Reinforcement Learning for Mobile Robotics Exploration: A Survey

Luíza Caetano Garaffa, Maik Basso, Andréa Aparecida Konzen, and Edison Pignaton de Freitas

***Abstract*— Efficient exploration of unknown environments is a fundamental precondition for modern autonomous mobile robot applications. Aiming to design robust and effective robotic exploration strategies, suitable to complex real-world scenarios, the academic community has increasingly investigated the integration of robotics with reinforcement learning (RL) techniques. This survey provides a comprehensive review of recent research works that use RL to design unknown environment exploration strategies for single and multirobots. The primary purpose of this study is to facilitate future research by compiling and analyzing the current state of works that link these two knowledge domains. This survey summarizes: what are the employed RL algorithms and how they compose the so far proposed mobile robot exploration strategies; how robotic exploration solutions are addressing typical RL problems like the exploration-exploitation dilemma, the curse of dimensionality, reward shaping, and slow learning convergence; and what are the performed experiments and software tools used for learning and testing. Achieved progress is described, and a discussion about remaining limitations and future perspectives is presented.**

***Index Terms*— Cooperative exploration, mobile robot exploration, multirobot exploration, multirobot systems (MRSs), reinforcement learning (RL), single robot exploration.**

I. Introduction

IN RECENT years, the advancement of new technologies applied to robotics has boosted the interest in academic research and practical application of mobile robots in different domains. Such applications include, for example, search and rescue (SAR) missions, intelligence, surveillance and reconnaissance (ISR), and planetary exploration. Several situations require robotic autonomy when the map or model of the environment is not previously known. The employment of mobile robots in such complicated contexts depends on a robust and efficient exploration strategy. Therefore, in the past few decades different “human-designed” exploration methods were proposed, such as the artificial potential fields [1] and the well-known frontier-based exploration [2], enabling exploration for many applications. However, defining the functioning of an appropriate resilient exploration strategy is hard in complex and dynamic real-world settings.

Concurrently, reinforcement learning (RL) techniques are based on letting the agent acquire skills through environment interaction instead of explicitly designing the desired behaviors. This learning paradigm tries to emulate the human learning process, which occurs through trial and error. Hence, RL and deep-RL (DRL) are being highlighted as promising alternatives to develop solutions to complex robotics problems, including unknown environment exploration. However, applying RL to real robots is not a straightforward task, facing problems like the curse of dimensionality, reward shaping, and awkward sim-to-real transference.

This survey reviews the most important strategies proposed by the recently published academic works tackling the robotic exploration problem using RL. The goal is to understand how the two fields are being integrated, which approaches are used to solve the RL in robotics issues, what are the so-far achieved successes and the remaining challenges, besides discussing future possibilities for this promising research area.

This article is organized as follows. Section II contains a concise overview of the mobile robot exploration field. Section III presents a brief review of RL basic concepts. Section IV describes some common problems and solutions related to RL in general robotic tasks. The reasons for using RL in robotic exploration and a proposal of classes of exploration strategies are presented in Section V. Sections VI and VII represent this survey’s main contribution, presenting the recently proposed solutions for exploration with RL, highlighting the remaining challenges for single-robot and multirobot systems (MRSs), respectively. Finally, some discussion topics are proposed in Section VIII, and a conclusion is presented in Section IX.

II. Role of Exploration in Mobile Robotics

A mobile robot is a robotic platform that is able to move through an environment using locomotive elements (e.g., wheels, propellers, and legs) [3]. In the 1950s, mobile robots were first introduced in industrial production processes, being mainly automated guided vehicles (AGVs) that followed

Manuscript received 4 March 2021; revised 21 August 2021; accepted 27 October 2021. Date of publication 12 November 2021; date of current version 4 August 2023. This work was supported in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior—Brasil (CAPES)—Finance Code 001 and in part by the Conselho Nacional de Desenvolvimento Científico e Tecnológico—Brasil (CNPq) under Project 309505/2020-8, Project 420109/2018-8, and Project 132668/2020-3. *(Corresponding author: Luíza Caetano Garaffa.)*

Luíza Caetano Garaffa and Andréa Aparecida Konzen are with the Graduate Program in Computer Science, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre 91509-900, Brazil (e-mail: lcgaraffa@inf.ufrgs.br).

Maik Basso is with the Graduate Program in Electrical Engineering, Department of Electrical Engineering, Federal University of Rio Grande do Sul, Porto Alegre 91509-900, Brazil.

Edison Pignaton de Freitas is with the Graduate Program in Computer Science, Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre 91509-900, Brazil, and also with the Graduate Program in Electrical Engineering, Department of Electrical Engineering, Federal University of Rio Grande do Sul, Porto Alegre 91509-900, Brazil.

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3124466.

Digital Object Identifier 10.1109/TNNLS.2021.3124466

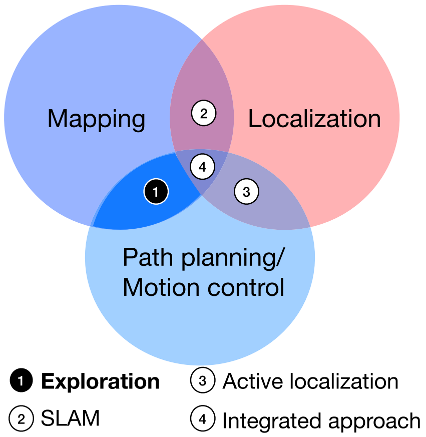

Fig. 1. Competences for robotic navigation, with highlight for the exploration task. Adapted from [8].

a predefined trajectory in order to transport tools [4]. Nowadays, however, mobile robots are operating in unstructured and dynamic environments, being employed in an increasing number of applications as medical care, personal services, planetary exploration, SAR operations, construction, entertainment, and surveillance [5]. This kind of application demand autonomous systems capable of choosing appropriate actions from their perception and interaction with the environment. Hence, the robotics research field is increasingly focusing on the development of robust software solutions that enable robots to autonomously transpose the challenges that arise from dynamic and unpredictable circumstances [6].

One of the biggest challenges to achieve full autonomy in unstructured environments is the *navigation problem*. Navigation comprehends the robot's ability to perform actions based on sensor measurements to reach its goal positions. Ideally, an autonomous agent should be able to explore the environment while simultaneously collecting information about the surrounding world, building an appropriate map, and localizing itself on this map [7]. Fig. 1 illustrates the backbone elements of robotic navigation, where each one represents a vast and challenging research field. The three basic navigation competencies—mapping, localization, and path planning, compose the simultaneous localization and mapping (SLAM), the active localization, and the integrated approach. However, this work focuses on an imperative aspect for mobile robot's autonomy: the exploratory behavior.

Autonomous exploration encompasses the robot's ability to autonomously move through an unknown environment while collecting the necessary information to accomplish a predefined goal. In the literature, robotic exploration was described as the attempt to answer the question "Given what you know about the world, where should you move to gain as much new information as possible?" [2]. Common problems solved by exploratory planning are map acquisition and single or multi targets identification. It is the exploratory behavior that makes human guiding or pre-planned trajectories unnecessary, being especially important in hostile or inaccessible environments [e.g., SAR in postdisaster scenes, planetary exploration, intelligence, surveillance, and reconnaissance (ISR)]. Defining an exploration strategy is not a straightforward process, given that an appropriate approach is widely dependent on the target application, and a compromise between map quality and coverage time is necessary. Therefore, different exploration methods and setups have been proposed and investigated over the years.

The exploration approaches can be classified according to the three most common paradigms for organizing intelligence in robotics: reactive, deliberative (or hierarchical), and hybrid. *reactive strategies* are behavior-based, which means that the robot exhibits behaviors as a reaction to events, without a planning stage [9]. Important reactive methods that have been employed for exploration include the random search [10], the subsumption architecture [11], and the potential fields methodology [1]. Purely reactive exploration is easy to implement and enables a considerable range of behaviors, but the absence of learning and planning abilities makes it usually inefficient for complex applications. On the other hand, *deliberative strategies* are based on a fixed event sequence: the agent senses the world, plans its actions, and then acts. The planning stage enhances the decision-making process, but its costly execution makes it difficult to respond appropriately to unexpected changes in the environment. Finally, there are exploration strategies that employ an *hybrid approach*, where the robot can perform planning but is also able to deal with the environment's unpredictability [3]. The hybrid paradigm has been the main focus of current research, and the great majority of recent works employ hybrid exploration strategies.

As mentioned above, several exploration strategies have been proposed over time. Well-known methods as the generalized Voronoi graph [12] and the widespread frontier-based approach [2] divide the world representation and rank the resulting regions to make logical decisions. Other strategies employ clustering algorithms such as K-means [13] or spectral clustering [14] to divide the map into different sectors. Whether it divides and ranks regions or not, the crucial part of an exploration strategy is deciding the next desirable destination or movement. Some frontier-based methods use interest attributes (e.g., nearest, farthest, most extensive frontier) to decide the next target region. A common alternative is the use of cost-utility models, like market-based [15] or next-best-view selection algorithms [16].

Another popular approach for the exploration decision-making process is modeling the problem as a Markov decision process (MDP) and solving it, for example, with dynamic programming algorithms, value iteration, or RL. Both MDP and RL are further discussed in the following sections. In methods where a target location is defined, the robot can navigate using reactive behaviors (e.g., move-to-goal, avoid-obstacles), plan a path and reactively execute it, or continuously replan and execute the path [17]. In strategies with path planning, graph search algorithms like RRT [18], A* [19], and Dijkstra [20] are commonly used. Some recent studies use Machine Learning techniques to plan the robot's path [21]. All methods above demonstrate that robotic exploration is a widely studied field. However, considering the growing demand for autonomous exploration in complex applications, the development of appropriate robotic exploration strategies

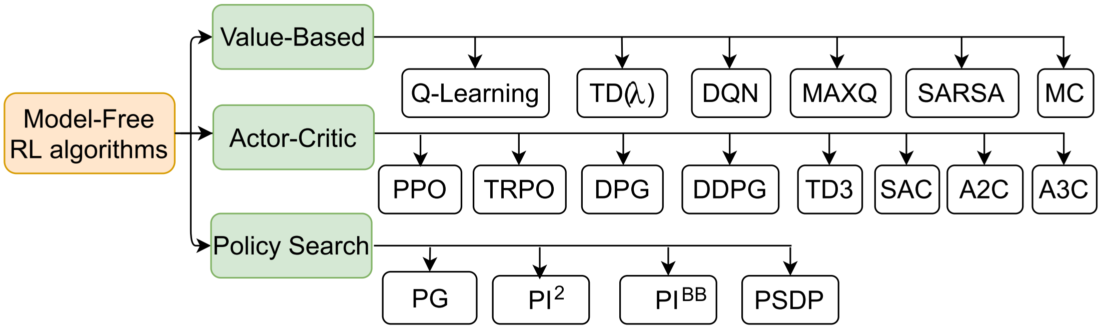

Fig. 2. Proposed taxonomy for the model-free RL algorithms, divided into value-based, policy search, and hybrid algorithms.

remains a challenge and a topic of broad and current research interest.

III. Reinforcement Learning in a Nutshell

RL is a computational approach to make an agent learn from interactions with the environment, without explicit examples or external instructors [22]. Through trial-and-error, the agent tries to maximize a numerical reward signal in order to learn an optimal policy (π^*) that maps situations into actions [23]. The reward function, which determines the agent's feedback depending on the selected action, is defined in accordance with the application goal. The fundamental elements in an RL problem are the environment and an agent capable of sensing the environment's state and taking actions that affect this state. The agent also must have one or more goals related to the state of the environment.

In RL problems, the environment is typically modeled as an MDP or a partially observable MDP (POMDP). Both MDP and POMDP are mathematical tools to model discrete decision-making and optimization problems. Generally, an MDP can be defined by a tuple (S, A, T, R), where S represents a set of states ($s \in S$), A represents a set of actions ($a \in A$), T is the transition probability associated with the states ($T(s_{t+1} \| s_t, a)$), and R is the reward function. The system attends the Markov property, which defines that only the current state impacts future decisions. POMDP is an extension of the MDP, and its main difference is the definition that states are not fully observable. They can be defined by a tuple (S, A, ω, T, O, R), in which ω is a set of observations and O is the probability distribution associated with the observations [24].

Some challenges arise from the RL basic definitions. For example, the agents trained through RL face a well-known trade-off called *exploration versus exploitation dilemma*. Since the agent must interact with the environment to learn, it must choose between exploring new actions to improve its knowledge or exploiting the current action-value estimates to get the most certain reward. The chosen method to solve this dilemma directly impacts learning efficiency. Another problem that RL methods must solve is the *credit assignment problem*, which refers to the problem of distributing credit for success among the many decisions involved in achieving it [22].

A large group of learning algorithms has been proposed to solve RL problems. These algorithms can be divided into two main classes: model-based and model-free. *Model-based* methods use learning and a model of the environment's transitions to approximate a global value or policy function. The model can be learned, meaning that the agent learns both the model and a value or policy, or the model can be known, where the agent knows the model and uses planning to learn a global value or policy [25]. On the other hand, *model-free* methods do not rely on an environment model, and the agent learns the policy or value directly through trial-and-error with the physical system [26]. Model-free RL approaches are able to solve problems that cannot be solved mathematically, but they return nonoptimal solutions. As is highlighted in Sections VI and VII, the great majority of surveyed works adopt Model-free approaches. This common choice may be justified by the targeted application, which encompasses exploring an unknown environment whose ground-truth model is usually previously unavailable. At the same time, the environments tend to be too complex to be precisely modeled by the agent. Therefore, although model-based RL can be used for robotic exploration, model-free algorithms are the focus of this section and are discussed in greater detail.

Fig. 2 illustrates a model-free RL taxonomy proposal that divides the algorithms into three subclasses: value-based, policy search, and actor–critic. The image does not contain all existing model-free algorithms but presents some prominent examples for each subclass. In *policy search* algorithms, the policy is learned directly from the agent interaction with the environment [27]. Some algorithms that fit into this category are the policy gradient (PG), which optimize parametrized policies using gradient descent [28]. The policy search by dynamic programming (PSDP) performs the policy search from a baseline policy distribution [29]. On the other hand, policy improvement with path integrals (PI^2) uses first-order principles of stochastic optimal control to learn a parametrized policy [30], and it has inspired other algorithms such as the PI^{BB}, which is a black-box optimization algorithm [31].

In *value-based* algorithms, the agent's trial-and-error process results in a value function from which the policy is derived. This value function estimates how advantageous it is for the agent to be in a specific state or to perform a given action considering a specific state [22]. Among the most famous algorithms are the classic Q-learning [32], which computes the quality of the taken actions, and the deep Q-networks (DQNs), which use deep learning to estimate the value function in a Q-learning framework [33]. MAXQ algorithm represents a hierarchical RL method that decomposes the target MDP into a hierarchy of smaller MDPs [34]. Other important value-based algorithms use the action performed by the current policy to learn the value function, which is the case of Monte Carlo (MC) [35], state–action–reward–state–action (SARSA) [36] and its variations like the SARSA(λ).

Finally, *actor–critic* algorithms adopt a hybrid approach, where the policy is not derived exclusively from a value function, but the value function is computed and influences the policy learning process. The actor is the policy structure that predicts the actions, and the critic computes the value function that evaluates the policy choices. Some widespread actor–critic algorithms include the soft actor–critic (SAC) [37], asynchronous advantage actor–critic (A3C) [38], and the advantage actor–critic (A2C), which is derived from A3C. Both proximal policy optimization (PPO) [39] and trust region policy optimization (TRPO) [40] are PG algorithms that improve the policy based on available data, without extreme parameter updates to avoid performance collapse. Other examples include the deterministic PG (DPG) [41] and its variations like the deep deterministic PG (DDPG), [42] and the twin delayed deep deterministic PGs (TD3) [43].

IV. Reinforcement Learning in Robotic Applications

As remarkably described in [44], the disciplines of RL and robotics compose a promising relationship, given that RL makes hard-to-engineer behaviors feasible for robotics applications, while robotics challenges inspire and validate RL solutions. The intrinsic function of robots is to replicate animal behavior to assist or replace humans in different tasks. Therefore, the idea of integrating learning capability to robotic devices arises almost naturally. Simultaneously, RL tries to emulate how humans and other animals learn through trial-and-error interactions with the environment, being even used to study the brain functioning in research fields like neuroinformatics. Hence, the application of RL techniques to robotics is increasingly being investigated by academics, motivated by the goal of letting the robot autonomously learn how to plan and control its actions in complex and dynamic tasks.

However, robotics differ in several essential aspects compared with domains where RL was already successfully employed, such as video games. Robotics applications take place in the real-world, which means that the agent must cope with partially observable systems, measurement noise and delays, impossibility to speed up the training phase in the real environment, and expensive hardware that requires safe exploration. The most commonly adopted alternative to limit the interaction with the real world is to perform the training phase through simulation, which also presents issues. Transferring the behaviors learned through simulation to the real robot is not usually a straightforward task, given that errors in the simulated environment model can easily accumulate and cause the behavior to diverge from the expected [45]. Other RL problems that are accentuated in robotics are the definition of appropriate reward functions, also called reward shaping, and the curse of dimensionality, which refers to the exponential rise of computational effort given the increase of spatial dimensions [46].

In order to solve the problems mentioned above and make the employment of RL in real robotic applications viable, several tools and strategies have been exploited. It is possible to reduce the dimensionality curse impact and enhance the RL algorithm convergence and generalization with smart approaches to discretize the state or action spaces [47], [48]. Function approximation can also be used to ease the dimensionality problem, besides helping to predict the reward functions [49] and to making value-based RL algorithms suitable for robotics [50] using methods like Gaussian processes and neural networks. In the cases where function approximation is not sufficient to reduce the dimensionality (e.g., balancing and walking applications like humanoid robotics), different alternatives are being investigated, like the central pattern generator based RL [51] and the combination of symbolic inverse kinematic with RL [52]. An alternative to speed up training and increase the convergence probability is transferring auxiliary information or knowledge to the agent before or during the learning phase. This can be accomplished by, for example, employing Transfer Learning techniques [53], using demonstration [54], [55], learning forward environment models [56], [57], incorporating human feedback during training [58], and decomposing a task into simpler components [59], [60]. However, despite existing solutions, it is possible to state that applying RL to robotics remains a challenge.

V. Robotic Exploration Using Reinforcement Learning

Considering the discussed limitations in robotic applications, what does justify the use of RL to tackle autonomous robotic exploration? The fact is that “human-designed” exploration techniques usually make strong assumptions about the environments and the tasks, which may restrict their adaptability to complex dynamic environments and thus limit their application in real-world practices. The need for robust and flexible solutions to robotic exploration and the significant advances in machine learning techniques have made the combination of the two research fields a hot topic in recent years. In that context, the employment of RL for robotic exploration tasks is being increasingly investigated, driven by the idea of letting the agents automatically learn skills from environment interaction instead of receiving explicit instructions. Furthermore, RL does not require dataset labeling, which represents a large cost for Supervised Learning solutions.

Sections VI and VII contain an overview of the most recent academic researches that propose exploration strategies of single and multiple robots employing RL techniques. An environment was considered unknown if the robots did not have access to a world’s map previously to the exploration. An application was considered an exploration task when the static or dynamic environment is unknown or when the targets positions are unknown. Therefore, even though some works use the terms *navigation* or *path planning* to describe the proposed strategy, they were considered suitable for the current study if the level

TABLE I

RL Algorithms Used in Single-Agent Strategies for Unknown Environment Exploration Tasks

	Model-Free						
	Value-based		Actor-Critic				
RL Algorithm	**Q-Learning**	**Deep Q-Learning**	**A3C**	**DPG**	**DDPG**	**SAC**	**TD3**
Research Work	[61], [62], [63], [64]	[65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75]	[76], [77], [78], [79]	[80]	[81]	[82]	[83]

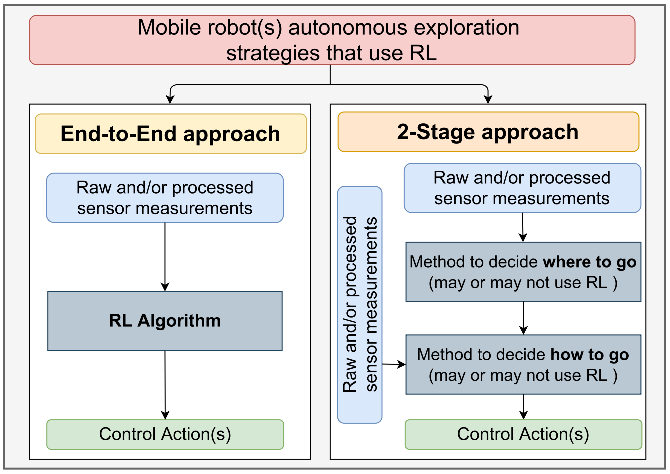
Mobile robot(s) autonomous exploration strategies that use RL
End-to-End approach
Raw and/or processed sensor measurements
RL Algorithm
Control Action(s)
2-Stage approach
Raw and/or processed sensor measurements
Method to decide where to go (may or may not use RL)
Raw and/or processed sensor measurements
Method to decide how to go (may or may not use RL)
Control Action(s)

Fig. 3. Classification of mobile-robot exploration strategies that employ RL techniques, including end-to-end and two-stage approaches.

of environment information met the requirements mentioned above. Some analyzed applications include unknown environment mapping and coverage, single or multitarget search with static, dynamic, known and unknown targets, and crossing unknown environments in a map-less fashion to reach a goal position.

The great majority of the recently proposed exploration techniques fit into the hybrid paradigm, using global environment information to plan the following actions and reacting to unseen events. Therefore, another classification was developed to facilitate the synthesis of how the RL algorithms usually compose the proposed exploration strategies. The proposed classification divides the methods into two categories, as illustrated in Fig. 3. In both classes, the inputs are composed of raw sensor measurements, such as camera images and sets of distances or velocities, by processed sensor measurements, like partial maps, robot trajectory, and robot pose, or by a combination of both. In some less common cases, human feedback is also used as input.

Strategies with an *end-to-end* approach accomplish robot exploration tasks as a black box. The inputs are fed to the RL or deep RL algorithm, which directly returns the robot control actions, like linear and angular velocities or the movement the robot must perform (e.g. move forward, backward, right, or left). As the name states, the *two-Stage* strategies divide the robot decisions into two parts and can integrate RL with nonlearning-based methods. First, the inputs are used by an algorithm that decides the next robot's target position. Then, the selected location and other sensory inputs are used to perform the path planning and guide the agent from the current to the target position. RL algorithms can be employed in the first, second, or both stages. In the following sections, the stated classification is taken as a reference for evaluating the research works.

VI. Single Robot Exploration

This section reviews the state-of-the-art exploration strategies designed for a single robotic agent using RL techniques. The key analyzed aspects are: 1) which RL algorithms compose and what is their role on the most recent exploration strategies; 2) how current works deal with problems like the exploration-exploitation dilemma, curse of dimensionality, reward shaping, and slow learning convergence; 3) what are the commonly employed simulation tools, simulation and real-world experiments, achieved progress, and remaining limitations. The reviewed research works projected strategies focusing on unmanned aerial vehicles (UAVs) and unmanned ground vehicles (UGVs), and to a lesser extent focusing on autonomous underwater vehicles (AUVs).

The applications targeted by the academic works are divided into *goal-guided*, regarding tasks whose goal is to find or reach a specified target, and *nonguided*, regarding tasks whose main goal is to cover a whole area, usually with mapping purposes. The most common identified nonguided application is mapping unknown indoor environments [63], [65], [66], [74]. For instance, in [77] the problem of mapping an unknown office is tackled, and in [64] the exploration is used for learning a saliency map of the environment. Currently, goal-guided applications are being studied in a larger quantity and with a greater task diversity. Some examples include finding victims in postdisaster scenes [62], [76], finding the original location of a chemical leaking source in underwater environments [80], traversal of land vehicles in undiscovered tracks [71], and goal-driven map-less navigation [78], [79], [81], [84]. The great majority of both kinds of applications include obstacle avoidance.

A. RL Algorithms in the Exploration Strategies

Table I contains a summary of the most common RL algorithms employed in single-agent exploration strategies. As already mentioned in Section III, the fact that all the identified algorithms are model-free derives from the nature of exploration tasks, in which the environment model is usually unknown or the environment is too complex to be modeled. It is possible to note that value-based algorithms are the most commonly adopted alternative, especially the deep Q-learning methods. This prevalence may be explained by the fact that deep neural networks enabled robust function approximation, making the employment of the well-known classic Q-learning in complex robotic applications more realistic. Actor–critic algorithms have also been considerably used

TABLE II

HOW RL ALGORITHMS COMPOSE TWO-STAGE SINGLE-AGENT EXPLORATION STRATEGIES

		Application Goal	Research Work	RL Algorithm	DRL	State Space	Action Space	Reward	Complementary Method
2-Stage Single Agent Exploration Strategies	RL algorithm to decide **where to go**	Goal-Guided	[76]	A3C	Yes*	Discrete	Discrete	Sparse	A* to find path and ROS Move Base package to generate control movements
		Non-Guided	[77]	A3C	Yes*	Discrete	Discrete	Dense	Next Best View with Bayesian Optimization to choose a point in the region selected by the DRL, and A* for path planning
			[67]	DQN	Yes*	Discrete	Discrete	Dense	A* for path planning and Timed-Elastic-Band (TEB) for setting the robot's velocities
			[64]	Q-Learning	No	Discrete	Discrete	Sparse	Method to control robot movements not specified
	RL algorithm to decide **how to go**	Non-Guided	[65]	OS-Q-ELM Network	Yes	Continuous	Discrete	Dense	Choose the furthest point measured by RPLIDAR

* Neural Netowrks with Convolutional Layers.

TABLE III

HOW RL ALGORITHMS COMPOSE END-TO-END SINGLE-AGENT EXPLORATION STRATEGIES

	Application Goal	Research Work	RL Algorithm	DRL	State Space	Action Space	Reward
End-to-End Single Agent Exploration Strategies	Non-Guided	[61]	Q-Learning	No	Discrete	Discrete	Dense
		[66]	DQN	Yes*	Discrete	Discrete	Dense
		[74]	DQN	Yes*	Discrete	Discrete	Sparse
	Goal-Guided	[68]	Double DQN	Yes*	Discrete	Discrete	Dense
		[69]	DQN	Yes	Discrete	Discrete	Dense
		[70]	DQN	Yes*	Continuous	Discrete	Dense
		[62]	Q-Learning	No	Discrete	Discrete	Dense
		[80]	DPG	Yes	Continuous	Continuous	Sparse
		[71]	DQN with Rainbow network	Yes*	Discrete	Discrete	Dense
		[72]	Dueling Double DQN (D3QN)	Yes*	Discrete	Discrete	Dense
		[78]	A3C	Yes*	Discrete	Discrete	Sparse
		[73]	Double DQN	Yes*	Discrete	Discrete	Dense
		[82]	SAC	Yes*	Continuous	Continuous	Dense
		[81]	Asynchronous DDPG	Yes	Continuous	Continuous	Dense
		[79]	A3C	Yes*	Continuous	Discrete	Dense
		[75]	DQN	Yes*	Continuous	Discrete	Sparse
		[84]	Deep Siamese Actor-Critic Network	Yes*	Discrete	Discrete	Dense
		[83]	TD3	Yes	Continuous	Continuous	Sparse
	Non-Guided and Goal-Guided	[63]	Q-Learning	No	Discrete	Discrete	Dense

* Neural Netowrks with Convolutional Layers.

in the exploration strategies, particularly the asynchronous advantage actor–critic (A3C), developed by Google Deep Mind in 2016 [38]. No Policy Search algorithm was identified in the most critical recent academic works for single-robot exploration approaches.

Tables II and III summarize how the identified RL algorithms compose, respectively, the proposed two-stage and end-to-end single-agent exploration strategies. Two approaches were identified in the two-stage strategies. The first and more common one is using the RL algorithm to decide the next location the robot should move to and then employing complementary methods to guide the robot toward the target point. In [76], DRL is combined with the classic frontier-based exploration. An A3C network receives the known map, the robot location, and the frontiers locations, returning the coordinates of the next goal frontier. Similarly, an A3C network receives the current map, the agent's location and orientation in [77], and returns the next visiting direction, given that the space around the agent is equally divided into six sectors. In [67], a DQN returns the goal points in the grid map, using as input the map, the current, and the historical robot positions. The three mentioned works employ the well-studied A* algorithm for path planning. In [64], Q-learning is used to decide whether the agent must move to one of its four adjacent nodes or if it must learn from the current node by capturing useful images to create a saliency map of the environment. The reviewed works prove that this approach is a smart way to discretize and reduce the action space while using RL to solve the main exploration problem, which is to decide the next region the robot should visit. However, a possible drawback is that the methods focus on the trajectory generation from one point to another, not prioritizing the full coverage of explored areas in the path planning process.

Another approach in two-stage strategies is employing a nonlearning method to decide the next goal location and then using RL to guide the robot toward the selected point. For single agents, this approach is less common. In [65], the goal point is randomly selected between the set of the furthest points measured by an RPLIDAR sensor with scanning range 360°. An online sequential extreme learning machine (OS-ELM) is used to estimate the Q-values and lead the robot toward the target with object avoidance. The network receives the sensors' measurements, the distance between robot and goal, the angle between robot orientation and goal, and returns the robot action, which can be moving 0.3 m, turning left, or turning right. The downside of this approach is that it does not use the RL algorithm for the critical exploration decision-making process, but to plan the trajectory between two known points. At the same time, there are much more efficient and well-established path planning methods in the literature.

Table III reveals the most investigated approach for single robot exploration using RL: tackling goal-guided problems in an end-to-end fashion. Within this context, many works use neural networks with convolutional layers aiming to reproduce human behavior and directly translate pixels into actions. For example, the Rainbow network architecture was used to generate local paths toward goal positions in unknown rough terrain environments while maintaining the UGV's safety [71]. Some works employ DQN algorithms' variations to improve their performance and avoid problems caused by estimation errors. double DQNs, proposed by Hasselt [85] and based on decoupling action selection from evaluation, were used to define the next robot action between five possible movement controls [68], [73]. A dualing architecture-based deep double Q network (D3QN), which combines double DQN and dueling network architectures, was used with a similar action space [72]. To tackle the common RL problem of lack of generalization capability to new goals, Zhu *et al.* [84] proposed a new deep neural network called siamese actor–critic network, which receives an RGB image of the current environment observation and an RGB image of the target, and returns the movement the agent should perform next. Other works also propose end-to-end exploration solutions combining convolutional layers for feature extraction with the A3C algorithm [78], [79] and the SAC [82]. As previously mentioned, these methods take images as inputs. A limitation of all mentioned works is that the models are trained with synthetic, rendered scenarios, which generally do not generalize appropriately to real-world images.

Within the goal-guided end-to-end solutions, there are also the ones that do not employ convolutional neural networks (CNNs). In order to guide the robot to reach a specific point at an unknown environment with obstacle avoidance, a back propagation (BP) network is used in [69] to estimate the Q-values. A continuous action space for the robot motion control is employed in [80] and [81] by using, respectively, an actor–critic model with deterministic PG (DPG) and asynchronous deep DPG. End-to-end solutions for nonguided applications are not as numerous in the literature. In both [66] and [74], CNNs are used to extract features from raw RGB images and depth information from an RGB-D sensor, respectively, and DQN define the next agent's movement to explore an unknown environment entirely. With the same goal, classic Q-learning is used in [61]. Finally, few solutions adapt for both goal-guided and nonguided applications. In [63], a Q-learning algorithm uses the UAV position, a binary parameter that indicates the presence or absence of a signal source (target) in the environment, and each grid cell's state to guide the robot in order to map the environment and detect targets. The integrated approach of end-to-end solutions also has its limitations. The main downfall is the need to control the vehicle reactively through on-board real-time computing. Moreover, end-to-end methods still experience limited generalization capabilities and they are tightly dependent on the system (e.g., type of vehicle and sensors) [86].

B. Approaching Common Problems

1) Exploration–Exploitation Dilemma: The great majority of academic works about single mobile robot exploration using RL employ the classic ϵ-greedy or some simple variations of the algorithm to handle the exploration-exploitation dilemma. Considering ϵ a positive scalar between 0 and 1, the agent selects the action with maximum value with probability $1-\epsilon$, and selects a random action with probability ϵ. If the ϵ value increases during training, the exploratory behavior is more stimulated in the early learning stages, and as time passes, the agent selects the optimal action more frequently. Furthermore, a fewer number of works employ different techniques, but with a similar approach. Stochastic switching is used in [83], and in [70] the Boltzmann distribution is employed to define the probability of choosing one action given the current state. In [82], the temperature decay training paradigm is proposed, working similarly to ϵ-greedy, but adapted to obtain an uncertainty-averse behavior.

2) Curse of Dimensionality: To avoid the curse of dimensionality, the most common approach is to employ deep neural networks to perform function approximation. Another function approximation technique used in [62] is called fixed sparse representation (FSR), and maps the original Q table to a parameter vector. The discretization of the states and action spaces is also a commonly adopted alternative, as can be observed in both Tables II and III. In the analyzed works, the algorithms that directly return motion controls with a discrete set of actions used between 3 and 9 possible alternatives. It is important to observe that although adopting a small and discrete set of actions accelerates training convergence, in motion control applications it may limit the robot performance, resulting in tortuous paths.

3) Reward Shaping: In general, the examined research works use heuristic strategies to define the reward functions for the robotic exploration applications. The application goal directly influences how the rewards are modeled: in a SAR mission, the reward usually encourages the agent to find the most quantity of information in the early stages of exploration [62], for example. For goal-driven navigation, the reward encourages the distance narrowing between the agent and the target [69], [70]. Collisions with obstacles are associated with punishments in all strategies. The rewards can be either *dense*, which means they are assigned to the agent in many different states, or *sparse*, usually returning zero for most states and only rewarding the agent in a few states or events.

Tables II and III indicate the kind of reward used for each reviewed work. In all cases, the agent receives the most significant reward if it achieves the exploration goal and gets a considerable punishment if the mission fails (e.g., takes too much time, a collision happens). The difference is that dense strategies adopt intermediate rewards. Sparse rewards are easier to be defined and are adopted by many strategies [64], [74]–[76], [78], [80], [83], but they can increase the learning convergence time. Therefore, despite the difficulty to properly determine dense rewards, they are the most adopted option among the single robot exploration works. Common approaches include: punishing the agent at every time-step to decrease exploration time [62], [73], [84]; small positive rewards if getting away from obstacles or closer to target; minor punishments otherwise [61], [69]–[71], [81]. From the analyzed work, it is possible to conclude that reward shaping for single-robot exploration still represents a nontrivial, application-dependent task.

4) Learning Convergence: Different strategies to accelerate learning convergence were identified in the robotic exploration works. The solutions used to tackle all previously mentioned aspects—exploration-exploitation dilemma, the curse of dimensionality, and reward shaping—directly impact the learning convergence. The discretization of the state and action spaces, widely adopted in the reviewed works, contributes to faster convergence. Curiosity-driven intrinsic rewards are used in [78] and [79] to stimulate the agent to explore new areas after many failures in familiar positions, improving data efficiency and avoiding possible deadlocks.

A method that is especially present is the experience replay (EP), which employs past experiences in the training process to reduce the correlation between successive samples, make the learning process smoother, and improve data efficiency [66], [67], [73]–[75], [82]. An attention mechanism that analyses the importance of the algorithm's inputs is employed in [71], aiming to increase data efficiency. Transfer learning techniques were also employed in the single-robot exploration context. Dynamic programming was used to find approximate solutions to initialize the actor–critic networks [80]. Similarly, the weights of the CNN trained with RL were initialized using a supervised learning model trained with real-world data [74]. Another identified strategy is to increase the training difficulty gradually and share knowledge between different agents [84].

C. Simulation and Real-World Experiments

As well as most robotic RL projects, all the reviewed works perform the learning process through simulation, reducing interaction with the real world and avoiding problems like unsafe exploration and the impossibility to speed up training. Although not all works mention it, Table IV summarizes commonly adopted software tools for training and testing experiments. As the agent learns through environment interaction, the physics simulation engine directly interferes in the learned behaviors and the transference from simulation to real robots. Therefore, the heterogeneity of simulation tools can make the comparison between exploration strategies inaccurate. It is possible to note that Gazebo [87] stands out as the most adopted simulation environment, as well as ROS is broadly employed to develop robot communication and control software. In comparison, the adoption of platforms for developing RL algorithms is more distributed among the different alternatives.

TABLE IV

SOFTWARE TOOLS USED FOR TRAINING AND TESTING SINGLE-AGENT EXPLORATION STRATEGIES

	Name	Used in
Physics Simulation Engine	Stage Simulator	[67], [76], [82]
	V-Rep	[61], [81]
	Gazebo	[65], [66], [71] [72], [74], [78]
	Matlab	[62]
	AirSim	[73]
	Unity 3D	[84]
	Box2D	[83]
Platforms and Toolkits for RL and DRL algorithm development	DeepMind Lab	[75], [78]
	Caffe	[74]
	OpenAI Gym	[73]
	Keral-RL	[73]
	TensorFlow	[84]
Framework for robot software development	ROS	[65]–[67], [76] [74], [83]

TABLE V

RELATION OF WORKS THAT PERFORM AND DO NOT PERFORM TESTS IN DYNAMIC ENVIRONMENTS

Tests in dynamic environments	
Yes	**No**
[69] [80] [71] [73] [82] [83]	[76] [77] [61] [66] [67] [68] [65] [70] [62] [72] [78] [81] [79] [74] [75] [84] [63] [64]

For performance evaluation, the adopted metrics depend on the application's goal task. Nonguided applications usually evaluate the *explored region rate*, while goal-guided evaluate the *success rate* related to reaching or locating the targets. Some standard metrics for both kinds of works are the *path length* and *exploration efficiency*. When it comes to the considered baselines, the most common alternatives are classic techniques, such as frontier-based or artificial potential fields, simple methods like random walk, or other RL algorithms. In [76], for example, the proposed exploration strategy enabled 98% of explored region rate in simulation and 97% in a real robot with shorter paths than cost-utility frontier-based methods. The classic Q-learning is used as a baseline in [69], [70] and [75], and its performance is overcome by the proposed DRL methods in all cases. Table V indicates the number of works that perform tests in dynamic environments, with moving objects or changing environmental characteristics. The great majority of methods are not validated in dynamic scenarios, which is an essential step for fulfilling the strategies' purpose and solving complex real-world problems. Another aspect that can negatively affect the sim-to-real transfer of the

TABLE VI
RL ALGORITHMS USED IN MRS COORDINATION STRATEGIES FOR UNKNOWN ENVIRONMENT EXPLORATION TASKS

	Model-Free								
	Value Based				Policy Search	Actor-Critic			
RL Algorithm	**Q-learning**	**DQN**	**MAXQ**	**MC**	**PG**	**PPO**	**TRPO**	**DDPG**	**A3C**
Research Work	[88], [89], [90]	[90], [91], [92], [93], [94], [95]	[96], [97]	[98]	[99]	[100], [101]	[102]	[93], [103]	[90]

exploration strategies is that most methods do not consider the model's uncertainties. In order to develop a resilient exploration solution, Fan *et al.* [82] explicitly modeled the data uncertainty into the SAC network. However, precise representation of the uncertainties remains a challenge.

Although most works validate the proposed techniques only through simulation tests, the number of experiments in real robots has increased in the last years. In [76], experiments were performed in a real unknown cluttered urban SAR (USAR)-like scene with $15 \times 15\ m^2$, which was 97% covered by the agent. The map was acquired by visiting 44 frontier locations in 835 seconds, considering that the computation time to choose a frontier location was 1.2 s. Experiments in real indoor environments for map acquisition were also performed in [67]. The proposed method achieved lower exploration rates than frontier-based methods, but a higher exploration efficiency with shorter path lengths. In [62], a UAV performed goal-guided navigation in a closed room, discretized as a 5×5 board with static objects. Other works also perform experiment with real robots in static environments, such as [78], [81], [84] and [63]. The only experiment in a real dynamic environment is performed in [83], using an office with obstacles, tight turns, and dynamic human subjects. The proposed method achieved competitive results compared to the widely used ROS move-base planner.

It is possible to conclude that the single mobile robot exploration field combined with RL techniques has been evolving and achieving significant progress in recent years. This advancement has been driven mainly by RL union with deep learning techniques, which can handle the complex states and action spaces inherent to real-world applications. However, there are several remaining challenges. More tests considering dynamic environments and uncertainties such as noisy sensor measurements are needed to scale the solutions to real-world applications. In some cases, the performance improvement is not significant enough to justify the RL method's choice over simpler classic methods, considering the long training periods and high computational and memory costs. As highlighted in [84], another critical remaining issue is the lack of solutions capable of generalization between environments and targets. Furthermore, some possible future investigations include testing strategies with policy-based RL algorithms and testing other methods to balance exploration and exploitation besides ϵ-greedy.

VII. MULTIROBOT EXPLORATION

MRSs can be defined as a group of two or more robots that are able to cooperate or compete with each other in order to achieve a specified goal [104]. Research on MRS has attracted considerable attention in the past years, driven by its advantageous characteristics such as high levels of fault tolerance, increased efficiency in task accomplishment, situational awareness from multiple locations, greater flexibility in operations, and distributed payloads. This set of features make MRS suitable for several complex and important applications [105], including unknown environment exploration. However, the multirobot teams' success relies on the agents' autonomy skills and on the design of a proper cooperation strategy, which is a nontrivial task and represents the core challenge for multirobot cooperation viability. In this context, RL algorithms are being highlighted as a promising alternative to compose MRS coordination strategies. This section contains an overview of the most recent academic works that propose MRS cooperation strategies for unknown environment exploration using RL techniques.

The analyzed research works were recently published and encompass UAVs, UGVs, and AUVs applications. Some commonly identified nonguided MRS goals are collectively exploring or covering entire unknown areas and mapping a strange environment as soon as possible, which are suitable skills for applications such as exploring cluttered USAR scenes and uncertain environment patrolling. On the other hand, usual goal-guided applications include goal-driven map-less navigation through unknown complex environments and searching static or dynamic multitargets in unknown environments. The target searching goal was investigated in contexts such as underwater environments, victims identification in USAR scenes, and pursuit-evasion game, where a group of predator agents are trained to capture the prey agents cooperatively. Collision avoidance, either with objects or between agents, is a highlighted concern of all nonguided and goal-guided applications.

A. RL Algorithms in the Exploration Strategies

Table VI contains a summary of the most common RL algorithms employed in MRS cooperative exploration strategies. Similar to single-agent applications, due to the nature of exploration tasks, all the algorithms are model-free. Deep Q-learning methods are the most widely used, particularly DQN, and the majority of works employ value-based methods. As will be further discussed, this impacts the agent state and action spaces, since discretization or function approximation is required. Similar to the single-agent methods, the number of actor–critic algorithms is considerable. Only one work that employed policy search algorithm was identified.

Tables VII and VIII summarize how the identified RL algorithms compose, respectively, the so far proposed two-stage and end-to-end MRS exploration strategies. Three approaches were identified in two-stage strategies. First, there are works that use RL to decide where each agent should move next,

TABLE VII
HOW RL ALGORITHMS COMPOSE TWO-STAGE MRS EXPLORATION STRATEGIES

		Application Goal	**Research Work**	**RL Algorithm**	**DRL**	**State Space**	**Action Space**	**Reward**	**Complementary Method**
2-Stage Cooperation Strategies	RL algorithm to decide **where to go**	Non-Guided	[95]	DQN	Yes*	Discrete	Discrete	Not Specified	Any path planning algorithm that suits graph search
			[98]	Monte Carlo (DGSMCP)	No	Discrete	Discrete	Sparse	
	RL algorithm to decide **how to go**	Non-Guided	[103]	DDPG	Yes	Continuous	Continuous	Dense	Voronoi-based target selection
		Goal-Guided	[102]	TRPO	Yes	Continuous	Discrete	Dense	POMD solver
			[97]	MAXQ	No	Discrete	Discrete	Not Specified	Hungarian Method
	RL algorithm for **both decisions**	Goal-Guided	[93]	Where: DQN	Yes*	Discrete	Discrete	Dense	———
				How: DDPG	Yes	Continuous	Continuous		
			[90]	Where: A3C	Yes*	Discrete	Discrete	Dense	———
				How: DQN	Yes*	Continuous	Discrete		

* Neural Netowrks with Convolutional Layers.

TABLE VIII
HOW RL ALGORITHMS COMPOSE END-TO-END MRS EXPLORATION STRATEGIES

	Application Goal	**Research Work**	**RL Algorithm**	**DRL**	**State Space**	**Action Space**	**Reward**
End-to-End Cooperation Strategies	Non-Guided	**[101]**	PPO	Yes*	Continuous	Discrete	Dense
		[99]	PG	Yes	Discrete	Discrete	Dense
		[94]	DQN	Yes*	Discrete	Discrete	Dense
		[89]	Q-Learning	No	Discrete	Discrete	Dense
	Goal-Guided	**[88]**	Q-Learning	No	Discrete	Discrete	Dense
		[91]	Double DQN	Yes*	Discrete	Discrete	Sparse
		[92]	DQN	Yes*	Discrete	Discrete	Sparse
		[106]	TD Actor-Critic	Yes	Discrete	Discrete	Sparse
		[100]	PPO	Yes*	Continuous	Continuous	Dense

* Neural Netowrks with Convolutional Layers.

and employ a complementary method to perform the path planning between the current position and goal destination. Both Luo *et al.* [95] and Zhou *et al.* [98] proposed a coordination strategy for nonguided applications employing topological maps, in which the RL algorithm determines the next nodes the agents should move to. As the goal of the RL is to decide between a finite set of possible locations, the action spaces are inherently discrete. Again, a benefit of this approach is the employment of RL in the key decision-making process of exploration, while using a discrete action space. However, path planning does not focus on efficient area coverage.

When it comes to RL usage for guiding the agents toward selected locations, the proposed strategies are more diverse. In [103], each robot is assigned a different target location based on dynamic Voronoi partitions. As DDPG can handle continuous action spaces, it is used to decide the linear and angular velocities of the robots, increasing behavior possibilities and enabling smoother movements. With a focus on target searching, in [102] a POMDP solver defines if the agent should explore its current node or for which node it should move next, and in [97] the Hungarian method is used to obtain the best arrangement of cooperative subtasks and distribute it between the robots. In both works, the RL algorithm receives the next goal location and a set of sensor measurements, and returns the agents movement. Unlike [103], the action spaces are discrete and composed of 3 and 4 movements possibilities, respectively.

Finally, there are works that use two different RL algorithms to assign where to go and how to reach the defined locations. As DQN is more suited to discrete action spaces, in [93] it is used to indicate the coordinates of the next location, and DDPG is used to guide the agents selecting rotation angles in continuous action space. In [90], the traditional frontier exploration method is combined with DRL, where the A3C algorithm decides the next goal frontier and the DQN defines the movements to guide the agents. This approach results in smart path planning and cooperation strategies. However, the use of two deep neural networks can represent high computational costs that must be considered, especially in applications with energy, area, and computational constraints.

As previously defined, end-to-end systems are an integrated approach, taking raw and/or processed sensors measurements as inputs and returning the robots' control actions. The state spaces of the MRS end-to-end exploration strategies are in general a combination of some or all of the following parameters: the robot's pose, the sensor measurements union, the location of the other agents, and the known map. From Table VIII, it is possible to notice that all methods with continuous state spaces use neural networks with convolutional layers. These layers are able to extract features from the inputs, and are connected to a fully connected neural network that returns each agent actions. As for the action space, the great majority of research works employ a discrete space, defining, in average, between 3 and 9 possible moving directions or spinning angles. At the same time that this discretization makes the training time decrease and facilitates the algorithms convergence, it was repeatedly pointed as a limitation for the MRS performance, as it causes abrupt behaviors and tends to make the solutions less effective in realistic complex scenarios. Similar to single-robot end-to-end strategies, on-board real-time computing to control the vehicles' movements must be considered as a project requirement, which can limit its application.

B. Approaching Common Problems

1) Exploration-Exploitation Dilemma: Like the single-agent exploration works, ϵ-greedy is the most adopted method to balance exploration and exploitation during the learning process [89]–[91], [93], [95], [96]. In [90], a switching strategy based on collision risk is also used to choose the actions, in combination with a self-decay probability to smooth the switch. Alternatively, the Boltzmann distribution mechanism is used in [88] to determine the probability of choosing one action considering the current state. Given that many solutions to this dilemma have recently been proposed and proved successful in different applications [107]–[109], it is possible to conclude that there is space to investigate different methods in the robotic exploration context.

2) Curse of Dimensionality: To avoid the curse of dimensionality, one approach is to employ RL, especially the value-based algorithms, only for tasks that inherently have a limited and discrete set of states and actions, such as choosing for which node the agent should move next [93], [95], [98]. A widely used strategy is to efficiently reduce the space representation. In [89], FSR approximation maps the original Q-values to a low-dimension parameter vector. As already discussed, DRL solutions can be employed to learn the low-dimensional state features of the high-dimensional state from the sensory data, and provide robust function approximation. Due to its great performance improvements in recent years, CNNs are being increasingly adopted as feature extractors of raw images or other high-dimensional sensor data [90]–[92], [95], [100], [101].

3) Reward Shaping: When it comes to reward shaping, the approach of the MRS works is very similar to the identified in single-agent research. Heuristic functions are by far the most adopted approach, being designed in accordance with the application objective. For example, in goal-guided tasks it is usual to give a positive reward when an agent finds a target, as well as giving a positive reward for finding new unexplored areas in nonguided tasks. For both kinds of multirobot tasks, it is common to apply a negative reward in case of collision between agents or with objects and to specify rewards for maintaining connectivity or learning to cooperate with other agents. Again, the most common approach is employing dense rewards, probably because it usually accelerates the agent's comprehension of how it should behave.

4) Learning Convergence: In order to decrease the amount of time spent on training and to improve learning effectiveness, some MRS works make use of transfer learning techniques. One example is the *curriculum learning* (CL) techniques, in which the agents learn from increasingly difficult scenarios to progressively acquire complex skills. In [101], CL is applied to simplify and direct the learning process in a teacher–student fashion, exposing the agents to four different environments with different complexity or difficulty. The strategy is tested with and without CL, and only succeed when the technique is applied. In [99], new obstacles are gradually introduced to the training environment, stimulating the agents to explore with the smallest number of collisions when faced with different kinds of dynamic environments. A similar approach is used in [93], in which a target selection policy is pre-trained in obstacle-free environments, and then new obstacles are added during training. This technique makes the algorithm converge much faster than classic multiagent DDPG. In [91], the agents trained in sparse environments are able to quickly adapt to clutter scenarios. Some methods use *EP* to improve data efficiency. Both Venturini *et al.* [91] and Liu *et al.* [96] enable experience sharing between different robots, which allows faster convergence of the learning algorithms. Prioritized EP is proposed in [103], which samples important experience data more frequently by calculating the priority of each state transition.

TABLE IX

SOFTWARE TOOLS USED FOR TRAINING AND TESTING MRS EXPLORATION STRATEGIES

	Name	**Used in**
Physics Simulation Engine	Gazebo	[103] [102] [94] [106]
	Matlab	[88] [89]
	Unity	[101]
	MazeBase	[99]
	MAgent	[92]
Platforms and Toolkits for RL and DLR algorithm development	OpenAI Gym	[94]
	TensorFlow	[101]
	Matlab	[88] [89]
Framework for robot software development	ROS	[95] [103] [102] [94] [106]

C. Simulation and Real-World Experiments

Table IX summarizes commonly adopted software tools for both MRS training and testing experiments. It is possible to note that there is no unanimity for the employed simulation environments. This makes comparisons between strategies more difficult, once the simulators do not model the world physical aspects in the same way, and the world model directly interferes in the agents learned behavior. In the perspective of adopting standardized tools for scaling both single and multiagent RL solutions, Gazebo [87] can be highlighted as an often-used simulator with a growing user community. Similar to the single-agent analysis, ROS is established as the most used framework for robot software development. OpenAI Gym, TensorFlow, and MATLAB are employed for RL algorithm development.

To evaluate the learning process performance, some works adopt other RL algorithms as baselines for comparison. Commonly employed evaluation metrics for the training phase are learning convergence speed, mean cumulative rewards, and learning efficiency. For example, in [103] it is demonstrated that the proposed method requires 25% of the training time required by classic DDPG. In [93], classic multiagents DDPG is also used as baseline, and the mean cumulative reward is

TABLE X
RELATION OF MRS WORKS THAT PERFORM AND DO NOT PERFORM TESTS IN DYNAMIC ENVIRONMENTS

Tests in dynamic environments	
Yes	**No**
[103] [99] [90]	[88] [101] [95] [102] [94]
[92] [106] [98]	[93] [89] [96] [97] [91] [100]

higher in the proposed method. When it comes to the evaluation of the cooperation strategies performance in simulated tests, the adopted evaluation metrics can be summarized to the average moving distance (AMD), average mean reward, number of collisions, success rate, connectivity, and travel time, not necessarily with these specific names. Works that focus on nonguided tasks can also evaluate exploration ratio and speed, and the most common baselines are classic frontier-based methods [95], [99], [101]. For works focused on goal-guided tasks, the particle swarm optimization algorithm is the most common baseline [88], [90], and number of identified targets and average time to find them are usually evaluated.

Most works validate the strategy performance only through simulation, and not using real robots. The works that perform real-world tests generally evaluate whether the robot team is able to accomplish the goal task successfully or not, and the path taken by the agents. In [100], three ground robots navigate through different environments with static obstacles to reach a goal position. Each robot has an onboard computer (Nvidia Jetson TX2). Two underwater vehicles find two targets in a 10 m × 20 m pool with and without obstacles in [90], successfully avoiding collisions. In [89], two UAVs cover a part of the environment without overlap, and in [103] each robot have a Raspberry Pi 3 receiving control commands from a host computer with a GPU in order to successfully explore a dynamic room environment. In [106], experiments are performed with 2, 3, 4, and 5 ground robots, that are able to safely reach their goal positions without any collision. A great aspect of the latter work is that the policy trained in simulation is successfully applied to the real robots without any calibration or retraining.

Considering the difficulty in transferring behavior learned through simulation to real-world scenarios, especially to a team of robots, the aforementioned results illustrate the great advances of multirobot exploration using RL. However, there are still unresolved questions. Depending on a central host to send control actions may not be the best solution to a team of robots that can eventually lose communication. The end-to-end method tested in real world used computationally heavy CNNs, and had success by using on-board computers with GPUs in each robot. In ground robots that can be viable, but for applications with computational and energy constraints like, for example, lightweight and low-energy UAVs, it is safe to assume that this approach is not the best fit. These aspects of the works are not necessarily problems, but reinforce how the robotic RL solutions are still very heterogeneous and not directly suitable to different systems, even when the application is very similar or equal.

As aforementioned, direct comparisons between the proposed strategies are probably inaccurate because of the differences in the employed simulation environments, kinds of robots, computer hardware, and other project choices. However, it is possible to state that, in general, the strategies that use RL make more intelligent decisions than classic multirobot exploration techniques, which leads to faster and more efficient explorations. By letting the agents automatically learn to cooperate instead of explicitly designing these behaviors has resulted in successful complex team dynamics, which may include concerns about maintaining connectivity, avoiding collisions with obstacles and other agents, avoiding that the same area is explored by different agents multiple times, among others.

At the same time, it is clear that this research field is still in its early steps. Among the current limitations, there is a small number of works that explore continuous action spaces, which are more appropriate to robotic path planning in realistic complex scenarios. Also, the problems of sim-to-real transfer are still a reality. The majority of works are tested only in static environments, as demonstrated in Table X. Most works that only use simulation validation do not consider real-world constraints, like noisy sensor measurements or the possibility of losing communication with other agents, which raises the question of whether these methods would be effective in a real environment. Therefore, further research is needed to find multirobot exploration solutions with RL that are largely scalable to real MRSs.

VIII. DISCUSSIONS

From the review of single-robot and multirobots exploration strategies using RL techniques presented in the previous sections, some overall discussion topics can be raised:

1) From the authors' point-of-view, the employment of RL is still more justifiable in MRS coordinated exploration than in single-robots exploration. This happens because the complexity of MRS is inherently high, and RL enables team dynamics that are really difficult to design. For a relevant number of the analyzed single-robot works, the performance improvement is not significant enough to justify the RL method's choice over simpler classic methods, considering the long training periods and high computational costs.
2) The ϵ-greedy algorithm for balancing between exploration and exploitation during the learning process was almost unanimity in the researched works. Simultaneously, there are a large body of research pertaining to this problem [107], [108]. An important example is the "First return, then explore" [109] that, to the best of our knowledge, has not been applied in the context of robotic exploration. Since the approach employed to handle this dilemma directly impacts the learning efficiency, investigating different alternatives could improve the learning process.
3) The greatest identified field trend is combining DRL with CNN to design end-to-end solutions. The network receives large state spaces (usually images or other raw sensor measurements), and directly returns the agent's

movement control. Diverse works demonstrated successful experiments with this kind of system. However, it may not be the best fit for applications with weight, energy, and computational constraints, given the need to control the vehicle reactively through on-board real-time computing. The limited generalization capabilities of end-to-end strategies are also a remaining issue.

4) There is still space for works that investigate robotic exploration with RL using continuous action spaces. Discrete action spaces were largely adopted in the reviewed solutions, and they facilitate learning convergence. However, this discretization can limit the robots' performance, cause abrupt behaviors and make the solutions less effective in real and complex scenarios.
5) Transfer learning techniques were applied in greater quantity in MRS than in single-robot systems but is still not a widespread approach in the targeted application. Both types of systems can benefit from further study of different transfer learning techniques in the robotic exploration context. To properly quantify the impact of methods used for improving learning convergence and perform accurate comparisons, it is important to use methods to explicitly analyze the convergence in the RL algorithms, like the ones proposed in [110] and [111].
6) An essential next step of the research field is an increase in the number of strategies designed to consider real-world uncertainties, like noisy sensor measurements and dynamic environments. The lack of these considerations separates academic solutions from real applications.

IX. Conclusion

This survey reviewed the most recently published works that tackle the mobile robot exploration problem using RL techniques. Both single and multirobot solutions and challenges were compared and examined. Discussion topics about the current research status and possible future steps were presented. Much progress has been made, but it is clear that this research field is still in its early steps. Further research is needed to find single and multirobot exploration solutions with RL that are vastly scalable to real-world applications. The presented literature review and the provided discussions around it tried to shed light in directions for possible future works in this area.

References

[1] B. Krogh and C. Thorpe, "Integrated path planning and dynamic steering control for autonomous vehicles," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, Apr. 1986, pp. 1664–1669.

[2] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom. Towards New Comput. Princ. Robot. Autom. (CIRA)*, Jul. 1997, pp. 146–151.

[3] S. Tzafestas, *Introduction to Mobile Robot Control*. Amsterdam, The Netherlands: Elsevier, Oct. 2013.

[4] S. Li, J. Yan, and L. Li, "Automated guided vehicle: The direction of intelligent logistics," in *Proc. IEEE Int. Conf. Service Oper. Logistics, Informat. (SOLI)*, 2018, pp. 250–255.

[5] E. Garcia, M. A. Jimenez, P. G. D. Santos, and M. Armada, "The evolution of robotics research," *IEEE Robot. Autom. Mag.*, vol. 14, no. 1, pp. 90–103, Mar. 2007.

[6] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics* (Intelligent Robotics and Autonomous Agents). Cambridge, MA, USA: MIT Press, 2005.

[7] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Holland, MI, USA: Bradford Company, 2004.

[8] A. A. Makarenko, S. B. Williams, F. Bourgault, and H. F. Durrant-Whyte, "An experiment in integrated exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 1, Oct. 2002, pp. 534–539.

[9] R. C. Arkin and R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA, USA: MIT Press, 1998.

[10] M. Chupeau, O. Bénichou, and R. Voituriez, "Cover times of random searches," *Nature Phys.*, vol. 11, no. 10, pp. 844–847, Oct. 2015.

[11] R. Brooks, "A hardware retargetable distributed layered architecture for mobile robot control," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 4, Mar. 1987, pp. 106–110.

[12] S. Park and K. S. Roh, "Coarse-to-fine localization for a mobile robot based on place learning with a 2-D range scan," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 528–544, Jun. 2016.

[13] A. Solanas and M. A. Garcia, "Coordinated multi-robot exploration through unsupervised clustering of unknown space," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 1, Sep. 2004, pp. 717–721.

[14] B. Kaleci, C. M. Senler, O. Parlaktuna, and U. Gurel, "Constructing topological map from metric map using spectral clustering," in *Proc. IEEE 27th Int. Conf. Tools Artif. Intell. (ICTAI)*, Nov. 2015, pp. 139–145.

[15] M. Rappaport and C. Bettstetter, "Coordinated recharging of mobile robots during exploration," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 6809–6816.

[16] C. Wang, H. Ma, W. Chen, L. Liu, and M. Q.-H. Meng, "Efficient autonomous exploration with incrementally built topological map in 3-D environments," *IEEE Trans. Instrum. Meas.*, vol. 69, no. 12, pp. 9853–9865, Dec. 2020.

[17] R. R. Murphy, *Introduction to AI Robotics*, 1st ed. Cambridge, MA, USA: MIT Press, 2000.

[18] S. M. LaValle *et al.*, "Rapidly-exploring random trees: A new tool for path planning," Iowa State Univ., Ames, IA, USA, Tech. Rep. 98-11, 1998.

[19] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles*. Boston, MA, USA: Springer, 1997, pp. 203–220.

[20] M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," in *Proc. SMC Conf. IEEE Int. Conf. Syst., Man Cybern. Cybern. Evolving Syst., Humans Complex Interact.*, vol. 3, Oct. 2000, pp. 2316–2320.

[21] M. W. Otte, "A survey of machine learning approaches to robotic path-planning," Univ. Colorado Boulder, Boulder, CO, USA, Tech. Rep., 2015. [Online]. Available: http://ottelab.com/html_stuff/pdf_files/Otte.prelim08.pdf

[22] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[23] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.

[24] M. T. Spaan, "Partially observable Markov decision processes," in *Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 387–414.

[25] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," 2020, *arXiv:2006.16712*.

[26] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *J. Intell. Robot. Syst. Theory Appl.*, vol. 86, no. 2, pp. 153–173, May 2017.

[27] M. Sewak, *Deep Reinforcement Learning*. Singapore: Springer, 2019.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992.

[29] J. A. Bagnell, S. Kakade, A. Y. Ng, and J. Schneider, "Policy search by dynamic programming," in *Proc. 16th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Cambridge, MA, USA: MIT Press, 2003, pp. 831–838.

[30] E. Theodorou, J. Buchli, and S. Schaal, "Learning policy improvements with path integrals," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 828–835.

[31] F. Stulp and O. Sigaud, "Policy improvement methods: Between black-box optimization and episodic reinforcement learning," Robot. Comput. Vis., Paris, France, Tech. Rep., 2012, p. 34.

[32] C. J. C. H. Watkins, "Learning from delayed rewards," Ph.D. dissertation, Dept. King's College, Cambridge U.K., 1989.

[33] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*.

[34] T. G. Dietterich, "Hierarchical reinforcement learning with the MAXQ value function decomposition," *J. Artif. Intell. Res.*, vol. 13, pp. 227–303, Nov. 2000.

[35] B. Bouzy and G. Chaslot, "Monte-carlo go reinforcement learning experiments," in *Proc. IEEE Symp. Comput. Intell. Games*, May 2006, pp. 187–194.

[36] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," Cambridge Univ. Eng. Dept., Cambridge, U.K., Tech. Rep. CUED/F-INFENG-TR 166, 1994.

[37] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[38] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[39] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, *arXiv:1707.06347*.

[40] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1889–1897.

[41] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

[42] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[43] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.

[44] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013.

[45] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," 2019, *arXiv:1904.12901*.

[46] N. Venkat, "The curse of dimensionality: Inside out," Ph.D. dissertation, Dept. CSIS, BITS Pilani, Pilani, India, 2018.

[47] R. Akrour, F. Veiga, J. Peters, and G. Neumann, "Regularizing reinforcement learning with state abstraction," in *Proc. IEEE Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2018, pp. 534–539.

[48] T. Arai, Y. Toda, I. Mutsumi, S. Shao, R. Tonomura, and N. Kubota, "Reinforcement learning based on state space model using growing neural gas for a mobile robot," in *Proc. 10th Int. Conf. Soft Comput. Intell. Syst. (SCIS), 19th Int. Symp. Adv. Intell. Syst. (ISIS)*, Dec. 2018, pp. 1410–1413.

[49] J. Lim, S. Ha, and J. Choi, "Prediction of reward functions for deep reinforcement learning via Gaussian process regression," *IEEE/ASME Trans. Mechatronics*, vol. 25, no. 4, pp. 1739–1746, Aug. 2020.

[50] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Trans. Intell. Technol.*, vol. 5, no. 3, pp. 177–183, Sep. 2020.

[51] O. Tutsoy, "CPG based RL algorithm learns to control of a humanoid robot leg," *Int. J. Robot. Autom.*, vol. 30, no. 2, pp. 1–7, 2015.

[52] O. Tutsoy, D. E. Barkana, and S. Colak, "Learning to balance an NAO robot using reinforcement learning with symbolic inverse kinematic," *Trans. Inst. Meas. Control*, vol. 39, no. 11, pp. 1735–1748, Apr. 2017.

[53] E. Chalmers, E. B. Contreras, B. Robertson, A. Luczak, and A. Gruber, "Learning to predict consequences as a method of knowledge transfer in reinforcement learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2259–2270, Jun. 2018.

[54] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 6292–6299.

[55] T. Shimizu, R. Saegusa, S. Ikemoto, H. Ishiguro, and G. Metta, "Robust sensorimotor representation to physical interaction changes in humanoid motion learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 5, pp. 1035–1047, May 2015.

[56] K. Hirata, H. Iizuka, and M. Yamamoto, "Reinforcement learning method with internal world model training," in *Proc. IEEE/SICE Int. Symp. Syst. Integr. (SII)*, Jan. 2020, pp. 201–204.

[57] T. D. Le, A. T. Le, and D. T. Nguyen, "Model-based Q-learning for humanoid robots," in *Proc. 18th Int. Conf. Adv. Robot. (ICAR)*, Jul. 2017, pp. 608–613.

[58] R. Pérez-Dattari, C. Celemin, J. Ruiz-del-Solar, and J. Kober, "Continuous control for high-dimensional state spaces: An interactive learning approach," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, 2019, pp. 7611–7617.

[59] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 7551–7557.

[60] Z. Yang, K. Merrick, L. Jin, and H. A. Abbass, "Hierarchical deep reinforcement learning for continuous action control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5174–5184, Nov. 2018.

[61] G. Cardona *et al.*, "Autonomous navigation for exploration of unknown environments and collision avoidance in mobile robots using reinforcement learning," in *Proc. SoutheastCon*, 2019, pp. 1–7.

[62] H. X. Pham, H. M. La, D. Feil-Seifer, and L. Van Nguyen, "Reinforcement learning for autonomous UAV navigation using function approximation," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Aug. 2018, pp. 1–6.

[63] A. Guerra, F. Guidi, D. Dardari, and P. M. Djuric, "Reinforcement learning for UAV autonomous navigation, mapping and target detection," in *Proc. IEEE/ION Position, Location Navigat. Symp. (PLANS)*, Apr. 2020, pp. 1004–1013.

[64] C. Craye, D. Filliat, and J.-F. Goudou, "RL-IAC: An exploration policy for online saliency learning on an autonomous mobile robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 4877–4884.

[65] Y. Liu, H. Liu, and B. Wang, "Autonomous exploration for mobile robot using Q-learning," in *Proc. 2nd Int. Conf. Adv. Robot. Mechatronics (ICARM)*, Aug. 2017, pp. 614–619.

[66] L. Tai and M. Liu, "Mobile robots exploration through CNN-based reinforcement learning," *Robot. Biomimetics*, vol. 3, no. 1, pp. 1–8, Dec. 2016.

[67] H. Li, Z. Qichao, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2019.

[68] R. B. Issa, M. S. Rahman, M. Das, M. Barua, and M. G. R. Alam, "Reinforcement learning based autonomous vehicle for exploration and exploitation of undiscovered track," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2020, pp. 276–281.

[69] B. Zhou, W. Wang, Z. Wang, and B. Ding, "Neural Q learning algorithm based UAV obstacle avoidance," in *Proc. IEEE CSAA Guid., Navigat. Control Conf. (CGNCC)*, Aug. 2018, pp. 1–6.

[70] Z. Yijing, Z. Zheng, Z. Xiaoyi, and L. Yang, "Q learning algorithm based UAV path learning and obstacle avoidence approach," in *Proc. 36th Chin. Control Conf. (CCC)*, Jul. 2017, pp. 3397–3402.

[71] S. Josef and A. Degani, "Deep reinforcement learning for safe local planning of a ground vehicle in unknown rough terrain," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6748–6755, Oct. 2020.

[72] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile robot navigation based on deep reinforcement learning," in *Proc. Chin. Control Decis. Conf. (CCDC)*, 2019, pp. 6174–6178.

[73] E. Cetin, C. Barrado, G. Munoz, M. Macias, and E. Pastor, "Drone navigation and avoidance of obstacles through deep reinforcement learning," in *Proc. IEEE/AIAA 38th Digit. Avionics Syst. Conf. (DASC)*, Sep. 2019, pp. 1–7.

[74] L. Tai and M. Liu, "Towards cognitive exploration through deep reinforcement learning for mobile robots," 2016, *arXiv:1610.01733*.

[75] L. Jiang, H. Huang, and Z. Ding, "Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 4, pp. 1179–1189, Jul. 2019.

[76] F. Niroui, K. Zhang, Z. Kashino, and G. Nejat, "Deep reinforcement learning robot for search and rescue applications: Exploration in unknown cluttered environments," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 610–617, Apr. 2019.

[77] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, "Deep reinforcement learning supervised autonomous exploration in office environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 7548–7555.

[78] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2393–2402, Apr. 2019.

[79] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, "Curiosity-driven exploration for mapless navigation with deep reinforcement learning," 2018, *arXiv:1804.00456*.

[80] H. Hu, S. Song, and C. L. P. Chen, "Plume tracing via model-free reinforcement learning method," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 8, pp. 2515–2527, Aug. 2019.

[81] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 31–36.

[82] T. Fan, P. Long, W. Liu, J. Pan, R. Yang, and D. Manocha, "Learning resilient behaviors for navigation under uncertainty," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 5299–5305.

[83] K. Rana, B. Talbot, V. Dasagi, M. Milford, and N. Sünderhauf, "Residual reactive navigation: Combining classical and learned navigation strategies for deployment in unknown environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2020, pp. 11493–11499.

[84] Y. Zhu *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 3357–3364.

[85] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," in *Proc. AAAI Conf. Artif. Intell.*, 2016, vol. 30, no. 1, pp. 2094–2100.

[86] D. C. Guastella and G. Muscato, "Learning-based methods of perception and navigation for ground vehicles in unstructured environments: A review," *Sensors*, vol. 21, no. 1, p. 73, Dec. 2020.

[87] N. P. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, vol. 3, Sep. 2004, pp. 2149–2154.

[88] W. Yue, X. Guan, and Y. Xi, "Reinforcement learning based approach for multi-UAV cooperative searching in unknown environments," in *Proc. Chin. Autom. Congr. (CAC)*, Nov. 2019, pp. 2018–2023.

[89] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," 2018, *arXiv:1803.07250*.

[90] X. Cao, C. Sun, and M. Yan, "Target search control of AUV in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96549–96559, 2019.

[91] F. Venturini *et al.*, "Distributed reinforcement learning for flexible UAV swarm control with transfer learning capabilities," in *Proc. 6th ACM Workshop Micro Aerial Vehicle Netw., Syst., Appl.*, Jun. 2020, pp. 1–6.

[92] C. Yu, Y. Dong, Y. Li, and Y. Chen, "Distributed multi-agent deep reinforcement learning for cooperative multi-robot pursuit," *J. Eng.*, vol. 2020, no. 13, pp. 499–504, Jul. 2020.

[93] Y. Jin, Y. Zhang, J. Yuan, and X. Zhang, "Efficient multi-agent cooperative navigation in unknown environments with interlaced deep reinforcement learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2019, pp. 2897–2901.

[94] J. A. Cruz, H. L. Cardoso, L. P. Reis, and A. Sousa, "Reinforcement learning in navigation and cooperative mapping," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2020, pp. 200–205.

[95] T. Luo, B. Subagdja, D. Wang, and A.-H. Tan, "Multi-agent collaborative exploration through graph-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Agents (ICA)*, Oct. 2019, pp. 2–7.

[96] Y. Liu, G. Nejat, and J. Vilela, "Learning to cooperate together: A semi-autonomous control architecture for multi-robot teams in urban search and rescue," in *Proc. IEEE Int. Symp. Saf., Secur., Rescue Robot. (SSRR)*, Oct. 2013, pp. 1–6.

[97] Y. Cai, S. X. Yang, and X. Xu, "A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Apr. 2013, pp. 52–59.

[98] X. Zhou, W. Wang, T. Wang, Y. Lei, and F. Zhong, "Bayesian reinforcement learning for multi-robot decentralized patrolling in uncertain environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 12, pp. 11691–11703, Dec. 2019.

[99] M. Geng, K. Xu, X. Zhou, B. Ding, H. Wang, and L. Zhang, "Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration," *Entropy*, vol. 21, no. 3, p. 294, Mar. 2019.

[100] J. Lin, X. Yang, P. Zheng, and H. Cheng, "End-to-end decentralized multi-robot navigation in unknown complex environments via deep reinforcement learning," in *Proc. IEEE Int. Conf. Mechatronics Autom. (ICMA)*, Aug. 2019, pp. 2493–2500.

[101] Z. Chen, B. Subagdja, and A.-H. Tan, "End-to-end deep reinforcement learning for multi-agent collaborative exploration," in *Proc. IEEE Int. Conf. Agents (ICA)*, Oct. 2019, pp. 99–102.

[102] O. Walker, F. Vanegas, F. Gonzalez, and S. Koenig, "Multi-UAV target-finding in simulated indoor environments using deep reinforcement learning," in *Proc. IEEE Aerosp. Conf.*, Mar. 2020, pp. 1–9.

[103] J. Hu, H. Niu, J. Carrasco, B. Lennox, and F. Arvin, "Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 14413–14423, Oct. 2020.

[104] *A Review of Research in Multi-Robot Systems*, Birla Inst. Technol. Sci., Pilani, India, Aug. 2012.

[105] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *Int. J. Adv. Robotic Syst.*, vol. 10, no. 12, p. 399, Dec. 2013.

[106] H. W. Jun, H. J. Kim, and B. H. Lee, "Goal-driven navigation for non-holonomic multi-robot system by learning collision," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 1758–1764.

[107] T. Hester, M. Lopes, and P. Stone, "Learning exploration strategies in model-based reinforcement learning," in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst.*, vol. 2, May 2013, pp. 1069–1076.

[108] R. McFarlane, *A Survey of Exploration Strategies in Reinforcement Learning*. Montreal, QC, Canada: McGill Univ., 2018.

[109] A. Ecoffet, J. Huizinga, J. Lehman, K. O. Stanley, and J. Clune, "First return, then explore," *Nature*, vol. 590, no. 7847, pp. 580–586, Feb. 2021.

[110] O. Tutsoy and M. Brown, "Chaotic dynamics and convergence analysis of temporal difference algorithms with bang-bang control," *Optim. Control Appl. Methods*, vol. 37, no. 1, pp. 108–126, Jan. 2016.

[111] O. Tutsoy and M. Brown, "An analysis of value function learning with piecewise linear control," *J. Experim. Theor. Artif. Intell.*, vol. 28, no. 3, pp. 529–545, May 2016.

Luíza Caetano Garaffa received the B.Sc. degree in electrical engineering from the Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 2019, where she is currently pursuing the M.Sc. degree in computer science.

Her current research interests include machine learning, reinforcement learning, mobile robotics, and (multi)unmanned aerial vehicles. Other research interests also include machine learning applied to biomedical engineering and hardware security.

Maik Basso received the bachelor's degree in information systems from the Federal University of Santa Maria (UFSM), Santa Maria, Brazil, in 2015, and the M.Sc. degree in electrical engineering from the Systems of Automation, Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 2018, where he is currently pursuing the Ph.D. degree in graduate program in electrical engineering.

His current research interests include embedded systems for autonomous (multi) unmanned aerial vehicles (UAVs), image processing, artificial intelligence, WEB technologies, and WEB development.

Andréa Aparecida Konzen received the M.Sc. degree in computer science from the Pontifical Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre, Brazil, in 2002, the Ph.D. degree in informatics in education from Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, in 2015, and the Ph.D. degree in computer science from the University of Santa Cruz do Sul (UNISC), Santa Cruz do Sul, Brazil.

She is currently a Post-Doctoral Researcher with the Institute of Informatics, UFRGS. She is also a Researcher in artificial intelligence—machine learning, focusing on the development of navigation algorithms and autonomous exploration, and fusion of maps based on machine learning techniques.

Edison Pignaton de Freitas received the Ph.D. degree in computer engineering from the Military Institute of Engineering, Rio de Janeiro, Brazil, in 2003, the M.Sc. degree in computer science from Federal University of Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 2007, and the Ph.D. degree in computer science and engineering from Halmstad University, Halmstad, Sweden, in 2011.

He is currently an Associate Professor with UFRGS, acting in the Graduate Programs on Computer Science (PPGC) and Electrical Engineering (PPGEE), where he is developing research computer networks, real-time systems, and (multi)unmanned aerial vehicles.