we assume the agents to be rational, and, therefore, all the agents always follow their optimal strategies. This general SG setting can now be used to solve multi-agent reinforcement learning (MARL) problems.

In a cooperative setting, the agents have a common goal in mind. Most of the studies in the robotics literature that use MARL use such a cooperative setting. In this case, the agents have the same reward function, R. Given this, all the agents in N will have the same value function, and, consequently, the same Q-function. The Nash equilibrium will be the optimal solution for this problem. Two main types of learning frameworks are prevalent—independent and joint learners. In an independent learning scenario, each agent ignores the presence of other agents in the environment and considers their influence as noise. The biggest advantage is that each agent/robot can implement its own RL algorithm and there is no need for coordination and, consequently, a joint policy calculation [16,27]. Independent classic Q-learners have shown promising results in AI [29,30], as well as in robotics [31,32]. On the other hand, the joint learners aim to learn the joint optimal policy from $\mathbf{O}$ and $\mathbf{A}$. Typically, an explicit coordination, potentially via communication in an MRS, is in place and the agents learn a better joint policy compared to the independent learners [16,27]. However, the complexity increases exponentially with the number of agents causing these not to scale very well. The joint Q-learning algorithms are also popular in robotics [24,33,34], as well as in general AI [28,35]. A comprehensive survey for

MARL techniques can be found in [27,36]. The authors in [36] also discuss the application domains for MARL, which includes multi-robot teams. A specific relevant example that is discussed is multi-robot object transportation.

2.3. (Multi-Agent) Deep Q-Learning

As the state and the action spaces increase in size, maintaining a table for the Q-values for all possible state–action pairs might be infeasible. To tackle this challenge, Mnih et al. [6] have proposed a neural network-based approach to approximate the Q-values directly from the sensory inputs. This has given birth of 'deep' Q-learning, as the Q-values of the state–action pairs are updated using a deep neural network.

2.3.1. Q-Networks

In their seminal paper, Mnih et al. [6] have proposed DQN—a convolutional neural network (CNN) to approximate the Q-values for a single agent. This is called the Q-network, which is parameterized by θ. The current state s_t is passed as an input to the network that outputs the Q-values for all the possible actions. An action is chosen next based on the highest Q-value, i.e.,

$$a^* = \arg\max_{a \in A} Q(s_t, a) \tag{5}$$

To ensure that the agent explores the state space enough, a^* is chosen with probability ϵ and the agent takes a random action with $(1-\epsilon)$ probability. Due to this action, the state transitions to s_{t+1}. To avoid instability, a *target* network is maintained—it is identical to the Q network, but the parameter set θ is periodically copied to the parameters of this target network, θ^-. The state transitions are maintained in an experience replay buffer $\mathcal{D}$. Mini-batches from $\mathcal{D}$ are selected and target Q-values are predicted. θ is regressed toward the target values by finding the gradient descent of the following temporal loss function

$$\mathcal{L} = \mathbb{E}[(y_t - Q(s_t, a_t))^2] \tag{6}$$

$$y_t = R + \gamma Q(s_{t+1}, \arg\max Q(s_{t+1}, a_{t+1})) \tag{7}$$

One of the most popular extensions of DQN is Double DQN (DDQN) [37], which reduces overestimation in Q-learning. DDQN uses the Q-network for action selection following the ϵ-greedy policy, as mentioned above, but uses the target network for the evaluation of the state–action values. DQN and DDQN are extremely popular in robotics [13,38–41]. A visual working procedure of the generic DQN algorithm is presented in Figure 3.

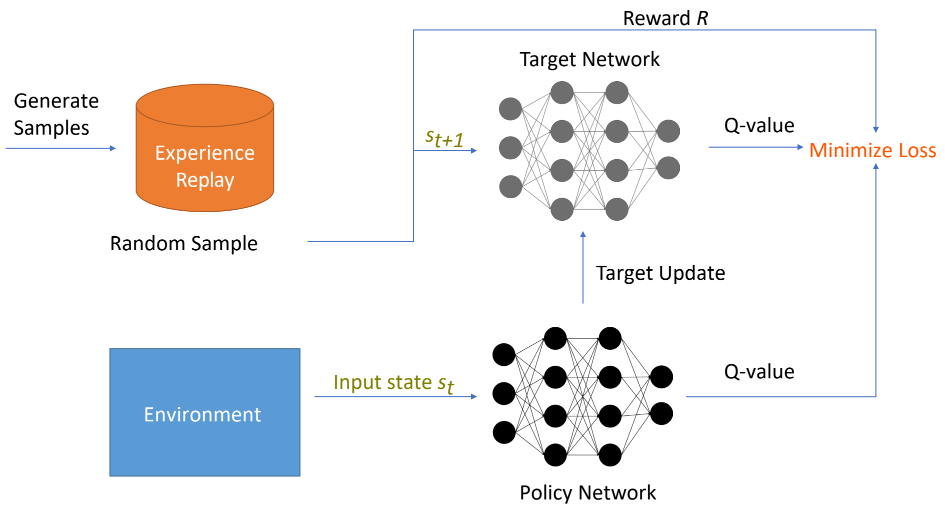

Figure 3. An illustration of the DQN architecture with a target network and an experience replay.

2.3.2. Policy Optimization Techniques

In policy optimization methods, the neural network outputs the probability distribution of these actions instead of outputting the Q-values of the available actions. Instead of using something like the ϵ-greedy strategy to derive a policy from the Q-values, the actions with higher probability outputs from the network will have higher chances of being selected. Let us denote a θ-parameterized policy by π_θ. The objective is to maximize the cumulative discounted reward

$$J(\theta) = \mathbb{E}[\mathcal{R}|\pi_\theta] \tag{8}$$

where $\mathcal{R}$ is the finite-horizon discounted cumulative reward. By optimizing the parameter set θ, e.g., by following the gradient of the policy, we aim to maximize the expected reward. Similar to the Q-networks, the learning happens in episodes. In general, the parameters in episode $i+1$, θ_{i+1}, will be an optimized version of θ_i as the following standard gradient ascent formula

$$\theta_{i+1} = \theta_i + \alpha \frac{\delta J(\theta_i)}{\delta \theta_i}. \tag{9}$$

In the vanilla form, similar to the Q-networks, the mean square error between the value of the policy (usually approximated using a neural network) and the reward-to-go (i.e., the sum of rewards received after every state transition so far) is calculated and the approximate value function parameters are regressed. Some of the popular policy optimization techniques include Deep Deterministic Policy Gradient (DDPG) [42], Proximal Policy Optimization (PPO) [43], Trust Region Policy Optimization (TRPO) [44], and Asynchronous Advantage Actor–Critic (A3C) [45], among others. Among these, DDPG is one of the most widely used for multi-robot applications [46–50]. It learns a Q-function similar to DQN and uses that to learn a policy. The policy DDPG learns is deterministic and the objective of this is to find actions that maximize the Q-values. As the action space A is assumed to be continuous, the Q-function is differentiable. To optimize θ and update the policy, we perform one-step policy ascent as follows:

$$\max_\theta \mathbb{E}_{s\in\mathcal{D}}[Q(s,\pi_\theta(s))]. \tag{10}$$

DDPG uses a sophisticated technique called actor–critic to achieve the successful combination of these two types of deep Q-learning. The actor essentially represents the policy and the critic represents the value network, respectively. The actor is updated towards the target and the critic is regressed by minimizing the error with the target [51]. The difference between the expected state value and the Q-value for an action a is called the *advantage*. One of the most popular algorithms that uses such an actor–critic framework is A3C [45]. In this algorithm, parallel actors explore the state space via different trajectories making the algorithm asynchronous; therefore, it does not require maintaining an experience replay. Another popular algorithm in the multi-robot domain is PPO, potentially because of its relatively simple implementation [43]. PPO-clip and PPO-penalty are its two primary variants that are used in robotics [52–57].

2.3.3. Extensions to Multi-Agent

As described earlier, in independent learning frameworks, any of the previously mentioned deep RL techniques, such as DQN, DDPG, A3C, or PPO, can be implemented on each agent. Note that no coordination mechanism is needed to be implemented for this [16,27,58].

For multi-agent DQN, a common experience memory can be used, which will combine the transitions of all the agents, and, consequently, they will learn from their global experiences while virtually emulating a stationary environment. Each agent can have its own network that will lead it to take an action from its Q-values [59]. Yang et al. [60] have proposed a mean field Q-learning algorithm for large-scale multi-agent learning applications. A mean-field formulation essentially brings down the complexity of an n-agent learning problem to a 2-agent learning problem by creating a virtual mean agent from the other $(n-1)$ agents in the environment. In [61], the authors have introduced the multi-agent extension of DDPG (MADDPG). Here, the actor remains decentralized, but the critic is centralized. Therefore, the critic needs information on the actions, observations, and target policies of all of the agents to evaluate the quality of the joint actions. Figure 4 shows an illustration of this process. Yu et al. [62] have proposed a multi-agent extension of PPO in cooperative settings (MAPPO). Similar to MADDPG, it uses centralized training with a decentralized execution strategy.

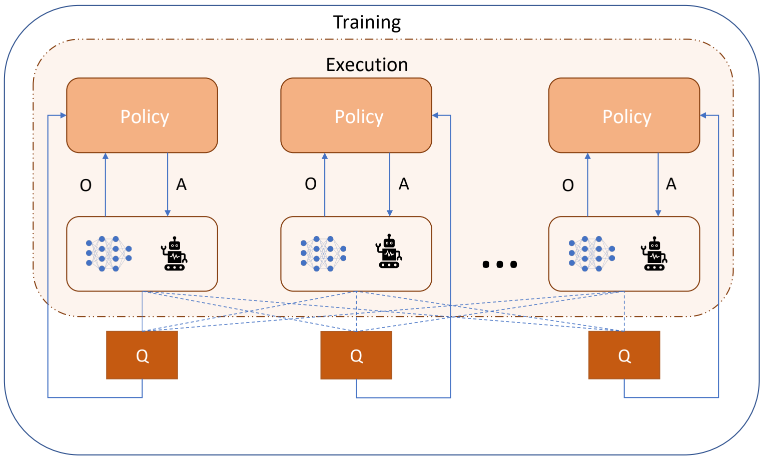

Figure 4. Illustration of multi-agent DDPG (MADDPG) [61].

Another approach to extending a single-agent DRL algorithm to a multi-agent setting is to model it as a centralized RL, where all the information from agents is input together. This might create an infeasibly large state and action space for the joint agent. To alleviate this, researchers have looked into how to find each agent's contribution to the joint reward. This is named Value Function Factorization. VDN [63] is one such algorithm for cooperative settings where the joint Q-value is the addition of the local Q-values of the agents. A summary of the main types of RL algorithms used in multi-robot applications is presented in Table 1. The reader is referred to [64,65] for recent comprehensive surveys on state-of-the-art MADRL techniques and challenges. Furthermore, Oroojlooy and Hajinezhad [66] have recently published a survey paper reviewing the state-of-the-art MADRL algorithms specifically for cooperative multi-agent systems. As in most of the scenarios, the robots in an MRS work together towards solving a common problem, we believe that the survey in [66] would be a valuable asset for the robotics community.

Table 1. Types of deep RL algorithms used in the surveyed papers are listed. If a popular algorithm is used as a foundation, the algorithm's name is also mentioned within parentheses.

Q-Networks	**Policy Gradients**		
	DDPG	**PPO**	**Other**
[23,24,39,40,59,67–92] (QMIX), [38] (DDQN), [93] (DDQN), [94] (DDQN), [95] (DQN), [96] (DQN)	[46–50,97–106],	[22,52–57,107–128]	[86,88,129–140] (TRPO), [141] (TRPO), [81] (TRPO), [142] (TD3), [143] (SAC), [144] (SAC)

3. Multi-Robot System Applications of Multi-Agent Deep Reinforcement Learning

A summary of the discussed multi-robot applications is presented in Figure 5.

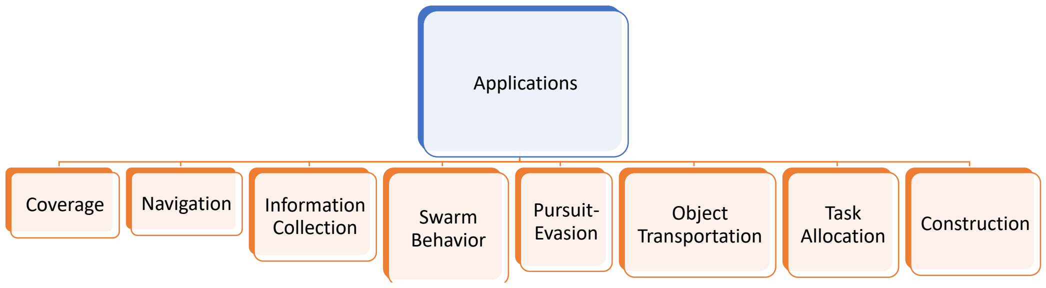

Figure 5. Multi-robot system applications that primarily use MADRL techniques.

3.1. Coverage and Exploration

The goal of an MRS in a coverage path planning (CPP) application is that every point in the environment is visited by at least one robot while some constraints are satisfied (e.g., no collision among the robots) and user-defined criteria are optimized (e.g., minimizing the travel time) [145]. CPP is one of the most popular topics in robotics. For multi-robot coverage, several popular algorithms exist even with performance guarantees and worst-case time bounds [146–149]. In exploration, however, the objective might not be the same as the multi-robot CPP problem. It is assumed that the sensor radius $r > 0$, and, therefore, the robots do not need to visit all the points on the plane. For example, the robots might be equipped with magnetic, acoustic, or infrared sensors in ground and aerial applications whereas a group of underwater vehicles might be equipped with water temperature and current measuring sensors. The robots will need GPS for outdoor localization. Such exploration can be used for mapping and searching applications among others [150–152]. Constraints such as maintaining wireless connectivity for robots with limited communication ranges might be present [153]. Inter-robot communication can be achieved via ZigBee or Wi-Fi. An example is shown in Figure 6.

Mou et al. [68] studied area coverage problems and proposed deep reinforcement learning for UAV swarms to efficiently cover irregular three-dimensional terrain. The basis of their UAV swarm structure is with the leader and the follower UAVs. The authors implement an observation history model based on convolutional neural networks and a mean embedding method to address limited communication. Li et al. [69] proposed the use of DDQN to train individual agents in a simulated grid-world environment. Then during the decision-making stage, where previously trained agents are placed in a test environment, the authors use their proposed multi-robot deduction method, which has foundations in Monte Carlo Tree Search. Zhou et al. [154] have developed a multi-robot coverage path planning mechanism that incorporates four different modules: (1) a map module, (2) a communication module, (3) a motion control module, and (4) a path generation module. They implement an actor–critic framework and natural gradient for updating the network. Up to three robots have been used in a simulation for testing the proposed coverage technique in a grid world. The two cornerstones of the study by Hu et al. [155] are (1) Voronoi partitioning-based area assignment to the robots and (2) the proposed DDPG-based DRL technique for the robots to have a collision-avoidance policy and evade objects in the field. The control of the robots is provided by the underlying neural network. The authors use a Prioritised Experience Replay (PER) [156] to store human demonstrations. The simulation was performed within Gazebo [157] and three Turtlebot3 Waffle Pi mobile robots were used to explore an unknown room during validation. Bromo [53], in his thesis, used MADRL on a team of UAVs using a modified version of PPO to map an area. During training, the policy function is shared among the robots and updated based on their current paths.

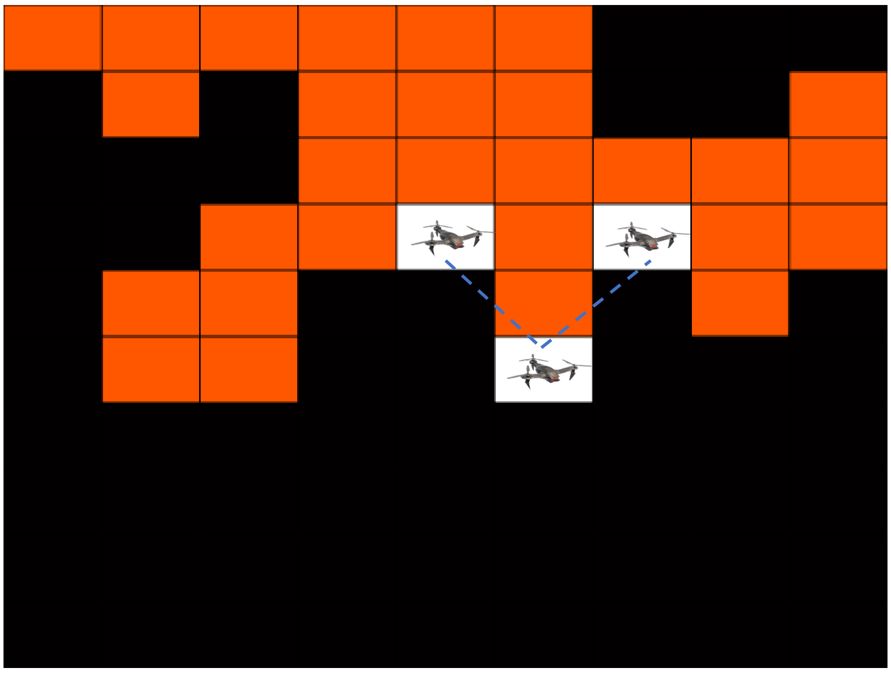

Figure 6. A multi-robot coverage scenario under continuous connectivity: the black, orange, and white cells represent the unvisited, previously visited, and the current cells of robots, respectively. The dotted lines represent the available communication channels among the robots. The goal is to cover the entire environment while maintaining a connected communication network throughout the mission.

For multi-UAV coverage, Tolstaya et al. [110] use graph neural networks (GNNs) [158] as a way for the robots to learn the environment through the abstractions of nodes and edges. GNNs have been successfully used in various coordination problems for multi-robot systems, and more recently, Graph Convolution Networks (GCNs) have been used [158]. The authors in this paper use "behavior cloning" as a heuristic to train the GNN on robots' previous experiences. In order for the individual UAVs to learn information distant from their position, they use up to 19 graph operation layers. PPO is the base DRL algorithm in this paper. Aydemir and Cetin [159] proposed a distributed system for the multi-UAV coverage in partially observable environments using DRL. Only the nearby robots share their state information and observations with each other. Blumenkamp et al. [111] developed a framework for decentralized coordination of an MRS. The RL aspect of their system uses GNNs and PPO. The agents train and develop a policy within a simulated environment and then the physical implementation of the policy with the robots occurs in a test environment. The authors also compare centralized control and communication levels to decentralized decision-making.

Similarly, Zhang et al. [160] have also proposed to employ graph neural networks for multi-robot exploration. The authors emphasize the "coarse-to-fine" exploration method of the robots, where the graph representation of the state space to be explored is explored in "hops" of greater detail. Simulation experiments involved up to 100 robots. Exploration can also be used for searching for a target asset. Liu et al. [84] have proposed a novel algorithm for cooperative search missions with a group of unmanned surface vehicles. Their algorithm makes use of two modules based on a divide-and-conquer architecture: an environmental sense module that utilizes sensing information and a policy module that is responsible for the optimal policy of the robots. Gao and Zhang [161] study a cooperative search problem while using MADRL as the solution method. The authors use independent learners on the robots to find the Nash equilibrium solution with the incomplete information available to the robots. Setyawan et al. [101] also use MADRL for multi-robot search and exploration. Unlike the previously mentioned studies, the authors have adopted a hierarchical RL approach, where they break down an abstraction of the global problem space into smaller sub-problem levels in order for the robot system to more efficiently learn in an actor–critic style. The lowest level in this order decides the robots' motor actions in the field. Sheng et al. [162] propose a novel probability density factorized multi-agent DRL method for solving the multi-robot reliable search problem. According to this study, when each robot follows its own policy to maximize its own reliability metric (e.g., probability of finding the target), the global reliability metric is also maximized. The authors implement the proposed technique on multiple simulated search environments including offices and museums, as well as on real robots. Another study in a similar application domain is done by Xia et al. [127]. Specifically, the authors have used MADRL for the multi-agent multi-target hunting problem. The authors make use of a feature embedding block to extract features from the agents' observations. The neural network architecture uses fully connected layers and a Gated Recurrent Unit (GRU) [163]. Simulation experiments included up to 24 robots and 12 targets. Caccavale et al. [96] proposed a DRL framework for a multi-robot system to clean and sanitize a railway station by coordinating the robots' efforts for maximum coverage. Their approach is decentralized where each robot runs its own CNN and the foundation of their technique is DQN. Note that the robots learn to cooperate online while taking the presence of the passengers into account.

Not only with the ground and aerial vehicles, but MADRL has also been used for ocean monitoring with a team of floating buoys as well. Kouzehgar et al. [105] proposed two area coverage approaches for such monitoring: (1) swarm-based (i.e., the robots follow simple swarming rules [164]) and (2) coverage-range-based (i.e., the robots with fixed sensing radius). The swarm-based model was trained using MADDPG and the latter model MARL was trained using a modified (consisting of eliminating reward sharing, collective reward, sensing their own share of the reward function, and independence based on individual reward) MADDPG algorithm.

Communication is one of the most important methods of coordination among a group of robots. More often than not, when and with whom the communication will happen is pre-defined. However, if the robots are non-cooperative, such an assumption does not work. Blumenkamp and Prorok [118] propose a learning model based on reinforcement learning that allows individual, potentially non-cooperative, agents to manipulate communication policies while the robots share a differentiable communication channel. The authors use GNN with PPO in their method. The proposed technique has also been successfully employed for multi-robot path planning. Along a similar path, Liang et al. [165] proposed the use of DRL to learn a high-level communication strategy. The authors presume the environment to be partially observable and they take a hierarchical learning approach. The implemented application is a cooperative patrolling field with moving targets. Meng and Kan [102] also put multi-robot communication at the forefront of their study while tackling the coverage problem. The goal of the robots has to cover an entire environment while maintaining connectivity in the team, e.g., via a tree topology. The authors use a modified version of MADDPG to solve the stated problem.

MADRL has also been used for sensor coverage, alongside area coverage [166]. In sensor-based coverage, the objective is to cover all the points in an environment with a sensor footprint. An example of this is communication coverage, where the goal of a team of UAVs is to provide Wi-Fi access to all the locations in a particular region. This might be extremely valuable after losing communication in a natural disaster, for example. The authors in [167] presented a solution for UAV coverage using mean field games [168]. This study was targeted toward UAVs that provide network coverage when network availability is down due to natural disasters. The authors constructed the Hamilton–Jacobi–Bellman [169] and Fokker–Planck–Kolmogorov [170] equations via mean field games. Their proposed neural network-based learning method is a modification of TRPO [44] and named mean-field trust region policy optimization (MFTRPO). Liu et al. [104] proposed a coverage method to have a system of UAVs cover an area and provide communication connectivity while maintaining energy efficiency and fairness of coverage. The authors utilize an actor–critic-based DDPG algorithm. Simulation experiments were carried out with up to 10 UAVs. Similar to these, Nemer et al. [171] proposed a DDPG-based MADRL framework for multi-UAV systems to provide better coverage, efficiency, and fairness for network coverage of an area. One of the key differentiating factors of this paper is that the authors also model energy-efficient controls of the UAVs to reduce the overall energy consumption by them during the mission. For a similar communication coverage application, Liu et al. [172] proposed that the UAVs have their own actor-critic networks for a fully-distributed control framework to maximize temporal mean coverage reward.

3.2. Path Planning and Navigation

In multi-robot path planning (or path finding), each robot is given a unique start and a goal location. Their objective is to plan a set of joint paths from the start to the goal, such that some pre-defined criteria, such as time and/or distance, are optimized and the robots avoid colliding with each other while following the paths. An illustration is presented in Figure 7. Planning such paths optimally has been proven to be NP-complete [173]. Like A^* [174], which is used for single-agent path planning in a discreet space, M^* [175] can be used for an MRS. Unfortunately, M^* lacks scalability. There exist numerous heuristic solutions for such multi-robot planning that scale well [176–179]. Overhead cameras and GPS can be used to localize the robots in indoor and outdoor applications, respectively. In GPS and communication-denied environments, vision systems can be used as a proxy [180]. Recently, researchers have started looking into deep reinforcement learning solutions to solve this notoriously difficult problem.

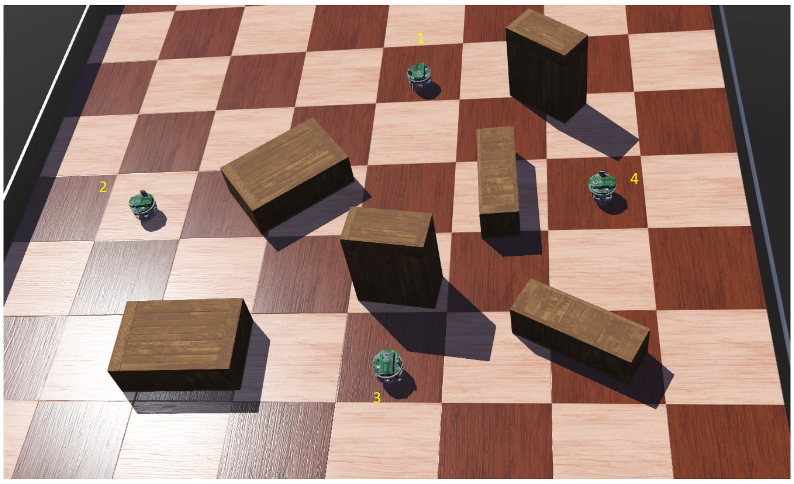

Figure 7. An illustration of multi-robot path planning with 4 e-puck robots, where the starting positions are shown and each robot's goal position is its orthogonal robot's starting location. The boxes represent the obstacles in the environment.

One of the most popular works that use MADRL for collision avoidance is due to Long et al. [22]. They propose a decentralized method using PPO while using CNNs to train the robots, which use their onboard sensors to detect obstacles. Up to 100 robots were trained and tested via simulation. Lin et al. [109] proposed a novel approach for centralized training and decentralized execution for a team of robots that need to concurrently reach a destination while avoiding objects in the environment. The authors implement their method using CNNs and PPO as well. The learned policy maps LiDAR measurements to the controls of the robots. Bae et al. [72] also use CNNs to train multiple robots to plan paths. The environment is treated as an image where the CNN extracts the features from the environment, and the robots share the network parameters.

Fan et al. [107] have proposed a DRL model technique using the policy gradient method to train the robots to avoid collisions with each other while navigating in an environment. The authors use LiDAR data for training, and, during testing, this drives the decision-making process to avoid collisions. The authors then transfer the learned policy to physical robots for real-world feasibility testing. The simulation included up to 100 robots with the objective of avoiding collisions with each other, static objects, and, finally, pedestrians. It builds on their previous work from 2018 [131]. Wang et al. [181] also use CNNs for multi-robot collision avoidance and coordination. The authors also use a recurrent module, namely Long Short Term Memory (LSTM) [182] to memorize the actions of the robots to smooth the trajectories. The authors have shown that the combined use of CNN and LSTM can produce smoother paths for the robots in a continuous domain.

Yang et al. [71] use a priori knowledge to augment the DDQN algorithm to improve the learning efficiency in multi-robot path planning. To avoid random exploration at the beginning of the learning process, the authors have used A^* [174] paths for single robots in static environments. This provides better preliminary Q-values to the networks, and, thus, the overall learning process converges relatively quickly. Wang and Deng [39] propose a novel neural network structure for task assignment and path planning where one network processes a top–down view of the environment and another network processes the first-person view of the robot. The foundation of the algorithm is also based on DQN. Na et al. [49] have used MADRL for collision avoidance among autonomous vehicles via modeling virtual pheromones inspired by nature. The authors also used a similar pheromone-based technique, along with a modified version of PPO in [55] for the same objective. Ourari et al. [123] also used a biologically-inspired method (specifically from the behavior of flocks of starlings) for multi-robot collision avoidance while a DRL method, namely PPO, is at its foundation. Their method is executed in a distributed manner and each robot incorporates information from k-nearest neighbors.

For multi-robot target assignment and navigation, Han, Chen, and Hao [117] proposed to train the policy in a simulated environment using randomization to decrease the performance transfer from simulation to the real world. The architecture they developed utilized communication amongst the robots to share experiences. They also developed a training algorithm for navigation policy, target allocation, and collision avoidance. It uses PPO as a foundation. Moon et al. [38] used MADRL for the coordination of multiple UAVs that track first responders in an emergency response situation. One of the key ideas behind their method is the inclusion of the Cramér–Rao lower bound into the learning process. The intent of the authors was to use the DRL-based UAV control algorithm to accurately track the target(s) of the UAV system. They used DDQN as their foundation technique.

Marchesini and Farinelli [74] extended their work in [75] by incorporating an Evolutionary Policy Search (EPS) for multi-robot navigation. It had two main components: navigation (reaching a target) and avoiding collisions. They extended their prior work [75] (using DDQN and LSTM at its core) by incorporating the EPS, which integrated randomization and genetic learning into the MARL technique to enhance the ability for the policy to explore and help the robots learn to navigate better.

Lin et al. [112] developed a novel deep reinforcement learning approach for coordinating the movements of an MRS such that the geometric center of the robots reached a target destination while maintaining a connected communication graph throughout the mission. Similarly, Li et al. [183] proposed a DRL method for multi-robot navigation while maintaining connectivity among the robots. The presented technique used constrained policy optimization [184] and behavior cloning. Real-world experiments with five ground robots show the efficacy of the proposed method. Maintaining such connectivity has previously been studied in an information collection application [185] applied to precision agriculture, albeit from a combinatorial optimization perspective [18].

On the other hand, Huang et al. [167] proposed a deep Q-learning method for maintaining connectivity between leader and follower robots. Interestingly, the authors do not use CNNs, instead, they rely only on dense fully connected layers in their network. Similar to these, Challita et al. [167] developed a novel DRL framework for UAVs to learn an optimal joint path while maintaining cellular connectivity. Their main contribution is founded in game theory. The authors used an Echo State Network (ESN), a type of recurrent neural network. In a similar setting, the authors' other work [186] studied minimizing interference from the cellular network using MADRL. Wang et al. [187] proposed to incorporate environmental spatiotemporal information. The proposed method used a global path planning algorithm with reinforcement learning at the local level via DDQN combined with an LSTM module. Choi et al. [92] also used a recurrent module, namely GRU along with CNN for the multi-agent path planning problem in an autonomous warehouse setting. The base of their work was the popular QMIX [188] algorithm, a form of value function factorization algorithm similar to VDN [63]. Another study of multi-robot path planning for warehouse production scenarios was carried out by Li and Guo [128]. They proposed a supervised DRL approach efficient path planning and collision avoidance. More specifically, using imitation learning and PPO, Li and Guo aimed to increase the learning performance of the vehicles in object transportation tasks.

Yao et al. [115] developed a map-based deep reinforcement learning approach for multi-robot collision avoidance, where the robots do not communicate with one another for coordination. The authors used an egocentric map as the basis of information that the robots use to avoid collisions. Three robots have been used for real-world implementations. Similar to this, Chen et al. [189] also did not rely on inter-robot communication for multi-robot coordinated path planning and collision avoidance while also navigating around pedestrians. Chen et al. [94]'s study on multi-robot path planning also considered non-communicating and decentralized agents using DDQN. Simulation experiments involved up to 96 robots.

Tan et al. [116] have developed a novel algorithm, called DeepMNavigate that uses local and global map information, PPO, and CNNs for navigation and collision avoidance learning. Their algorithm also makes use of multi-staged training for robots. Simulation experiments involved up to 90 robots. Chen et al. [190] proposed a method of using DRL in order for robots to learn human social patterns to better avoid collisions. As human behaviors are difficult to model mathematically, the authors noted that social rules usually emerge from local interactions, which drives the formulation of the problem. Chen et al. [87] proposed a novel DRL framework using hot-supervised contrastive loss (via supervised contrastive learning) combined with DRL loss for pathfinding. The robots do not use communication. They also incorporated a self-attention mechanism in the training. Their network structure used CNNs with DQN while up to 64 agents have been used for testing the approach in simulation. Navigation control using MADRL was also studied in [88], where the authors showed that the robots could recover from reaching a dead end. Alon and Zhou [135] have proposed a multi-critic architecture that also included multiple value networks. Path planning is also important in delivering products to the correct destinations. Ding et al. [91] have proposed a DQN-based MADRL technique for this specific application while combining it with a classic search technique, namely the Warshall–Floyd algorithm.

Transfer learning and federated deep learning have also been used for multi-robot path planning. In transfer learning, the assumption that the training and the testing data come from the same domain does not need to hold, which makes it attractive in many real-world scenarios, including robotics [191]. The objective here is to *transfer* the learning from one or more source domains to a potentially different target domain. Wen et al. [133] developed two novel reinforcement learning frameworks that extend the PPO algorithm and incorporate transfer learning via meta-learning for path planning. The robots learn policies in the source environments and obtain their policies following the proposed training algorithm. Next, this learning is then transferred to target environments, which might have more complex obstacle configurations. This increases the efficiency of finding the solutions in the target environments. The authors used LSTM in their neural network for memorizing the history of robot actions. In federated deep learning, training data might still be limited similar to the transfer learning applications. In this case, each agent has its own training data instead of using data shared by a central observer [192]. For example, each robot might have access to a portion of the environment, and they are not allowed to share the local images with each other, where the objective is still to train a high-quality global model. Luo et al. [193] have employed such a federated deep RL technique for multi-robot communication. The authors, in this paper, avoid blockages in communication signals due to large obstacles while avoiding inter-robot collisions. It has been shown that the proposed semi-distributed optimization technique is 86% more efficient than a central RL technique. Another federated learning-based path planning technique can be found in [130]. To reduce the volume of exchanged data between a central server and an individual robot, the proposed technique only shares the weights and biases of the networks from each agent. This might be significant in scenarios where the communication bandwidth is limited. The authors show that the presented technique in their paper offers higher robustness than a centralized training model.

PRIMAL is a multi-agent path-finding framework that uses MADRL and is proposed by Sartoretti et al. [194]. PRIMAL used the A3C [45] algorithm and an LSTM module. It also makes use of imitation learning whereby each agent can be given a copy of the centrally trained policy by an expert [195]. One of the highlights of this paper is that the proposed technique could scale up to 1024 robots albeit in simulation. PRIMAL$_2$ [196] is the advanced version of PRIMAL and was proposed by Damani et al. in 2021. It also uses A3C as its predecessor, offers real-time path re-planning, and scales up to 2048 robots—double that which PRIMAL could do.

Curriculum learning [197] has also been used for multi-robot path planning in [198], where the path planning is modeled as a lesson, going from easy to hard difficulty levels. An end-to-end MADRL system for multi-UAV collision avoidance using PPO has been proposed by Wang et al. [57]. Asayesh et al. [137] proposed a novel module for safety control of a system of robots to avoid collisions. The authors use LSTM and a Variational Auto-Encoder [199]. Li [200] has proposed using a lightweight decentralized learning framework for multi-agent collision avoidance by using only a two-layer neural network. Thumiger and Deghat [56] used PPO with an LSTM module for multi-UAV decentralized collision avoidance. Along the same line, Han et al. [54] used GRUs and their proposed reward function used reciprocal velocity obstacle for distributed collision avoidance.

For collaborative motion planning with multiple manipulators, Zhao et al. [108] proposed a PPO-based technique. The manipulators learned from their own experiences, and then, a common policy was updated while the arms continued to learn from individual experiences. This created differences in accuracy or actuator ability among the manipulators. Similarly, Gu et al. [50] proposed a method for asynchronous training of manipulator arms using DDPG and Normalized Advantage Function (NAF). Real-world experiments were carried out with two manipulators. Prianto et al. [143] proposed the use of the Soft Actor–Critic (SAC) algorithm [14] due to its efficiency in exploring large state spaces for path planning with a multi-arm manipulator system, i.e., each arm has its own unique start and goal configurations. Unlike the previous works in this domain, the authors used Hindsight Experience Replay (HER) [201] for sample-efficient training. On the other hand, Cao et al. [144] proposed a DRL framework for a multi-arm manipulator to track trajectories. Similarly to [143], Cao et al. also used SAC as their base algorithm. The main distinguishing factor of this study is that the multiple manipulator arms were capturing a non-cooperative object. Results show that the dual-arm manipulator can capture a rotating object in space with variable rotating speeds. An illustration of such a dual-arm manipulation application is shown in Figure 8.

Everett et al. [202] have proposed to use LSTM and extend their previous DRL algorithm [189] for multi-robot path planning to enhance the ability of the robots to avoid collisions. Semnani et al. [203] proposed an extension of the work proposed in [202] by using a new reward function for multi-agent motion planning in three-dimensional dense spaces. They used a hybrid control framework by combining DRL and force-based motion planning. Khan et al. [136] have proposed using GCN and a DRL algorithm called Graph Policy Gradients [134] for unlabeled motion planning of a system of robots. The multi-robot system must find the goal assignments while optimizing their trajectories.

Song et al. [90] designed a new actor–critic algorithm and a method for extracting the state features via a local-and-global attention module for a more robust MADRL method with an increasing number of agents present in the environment. The simulated experiments used dynamic environments with simulated pedestrians. Zhang et al. [204] proposed a method for using a place-timed Petri net and DRL for the multi-vehicle path planning problem. They used a curriculum-based DRL model. Huang et al. [205] proposed a vision-based decentralized policy for path planning. The authors use Soft Actor–Critic with auto encoders [206] as their deep RL technique for training a multi-UAV system. The 3D images captured by the UAVs and their inertial measurement values were used as inputs, whereas the control commands were rejected by the neural network. Simulation experiments with up to 14 UAVs were performed within the Airsim simulator. Jeon et al. [207] proposed to use MADRL to improve the energy efficiency of coordinating multiple UAVs within a logistic delivery service. The authors show that their model performs better in terms of consumed energy while delivering similar numbers of goods.

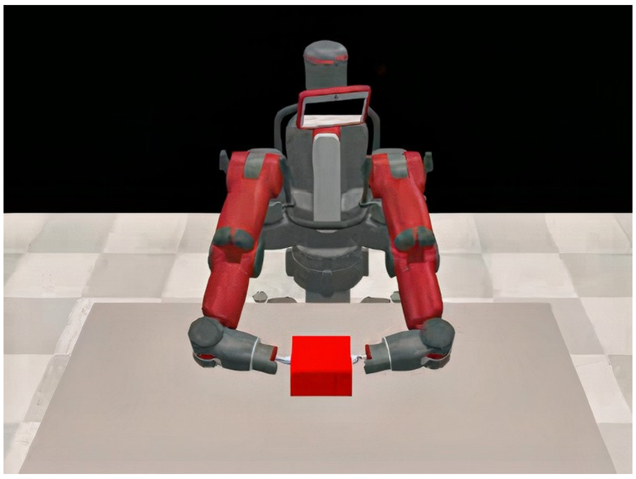
Figure 8. An illustration of dual-arm manipulation is shown using a Baxter robot where each arm might act as an RL agent.

MADRL has also found its way into coordinating multiple autonomous vehicles. The authors in [208] provide a solution to the “double merge” scenario for autonomous driving cars that consists of three primary contributions in this field: (1) the variance of the gradient estimate can be minimized without Markovian assumptions, (2) trajectory planning with hard constraints to maintain the safety of the maneuver [209], and (3) introduction of a hierarchical temporal abstraction [25] that they call an “Option Graph” to reduce the effective horizon which ultimately reduces the variance of the gradient estimation [210,211]. Similar to this, Liang et al. [212] have modeled the cooperative lane changing problem among autonomous cars as a multi-agent cooperation problem and solved it via MADRL. Specifically, the authors have used a hierarchical DRL method that breaks down the problem into “high-level option selection” and “low-level control” of the agent. Real-world experiments were performed using a robotic test track with four robots, where two of them performed the cooperative lane change.

Finally, Sivanathan et al. [119] proposed a decentralized motion planning framework and a Unity-based simulator specifically for a multi-robot system that uses DRL. The simulator can handle both independent learners and common policies. The simulator was tested with up to four cooperative non-holonomic robots that shared limited information. PPO was used as the base algorithm to train the policies.

3.3. Swarm Behavior Modeling

Navigation of a swarm of robots through a complex environment is one of the most researched topics in swarm robotics. To have a stable formation, each robot should be aware of the positions of the nearby robots. A swarm consisting of miniature robots might not have a sophisticated set of sensors available. For example, a compass can be used to know the heading of the robot. Additionally, range and bearing sensors can also be available [213,214]. Infrared sensors can be used for communication in such a swarm system [215]. Inspired by swarms of birds or schools of fish, robots usually follow three simple rules to maintain such formations: cohesion, collision avoidance, and velocity alignment [164]. It is no surprise that multi-agent deep reinforcement learning techniques have been extensively employed to mimic such swarm behaviors and solve similar problems. An illustration of forming a circle with a swarm of five e-puck robots is presented in Figure 9.

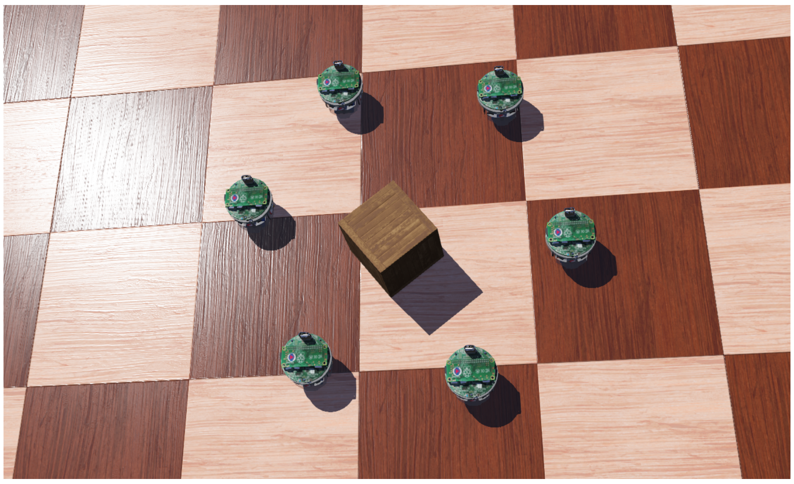

Figure 9. An illustration of pattern (circle) formation with five e-puck robots which are guarding a box in the center.

Zhu et al. [216] proposed a novel algorithm for multi-robot flocking. The algorithm builds on MADDPG and uses PER. Results from three robots show that the proposed algorithm improves over the standard MADDPG. Similarly, Salimi and Pasquier [106] have proposed the use of DDPG with centralized training and a decentralized execution mechanism to train the flocking policy for a system of UAVs. Such flocking with UAVs might be challenging due to complex kinematics. The authors show that the UAVs reach the flocking formation using a leader–follower technique without any parameter tuning. Lan et al. [217] developed a control scheme for the cooperative behavior of a swarm. The basis of their control scheme is pulled from joint multi-agent reinforcement learning theory, where the robots not only share state information, but also a performance index designed by the authors. Notably, the convergence of the policy and the value networks is theoretically guaranteed. Following the above-mentioned works, Kheawkhem and Khuankrue [99] also proposed using MADDPG to solve the multi-agent flocking control problem.

Qiu et al. [100] used MADRL to improve sample efficiency, reduce overfitting, and allow better performance, even when agents had little or “bad” sample data in a flocking application. The main idea was to train a swarm offline with demonstration data for pre-training. The presented method is based on MADDPG. Not only for coverage as described earlier, but GNNs are also popular in general for coordination in a swarm system, especially in spatial domains. For example, Kortvelesy and Prorok [218] developed a framework, called ModGNN, which aimed to provide a generalized, neural network framework, that can be applied to varying multi-robot applications. The architecture is modular in nature. They tested the framework for a UAV flocking application with 32 simulated robots.

Yan et al. [219] studied flocking in a swarm of fixed-wing UAVs operating in a continuous space. Similar studies on flocking can also be found in more recent papers from these authors [83,220]. Similar to Yan et al.’s body of work, Wang et al. [142] proposed a TD3-based [221] solution for a similar application—flocking with fixed-wing UAVs where the authors test the method with up to 30 simulated UAVs. Not strictly for swarms, Lyu et al. [47]

addressed the multi-agent flocking control problem specifically for a multi-vehicle system using DDPG with centralized training and decentralized execution. Notably, the authors take connectivity preservation into account while designing their reward function—the maximum distance could not go beyond the communication range and the minimum distance was kept at d_s, a physically safe distance between two vehicles. Interestingly, the mission waypoints are pre-defined in this paper. Bezcioglu et al. [48] also study flocking in a swarm system using DDPG and CNN, and tested it with up to 100 robots. The authors have used bio-inspired self-organizing dynamics for the joint motion of the robots.

Wang et al. [113] used MADRL to organize a swarm in specific patterns using auto-encoders [222] to learn compressed versions of the states and they tested the presented solution with up to 20 robots. Li et al. [46] proposed using a policy gradient method, namely MADDPG, with an actor–critic structure for circle formation control with a swarm of quad-rotors. Although circle formation is a popular application [223–226], this is one of the few studies that employed MADRL techniques. Sadhukhan and Selmic [121] have used PPO in order to train a multi-robot system to navigate through narrow spaces and reform into a designated formation. They used two reward schemes (one individual to the agents and one depending on the contributions to the team) and the system was centrally trained. In [125], Sadhukhan and Selmic extended their prior works by proposing a bearing-based reward function for training the swarm system, which utilizes a single policy shared among the robots.

Chen et al. [97] have developed an improved DDPG to enhance the ability of a robot to learn human intuition-style navigation without using a map. Furthermore, they create a parallel version of DDPG to extend their algorithm to a multi-robot application. Thereby, providing the robots with a method of sharing information/experiences in order to maintain formation, navigate an indoor environment, and avoid collisions. Qamar et al. [138] proposed novel reward functions and an island policy-based optimization framework for multiple target tracking using a swarm system. Along a similar line, Ma et al. [98] developed a DDPG-based algorithm for multi-robot formation control around a target, particularly in a circle around a designated object. The algorithm allows the robots to independently control their actions using local teammates' information.

Recently, Zhang et al. [124] have also proposed a target encirclement solution that uses a decentralized DRL technique. The main contribution of their work is the use of three relational graphs among the robots and other entities in the system designed using a graph attention network [227]. In their simulation experiments, the authors use six robots encircling two targets. Similarly, Khan et al. [134] have used a graph representation of the robot formation and proposed using graph convolutional neural networks [158,228,229] to extract features, i.e., local features of robot formations, for policy learning. Simulation policies were trained on three robots and then the policy is transferred to over 100 robots for testing. The robots are initialized to certain positions and are to form a specific formation while reaching an end goal.

Zhou et al. [230] recognized the problem of computational complexity with existing MADRL methods for multi-UAV multi-target tracking while proposing a decentralized solution. Their proposed solution has its root in the reciprocal altruism mechanism of cooperation theory [231]. The experience replay is shared among the UAVs in this work. Zhou et al. [139] also study target tracking with a swarm of UAVs. Not only do they learn to track a target, but the robots also learn to communicate better (i.e., the content of the message) for such tracking following the proposed policy gradient technique.

Yasuda and Ohkura [78] used a shared replay memory along with DQN to accelerate the training process for the swarm with regard to path planning. By using more robots contributing their individual experiences to the replay memory, the swarm system was able to learn the joint policy faster. Communication is an important aspect of swarm systems. Usually, researchers use pre-defined communication protocols for coordination among the swarm robots. Hüttenrauch et al. [140] proposed a histogram-based communication protocol for swarm coordination, where the robots use DRL to learn decentralized policies using TRPO [44]. An example task is graph building formation, where the robots aim to cover a certain area through coordination. Another considered task is establishing a communication link between the robots and connecting two points on a map. Along the same line, in 2019, Hüttenrauch et al. [141] used TRPO again to find MADRL-based solutions for rendezvous and pursuit–evasion in a swarm system. The main contribution of their work is the incorporation of Mean Embedding [232] into the DRL method they use to simplify the state information each agent obtains from other agents. Up to 100 robots were used in simulation experiments.

3.4. Pursuit-Evasion

In a pursuit–evasion game, usually, multiple pursuers try to capture potentially multiple evaders. When all the evaders are captured or a given maximum time elapses, the game finishes [233–235]. For a detailed taxonomy of such problems, the reader is referred to [233]. Some of the sensors that the robots might use in this application include sonar, LiDAR, and 3D cameras, among others. A unified model to analyze data from a suit of sensors can also be used [236]. An illustration is shown in Figure 10.

Egorov [59] proposed a solution for the classic pursuit–evasion problem [233] using an extension of single-agent DQN, called multi-agent DQN (MADQN). The state space is represented as a four-channel image consisting of a map, opponent location(s), ally location(s), and a self-channel. Yu et al. [40] proposed the use of a decentralized training method for pursuit evasion where each agent learns its policy individually and used limited communication with other agents during the training process. This is unlike traditional MADRL techniques where the training is centralized. The execution of the policy for each agent is also decentralized.

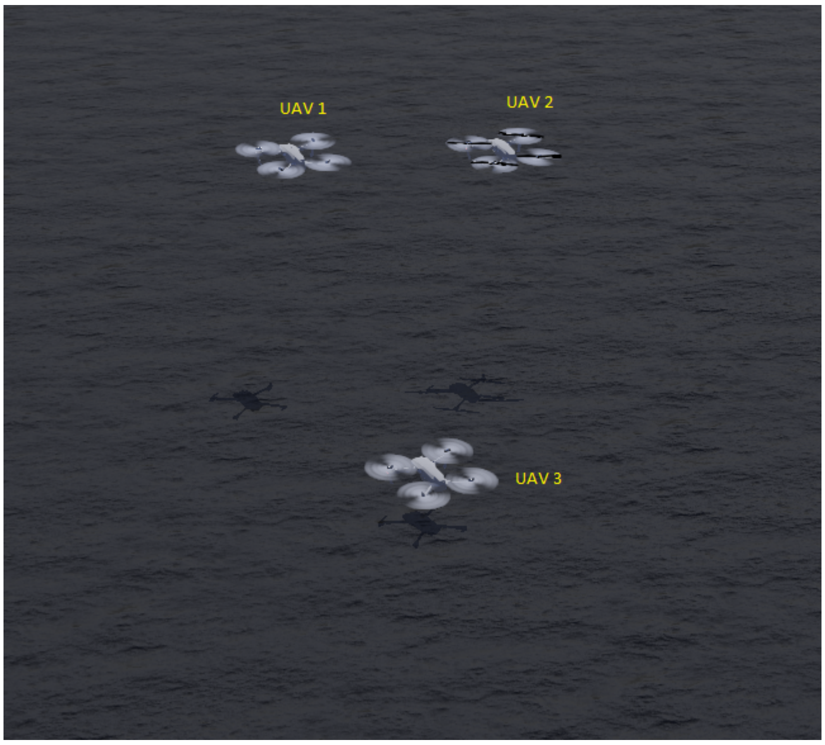

Figure 10. An illustration of multi-robot pursuit–evasion scenario where UAVs 1 and 2 have captured and “grounded” UAV 3, which is an evader robot.

Wang et al. [23] proposed to extend a MARL algorithm called cooperative double Q-learning (Co-DQL) for the multi-UAV pursuit–evasion problem. The foundation of Co-DQL is Q-networks with multi-layer perceptrons. Unlike traditional applications where the evader might move around randomly, in this paper, the authors assume that the target also learns to move intelligently up to a certain degree via RL. In [237], the authors consider a setup with one superior evader and multiple pursuers. They use a centralized critic model, where the actors are distributed. Unlike traditional broadcasting techniques, the authors smartly use a leader–follower line topology network for inter-robot communication that reduces the communication cost drastically. Although not strictly pursuit-evasion, Zhang et al. [76] use MADRL for coordinated territory defense, which is modeled as a game where two defender robots coordinate to block an intruder from entering a target zone.

Gupta et al. [81] argue that instead of using a centralized multi-agent DRL framework, where the model learns joint actions from joint states and observations, a more sophisticated parameter-sharing approach can be used. A drawback of the centralized learning system is that the complexity grows exponentially with the number of agents. The authors use TRPO as their base algorithm and the policy is trained with the experiences of all agents simultaneously via parameter sharing. The multi-agent scenarios they use for testing the quality of the proposed solution are pursuit–evasion and a multi-walker system with bipedal walkers.

3.5. Information Collection

The objective of information gathering about an ambient phenomenon (e.g., temperature monitoring or weed mapping) using a group of mobile robots is to explore parts of an unknown environment, such that uncertainty about the unseen locations is minimized. Relevant sensors for information gathering include RGB, Normalized Difference Vegetation Index (NDVI), or multi-spectral cameras, and thermal and humidity sensors, among others. This is unlike coverage, where the goal is to visit all the locations. There are two main reasons for this: (1) information (e.g., temperature measurements) in nearby points are highly correlated, and, therefore, the robots do not need to go to all the locations within a neighborhood [238]; and (2) the robot might not have enough battery power to cover the entire environment. This is especially true in precision agriculture, where the fields are usually too large to cover [18]. An illustration is shown in Figure 11, where the robots are tasked with collecting information from their unique sub-regions, and through communication, they will need to learn the underlying model.

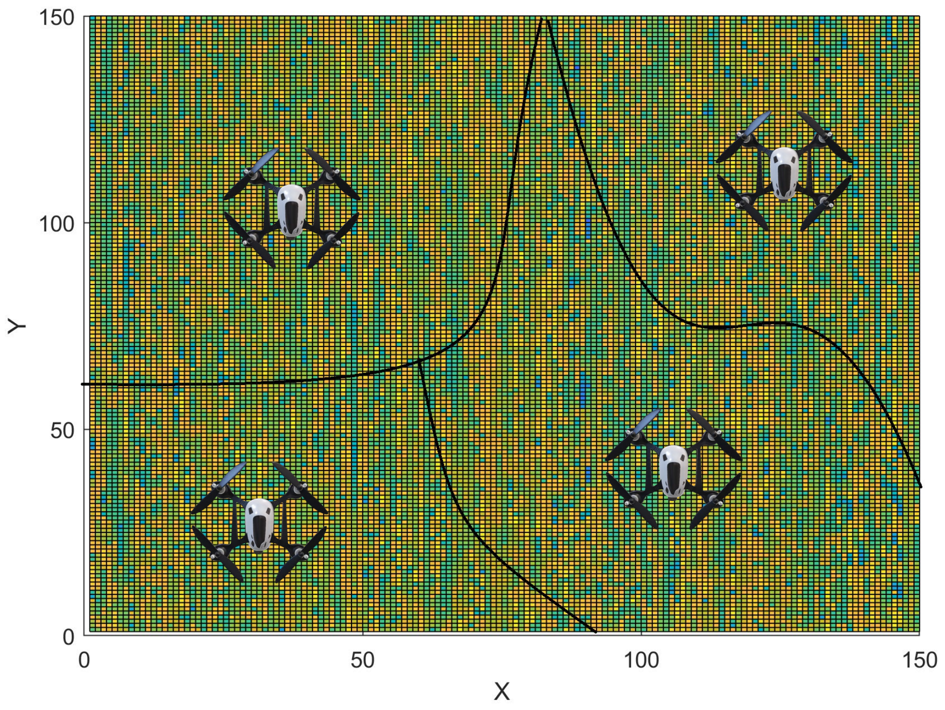

Figure 11. An illustration of multi-robot information collection. The environment is divided into four sub-regions and the underlying heatmap represents measures of soil acidity in parts of the USA [239].

Viseras and Garcia [240] have developed a novel DRL algorithm based on the popular A3C [45] algorithm. They also provide a model-based version of their original algorithm for gathering information, which uses CNNs. Said et al. [241] have proposed a mean field-based DRL technique that uses an LSTM module—a type of recurrent neural network for multi-robot information collection about an unknown ambient phenomenon. The robots are battery-powered with limited travel ranges. Recently, Wei and Zheng [67] also used MADRL for multi-robot informative path planning. They develop two strategies for cooperative learning: (1) independent Q-learning with credit assignment [4], and (2) sequential rollout using a GRU. Along the same line, Viseras et al. [85] have proposed using a MADRL framework for a multi-robot team to monitor a wildfire front. The two main components in this framework are (1) individually-trained Q-learning robots and (2) value decomposition networks. The authors have used up to 9 UAVs for testing the efficiency of their presented work.

3.6. Task Allocation

Multi-robot task allocation (MRTA) is a combinatorial optimization problem. Given a set of n robots and m tasks, the goal is to allocate the robots to the tasks such that a given utility function is optimized. Now, if multiple robots need to form a team to complete a single task, then it is a single-task, multi-robot allocation problem. On the other hand, if one robot can offer its services to multiple tasks, then it is called a single-robot, multi-task allocation problem. The robots might be connected to a central server via Wi-Fi, e.g., in a warehouse setting, and can receive information about tasks and other robots. Similarly, communication can happen with other robots via this central server using Wi-Fi as well. Overhead cameras or tracking systems can be used for robot localization in such a scenario. Comprehensive reviews about such MRTA concepts and solutions can be found in [242,243]. An example task allocation scenario is presented in Figure 12.

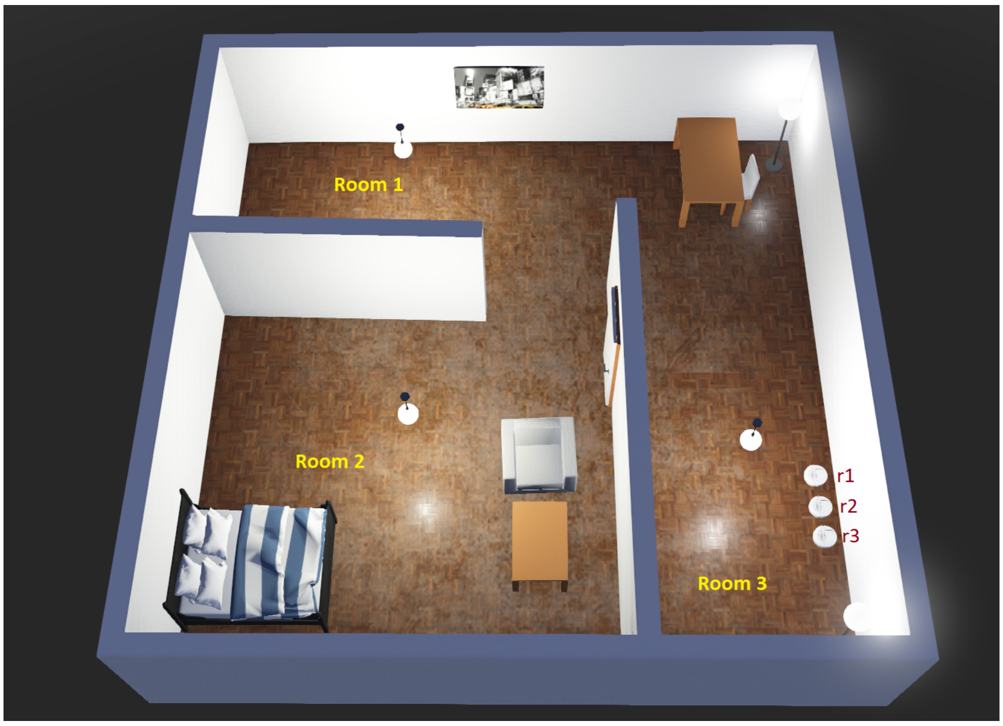

Figure 12. An illustration of multi-robot task allocation: there are 3 iRobot Roombas ($r1$–$r3$) and 3 rooms to clean. In a one-to-one matching scenario, the objective would be to assign one Roomba to a certain room. However, as room 2 is larger in size, two robots might be needed to clean it, whereas the third robot ($r3$) might be assigned to rooms 1 and 3.

Elfakharany and Ismail [132] developed a novel multi-robot task allocation and navigation method. This is the first work to propose a MADRL method to tackle task allocation, as well as the navigation problem. They use PPO with actor–critic. Their centralized training and decentralized execution method uses CNNs. Paul et al. [129] proposed to use DRL for multi-robot task allocation. They proposed a neural network architecture that they called a Capsule Attention-based Mechanism, which contains a Graph Capsule Convolutional Neural Network (GCapCN) [244] and a Multi-head Attention mechanism (MHA) [245,246]. The underlying architecture is a GNN. The task graph is encoded using GCapCN and combined with the context, which contains information on the robot, time, and neighboring robots. This information is then decoded with the MHA. Although not strictly task assignment, MADRL has been used for forming teams of heterogeneous agents (such as ambulance and fire brigade in a rescue operation) to complete a given task by Goyal [86]. Goyal has applied this technique for training a team of fire brigades to collaboratively extinguish a fire in a city within the Robocup Rescue Simulator.

Devin et al. [247] developed a novel method of compartmentalizing a trained deep reinforcement learning model into task-specific and robot-specific components. Due to this, the policies can be transferred between robots and/or tasks. Park et al. [114] propose a PPO-based DRL technique for task allocation. Their solution is tested with single-task, multi-robot, and time-extended assignments. They use an encoder–decoder architecture to represent robots and tasks, where a cross-attention layer is used to derive the relative importance of the tasks for the robots.

Scheduling tasks is another important aspect of task planning. Wang and Gombolay [82] used GNNs and imitation learning for a multi-robot system to learn a policy for task scheduling. The proposed model is based on graph attention networks [227]. The scheduling policy is first learned using a Q-network with two fully-connected layers. Imitation learning is then used to train the network from an expert dataset that contains schedules from other solutions. On the other hand, Johnson et al. [93] study the problem of dynamic flexible job shop scheduling, where an assembly line of robots must dynamically change tasks for a new job series over time. The robots learn to coordinate their actions in the assembly line. Agrawal et al. [52] performed a case study on a DRL approach to handling a homogeneous multi-robot system that can communicate while operating in an industry setting. PPO is used as the foundation algorithm. The objective of this work is to train the robots to work with each other to increase throughput and minimize the travel distances to the allocated tasks while taking the current states of the robots and the machines on the floor into account.

One of the most recent studies on deep RL-based MRTA is due to [89], which aims to use DRL for the parallelization of processing tasks for MRTA. The authors base their method on Branching Dueling Q-Network [248] with respect to multi-robot search and rescue tasks. In such a network, multiple branches of a network shares a common decision-making module where each branch handles one action dimension. This helps to reduce the curse of dimensionality in the action space. In total, 20 robots have been used within a simulation to test the feasibility of the proposed technique.

A very different and interesting task assignment application in defense systems is studied by Liu et al. [120]. The authors presented a DRL framework for multi-agent task allocation for weapon target assignment in air defense systems. They use PPO-clip along with a multi-head attention mechanism for task assignments of a (army) general and multiple narrow agents. The neural network architecture uses fully connected layers and a GRU. The major aim of this work is to increase processing efficiency and solution speed of the multi-agent task assignment problem at a large scale. Simulation experiments were carried out in a virtual digital battlefield. The experimental setup includes offensive forces and defensive forces. The defensive forces have places to protect and need to make real-time task allocation decisions for defense purposes. The defensive forces are tested with 12 and the offensive forces are tested with a total of 32 agents.

3.7. Object Transportation

To transport an object using two or more cooperative mobile robots, the goal is to design a strategy where the robots' actions are highly coordinated. Communication among the robots may or may not be possible. The robots can use depth cameras or laser scanners for avoiding obstacles. On the other hand, an optic-flow sensor can be used to determine if the pushing force from the robot has resulted in any object movement or not [249]. A force-torque sensor can be used on the robot to measure the force amount placed on the object. For a comprehensive review of this topic, please refer to [250]. An illustration is shown in Figure 13.

Zhang et al. [73] have used a modified version of DQN that controls each robot individually without a centralized controller or a decision maker. To quantitatively measure how well the robots are working together, they use the absolute error of estimated state–action values. The main idea is to use DQN to have homogeneous robots carry a rod to a target location. Each robot acts independently with neither leading nor following. Niwa et al. [251] proposed a MADRL-based solution to the cooperative transportation and obstacle removal problem. The basis of their solution is to use MARL to train individual robots' decentralized policies in a virtual environment. The policies are trained using MADDPG [61]. The authors then use the trained policies on real teams of robots to validate the effectiveness. The robots are supposed to push a target object to a final waypoint while moving a physical barrier out of the way to accomplish the task. Manko et al. [77] used CNN-based DRL architecture for multi-robot collaborative transportation where the objective is to carry an object from the start to the goal location. Eoh and Park [79] proposed a curriculum-based deep reinforcement learning method for training robots to cooperatively transport an object. In a curriculum-based RL, past experiences are organized and sorted to improve training efficiency [252]. In this paper, a region-based curriculum starts by training robots in a smaller area, before transitioning to a larger area and a single-robot to multi-robot curriculum begins by training a single robot to move an object, then transferring that learned policy to multiple robots for multi-robot transportation.

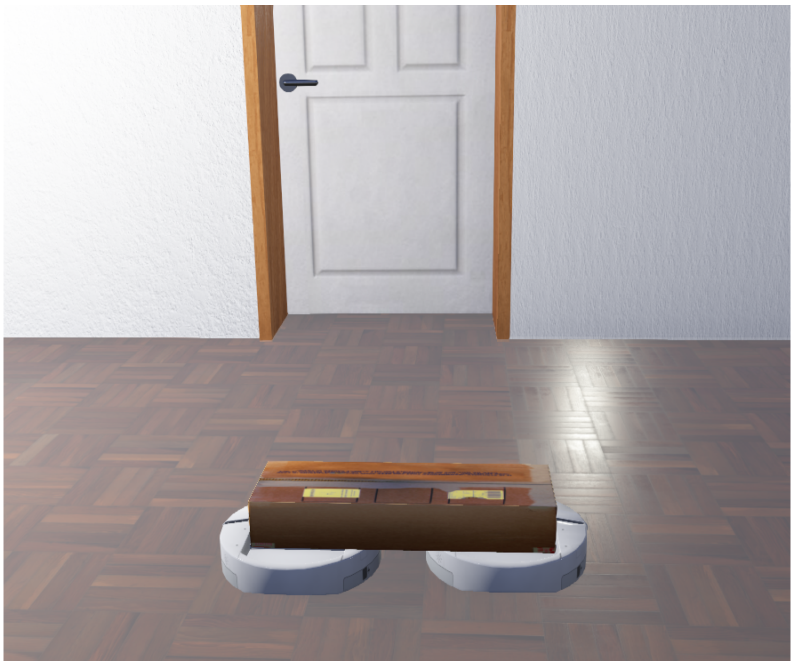

Figure 13. An illustration of multi-robot object transportation is presented where 2 iRobot Create robots carry a cardboard box and plan to go through a door in front of them.

3.8. Collective Construction

In a collective construction setup, multiple cooperative mobile robots are required. The robots might have heterogeneous properties [253]. The robots can follow simple rules and only rely on local information [254]. In the popular TERMES project from Harvard University [254], a large number of simple robots collect, carry, and place building blocks to develop a user-specified 3D structure. The robots might use onboard vision systems to access the progress in construction. A force sensor-equipped gripper can be used for holding the materials. Furthermore, a distance sensor, e.g., sonar can be used for maintaining a safe distance from the construction as well as other robots [255].

Sartoretti et al. [256] developed a framework using A3C to train robots to coordinate the construction of a user-defined structure. The proposed neural network architecture includes CNNs and an LSTM module. Each robot runs its own copy of the policy without communicating with other agents during testing.

A summary of the state and action spaces and reward functions used in some of the papers reviewed in this article are listed in Table 2.

Table 2. Examples of state and action spaces and reward functions used in prior studies.

Refs.	State	Action	Reward
[59]	Map of the environment and robots' locations with 4 channels	Discreet	Based on the locations of the robots
[67]	Robot locations and budgets	Discreet	Based on collected sensor data
[68]	Position of the leader UAVs, the coverage map, and the connection network	Discreet	Based on the overall coverage and connectivity of the Leader UAVs.
[69]	Map with obstacles and the coverage area	Discreet	Based on the robot reaching a coverage region within its task area.
[24]	Map with covered area	Discreet	Based on robot coverage.
[71]	Map of the environment	Discreet	Based on the robot movements and reaching the target without collisions.
[70]	Map with robots' positions	Discreet	Based on the herding pattern.
[72]	Map with robots' positions and target locations	Discreet	Distance from the goal and collision status.
[39]	Map with robots' positions and target locations	Discreet	Distance from the goal and collision status.
[40]	Pursuer and evader positions	Discreet	Collision status and time to capture the predator.
[73]	The map, locations, and orientations of the robots, and the objects the robots are connected to	Discreet	Based on the position of the object and the robots hitting the boundaries.
[74]	The map, and the locations of the robots	Discreet	Based on distance from the target and collisions.
[76]	The regions of the robots, positions of the defender and the attacker UAVs and the intruder	Discreet	Based on distance.
[77]	The distance from the MRS center to the goal, the difference in orientation of the direction of MRS to the goal, and the distance between the robots	Discreet	Based on the distance to the goal, orientation to the goal, proximity of obstacles, and the distance between the robots.
[78]	Sensor input information that includes distance to other robots and the target landmarks	Discreet	Based on becoming closer to the target landmark.
[79]	Spatial information on the robots, the object, and the goal	Discreet	Based on the object reaching the goal while avoiding collisions.
[80]	Robot position and velocity	Discreet	Based on the robots being within sensing range of one another.
[38]	The positions and speed of the first responders and UAVs	Discreet	Based on the Cramér–Rao lower bound (CRLB) for the whole system.
[93]	The agents' positions, types, and remaining jobs	Discreet	Based on minimizing the makespan.
[83]	The position and direction of the leader and the followers	Discreet	Based on the distance from followers to leaders and collision status.
[84]	Information on the target, other agents, maps, and collisions	Discreet	Based on finding targets and avoiding obstacles.
[85]	The robot's position, position relative to other robots, and angle and direction of the robots	Discreet	Based on covering a location on fire.

Table 2. *Cont.*

Ref.	State	Action	Reward
[23]	The positions, velocities, and distances between UAVs	Discreet	Determined by the distance from the target and the evader being reached by the pursuer.
[87]	Consists of static obstacle locations and the locations of other agents	Discreet	Based on the robots' movements toward the goal while avoiding collisions.
[94]	Contains the sensor data for the location of the target relative to the robot and the last action done by the robot	Discreet	Determined by the robot reaching the goal, reducing the number of direction changes, and avoiding collisions.
[95]	A map that includes the agent's locations, empty cells, obstacle cells, and the location of the tasks	Discreet	Determined by laying pieces of flooring in the installation area.
[110]	Map of the environment represented with waypoints, locations of the UAVs, and points of interest	Discreet	Based on the coverage of the team of robots.
[114]	The positions and tasks of the robots, the state of the robot	Discreet	Based on minimizing the number of timesteps in an episode.
[96]	Map of the area to be sterilized and the positions of the agents, the cleaning priority, size, and area of the cleaning zone	Discreet	Based on the agents cleaning priority areas for sanitation.
[86]	Temperature and "fieryness" of a building, location of the robots, water in the tanks, and busy or idle status	Discreet	Based on keeping the fires to a minimum "fieryness" level.
[53]	Sensory information on obstacles	Discreet	Based on the UAV's coverage of the area.
[52]	Robots' positions and velocities and the machine status	Discreet	Determined by robots completing machine jobs to meet the throughput goal, and their motions while avoiding collisions.
[107]	Includes the laser readings of the robots, the goal position, and the robot's velocity	Continuous	Based on the smooth movements of the robots while avoiding collisions.
[22]	Includes the laser readings of the robots, the goal position, and the robot's velocity	Continuous	Based on the time to reach the target while avoiding collisions.
[108]	An environment that includes the coordinates of the manipulator arm gripper	Continuous	Based on reaching the target object.
[109]	Laser measurements of the robots and their velocities	Continuous	Based on the centroid of the robot team reaching the goal.
[117]	The state of the robot, other robots, obstacles, and the target position	Continuous	Based on the robots' relative distance from the target location.
[97]	Sensed LiDAR data	Continuous	Based on the robot approaching and arriving at the target, avoiding collisions and the formation of the robots.
[132]	Consists of the goal positions, the robots' positions, past observations	Continuous	Based on the robot moving towards the goal in the shortest amount of time.
[133]	Contains laser data, speeds and positions of the robots, and the target position	Continuous	Based on arriving at the target, avoiding collisions, and relative position to other robots.
[113]	The position information of other robots (three consecutive frames)	Continuous	Based on time for formation, collisions, and the formation progress.
[115]	The most recent three frames of the map, local goals that include positions and directions	Continuous	Based on minimizing the arrival time of each robot while avoiding collisions.
[116]	Map of the environment, robot positions and velocities, and laser scans	Continuous	Based on the arrival time of the robot to the destination, avoiding collisions, and smoothness of travel.
[104]	The coverage score and coverage state for each point of interest and the energy consumption of each UAV	Continuous	Defined by coverage score, connectivity fairness, and energy consumption.
[134]	Robot's relative position to the goal and its velocity	Continuous	Based on the robots having collisions.
[88]	Robot motion parameters, relative distance and orientation to the goal, and their laser scanner data	Discreet/ Continuous	Determined by reaching the goal without timing out and avoiding collisions.

4. Challenges and Discussion

Although we find that a plethora of studies have used multi-agent deep reinforcement learning techniques in recent years, a number of challenges remain before we can expect wide adaptation of them in academia as well as commercially. One of the biggest challenges that we identify is scalability. Most of the papers reviewed in this article do not scale beyond tens of robots. This limits real-world adaptation. Although this is an issue with multi-robot systems in general, the data-hungry nature of most of today's DRL techniques makes the situation worse. In the future, the research community needs to come up with lightweight techniques that potentially are inspired by nature, such as swarming in biology or particle physics while making necessary changes to the underlying RL technique to fit these appropriately.

The second drawback we found in most of the studies is the lack of resources to make them reproducible. One of the overarching goals of academic research is that researchers across the world should be able to reproduce the results reported in one paper and propose a novel technique that potentially advances the field. In the current setup, most papers employing MADRL use their own (simulation) environments for their robots, which makes it extremely difficult for others to reproduce the results. As a community, we need to come up with an accepted set of benchmarks and/or simulators that the majority of the researchers can use for method design and experiments, which, in turn, will advance the field.

The next challenge is to transfer the learned models to real robots and real-world applications. We find that most experiments in the literature are conducted virtually, i.e., in simulation, rather than with physical robots. This leads to a gap in understanding the feasibility. This corroborates the finding by Liang et al. [257]. Unless we can readily use the learned models on real robots in real-world situations, we might not be able to widely adopt such techniques. It is tied up with the previously-mentioned issue of scalability. Additionally, in the deployment phase, the algorithms need to be lightweight while considering the bandwidth limitation for communication among the robots.

Software plays a significant role in developing and testing novel techniques in any robotic domain and applications of MADRL are no different. Here, we discuss some software that are popularly used for testing the feasibility of the proposed techniques in simulation.

- VMAS: Vectorized Multi-Agent Simulator for Collective Robot Learning (VMAS) is an open-source software for multi-robot application benchmarking [258]. Some applications that are part of the software include swarm behaviors, such as flocking and dispersion, as well as object transportation and multi-robot football. Note that it is a 2D physics simulator powered by PyTorch [259].
- MultiRoboLearn: Similar to VMAS, this is an open-source framework for multi-robot deep reinforcement learning applications [260]. The authors aim to unify the simulation and the real-world experiments with multiple robots via this presented software tool, which is accomplished by integrating ROS into the simulator. Mostly multi-robot navigation scenarios were tested. It would be interesting to extend this software to other multi-robot applications, especially where the robots might be static.
- MARLlib: Although not strictly built for robots, Multi-Agent RLlib (MARLlib) [261] is a multi-agent DRL software framework that is built upon Ray [262] and its toolkit RLlib [263]. This is a rich open-source software that follows Open AI Gym standards and provides frameworks for not only cooperative tasks, but for competitive multi-robot applications as well. Currently, ten environments are supported by MARLlib among which the grid world environment might be the most relevant one to the multi-robot researchers. Many baseline algorithms including the ones that are highly popular among roboticists, e.g., DDPG, PPO, and TRPO are available as baselines. The authors also show that this software is much more versatile than some of the existing ones including [264,265].

Not only these specialized ones, but other traditional robot simulators, such as Webots [266], V-rep [267], and Gazebo [157], can also be used for training and testing multiple robots. These established software platforms provide close-to-reality simulation models for many popular robot platforms. This is especially useful for robotics researchers as we have seen in this survey that MADRL applications range from aerial and ground robots to underwater robots and manipulators. Table 3 summarizes the main types of robots that have been used for MADRL applications. Not only software development, but another challenge is training data. As most of the state-of-the-art algorithms rely on massive amounts of training data, it is not always easy to train a robot with sufficient data. Dasari et al. [268] have created an open-source database for sharing robotic experiences. It contains 15 million video frames from 7 different robot manipulators including Baxter, Sawyer, Kuka, and Fetch arms. Researchers can use this dataset for efficient training while adding new experiences from their experiments to the dataset itself.

Table 3. Types of robots in the reviewed papers. If the type is not specified in the paper, it is not listed here.

Aerial Robots	Ground Robots	Manipulators
[23,24,38,46,53,56,57,68,76,83,85,91,104,106,110,110,120,123,134,135,138,139,141,142,161,167,171,172,186,205,207,218–220,230,240,269]	[22,39,49,52,54,55,59,70,71,73–75,77–79,82,87,90,92,97,107,109,111,112,115,117,120,124,128,130,132,133,137,155,162,183,189,190,194,202,212,251]	[50,93,108,131,143,144]

As we have seen, sensors play a major role in creating the perception about the environment, as well as aiding the robots with communication capabilities. The robots might need to collect multi-modal sensor data and fuse them for better perceptions. These sensory observations about the environment can then be used as state inputs to the deep neural networks. Modern sensors have high sampling rates, e.g., standard LiDAR samples over 1 million data points per second. Without state-of-art learning mechanisms, it would have been almost infeasible to process and extract meaningful information from such large amounts of data (for tasks such as target recognition, classification, and semantic feature analysis, among others).

Although many robotic applications are utilizing the progress in multi-agent reinforcement learning, we have not seen any paper on modular self-reconfigurable robotics (MSRs) [270–273] where MADRL has been utilized. We believe that the field of modular robots can benefit from these developments especially given the fact that MSRs can change their shapes and the new shape might not have been pre-defined. Therefore, its control is undefined as well and it might need to learn to move around and complete tasks on-the-fly using techniques, such as MADRL, where each module acts as an intelligent agent.

On the other hand, we have found MADRL-based solutions for manipulation and motion separately. The next question that should be answered is how one can simultaneously learn those two actions where they might affect each other. For example, in a scenario, where multiple UAVs are learning to maintain a formation while manipulating an object with their onboard manipulators. This task would potentially require the robots to learn two actions simultaneously. The research question then would be how to best model the agents, their goals, and the rewards in this complex scenario.

5. Conclusions

In this paper, we have reviewed state-of-the-art studies that use multi-agent deep reinforcement learning techniques for multi-robot system applications. The types of such applications range from exploration and path planning to manipulation and object transportation. The types of robots that have been used encompass ground, aerial, and underwater applications. Although most applications involve mobile robots, we reviewed a few papers that use non-mobile (manipulator) robots as well. Most of the reviewed papers have used convolutional neural networks, potentially combining them with fully connected layers, recurrent layers, and/or graph neural networks. It is worth investigating such reinforcement learning techniques for robotics as they have the potential to learn high-level causal relationships among the robots, as well as between the robots and their environment, which might have been extremely difficult to model using a non-learning approach. As better hardware is available on a smaller scale and at a lower price, we expect to see significant growth in novel multi-robot system applications that use multi-agent reinforcement learning techniques. Furthermore, with the progress of the field of artificial intelligence in general, we expect that more studies will have theoretical underpinnings along with their showcased empirical advancements. Although a number of challenges remain to be solved, we are perhaps not too far away from seeing autonomous robots tightly integrated into our daily lives.

Author Contributions: Conceptualization, J.O. and A.D.; methodology, J.O. and A.D.; investigation, J.O. and A.D.; resources, J.O. and A.D.; data curation, J.O. and A.D.; writing—original draft preparation, J.O. and A.D.; writing—review and editing, J.O. and A.D.; visualization, A.D.; supervision, A.D.; project administration, A.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

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

References

1. Arai, T.; Pagello, E.; Parker, L.E. Advances in multi-robot systems. *IEEE Trans. Robot. Autom.* **2002**, *18*, 655–661. [CrossRef]
2. Gautam, A.; Mohan, S. A review of research in multi-robot systems. In Proceedings of the 7th IEEE International Conference on Industrial and Information Systems (ICIIS), Chennai, India, 6–9 August 2012; pp. 1–5.
3. Rizk, Y.; Awad, M.; Tunstel, E.W. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–31. [CrossRef]
4. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An introduction*; MIT Press: Cambridge, MA, USA, 2018.
5. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
6. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
7. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef]
8. Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; Ruiz, F.J.R.; Schrittwieser, J.; Swirszcz, G.; et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **2022**, *610*, 47–53. [CrossRef]
9. Popova, M.; Isayev, O.; Tropsha, A. Deep reinforcement learning for de novo drug design. *Sci. Adv.* **2018**, *4*, eaap7885. [CrossRef]
10. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
11. Mammeri, Z. Reinforcement learning based routing in networks: Review and classification of approaches. *IEEE Access* **2019**, *7*, 55916–55950. [CrossRef]
12. Panov, A.I.; Yakovlev, K.S.; Suvorov, R. Grid path planning with deep reinforcement learning: Preliminary results. *Procedia Comput. Sci.* **2018**, *123*, 347–353. [CrossRef]
13. Theile, M.; Bayerlein, H.; Nai, R.; Gesbert, D.; Caccamo, M. UAV coverage path planning under varying power constraints using deep reinforcement learning. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 1444–1449.
14. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1861–1870.
15. Nguyen, H.; La, H. Review of deep reinforcement learning for robot manipulation. In Proceedings of the 3rd IEEE International Conference on Robotic Computing (IRC), Naples, Italy, 25–27 February 2019; pp. 590–595.
16. Busoniu, L.; Babuska, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.)* **2008**, *38*, 156–172. [CrossRef]
17. Yang, E.; Gu, D. *Multiagent Reinforcement Learning for Multi-Robot Systems: A Survey*; Technical Report of the Department of Computer Science; 2004.
18. Dutta, A.; Roy, S.; Kreidl, O.P.; Bölöni, L. Multi-Robot Information Gathering for Precision Agriculture: Current State, Scope, and Challenges. *IEEE Access* **2021**, *9*, 161416–161430. [CrossRef]
19. Zhou, Z.; Liu, J.; Yu, J. A survey of underwater multi-robot systems. *IEEE/CAA J. Autom. Sin.* **2021**, *9*, 1–18. [CrossRef]

20. Queralta, J.P.; Taipalmaa, J.; Pullinen, B.C.; Sarker, V.K.; Gia, T.N.; Tenhunen, H.; Gabbouj, M.; Raitoharju, J.; Westerlund, T. Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision. *IEEE Access* **2020**, *8*, 191617–191643. [CrossRef]
21. Yliniemi, L.; Agogino, A.K.; Tumer, K. Multirobot coordination for space exploration. *AI Mag.* **2014**, *35*, 61–74. [CrossRef]
22. Long, P.; Fan, T.; Liao, X.; Liu, W.; Zhang, H.; Pan, J. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, QLD, Australia, 21–25 May 2018; pp. 6252–6259.
23. Wang, X.; Xuan, S.; Ke, L. Cooperatively pursuing a target unmanned aerial vehicle by multiple unmanned aerial vehicles based on multiagent reinforcement learning. *Adv. Control Appl. Eng. Ind. Syst.* **2020**, *2*, e27. [CrossRef]
24. Pham, H.X.; La, H.M.; Feil-Seifer, D.; Nefian, A. Cooperative and distributed reinforcement learning of drones for field coverage. *arXiv* **2018**, arXiv:1803.07250.
25. Sutton, R.S.; Precup, D.; Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artif. Intell.* **1999**, *112*, 181–211. [CrossRef]
26. Kober, J.; Bagnell, J.A.; Peters, J. Reinforcement learning in robotics: A survey. *Int. J. Robot. Res.* **2013**, *32*, 1238–1274. [CrossRef]
27. Bloembergen, D.; Tuyls, K.; Hennes, D.; Kaisers, M. Evolutionary dynamics of multi-agent learning: A survey. *J. Artif. Intell. Res.* **2015**, *53*, 659–697. [CrossRef]
28. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 157–163.
29. Bowling, M.; Veloso, M. Multiagent learning using a variable learning rate. *Artif. Intell.* **2002**, *136*, 215–250. [CrossRef]
30. Kaisers, M.; Tuyls, K. Frequency adjusted multi-agent Q-learning. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, ON, Canada, 10–14 May 2010; Volume 1, pp. 309–316.
31. Dutta, A.; Dasgupta, P.; Nelson, C. Adaptive locomotion learning in modular self-reconfigurable robots: A game theoretic approach. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 3556–3561.
32. Matignon, L.; Laurent, G.J.; Le Fort-Piat, N. Hysteretic q-learning: An algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, USA, 29 October–2 November 2007; pp. 64–69.
33. Dutta, A.; Dasgupta, P.; Nelson, C. Distributed adaptive locomotion learning in modred modular self-reconfigurable robot. In *Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 345–357.
34. Sadhu, A.K.; Konar, A. Improving the speed of convergence of multi-agent Q-learning for cooperative task-planning by a robot-team. *Robot. Auton. Syst.* **2017**, *92*, 66–80. [CrossRef]
35. Hu, J.; Wellman, M.P. Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.* **2003**, *4*, 1039–1069.
36. Buşoniu, L.; Babuška, R.; Schutter, B.D. Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications—1*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–221.
37. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
38. Moon, J.; Papaioannou, S.; Laoudias, C.; Kolios, P.; Kim, S. Deep reinforcement learning multi-UAV trajectory control for target tracking. *IEEE Internet Things J.* **2021**, *8*, 15441–15455. [CrossRef]
39. Wang, D.; Deng, H. Multirobot coordination with deep reinforcement learning in complex environments. *Expert Syst. Appl.* **2021**, *180*, 115128. [CrossRef]
40. Yu, C.; Dong, Y.; Li, Y.; Chen, Y. Distributed multi-agent deep reinforcement learning for cooperative multi-robot pursuit. *J. Eng.* **2020**, *2020*, 499–504. [CrossRef]
41. Zellner, A.; Dutta, A.; Kulbaka, I.; Sharma, G. Deep Recurrent Q-learning for Energy-constrained Coverage with a Mobile Robot. *arXiv* **2022**, arXiv:2210.00327.
42. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
43. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
44. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 1889–1897.
45. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
46. Li, B.; Li, S.; Wang, C.; Fan, R.; Shao, J.; Xie, G. Distributed Circle Formation Control for Quadrotors Based on Multi-agent Deep Reinforcement Learning. In Proceedings of the 2021 IEEE China Automation Congress (CAC), Beijing, China, 22–24 October 2021; pp. 4750–4755.
47. Xu, Z.; Lyu, Y.; Pan, Q.; Hu, J.; Zhao, C.; Liu, S. Multi-vehicle flocking control with deep deterministic policy gradient method. In Proceedings of the 14th IEEE International Conference on Control and Automation (ICCA), Anchorage, AK, USA, 2–15 June 2018; pp. 306–311.

48. Bezcioglu, M.B.; Lennox, B.; Arvin, F. Self-Organised Swarm Flocking with Deep Reinforcement Learning. In Proceedings of the 7th IEEE International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 4–6 February 2021; pp. 226–230.
49. Na, S.; Niu, H.; Lennox, B.; Arvin, F. Bio-Inspired Collision Avoidance in Swarm Systems via Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2022**, *71*, 2511–2526. [CrossRef]
50. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396.
51. Li, Y. Deep reinforcement learning: An overview. *arXiv* **2017**, arXiv:1701.07274.
52. Agrawal, A.; Won, S.J.; Sharma, T.; Deshpande, M.; McComb, C. A multi-agent reinforcement learning framework for intelligent manufacturing with autonomous mobile robots. *Proc. Des. Soc.* **2021**, *1*, 161–170. [CrossRef]
53. Bromo, C. Reinforcement Learning Based Strategic Exploration Algorithm for UAVs Fleets. Ph.D. Thesis, Politecnico di Torino, Turin, Italy, 2022.
54. Han, R.; Chen, S.; Wang, S.; Zhang, Z.; Gao, R.; Hao, Q.; Pan, J. Reinforcement Learned Distributed Multi-Robot Navigation With Reciprocal Velocity Obstacle Shaped Rewards. *IEEE Robot. Autom. Lett.* **2022**, *7*, 5896–5903. [CrossRef]
55. Na, S.; Niu, H.; Lennox, B.; Arvin, F. Universal artificial pheromone framework with deep reinforcement learning for robotic systems. In Proceedings of the 6th IEEE International Conference on Control and Robotics Engineering (ICCRE), Beijing, China, 16–18 April 2021; pp. 28–32.
56. Thumiger, N.; Deghat, M. A Multi-Agent Deep Reinforcement Learning Approach for Practical Decentralized UAV Collision Avoidance. *IEEE Control Syst. Lett.* **2021**, *6*, 2174–2179. [CrossRef]
57. Wang, G.; Liu, Z.; Xiao, K.; Xu, Y.; Yang, L.; Wang, X. Collision Detection and Avoidance for Multi-UAV based on Deep Reinforcement Learning. In Proceedings of the 40th IEEE Chinese Control Conference (CCC), Shanghai, China, 26–28 July 2021; pp. 7783–7789.
58. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [CrossRef] [PubMed]
59. Egorov, M. Multi-agent deep reinforcement learning. In *CS231n: Convolutional Neural Networks for Visual Recognition*; 2016; pp. 1–8.
60. Yang, Y.; Luo, R.; Li, M.; Zhou, M.; Zhang, W.; Wang, J. Mean field multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Playa Blanca, Spain, 9–11 April 2018; pp. 5571–5580.
61. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6382–6393.
62. Yu, C.; Velu, A.; Vinitsky, E.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative, multi-agent games. *arXiv* **2021**, arXiv:2103.01955.
63. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296.
64. Du, W.; Ding, S. A survey on multi-agent deep reinforcement learning: From the perspective of challenges and applications. *Artif. Intell. Rev.* **2021**, *54*, 3215–3238. [CrossRef]
65. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef]
66. OroojlooyJadid, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *arXiv* **2019**, arXiv:1908.03963.
67. Wei, Y.; Zheng, R. Multi-Robot Path Planning for Mobile Sensing through Deep Reinforcement Learning. In Proceedings of the INFOCOM 2021-IEEE Conference on Computer Communications, Vancouver, BC, Canada, 10–13 May 2021; pp. 1–10.
68. Mou, Z.; Zhang, Y.; Gao, F.; Wang, H.; Zhang, T.; Han, Z. Deep reinforcement learning based three-dimensional area coverage with UAV swarm. *IEEE J. Sel. Areas Commun.* **2021**, *39*, 3160–3176. [CrossRef]
69. Li, W.; Zhao, T.; Dian, S. Multirobot Coverage Path Planning Based on Deep Q-Network in Unknown Environment. *J. Robot.* **2022**, *2022*, 6825902. [CrossRef]
70. Kakish, Z.; Elamvazhuthi, K.; Berman, S. Using reinforcement learning to herd a robotic swarm to a target distribution. In *Proceedings of the International Symposium Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 401–414.
71. Yang, Y.; Juntao, L.; Lingling, P. Multi-robot path planning based on a deep reinforcement learning DQN algorithm. *CAAI Trans. Intell. Technol.* **2020**, *5*, 177–183. [CrossRef]
72. Bae, H.; Kim, G.; Kim, J.; Qian, D.; Lee, S. Multi-robot path planning method using reinforcement learning. *Appl. Sci.* **2019**, *9*, 3057. [CrossRef]
73. Zhang, L.; Sun, Y.; Barth, A.; Ma, O. Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning. *IEEE Access* **2020**, *8*, 184109–184119. [CrossRef]
74. Marchesini, E.; Farinelli, A. Enhancing deep reinforcement learning approaches for multi-robot navigation via single-robot evolutionary policy search. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 5525–5531.

75. Marchesini, E.; Farinelli, A. 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 (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4583–4588.
76. Zhang, H.; Li, D.; He, Y. Multi-robot cooperation strategy in game environment using deep reinforcement learning. In Proceedings of the 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), Kuala Lumpur, Malaysia, 12–15 December 2018; pp. 886–891.
77. Manko, S.V.; Diane, S.A.; Krivoshatskiy, A.E.; Margolin, I.D.; Slepynina, E.A. Adaptive control of a multi-robot system for transportation of large-sized objects based on reinforcement learning. In Proceedings of the 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), Moscow and St. Petersburg, Russia, 29 January–1 February 2018; pp. 923–927.
78. Yasuda, T.; Ohkura, K. Collective behavior acquisition of real robotic swarms using deep reinforcement learning. In Proceedings of the 2nd IEEE International Conference on Robotic Computing (IRC), Laguna Hills, CA, USA, 31 January–2 February 2018; pp. 179–180.
79. Eoh, G.; Park, T.H. Cooperative object transportation using curriculum-based deep reinforcement learning. *Sensors* **2021**, *21*, 4780. [CrossRef]
80. Huang, W.; Wang, Y.; Yi, X. Deep q-learning to preserve connectivity in multi-robot systems. In Proceedings of the 9th International Conference on Signal Processing Systems, ICSPS 2017, Auckland, New Zealand, 27–30 November 2017; pp. 45–50.
81. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 66–83.
82. Wang, Z.; Gombolay, M. Learning scheduling policies for multi-robot coordination with graph attention networks. *IEEE Robot. Autom. Lett.* **2020**, *5*, 4509–4516. [CrossRef]
83. Yan, C.; Wang, C.; Xiang, X.; Lan, Z.; Jiang, Y. Deep reinforcement learning of collision-free flocking policies for multiple fixed-wing uavs using local situation maps. *IEEE Trans. Ind. Inform.* **2021**, *18*, 1260–1270. [CrossRef]
84. Liu, Y.; Peng, Y.; Wang, M.; Xie, J.; Zhou, R. Multi-usv system cooperative underwater target search based on reinforcement learning and probability map. *Math. Probl. Eng.* **2020**, *2020*, 7842768. [CrossRef]
85. Viseras, A.; Meissner, M.; Marchal, J. Wildfire front monitoring with multiple uavs using deep q-learning. *IEEE Access* **2021**. [CrossRef]
86. Goyal, A. Multi-Agent Deep Reinforcement Learning for Robocup Rescue Simulator. Ph.D. Thesis, The University of Texas, Austin, TX, USA, 2020.
87. Chen, L.; Wang, Y.; Mo, Y.; Miao, Z.; Wang, H.; Feng, M.; Wang, S. Multi-Agent Path Finding Using Deep Reinforcement Learning Coupled With Hot Supervision Contrastive Loss. *IEEE Trans. Ind. Electron.* **2022**, *70*, 7032–7040. [CrossRef]
88. Jestel, C.; Surmann, H.; Stenzel, J.; Urbann, O.; Brehler, M. Obtaining Robust Control and Navigation Policies for Multi-robot Navigation via Deep Reinforcement Learning. In Proceedings of the 7th IEEE International Conference on Automation, Robotics and Applications (ICARA), Prague, Czech Republic, 4–6 February 2021; pp. 48–54.
89. Gautier, P.; Laurent, J.; Diguet, J.P. Deep Q-Learning-Based Dynamic Management of a Robotic Cluster. *IEEE Trans. Autom. Sci. Eng.* **2022**, 1–13. [CrossRef]
90. Song, C.; He, Z.; Dong, L. A Local-and-Global Attention Reinforcement Learning Algorithm for Multiagent Cooperative Navigation. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, 1–11. [CrossRef] [PubMed]
91. Ding, S.; Aoyama, H.; Lin, D. Combining Multiagent Reinforcement Learning and Search Method for Drone Delivery on a Non-grid Graph. In Proceedings of the International Conference on Practical Applications of Agents and Multi-Agent Systems; Springer: Berlin/Heidelberg, Germany, 2022; pp. 112–126.
92. Choi, H.B.; Kim, J.B.; Ji, C.H.; Ihsan, U.; Han, Y.H.; Oh, S.W.; Kim, K.H.; Pyo, C.S. MARL-based Optimal Route Control in Multi-AGV Warehouses. In Proceedings of the 2022 IEEE International Conference on Artificial Intelligence in Information and Communication (ICAIIC), Jeju Island, Republic of Korea, 21–24 February 2022; pp. 333–338.
93. Johnson, D.; Chen, G.; Lu, Y. Multi-Agent Reinforcement Learning for Real-Time Dynamic Production Scheduling in a Robot Assembly Cell. *IEEE Robot. Autom. Lett.* **2022**, *7*, 7684–7691. [CrossRef]
94. Chen, L.; Zhao, Y.; Zhao, H.; Zheng, B. Non-communication decentralized multi-robot collision avoidance in grid map workspace with double deep Q-network. *Sensors* **2021**, *21*, 841. [CrossRef]
95. Miyashita, Y.; Sugawara, T. Analysis of coordinated behavior structures with multi-agent deep reinforcement learning. *Appl. Intell.* **2021**, *51*, 1069–1085. [CrossRef]
96. Caccavale, R.; Calà, V.; Ermini, M.; Finzi, A.; Lippiello, V.; Tavano, F. Multi-robot Sanitization of Railway Stations Based on Deep Q-Learning. In Proceedings of the 8th Italian Workshop on AI and Robotics (AIRO), Online, 30 November 2021.
97. Chen, W.; Zhou, S.; Pan, Z.; Zheng, H.; Liu, Y. Mapless collaborative navigation for a multi-robot system based on the deep reinforcement learning. *Appl. Sci.* **2019**, *9*, 4198. [CrossRef]
98. Ma, J.; Lu, H.; Xiao, J.; Zeng, Z.; Zheng, Z. Multi-robot target encirclement control with collision avoidance via deep reinforcement learning. *J. Intell. Robot. Syst.* **2020**, *99*, 371–386. [CrossRef]
99. Kheawkhem, P.; Khuankrue, I. Study on Deep Reinforcement Learning for Mobile Robots Flocking Control in Certainty Situations. In Proceedings of the 19th IEEE International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), Prachuap Khiri Khan, Thailand, 24–27 May 2022; pp. 1–4.

100. Qiu, Y.; Zhan, Y.; Jin, Y.; Wang, J.; Zhang, X. Sample-Efficient Multi-Agent Reinforcement Learning with Demonstrations for Flocking Control. *arXiv* **2022**, *arXiv:2209.08351*.
101. Setyawan, G.E.; Hartono, P.; Sawada, H. Cooperative Multi-Robot Hierarchical Reinforcement Learning. *Int. J. Adv. Comput. Sci. Appl.* **2022**, *13*, 35–44. [CrossRef]
102. Meng, S.; Kan, Z. Deep reinforcement learning-based effective coverage control with connectivity constraints. *IEEE Control Syst. Lett.* **2021**, *6*, 283–288. [CrossRef]
103. Hamed, O.; Hamlich, M. Hybrid Formation Control for Multi-Robot Hunters Based on Multi-Agent Deep Deterministic Policy Gradient. *Mendel* **2021**, *27*, 23–29. [CrossRef]
104. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
105. Kouzehgar, M.; Meghjani, M.; Bouffanais, R. Multi-agent reinforcement learning for dynamic ocean monitoring by a swarm of buoys. In Proceedings of the Global Oceans 2020: Singapore–US Gulf Coast, IEEE, Biloxi, MS, USA, 5–30 October 2020; pp. 1–8.
106. Salimi, M.; Pasquier, P. Deep Reinforcement Learning for Flocking Control of UAVs in Complex Environments. In Proceedings of the 6th IEEE International Conference on Robotics and Automation Engineering (ICRAE), Guangzhou, China, 19–22 November 2021; pp. 344–352.
107. Fan, T.; Long, P.; Liu, W.; Pan, J. Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *Int. J. Robot. Res.* **2020**, *39*, 856–892. [CrossRef]
108. Zhao, W.; Queralta, J.P.; Qingqing, L.; Westerlund, T. Towards closing the sim-to-real gap in collaborative multi-robot deep reinforcement learning. In Proceedings of the 5th IEEE International Conference on Robotics and Automation Engineering (ICRAE), Singapore, 20–22 November 2020; pp. 7–12.
109. Lin, J.; Yang, X.; Zheng, P.; Cheng, H. End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning. In Proceedings of the 2019 IEEE International Conference on Mechatronics and Automation (ICMA), Tianjin, China, 4–7 August 2019; pp. 2493–2500.
110. Tolstaya, E.; Paulos, J.; Kumar, V.; Ribeiro, A. Multi-robot coverage and exploration using spatial graph neural networks. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 8944–8950.
111. Blumenkamp, J.; Morad, S.; Gielis, J.; Li, Q.; Prorok, A. A framework for real-world multi-robot systems running decentralized GNN-based policies. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 8772–8778.
112. Lin, J.; Yang, X.; Zheng, P.; Cheng, H. Connectivity guaranteed multi-robot navigation via deep reinforcement learning. In Proceedings of the Conference on Robot Learning, PMLR, Virtual, 16–18 November 2020; pp. 661–670.
113. Wang, J.; Cao, J.; Stojmenovic, M.; Zhao, M.; Chen, J.; Jiang, S. Pattern-rl: Multi-robot cooperative pattern formation via deep reinforcement learning. In Proceedings of the 18th IEEE International Conference On Machine Learning And Applications (ICMLA), Boca Raton, FL, USA, 16–19 December 2019; pp. 210–215.
114. Park, B.; Kang, C.; Choi, J. Cooperative Multi-Robot Task Allocation with Reinforcement Learning. *Appl. Sci.* **2021**, *12*, 272. [CrossRef]
115. Yao, S.; Chen, G.; Pan, L.; Ma, J.; Ji, J.; Chen, X. Multi-robot collision avoidance with map-based deep reinforcement learning. In Proceedings of the 32nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI), Baltimore, MD, USA, 9–11 November 2020; pp. 532–539.
116. Tan, Q.; Fan, T.; Pan, J.; Manocha, D. DeepMNavigate: Deep reinforced multi-robot navigation unifying local & global collision avoidance. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October–24 January 2020; pp. 6952–6959.
117. Han, R.; Chen, S.; Hao, Q. Cooperative multi-robot navigation in dynamic environment with deep reinforcement learning. In Proceedings of the 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 31 May–31 August 2020; pp. 448–454.
118. Blumenkamp, J.; Prorok, A. The emergence of adversarial communication in multi-agent reinforcement learning. *arXiv* **2020**, arXiv:2008.02616.
119. Sivanathan, K.; Vinayagam, B.; Samak, T.; Samak, C. Decentralized motion planning for multi-robot navigation using deep reinforcement learning. In Proceedings of the 3rd IEEE International Conference on Intelligent Sustainable Systems (ICISS), Thoothukudi, India, 3–5 December 2020; pp. 709–716.
120. Liu, J.y.; Wang, G.; Fu, Q.; Yue, S.h.; Wang, S.y. Task assignment in ground-to-air confrontation based on multiagent deep reinforcement learning. *Def. Technol.* **2022**, *19*, 210–219. [CrossRef]
121. Sadhukhan, P.; Selmic, R.R. Multi-agent formation control with obstacle avoidance using proximal policy optimization. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 2694–2699.
122. Sadhukhan, P. Proximal Policy Optimization for Formation Control and Obstacle Avoidance in Multi-Agent Systems. Ph.D. Thesis, Concordia University, Montreal, QC, Canada, 2021.
123. Ourari, R.; Cui, K.; Koeppl, H. Decentralized swarm collision avoidance for quadrotors via end-to-end reinforcement learning. *arXiv* **2021**, arXiv:2104.14912.

124. Zhang, T.; Liu, Z.; Pu, Z.; Yi, J. Multi-Target Encirclement with Collision Avoidance via Deep Reinforcement Learning using Relational Graphs. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 8794–8800.
125. Sadhukhan, P.; Selmic, R.R. Proximal policy optimization for formation navigation and obstacle avoidance. *Int. J. Intell. Robot. Appl.* **2022**, *6*, 746–759. [CrossRef]
126. Allen, R.E.; Gupta, J.K.; Pena, J.; Zhou, Y.; Bear, J.W.; Kochenderfer, M.J. Health-Informed Policy Gradients for Multi-Agent Reinforcement Learning. *arXiv* **2019**, arXiv:1908.01022.
127. Xia, J.; Luo, Y.; Liu, Z.; Zhang, Y.; Shi, H.; Liu, Z. Cooperative multi-target hunting by unmanned surface vehicles based on multi-agent reinforcement learning. *Defence Technol.* 2022, *in press*. [CrossRef]
128. Li, S.; Guo, W. Supervised Reinforcement Learning for ULV Path Planning in Complex Warehouse Environment. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 4384954. [CrossRef]
129. Paull, S.; Ghassemi, P.; Chowdhury, S. Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 8815–8822.
130. Na, S.; Krajník, T.; Lennox, B.; Arvin, F. Federated Reinforcement Learning for Collective Navigation of Robotic Swarms. *arXiv* **2022**, arXiv:2202.01141.
131. Fan, T.; Long, P.; Liu, W.; Pan, J. Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios. *arXiv* **2018**, arXiv:1808.03841.
132. Elfakharany, A.; Ismail, Z.H. End-to-end deep reinforcement learning for decentralized task allocation and navigation for a multi-robot system. *Appl. Sci.* **2021**, *11*, 2895. [CrossRef]
133. Wen, S.; Wen, Z.; Zhang, D.; Zhang, H.; Wang, T. A multi-robot path-planning algorithm for autonomous navigation using meta-reinforcement learning based on transfer learning. *Appl. Soft Comput.* **2021**, *110*, 107605. [CrossRef]
134. Khan, A.; Tolstaya, E.; Ribeiro, A.; Kumar, V. Graph policy gradients for large scale robot control. In Proceedings of the Conference on Robot Learning, PMLR, Virtual, 16–18 November 2020; pp. 823–834.
135. Alon, Y.; Zhou, H. Multi-agent reinforcement learning for unmanned aerial vehicle coordination by multi-critic policy gradient optimization. *arXiv* **2020**, arXiv:2012.15472.
136. Khan, A.; Kumar, V.; Ribeiro, A. Graph policy gradients for large scale unlabeled motion planning with constraints. *arXiv* **2019**, arXiv:1909.10704.
137. Asayesh, S.; Chen, M.; Mehrandezh, M.; Gupta, K. Least-restrictive multi-agent collision avoidance via deep meta reinforcement learning and optimal control. *arXiv* **2021**, arXiv:2106.00936.
138. Qamar, S.; Khan, S.H.; Arshad, M.A.; Qamar, M.; Gwak, J.; Khan, A. Autonomous Drone Swarm Navigation and Multi-target Tracking with Island Policy-based Optimization Framework. *IEEE Access* **2022**, *10*, 91073–91091. [CrossRef]
139. Zhou, W.; Li, J.; Zhang, Q. Joint Communication and Action Learning in Multi-Target Tracking of UAV Swarms with Deep Reinforcement Learning. *Drones* **2022**, *6*, 339. [CrossRef]
140. Hüttenrauch, M.; Šošić, A.; Neumann, G. Local communication protocols for learning complex swarm behaviors with deep reinforcement learning. In *Proceedings of the International Conference on Swarm Intelligence*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 71–83.
141. Hüttenrauch, M.; Adrian, S.; Neumann, G. Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.* **2019**, *20*, 1–31.
142. Wang, W.; Wang, L.; Wu, J.; Tao, X.; Wu, H. Oracle-Guided Deep Reinforcement Learning for Large-Scale Multi-UAVs Flocking and Navigation. *IEEE Trans. Veh. Technol.* **2022**, *71*, 10280–10292. [CrossRef]
143. Prianto, E.; Kim, M.; Park, J.H.; Bae, J.H.; Kim, J.S. Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay. *Sensors* **2020**, *20*, 5911. [CrossRef]
144. Cao, Y.; Wang, S.; Zheng, X.; Ma, W.; Xie, X.; Liu, L. Reinforcement Learning with Prior Policy Guidance for Motion Planning of Dual-Arm Free-Floating Space Robot. *arXiv* **2022**, arXiv:2209.01434.
145. Galceran, E.; Carreras, M. A survey on coverage path planning for robotics. *Robot. Auton. Syst.* **2013**, *61*, 1258–1276. [CrossRef]
146. Agmon, N.; Hazon, N.; Kaminka, G.A. Constructing spanning trees for efficient multi-robot coverage. In Proceedings of the IEEE International Conference on Robotics and Automation, ICRA 2006, Orlando, FL, USA, 15–19 May 2006; pp. 1698–1703.
147. Kapoutsis, A.C.; Chatzichristofis, S.A.; Kosmatopoulos, E.B. DARP: Divide areas algorithm for optimal multi-robot coverage path planning. *J. Intell. Robot. Syst.* **2017**, *86*, 663–680. [CrossRef]
148. Rekleitis, I.; New, A.P.; Rankin, E.S.; Choset, H. Efficient boustrophedon multi-robot coverage: An algorithmic approach. *Ann. Math. Artif. Intell.* **2008**, *52*, 109–142. [CrossRef]
149. Zheng, X.; Jain, S.; Koenig, S.; Kempe, D. Multi-robot forest coverage. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 3852–3857.
150. Marjovi, A.; Nunes, J.G.; Marques, L.; De Almeida, A. Multi-robot exploration and fire searching. In Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, St. Louis, MO, USA, 10–15 October 2009; pp. 1929–1934.
151. Nieto-Granda, C.; Rogers III, J.G.; Christensen, H.I. Coordination strategies for multi-robot exploration and mapping. *Int. J. Robot. Res.* **2014**, *33*, 519–533. [CrossRef]

152. Simmons, R.; Apfelbaum, D.; Burgard, W.; Fox, D.; Moors, M.; Thrun, S.; Younes, H. Coordination for multi-robot exploration and mapping. In Proceedings of the 17th National Conference on Artificial Intelligence (AAAI-00), Austin, TX, USA, 30 July–3 August 2000; pp. 852–858.
153. Rooker, M.N.; Birk, A. Multi-robot exploration under the constraints of wireless networking. *Control Eng. Pract.* **2007**, *15*, 435–445. [CrossRef]
154. Zhou, X.; Liu, X.; Wang, X.; Wu, S.; Sun, M. Multi-Robot Coverage Path Planning based on Deep Reinforcement Learning. In Proceedings of the 24th IEEE International Conference on Computational Science and Engineering (CSE), Shenyang, China, 20–22 October 2021; pp. 35–42.
155. Hu, J.; Niu, H.; Carrasco, J.; Lennox, B.; Arvin, F. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 14413–14423. [CrossRef]
156. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
157. Koenig, N.; Howard, A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566), Sendai, Japan, 28 September–2 October 2004; Volume 3, pp. 2149–2154.
158. Gama, F.; Marques, A.G.; Leus, G.; Ribeiro, A. Convolutional neural network architectures for signals supported on graphs. *IEEE Trans. Signal Process.* **2018**, *67*, 1034–1049. [CrossRef]
159. Aydemir, F.; Cetin, A. Multi-Agent Dynamic Area Coverage Based on Reinforcement Learning with Connected Agents. *Comput. Syst. Sci. Eng.* **2023**, *45*, 215–230. [CrossRef]
160. Zhang, H.; Cheng, J.; Zhang, L.; Li, Y.; Zhang, W. H2GNN: Hierarchical-Hops Graph Neural Networks for Multi-Robot Exploration in Unknown Environments. *IEEE Robot. Autom. Lett.* **2022**, *7*, 3435–3442. [CrossRef]
161. Gao, M.; Zhang, X. Cooperative Search Method for Multiple UAVs Based on Deep Reinforcement Learning. *Sensors* **2022**, *22*, 6737. [CrossRef]
162. Sheng, W.; Guo, H.; Yau, W.Y.; Zhou, Y. PD-FAC: Probability Density Factorized Multi-Agent Distributional Reinforcement Learning for Multi-Robot Reliable Search. *IEEE Robot. Autom. Lett.* **2022**, *7*, 8869–8876. [CrossRef]
163. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
164. Reynolds, C.W. Flocks, herds and schools: A distributed behavioral model. In Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, Anaheim, CA, USA, 27–31 July 1987; pp. 25–34.
165. Liang, Z.; Cao, J.; Lin, W.; Chen, J.; Xu, H. Hierarchical Deep Reinforcement Learning for Multi-robot Cooperation in Partially Observable Environment. In Proceedings of the 3rd IEEE International Conference on Cognitive Machine Intelligence (CogMI), Atlanta, GA, USA, 13–15 December 2021; pp. 272–281.
166. Acar, E.U.; Choset, H.; Lee, J.Y. Sensor-based coverage with extended range detectors. *IEEE Trans. Robot.* **2006**, *22*, 189–198. [CrossRef]
167. Chen, D.; Qi, Q.; Zhuang, Z.; Wang, J.; Liao, J.; Han, Z. Mean field deep reinforcement learning for fair and efficient UAV control. *IEEE Internet Things J.* **2020**, *8*, 813–828. [CrossRef]
168. Zhang, Y.; Yang, C.; Li, J.; Han, Z. Distributed interference-aware traffic offloading and power control in ultra-dense networks: Mean field game with dominating player. *IEEE Trans. Veh. Technol.* **2019**, *68*, 8814–8826. [CrossRef]
169. Guéant, O.; Lasry, J.M.; Lions, P.L. Mean field games and applications. In *Paris-Princeton Lectures on Mathematical Finance 2010*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 205–266.
170. Kadanoff, L.P.; Martin, P.C. Statistical physics: Statics, dynamics, and renormalization. *Phys. Today* **2001**, *54*, 54–55.
171. Nemer, I.A.; Sheltami, T.R.; Belhaiza, S.; Mahmoud, A.S. Energy-Efficient UAV Movement Control for Fair Communication Coverage: A Deep Reinforcement Learning Approach. *Sensors* **2022**, *22*, 1919. [CrossRef] [PubMed]
172. Liu, C.H.; Ma, X.; Gao, X.; Tang, J. Distributed energy-efficient multi-UAV navigation for long-term communication coverage by deep reinforcement learning. *IEEE Trans. Mob. Comput.* **2019**, *19*, 1274–1285. [CrossRef]
173. Surynek, P. An optimization variant of multi-robot path planning is intractable. In Proceedings of the AAAI Conference on Artificial Intelligence, Atlanta, GA, USA, 11–15 July 2010; Volume 24, pp. 1261–1263.
174. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
175. Wagner, G.; Choset, H. Subdimensional expansion for multirobot path planning. *Artif. Intell.* **2015**, *219*, 1–24. [CrossRef]
176. Bennewitz, M.; Burgard, W.; Thrun, S. Finding and optimizing solvable priority schemes for decoupled path planning techniques for teams of mobile robots. *Robot. Auton. Syst.* **2002**, *41*, 89–99. [CrossRef]
177. Dutta, A.; Dasgupta, P. Bipartite graph matching-based coordination mechanism for multi-robot path planning under communication constraints. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 857–862.
178. Kimmel, A.; Bekris, K. Decentralized multi-agent path selection using minimal information. In *Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 341–356.
179. Yu, J.; LaValle, S.M. Multi-agent path planning and network flow. In *Algorithmic Foundations of Robotics X*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 157–173.

180. Xu, Y.; Wei, Y.; Wang, D.; Jiang, K.; Deng, H. Multi-UAV Path Planning in GPS and Communication Denial Environment. *Sensors* **2023**, *23*, 2997. [CrossRef]
181. Wang, D.; Deng, H.; Pan, Z. Mrcdrl: Multi-robot coordination with deep reinforcement learning. *Neurocomputing* **2020**, *406*, 68–76. [CrossRef]
182. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
183. Li, M.; Jie, Y.; Kong, Y.; Cheng, H. Decentralized Global Connectivity Maintenance for Multi-Robot Navigation: A Reinforcement Learning Approach. In Proceedings of the 2022 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2022; pp. 8801–8807.
184. Achiam, J.; Held, D.; Tamar, A.; Abbeel, P. Constrained policy optimization. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 22–31.
185. Dutta, A.; Ghosh, A.; Kreidl, O.P. Multi-robot informative path planning with continuous connectivity constraints. In Proceedings of the 2019 IEEE International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 3245–3251.
186. Challita, U.; Saad, W.; Bettstetter, C. Interference management for cellular-connected UAVs: A deep reinforcement learning approach. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 2125–2140. [CrossRef]
187. Wang, B.; Liu, Z.; Li, Q.; Prorok, A. Mobile robot path planning in dynamic environments through globally guided reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 6932–6939. [CrossRef]
188. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Playa Blanca, Spain, 9–11 April 2018; pp. 4295–4304.
189. Chen, Y.F.; Liu, M.; Everett, M.; How, J.P. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 285–292.
190. Chen, Y.F.; Everett, M.; Liu, M.; How, J.P. Socially aware motion planning with deep reinforcement learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1343–1350.
191. Weiss, K.; Khoshgoftaar, T.M.; Wang, D. A survey of transfer learning. *J. Big Data* **2016**, *3*, 1–40. [CrossRef]
192. Konečnỳ, J.; McMahan, B.; Ramage, D. Federated optimization: Distributed optimization beyond the datacenter. *arXiv* **2015**, arXiv:1511.03575.
193. Luo, R.; Ni, W.; Tian, H.; Cheng, J. Federated Deep Reinforcement Learning for RIS-Assisted Indoor Multi-Robot Communication Systems. *IEEE Trans. Veh. Technol.* **2022**, *71*, 12321–12326. [CrossRef]
194. Sartoretti, G.; Kerr, J.; Shi, Y.; Wagner, G.; Kumar, T.S.; Koenig, S.; Choset, H. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 2378–2385. [CrossRef]
195. Ross, S.; Gordon, G.; Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics. JMLR Workshop and Conference Proceedings, Fort Lauderdale, FL, USA, 11–13 April 2011; pp. 627–635.
196. Damani, M.; Luo, Z.; Wenzel, E.; Sartoretti, G. PRIMAL _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robot. Autom. Lett.* **2021**, *6*, 2666–2673. [CrossRef]
197. Bengio, Y.; Louradour, J.; Collobert, R.; Weston, J. Curriculum learning. In Proceedings of the 26th Annual International Conference on Machine Learning, Montreal, QC, Canada, 14–18 June 2009; pp. 41–48.
198. Sun, L.; Yan, J.; Qin, W. Path planning for multiple agents in an unknown environment using soft actor critic and curriculum learning. *Comput. Animat. Virtual Worlds* **2023**, *34*, e2113. [CrossRef]
199. Pu, Y.; Gan, Z.; Henao, R.; Yuan, X.; Li, C.; Stevens, A.; Carin, L. Variational autoencoder for deep learning of images, labels and captions. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS'16, Barcelona, Spain, 5–10 December 2016; Volume 29.
200. Li, H. Decentralized Multi-Agent Collision Avoidance and Reinforcement Learning. Ph.D. Thesis, The Ohio State University, Columbus, OH, USA, 2021.
201. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5048–5058.
202. Everett, M.; Chen, Y.F.; How, J.P. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 3052–3059.
203. Semnani, S.H.; Liu, H.; Everett, M.; De Ruiter, A.; How, J.P. Multi-agent motion planning for dense and dynamic environments via deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3221–3226. [CrossRef]
204. Zhang, H.; Luo, J.; Lin, X.; Tan, K.; Pan, C. Dispatching and Path Planning of Automated Guided Vehicles based on Petri Nets and Deep Reinforcement Learning. In Proceedings of the 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), Xiamen, China, 3–5 December 2021; Volume 1, pp. 1–6.
205. Huang, H.; Zhu, G.; Fan, Z.; Zhai, H.; Cai, Y.; Shi, Z.; Dong, Z.; Hao, Z. Vision-based Distributed Multi-UAV Collision Avoidance via Deep Reinforcement Learning for Navigation. *arXiv* **2022**, arXiv:2203.02650.

206. Yarats, D.; Zhang, A.; Kostrikov, I.; Amos, B.; Pineau, J.; Fergus, R. Improving sample efficiency in model-free reinforcement learning from images. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 10674–10681.
207. Jeon, S.; Lee, H.; Kaliappan, V.K.; Nguyen, T.A.; Jo, H.; Cho, H.; Min, D. Multiagent Reinforcement Learning Based on Fusion-Multiactor-Attention-Critic for Multiple-Unmanned-Aerial-Vehicle Navigation Control. *Energies* **2022**, *15*, 7426. [CrossRef]
208. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv* **2016**, arXiv:1610.03295.
209. Ammar, H.B.; Tutunov, R.; Eaton, E. Safe policy search for lifelong reinforcement learning with sublinear regret. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 2361–2369.
210. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. *arXiv* **2015**, arXiv:1506.02438.
211. Taskar, B.; Chatalbashev, V.; Koller, D.; Guestrin, C. Learning structured prediction models: A large margin approach. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 896–903.
212. Liang, Z.; Cao, J.; Jiang, S.; Saxena, D.; Xu, H. Hierarchical Reinforcement Learning with Opponent Modeling for Distributed Multi-agent Cooperation. *arXiv* **2022**, arXiv:2206.12718.
213. Farrow, N.; Klingner, J.; Reishus, D.; Correll, N. Miniature six-channel range and bearing system: Algorithm, analysis and experimental validation. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6180–6185.
214. Shiell, N.; Vardy, A. A bearing-only pattern formation algorithm for swarm robotics. In Proceedings of the Swarm Intelligence: 10th International Conference, ANTS 2016, Brussels, Belgium, 7–9 September 2016; pp. 3–14.
215. Rubenstein, M.; Cornejo, A.; Nagpal, R. Programmable self-assembly in a thousand-robot swarm. *Science* **2014**, *345*, 795–799. [CrossRef]
216. Zhu, P.; Dai, W.; Yao, W.; Ma, J.; Zeng, Z.; Lu, H. Multi-robot flocking control based on deep reinforcement learning. *IEEE Access* **2020**, *8*, 150397–150406. [CrossRef]
217. Lan, X.; Liu, Y.; Zhao, Z. Cooperative control for swarming systems based on reinforcement learning in unknown dynamic environment. *Neurocomputing* **2020**, *410*, 410–418. [CrossRef]
218. Kortvelesy, R.; Prorok, A. ModGNN: Expert policy approximation in multi-agent systems with a modular graph neural network architecture. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 9161–9167.
219. Yan, C.; Xiang, X.; Wang, C. Fixed-Wing UAVs flocking in continuous spaces: A deep reinforcement learning approach. *Robot. Auton. Syst.* **2020**, *131*, 103594. [CrossRef]
220. Yan, C.; Xiang, X.; Wang, C.; Lan, Z. Flocking and Collision Avoidance for a Dynamic Squad of Fixed-Wing UAVs Using Deep Reinforcement Learning. In Proceedings of the 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Prague, Czech Republic, 27 September–1 October 2021; pp. 4738–4744.
221. Fujimoto, S.; Hoof, H.; Meger, D. Addressing function approximation error in actor-critic methods. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 1587–1596.
222. Ng, A. Sparse Autoencoder. CS294A Lecture Notes. 2011; Volume 72, pp. 1–19.
223. Bhagat, S.; Das, B.; Chakraborty, A.; Mukhopadhyaya, K. k-Circle Formation and k-epf by Asynchronous Robots. *Algorithms* **2021**, *14*, 62. [CrossRef]
224. Datta, S.; Dutta, A.; Gan Chaudhuri, S.; Mukhopadhyaya, K. Circle formation by asynchronous transparent fat robots. In *Proceedings of the International Conference on Distributed Computing and Internet Technology*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 195–207.
225. Dutta, A.; Gan Chaudhuri, S.; Datta, S.; Mukhopadhyaya, K. Circle formation by asynchronous fat robots with limited visibility. In *Proceedings of the International Conference on Distributed Computing and Internet Technology*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 83–93.
226. Flocchini, P.; Prencipe, G.; Santoro, N.; Viglietta, G. Distributed computing by mobile robots: Uniform circle formation. *Distrib. Comput.* **2017**, *30*, 413–457. [CrossRef]
227. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; Bengio, Y. Graph attention networks. *arXiv* **2017**, arXiv:1710.10903.
228. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
229. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Philip, S.Y. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 4–24. [CrossRef] [PubMed]
230. Wenhong, Z.; Jie, L.; Zhihong, L.; Lincheng, S. Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning. *Chin. J. Aeronaut.* **2022**, *35*, 100–112.
231. Nowak, M.A. Five rules for the evolution of cooperation. *Science* **2006**, *314*, 1560–1563. [CrossRef]
232. Smola, A.; Gretton, A.; Song, L.; Schölkopf, B. A Hilbert space embedding for distributions. In *Proceedings of the International Conference on Algorithmic Learning Theory*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 13–31.
233. Chung, T.H.; Hollinger, G.A.; Isler, V. Search and pursuit-evasion in mobile robotics. *Auton. Robot.* **2011**, *31*, 299–316. [CrossRef]
234. Sincák, D. Multi–robot control system for pursuit–evasion problem. *J. Electr. Eng* **2009**, *60*, 143–148.

235. Stiffler, N.M.; O'Kane, J.M. A sampling-based algorithm for multi-robot visibility-based pursuit-evasion. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 1782–1789.
236. Oh, S.; Schenato, L.; Chen, P.; Sastry, S. Tracking and coordination of multiple agents using sensor networks: System design, algorithms and experiments. *Proc. IEEE* **2007**, *95*, 234–254. [CrossRef]
237. Wang, Y.; Dong, L.; Sun, C. Cooperative control for multi-player pursuit-evasion games with reinforcement learning. *Neurocomputing* **2020**, *412*, 101–114. [CrossRef]
238. Tokekar, P.; Vander Hook, J.; Mulla, D.; Isler, V. Sensor planning for a symbiotic UAV and UGV system for precision agriculture. *IEEE Trans. Robot.* **2016**, *32*, 1498–1511. [CrossRef]
239. Batjes, N.H.; Ribeiro, E.; Van Oostrum, A.; Leenaars, J.; Hengl, T.; Mendes de Jesus, J. WoSIS: Providing standardised soil profile data for the world. *Earth Syst. Sci. Data* **2017**, *9*, 1–14. [CrossRef]
240. Viseras, A.; Garcia, R. DeepIG: Multi-robot information gathering with deep reinforcement learning. *IEEE Robot. Autom. Lett.* **2019**, *4*, 3059–3066. [CrossRef]
241. Said, T.; Wolbert, J.; Khodadadeh, S.; Dutta, A.; Kreidl, O.P.; Bölöni, L.; Roy, S. Multi-robot information sampling using deep mean field reinforcement learning. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 1215–1220.
242. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. *Coop. Robot. Sens. Netw.* **2015**, *2015*, 31–51.
243. Korsah, G.A.; Stentz, A.; Dias, M.B. A comprehensive taxonomy for multi-robot task allocation. *Int. J. Robot. Res.* **2013**, *32*, 1495–1512. [CrossRef]
244. Verma, S.; Zhang, Z.L. Graph capsule convolutional neural networks. *arXiv* **2018**, arXiv:1805.08090.
245. Kool, W.; Van Hoof, H.; Welling, M. Attention, learn to solve routing problems! *arXiv* **2018**, arXiv:1803.08475.
246. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
247. Devin, C.; Gupta, A.; Darrell, T.; Abbeel, P.; Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 2169–2176.
248. Tavakoli, A.; Pardo, F.; Kormushev, P. Action branching architectures for deep reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LO, USA, 2–7 February 2018; Volume 32.
249. Alkilabi, M.H.M.; Narayan, A.; Tuci, E. Cooperative object transport with a swarm of e-puck robots: Robustness and scalability of evolved collective strategies. *Swarm Intell.* **2017**, *11*, 185–209. [CrossRef]
250. Tuci, E.; Alkilabi, M.H.; Akanyeti, O. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Front. Robot. AI* **2018**, *5*, 59. [CrossRef] [PubMed]
251. Niwa, T.; Shibata, K.; Jimbo, T. Multi-agent Reinforcement Learning and Individuality Analysis for Cooperative Transportation with Obstacle Removal. In *Proceedings of the International Symposium Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 202–213.
252. Narvekar, S.; Peng, B.; Leonetti, M.; Sinapov, J.; Taylor, M.E.; Stone, P. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *J. Mach. Learn. Res.* **2020**, *21*, 181:1–181:50.
253. Stroupe, A.; Huntsberger, T.; Okon, A.; Aghazarian, H.; Robinson, M. Behavior-based multi-robot collaboration for autonomous construction tasks. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 1495–1500.
254. Werfel, J.K.; Petersen, K.; Nagpal, R. Distributed multi-robot algorithms for the TERMES 3D collective construction system. In Proceedings of the Robotics: Science and Systems VII, Institute of Electrical and Electronics Engineers, Los Angeles, CA, USA, 25–30 September 2011.
255. Werfel, J.; Petersen, K.; Nagpal, R. Designing collective behavior in a termite-inspired robot construction team. *Science* **2014**, *343*, 754–758. [CrossRef] [PubMed]
256. Sartoretti, G.; Wu, Y.; Paivine, W.; Kumar, T.; Koenig, S.; Choset, H. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Distributed Autonomous Robotic Systems*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 35–49.
257. Liang, Z.; Cao, J.; Jiang, S.; Saxena, D.; Chen, J.; Xu, H. From Multi-agent to Multi-robot: A Scalable Training and Evaluation Platform for Multi-robot Reinforcement Learning. *arXiv* **2022**, arXiv:2206.09590.
258. Bettini, M.; Kortvelesy, R.; Blumenkamp, J.; Prorok, A. VMAS: A Vectorized Multi-Agent Simulator for Collective Robot Learning. *arXiv* **2022**, arXiv:2207.03530.
259. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. Pytorch: An imperative style, high-performance deep learning library. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 8024–8035.
260. Chen, J.; Deng, F.; Gao, Y.; Hu, J.; Guo, X.; Liang, G.; Lam, T.L. MultiRoboLearn: An open-source Framework for Multi-robot Deep Reinforcement Learning. *arXiv* **2022**, arXiv:2209.13760.
261. Hu, S.; Zhong, Y.; Gao, M.; Wang, W.; Dong, H.; Li, Z.; Liang, X.; Chang, X.; Yang, Y. MARLlib: Extending RLlib for Multi-agent Reinforcement Learning. *arXiv* **2022**, arXiv:2210.13708.

262. Moritz, P.; Nishihara, R.; Wang, S.; Tumanov, A.; Liaw, R.; Liang, E.; Elibol, M.; Yang, Z.; Paul, W.; Jordan, M.I.; et al. Ray: A distributed framework for emerging {AI} applications. In Proceedings of the 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), Carlsbad, CA, USA, 8–10 October 2018; pp. 561–577.
263. Liang, E.; Liaw, R.; Nishihara, R.; Moritz, P.; Fox, R.; Gonzalez, J.; Goldberg, K.; Stoica, I. Ray rllib: A composable and scalable reinforcement learning library. *arXiv* **2017**, arXiv:1712.09381.
264. Hu, J.; Jiang, S.; Harding, S.A.; Wu, H.; Liao, S.w. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2102.03479 .
265. Zhou, M.; Wan, Z.; Wang, H.; Wen, M.; Wu, R.; Wen, Y.; Yang, Y.; Zhang, W.; Wang, J. Malib: A parallel framework for population-based multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2106.07551.
266. Michel, O. Cyberbotics Ltd. Webots™: Professional mobile robot simulation. *Int. J. Adv. Robot. Syst.* **2004**, *1*, 5. [CrossRef]
267. Rohmer, E.; Singh, S.P.; Freese, M. V-REP: A versatile and scalable robot simulation framework. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, 3–7 November 2013; pp. 1321–1326.
268. Dasari, S.; Ebert, F.; Tian, S.; Nair, S.; Bucher, B.; Schmeckpeper, K.; Singh, S.; Levine, S.; Finn, C. Robonet: Large-scale multi-robot learning. *arXiv* **2019**, arXiv:1910.11215.
269. Challita, U.; Saad, W.; Bettstetter, C. Deep reinforcement learning for interference-aware path planning of cellular-connected UAVs. In Proceedings of the 2018 IEEE International Conference on Communications (ICC), Kansas City, MO, USA, 20–24 May 2018; pp. 1–7.
270. Baca, J.; Hossain, S.; Dasgupta, P.; Nelson, C.A.; Dutta, A. Modred: Hardware design and reconfiguration planning for a high dexterity modular self-reconfigurable robot for extra-terrestrial exploration. *Robot. Auton. Syst.* **2014**, *62*, 1002–1015. [CrossRef]
271. Chennareddy, S.; Agrawal, A.; Karuppiah, A. Modular self-reconfigurable robotic systems: A survey on hardware architectures. *J. Robot.* **2017**, *2017*.
272. Tan, N.; Hayat, A.A.; Elara, M.R.; Wood, K.L. A framework for taxonomy and evaluation of self-reconfigurable robotic systems. *IEEE Access* **2020**, *8*, 13969–13986. [CrossRef]
273. Yim, M.; Shen, W.M.; Salemi, B.; Rus, D.; Moll, M.; Lipson, H.; Klavins, E.; Chirikjian, G.S. Modular self-reconfigurable robot systems [grand challenges of robotics]. *IEEE Robot. Autom. Mag.* **2007**, *14*, 43–52. [CrossRef]