Received January 11, 2021, accepted January 19, 2021, date of publication January 21, 2021, date of current version January 29, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3053348

A Multiagent Deep Reinforcement Learning Approach for Path Planning in Autonomous Surface Vehicles: The Ypacaraí Lake Patrolling Case

SAMUEL YANES LUIS, DANIEL GUTIÉRREZ REINA, AND SERGIO L. TORAL MARÍN
Department of Electronic Engineering, Technical School of Engineering of Seville, 41092 Seville, Spain

Corresponding author: Samuel Yanes Luis (syanes@us.es)

This work was supported in part by the Universidad de Sevilla through the Contract ''Contratos de acceso al Sistema Español de Ciencia, Tecnología e Innovación para el desarrollo del programa propio de I+D+i de la Universidad de Sevilla'', in part by the Spanish ''Ministerio de Ciencia, Innovación y Universidades, Programa Estatal de I+D+i Orientada a los Retos de la Sociedad'' through the Project ''Despliegue Adaptativo de Vehículos no Tripulados para Gestión Ambiental en Escenarios Dinámicos" under Grant RTI2018-098964-B-I00, in part by the regional government Junta de Andalucía through the Project ''Despliegue Inteligente de una red de Vehículos Acuáticos no Tripulados para la monitorización de Recursos Hídricos'' under Grant US-1257508, and in part by the regional government Junta de Andalucía through the Project "Despliegue y Control de una Red Inteligente de Vehículos Autónomos Acuáticos para la Monitorización de Recursos Hídricos Andaluces" under Grant PY18-RE0009.

ABSTRACT Autonomous surfaces vehicles (ASVs) excel at monitoring and measuring aquatic nutrients due to their autonomy, mobility, and relatively low cost. When planning paths for such vehicles, the task of patrolling with multiple agents is usually addressed with heuristics approaches, such as Reinforcement Learning (RL), because of the complexity and high dimensionality of the problem. Not only do efficient paths have to be designed, but addressing disturbances in movement or the battery's performance is mandatory. For this multiagent patrolling task, the proposed approach is based on a centralized Convolutional Deep Q-Network, designed with a final independent dense layer for every agent to deal with scalability, with the hypothesis/assumption that every agent has the same properties and capabilities. For this purpose, a tailored reward function is created which penalizes illegal actions (such as collisions) and rewards visiting idle cells (cells that remains unvisited for a long time). A comparison with various multiagent Reinforcement Learning (MARL) algorithms has been done (Independent Q-Learning, Dueling Q-Network and multiagent Double Deep Q-Learning) in a case-study scenario like the Ypacaraí lake in Asunción (Paraguay). The training results in multiagent policy leads to an average improvement of 15% compared to lawn mower trajectories and a 6% improvement over the IDQL for the case-study considered. When evaluating the training speed, the proposed approach runs three times faster than the independent algorithm.

INDEX TERMS Deep reinforcement learning, multiagent learning, monitoring, path planning, autonomous surface vehicle, patrolling.

I. INTRODUCTION

Ypacaraí Lake is the largest body of water in Paraguay with more than 60 km^2 of navigable surface. It is located between the cities of San Bernardino (eastwards), Areguá (westwards), and Ypacaraí (southwards). It is the main source of water in the area. Recently, it has become a tourist attraction for swimming and other water sports. The lake's health is also important for the ecosystem in the wetlands of the surrounding basin. However, in the past 40 years, the continuous expansion of the agriculture around the lake, the lack of sewerage systems in the neighboring cities, and the disposals of wastes from industries located at the shore, among other factors, have caused eutrophication to occur in the lake. This unnatural process caused a green-blue algae (cyanobacteria) bloom, resulting in both heightened levels of toxins, harmful for both humans and fauna, and repulsive odor in the area [1].

The associate editor coordinating the review of this manuscript and approving it for publication was Youqing Wang.

Among the many actions carried out to improve the lake's situation, it is vital to monitor efficiently its environmental state in order to have an updated image of the biological status of algae blooms. This contamination map allows to analyze the performance of the environmental measures taken by the authorities and researchers. However, the manual monitoring task takes a lot of effort and human resources since it requires constant travels from the shore to the main blooms with motor boats and manual sampling of waters. In [2] the use of autonomous surface vehicles (ASVs) equipped with water quality sensors is proposed to substitute the manual sampling since they allow an automated sampling and surveillance of the water quality at a relative low cost. Most ASVs have the ability to operate via remote control (RC) or autonomously by travelling along a path of waypoints without any human crew. This means the energy consumption of the ASV could be decreased by using small electric vehicles. For a better efficiency, it is convenient to deploy a fleet of these ASVs so that each individual agent explores different zones and measures the quality of the water. This task requires coordination between the unmanned boats to avoid collisions and share the exploration zones according to some efficiency criteria. Depending on the size of the fleet, the problem could easily became too complex to handle. In this kind of problems, usually NP-hard, it is mandatory to look for Artificial Intelligence approaches that can handle both the scalability and the high dimension of this challenge [3].

The surveillance and patrolling task is focused in the useful redundancy of the information acquired along the path. Since there are several important magnitudes to monitor related to water quality (pH level, dissolved oxygen, etc) and these can change with time of the day and the year, the resulting paths generated by any suitable approach must take in consideration not only the different zones (with different levels of importance) but also the revisiting of those unsampled for a long time [4]. This problem, described later as the Patrolling Problem, requires a reactive, adaptive and coordinated approach especially in the multiagent case, due to the many possible solutions.

This work will address this challenge by using the trending Deep Reinforcement Learning techniques within a Fully Observable Markov Decision Problem framework (FOMDP). This mathematical formulation, in addition to parametric function approximators (such as Deep Neural Networks (DNN) for the Action-State function $Q(s, a)$ estimation), allows for an improvement without any previous model of the problem (off-model algorithms) [5]. For this particular multiagent case, it is relevant to consider also the scalability issue, since this particular scenario has a big extension, and multiple ASVs will be needed to monitor the whole area in a feasible amount of time. The multiagent case implies a much higher complexity in the planification, because it needs a path planning optimization and consideration of the possible interaction between the vehicles, as in [6], where the collisions of the vehicles are acknowledged. In addition to the interactions, the action space of the problem is significant and the number of different possibilities and the computational complexity to obtain the optimal result could easily become impossible to handle [7].

This problem, known as the *dimensionality curse* [8], is currently addressed by optimization algorithms, such as Evolutionary Computation (EC) or Reinforcement Learning (RL), due to its well-known capacity of solving high-dimensional problems [9], [10]. In this paper, the scalability in the learning is achieved, combined with a centralized Convolutional Neural Network, and an independent execution layer for every agent. This approach not only brings a method for the ASVs to learn from others experiences (as the agents capacities are equivalent), but also provides a reactive algorithm able to improve its own policy in response to possible on-line learning and reactivity to changes in the environment (changes in the fleet size, for example). As the EC algorithm uses a black-box formulation and learns only from the final episodic reward [11], DRL stands as a very convenient approach to generalize an optimal policy using only a RGB scenario representation [12].

As a matter of fact, this visual way of learning with a tailored reward function to evaluate every action taken by the ASVs allows for easier formulation of the objectives by imposing the rewards in terms of *what is considered acceptable behavior and what is not*. This work also proposes a state definition, which includes graphically temporally-dependent situations, such as how long it has been since any agent visited a particular zone as a method to keep the Markovian assumption true. Another novel proposal of this work is in the common-experience based optimization sustained in the homogeneous agents feature to address the training efficiency and scalability of the fleet. As the observation of every agents are interchangeable, every agent can take advantage of others experiences.

The main contributions of this paper are:

- A novel method for a multiagent path planner based on Deep Reinforcement Learning that faces the scalability issue in the context of the Ypacaraí Lake non-homogeneous patrolling.
- A comparison between commonly-used multiagent DRL (MADRL) approaches and a learning performance analysis (learning time).

This paper is structured as follows: Section II contains a brief analysis of the state of the art in Deep Reinforcement Learning (DRL) and multiagent Reinforcement Learning (MARL), where the advantages and drawbacks of the recently published methods are analyzed in the context of the problem. Section III formally describes the Ypacaraí monitoring problem, the scenario assumption (movement, simulation, batteries, etc), and the two proposed DRL approaches. In Section IV, the results of the proposed methods are discussed, and a comparison between our approach and others is done. Finally, Section V details the conclusion of the paper.

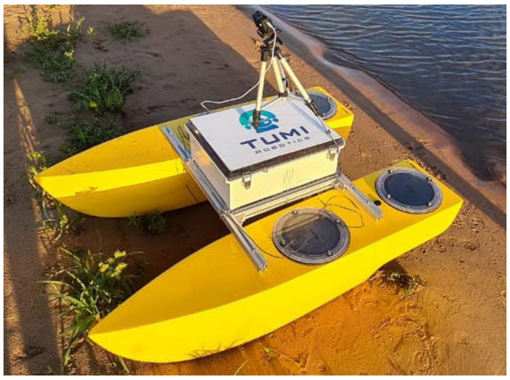
FIGURE 1. Prototype of ASV for the monitoring ask of Ypacaraí Lake.

II. RELATED WORK

The use of ASVs for monitoring endangered water resources is trending nowadays [3], [20] and is an efficient solution for automated coverage missions. In [2], an ASV is designed and proposed for monitoring the Ypacaraí Lake (see Fig. 1) using a set of water quality sensors such as PH-meter, Oxide-Potential Reduction sensor, thermometer, etc. The use of ASVs entails the need of motion planification and efficient control, since the limitation of the vehicle batteries requires finding the best trajectories to accomplish the monitoring task. This divides the motion planning into two main tasks [21]: local planification and global planification. In [22], a comparison between local planification algorithms has been done in the Ypacaraí environment and constitutes the lower level of planification which every ASV has, i.e., the local trajectory generator and obstacle avoidance with $A*$ or *RRT* techniques. The other challenge is the global path planning: generating a set of waypoints the ASV must visit in order to complete the monitoring. These waypoints must follow an efficiency criteria like in [23], where the focus is on minimizing the uncertainty within a surrogate model in order to find the maximum value.

The high dimensionality of the state-action domain from the many possible trajectories the ASVs can take is a computational and optimization challenge. The Evolutionary Computation algorithms provide a black-box approach to solve this problem, nonetheless they are non-reactive solutions of a poor sample efficiency [5], [11] and every change of the constraints requires a whole recalculation of the optimal trajectory. On the contrary, Deep Reinforcement Learning approaches usually can generalize the policy from the state thanks to the multiple previous experiences [24].

Path planning with RL tabular methods such as Q-Learning or Policy Iteration [5], suffers from dimensionality problems and a lack of generalization. Thus, Deep Reinforcement Learning (DRL) has become the most common way to deal with the function approximation (neural networks as non-linear parametric approximators). As a matter of fact, DRL has been recently one of the most common approaches for reactive path planning and collision avoidance with autonomous surface and underwater devices. In [25], for example, it is proposed a method for optimizing the movement planner of an underwater vehicle using a Q-Learning methodology as a successful application of RL in control of this kind of vehicles. In [26] there is another example of RL-based local planner for local planning. This approach focuses also in a static-obstacle environment, the most common solved problem with RL algorithms for this vehicles.

Recent works have extended the aforementioned single DRL problems to multiagent deep reinforcement learning (MADRL) scenarios [16], [27], [28] to avoid the previously mentioned dimensionality curse. Initial results report successes in complex multiagent scenarios. However, there are several challenges to be addressed [7], like multiagent credit assignment, global exploration, relative over-generalization, and scalability. It is remarkable, in the context of optimization of ASVs fleets, the contributions of [17], where a fleet meta-agent of three boat-like autonomous vehicles is trained using Deep Q-Learning (DQL) to perform swarm-cooperative trajectories. The multi agent local trajectory optimization is addressed also in [18], where the DRL goal is to optimize the policies of 3-5 agents to reach several final positions trough static obstacles.

Two main approaches can be observed in the recent literature: distributed learning [16], [29], [30] such as Independent Q-Learning (IQL), where every agent tries its best based only on its own experience, and centralized learning: a centralized network which tries to optimize the behavior of all agents. In the first approach, the scalability is feasible because every agent deals with its own behavior, but the environment becomes non-stationary [31] because every agent observes the others as a part of the environment. As each agent changes its policy over time, the learning stability is compromised, since the target optimal policy of every agent changes from one episode to another. In [16], a multiagent deep reinforcement learning (MDRL) approach is proposed with a focus on the scalability and learning stability: IQL is implemented with an inter-agent shared fingerprint in the buffer replay to avoid learning instabilities. In the centralized approach, the agents are modelled as a meta-agent with an action set A'. With A_i as the action set for an agent i and with N as the number of agents, the action joint dimension becomes $|A'| = |\prod_{i=1}^{N}(A_i)| = |\{a_1, a_2, \ldots, a_N\}|$. This dimension problem becomes unfeasible with a large number of agents, limiting the scalability of such methods as they are unable to deal with changes in the fleet size. This dimension problem becomes unfeasible with a large number of agents, limiting the scalability of such methods as they are unable to deal with changes in the fleet size. This is the case in [17], where the fleet size is fixed and the action space is small ($|A'| = 27$) and more vehicles will explode the scale of the problem.

Some researches try to deal with the drawbacks of both methodologies by designing combined methodologies between the pure independent approach and a centralized learning like [14], [15]. In [15], a centralized action-value function $Q(s, a)$ will serve the purpose of the *critic* and a distributed policy $\pi(s)$ will model agents' behavior as the

TABLE 1. Different MARL approaches in the literature.

Ref.	Scenario	Action	RL algorithm	Contribution
[13]	Aerial Drones (Grid map)	Discrete actions (directional movements)	Contextual Actors-Critic (Centralized Value Function + Distributed Policy Function)	- A multi-algorithm comparison. - The cA2C algorithm. - A Linear-Programming cA2C variation.
[14]	StarCraft MA benchmark	Discrete actions (directional and special actions)	Decomposed Off-Policy Gradient (Independent Q-Functions + Policy Gradient with composed Q)	- DOP algorithm. - An Centralized - Decentralized mismatch issue analysis.
[15]	Multiple benchmarks (Predator-Prey, Cooperative...)	Multiple (continous and discrete)	Decentralized Actor + Centralized Critic	- Algorithm suitable for both continous and discrete scenarios. - A comparison between Decentralized RL approaches.
[16]	StarCraft MA benchmark	Discrete actions (directional and special actions)	Independent Q-Learning with fingerprint method.	- Two methods to stabilize IQL: fingerprint method and prioritized replay.
[17]	Three ASVs with static obstacles	Discrete actions (angular speeds)	Meta-agent Double Deep Reinforcement Learning	- A method for tracking different goal position with ASV triangular squadron formation. - A comparison in a real-case maritime scenario.
[18]	Three drones with static environment.	Discrete actions (next discrete position)	Multi-agent Deep Policy Gradient	- A method for tracking different goal positions with different number of vehicles.
[19]	Platoon formation of vehicles	Discrete actions (transitions in a graph)	Q-Learning	- A platoon modelling of an autonomous car fleet. - A method for platoon path planning focused in the optimization of energy consumption.

actor in a multiagent Actors-Critic novel approach. This kind of method tends to avoid the local-minima problem generated by the Nash equilibrium observed in fully-independent and competitive agents [32] but are usually less sample efficient. In our approach, a different objective is pursued because the agents work in a full cooperative environment. In Table 1 are presented the most interesting approaches for the multiagent case of the recent literature. Note each one addresses a very different problem.

In the DRL path planning literature, in both multi agent and single agent case, it is common to find that the task to complete is the position tracking one like in [18]. This particular work put the effort in the multi agent task of reaching goal positions without collisions between agents and obstacles in a static environment. Our proposed method is able to deal not only with multiple agents but also with a *dynamic environment* with *different interest zones* and, in a further test, deal with the *battery issues*. Those characteristics are not usually addressed in the literature and, when they do [33], [34], is it infrequent to find those applied at the same time. Regarding deep network architectures of related works, the same goal-point tracking is addressed in [35] using convolutional neural networks (CNN) as it is presented in this approach. Nevertheless, a higher state-space domain is required to prove the method is scalable to bigger maps and more agents, as it is done in the with this proposal. In this regard, the proposed method finds a way to code all the information available in the environment, i.e., agent's positions, the physical boundaries of the environment and, as a novel proposal, the temporal dependencies in order to enhance the stability of the learning and avoiding the need of recurrent neural networks, like in [36].

When addressing the information availability and agent abilities, some multi agent approaches have focused mainly on pure distributed learning [16], [29], [30] because the agents are different between them, they pursue different objectives or are not always in communication. On the contrary, our approach puts the attention on the performance for equal, identical cooperative agents since every ASV experience is interchangeable. In this way, the efficiency of the learning is improved with a shared memory replay, and scalability is addressed, while other researchers are interested in an fully-independent execution like [15], [18], where the experiences are not shared and the gradient-descent steps are higher. This work uses also Value Iteration methods (DDQL), which are generally more sample efficient than Policy Iteration methods [4], [37] like in [28]. This work also uses an implementation where the actions are always computed with all the available shared information about the environment, differently from [15], where the centralized Q-function only is used in the training as the critic. Therefore, the novelty of this approach focuses also in obtaining good policies efficiently within the training, without too much computational complexity. This objective is not always pursued in the literature and the sample efficiency in the learning is neglected.

Furthermore, this novel methodology is able to work with a Dueling variation of the classic DDQL algorithm for this particular multi agent case. The Dueling architecture has resulted in splendid learning behaviors for the single agent case in learning agents for the ATARI games [12] but its

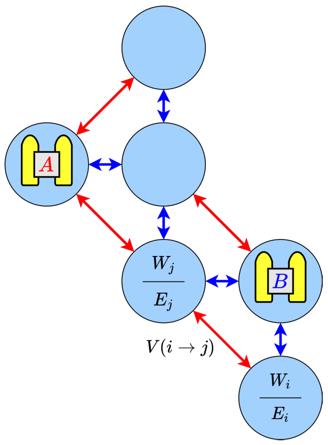

FIGURE 2. Example of graph for the patrolling problem.

advantages have not been sufficiently explored in the DRL path planning literature and, more specifically, in a multi agent dynamic scenario.

III. PRELIMINARIES

A. THE PATROLLING PROBLEM

The patrolling problem can be described in terms of an undirected graph $G(E, V, W)$ (see Fig. 2) [38], where every geographical zone of the lake worth of visiting V constitutes a node from M total nodes, and can be reached by an agent by moving along the edges E (following a metric assumption) once every timestep. Every node also has an *idleness*, noted by W, that counts the timesteps since the node's last visit. The patrolling problem consists of finding a policy π_i for every agent i that minimizes the average W given a number of timesteps:

$$G(E, V, W) \to \{\pi_1(E_1), \ldots, \pi_n(E_n)\}|\min \frac{1}{M}\sum_{k=1}^{M} W_k \quad (1)$$

In the case of Ypacarai Lake, it is well-known that there are highly-concentrated contamination areas (e.g. algae blooms) with special interest for researchers [2], [11]. These areas match with recreational port areas or industrial zones near the shore, such as *Puerto de San Blas*, in the south of the lake. Therefore, these relevant areas should be revisited with a higher frequency depending on the importance given to each zone. This implies that the idleness must be weighted relative to the importance of this node $\mathcal{I}$. This will be known as the **non-homogeneous patrolling problem**, and it is more suitable for the Ypacarai scenario as is explained in [4]:

$$G(E, V, W) \to \{\pi_1(E_1), \ldots, \pi_n(E_n)\}|\min \frac{1}{M}\sum_{k=1}^{M} W_k \times \mathcal{I}_k \quad (2)$$

Finally, an importance matrix $\mathcal{I}$ (see Fig. 3) is created to weight the idleness of every cell (node). It was generated with the sum of two Gaussian density distributions and a minimum

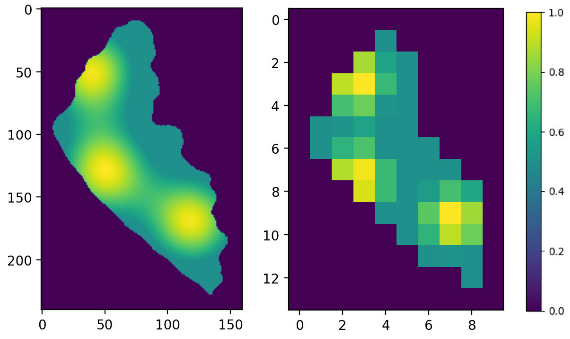

FIGURE 3. Importance map (left) and discretized importance map $\mathcal{I}$ (right).

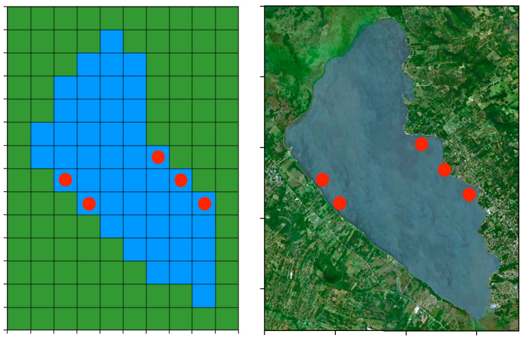

FIGURE 4. Discretized gridmap (left) and a satellite image of the Ypacaraí Lake (right). In red, the initial deploy position for the ASVs.

interest of 0.5 centered in three main focus of contamination using the information available in the International Hydroinformatics Center of Paraguay[1] as guideline.

B. SCENARIO AND ASSUMPTIONS

In order to train the agents for a real environment, a simulator has to be designed which must consider real constraints. In the following list, the many assumptions of this approach have been enumerated:

- The scenario has been discretized in a homogeneous grid. This discretization results in a 14 × 10 cell map, therefore each cell is $1100m \times 1100m$. A higher resolution will result in a more precise scenario, but slower and harder training because of the increased problem dimension. There are 5 possible zones for the deploy of the ASVs (red dots in Fig. 4).
- As all ASVs have the same capabilities, it is assumed they move synchronously: every ASV samples a cell (its current location) and, after that, performs a movement to the next cell at the same speed: with full battery, every ASV can travel through 30 cells (approx. 4 hours of battery at full speed).
- ASVs can perform 8 different movements: perpendiculars (N, S, E, W) and diagonals (NE, SE, SW, NW). They will work under a metric assumption, so the diagonal

[1]https://hidroinformatica.itaipu.gov.py/

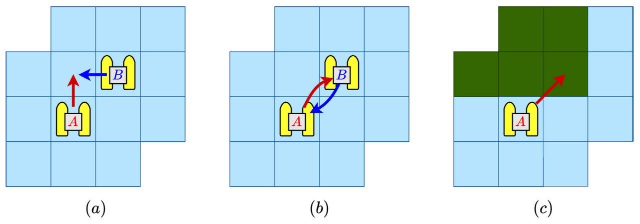

FIGURE 5. Examples of illegal movements. (a) and (b) are considered a collision. (c) is considered a non-possible movement.

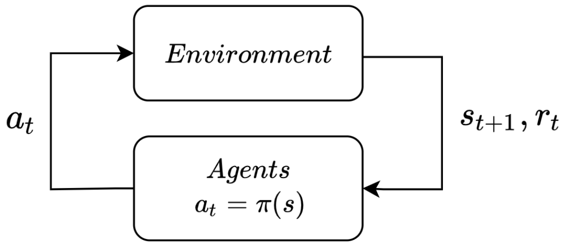

FIGURE 6. Fundamental reinforcement learning scheme. The agent/agents perform an action a_t using its policy $\pi(s)$ resulting in a reward r_t and a new state s_{t+1}.

movements will be punished proportionally to a perpendicular movement.

- There are different types of cells: occupied (by some agent), obstacle (a non-navigable cell), and free (a navigable cell, already visited or not). An obstacle cell cannot be a target of a movement (a penalization will be granted) and occupied cells can be a waypoint if the agent will free it in the following time step (Fig. 5b).
- Agents cannot occupy the same cell (because of the risk of collision). Furthermore, swapping positions is considered a risk of collision in travelling and will be penalized (Fig. 5a-b).
- It is assumed (for a better realism) that drifts and control fails could happen due poor path tracking or a disturbance rejection of the control loop. There will be a constant probability of 5% for a movement failure as a maximum expected value. The failure leads to a random movement if it occurs.
- Once the next waypoint is decided (the next movement for every ASV), a low-level control loop is in charge of the actuators' dynamics, i.e., it guides every ASV to the next physical point and rejects all possible disturbances. This will be referred to as the *local path planner loop* [22].
- As the battery duration for every ASV is an important limitation to consider, in a second round of training, a different battery level for every agent is assumed. Initial battery levels are chosen randomly from 100% to a minimum of 70%. It is reasonable to think that the initial charge of the ASVs could vary from one agent to other. With every movement, a 1/30 fraction of the battery is subtracted. When the battery reaches zero, the ASV is considered dead and will not perform any movement.

IV. METHODOLOGY

In this section, the methodology is presented on the mathematical framework of a Markov Decision Process (MDP). Every MDP is defined by a state of the environment s, an action taken by the agent/agents $a \in A$, a reward r resulting from this action, and a transition probability T of taking this certain action in the given state (Fig. 6). Reinforcement Learning can address the solution of an MDP to find the optimal policy $\pi^*(s)$ that maps the state of the environment into the action that will return the maximum reward on the long term. In this work, two different approaches are proposed, based on different improvements of the same Deep Q-Learning foundations: i) a Double Deep Q-Network and ii) a Dueling Architecture for Q-values optimization. In both algorithms, the Q-function is optimized by taking a descending gradient step in the loss function with respect to each parameter of the given deep network. The main difference rests in how the Q-values are computed: whereas in the DDQL the Q-values are estimated directly, with the dueling network, they are computed with an estimated advantage function $A(s, a)$ and the value function $V(s)$.

A. SINGLE AGENT DEEP Q-LEARNING

Deep Q-Learning (DQL) and its variation, Double Deep Q-Learning (DDQL) [12], are the most common algorithms for model-free reinforcement learning. Both algorithms are based off of collecting experiences and estimating the Q function (modelled with a DNN) to maximize the expected total return without any need of an environment model. In the single agent case, an action is usually selected with an ϵ-greedy policy given a state s. This action a makes the environment return a reward r, and a new state s' is processed. An experience (s, a, r, s') is generated and stored in a replay memory of size $\mathcal{M}$ for a future mini-batch training. With every episode (an episode is defined with many timesteps as the duration of a mission), some experiences are sampled in a mini-batch of size $\mathcal{B}$ and the neural network is trained by the time-difference (TD) method, taking a gradient descent step from the error of the calculated Q function (behavioral function) and the target Q function, which is the maximum expected discounted reward.

$$Q(s, a)^{t+1} = Q(s, a)^t + \alpha \times \left(r + \gamma \max_{a'} Q(s', a')^t - Q(s, a)^t \right) \tag{3}$$

The discount factor γ decreases the effect of the rewards far in the future, to prioritize the rewards at the beginning of the episode and decrease the importance of future rewards. As it is modeled in subsection IV-E, rewards in the last timesteps are less significant than in the early moments of the episode.

Generally, DQL is able to train a single agent in a stable way for many scenarios and assumptions, but for a better

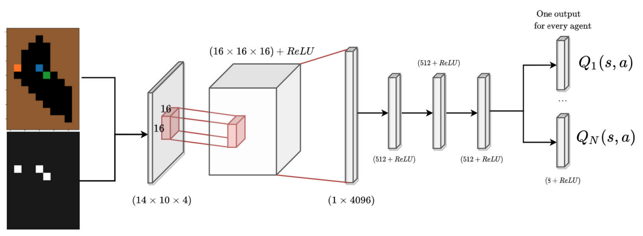

FIGURE 7. **Proposed centralised-distributed deep Q-Learning network.**

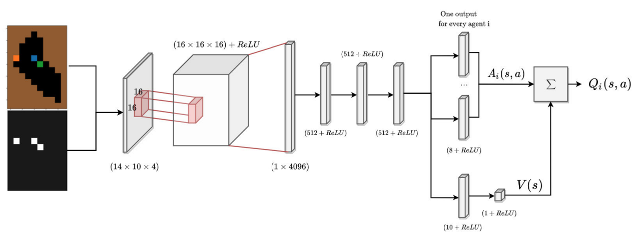

FIGURE 8. **Proposed centralized-distributed dueling Q-Learning network.**

convergence, a modification of the vanilla DQL is proposed in [12]. The authors of the paper solve the overestimation of the Q-values with a single Deep Q-Network by cloning the aforementioned $Q(s, a; \theta)$ to be the target or objective function $Q^*(s, a; \theta')$, with a transferring of the weights ($\theta \rightarrow \theta'$) only every few episodes. This results in a much better convergence and stability at almost zero cost.

B. MULTIAGENT DEEP Q-LEARNING

Therefore, in this work a Double Deep Q-Learning network (with some modifications) is proposed for the learning process (see Fig. 7). First, convolutional and fully-connected layers are designed to capture the features of the scenario (the positions of the agents, obstacles and already visited cells). The final layers as seen in Fig. 7 are designed in a distributed form: due to the scalability problem, modeling the multiagent case as a meta-agent, as mentioned in Section II, will result in an $|A|^N$ fully-connected layer with $|A|$ possible actions and N agents. This calculation is obviously unfeasible, even with a low value of N. Consequently, this paper proposes a decoupled final layer with an individual $|A|$ fully-connected layer for every agent (see Fig. 7). Since every agent has the same action set and is ruled under the same constraints, the experiences (s, a, r, s) of each agent must contribute equally to the learning process. In this way, the single-agent DDQL algorithm is extended to the multiagent paradigm with $|A| \times N$ neurons in the last layer. Additionally, the common centralized convolutional resolves possible sub-optimal actions from Nash Equilibrium by jointly deciding the next move: because two independent agents, only provided with their observation of the environment, could choose to visit the same zone in a greedy decision resulting in a collision penalization (as it happens in the well-known *Prisoner's Dilemma* [24]). When acting in a full competitive case, the optimal behavior will be to choose a zero reward instead of the greedy reward. This policy is discouraged when the decision is not purely individual like in cooperative games [37].

C. MULTIAGENT DUELING Q-LEARNING

Following the aforementioned design for the Q-Network, a modification of the Q-Network is proposed, as in [39]. In this paper, two different estimators for the Q-function optimization are presented: the State-Value Function $V(s)$ and the Advantage Function $A(s, a)$. The former returns the value of the current state s in terms of the future expected reward, and the latter evaluates the expected reward for an individual action a in a state s with respect to the other possible actions. Thus:

$$Q(s, a; \theta) = V(s; \theta') + A(s, a; \theta'') \tag{4}$$

It is stated that, to address the identifiability issue (in the sense that given Q we cannot recover V and A uniquely), modifying 4 is mandatory to add a baseline of A. This baseline is chosen to be the average value of A for every possible a in the given state:

$$Q(s, a; \theta) = V(s; \theta') + \left(A(s, a; \theta'') - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta'')\right) \tag{5}$$

In our approach, the use of the common centralized network to estimate the state-value function is intuitive, as it takes the whole scenario of the many agents into account. The individual output layers estimate the advantage function for every agent as they estimated Q values before. (see Fig. 8) This variant has two positive aspects: i) the DQN remains the same except for the $V(s)$ estimator (only an additional neuron and the aggregate layer) so the DDQL algorithm can be recycled and ii) the formulation of the learning problem suggests that this architecture can focus in collaborative actions which allows agents to use the estimated value function to improve their policies as suggested in [39]. In Algorithm 1 is presented the pseudo-code for both approaches. Note in the case of the Dueling architecture, the algorithm remains the same as the adversarial calculus of $Q(s, a)$ from $A(s, a)$ and $V(s)$ function is considered embedded into the function itself as stated in [12].

D. STATE REPRESENTATION

As mentioned before, the state is the representation of the information available for training. From the state, the scenario model must be inferred and processed to optimize the action-value function. A MDP framework is used based on the Markovian assumption: *the probability of transition for the environment from the actual state to the next one only depends on the actual* [5]. This requires the environment to be *memoryless*, which is not suitable for every real case. For the proposed Reinforcement Learning methods, it is intended to deal with the Markovian assumption with a complete information state, i.e., the state contains all the information from the previous timesteps so that the state groups the information of past actions without any time-dependencies left. In the proposed

Algorithm 1 Proposed DDQL Centralized Learning Algorithm

```
Initialize replay memory M; Initialize action-value
function $Q(s, a; \theta)$ with random weights $\theta$;
Clone $Q(s, a; \theta)$ function into target function
$Q^*(s, a; \theta')$;
$epoch \leftarrow 0$;
while $e \leq epochs$ do
    step $\leftarrow$ 0;
    Reset the environment;
    Get the initial state $s$;
    while $step \leq N_{steps}$ do
        With probability $\epsilon$ select a random action joint $a$
        otherwise select $a = argmax_a Q(s, a; \theta)$ Execute
        action joint $a$ and observe reward joint $r$ and
        next state $s'$ Store transition $(s, a, r, s')$ in $M$
        if $M \geq \mathcal{B}$ then
            Sample random *Batch Size* $(s, a, r, s')$
            experiences from $M$;
            for *every* $(s, a, r, s')$ *sampled* do
                for *every agent i* do
                    $q_i^* \approx r_i + \gamma \max_{a'} Q^*(s', a'; \theta')$;
                    Acumulate the Loss by
                    backpropagating
                    $L \leftarrow L + Loss(q_i^* - Q_i(s, a_i))$
                end
                Perform gradient descent step:
                $\theta \leftarrow \theta + \alpha \frac{\partial L}{\partial \theta}$
            end
        end
        $step \leftarrow step + 1$;
    end
    $epoch \leftarrow epoch + 1$;
    Every T epochs, updates $Q^*$ weights $\theta^- \leftarrow \theta$ ;
end
```

approach, the state will be a 3-channel (RGB) image. This state representation allows time-dependent information with the inclusion of a gray-scale tone representing the idleness of every visited zone, as was proposed and tested successfully in [4]. In different vivid colors, the agents are represented and, in brown, the non-visitable cells (see Fig. 9).

E. REWARD FUNCTION

The reward function models the objective of the agents. This function will consider the interactions between agents and also the individual returns of every moment an agent performs. A tailored reward function is designed with the aim of reducing the mean of idleness in the map (as mentioned in subsection III-A). Therefore, a positive reward must be awarded when an agent performs an action that results in a visitation of a cell with high idleness $\mathcal{W}$. Depending on the importance of the cell and the idleness, the reward must be modulated in consequence. Bad behaviors, like collisions

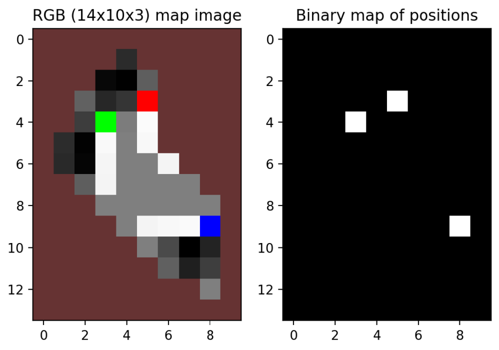

FIGURE 9. RGB state with 3 agents. In gray, the visitable. In brown, the land cells. In red, green and blue, the three agents.

between agents and movements outside the navigable zones, shall be punished. Given (x, y) the position result of the action a, the reward function r is defined in 6. This reward function was tested with successful results in [4].

$$r_i(s, a, (x, y)) = \begin{cases} -5 & \text{if a causes collision.} \\ -5 & \text{if a causes a move out lake.} \\ \mathcal{I}_{x,y} & \text{if (x,y) is not visited yet.} \\ \frac{\mathcal{I}_{x,y}}{max\ \mathcal{W}} \times \mathcal{W}_{x,y} & \text{if (x,y) is already visited.} \end{cases} \quad (6)$$

The aforementioned matrix $\mathcal{I}$, which symbolizes the *relative* importance of the given (x, y) visited cell, will weight the idleness of the zone. In the end, the maximum reward for a high-importance high-idleness cell will be 1. Finally, for a better coverage and in order to avoid trivial solutions around a certain zone, every position of $\mathcal{I}$ will be decreased a 20% in every visit, which means a cell (x, y) of $\mathcal{I}_{(x,y)} = 1$ will return a reward 5 times at most before its $\mathcal{I}_{(x,y)}$ becomes zero. For the quality measurement process this is a realistic assumption since sampling a certain zone more than 5 times is unnecessarily redundant. Finally, to adapt the reward function to the metric assumption (in diagonals movements we travel more distance), every diagonal movements will be penalized with -0.1.

V. RESULTS

This section contains the metrics to evaluate the performance of the algorithms, along with the results of the learning in both proposed approaches are depicted with its respective accumulated reward curves. Finally, a comparison between all the algorithms with IDQL and lawn mower included is presented. The battery-constrained case and a learning-time analysis of the proposed approach is also presented.

For the training, a simulator coded in Python[2] is designed and for the neural network optimization PyTorch[3] library is used. The code is available in a Github repository.[4] All simulations were executed in an AMD Ryzen 9 3900 (3.8 GHz)

[2]https://www.python.org/
[3]https://pytorch.org/
[4]https://github.com/derpberk/MARLYpacarai

with an Nvidia RTX 2080Super-8GB GPU and a 16 GB RAM memory module.

A. METRICS

For the evaluation of the performance in every case, some metrics are used:

- **Accumulated reward** $\mathcal{R}$: The accumulated reward shows performance with respect to the reward function. The higher the reward, the more the problem fits its designed objective. Also, the deviation of the reward gives an idea of the robustness of the inferred policy from one start condition to another. With N as the number of agents and t the timestep (from 0 to T possible timesteps):

$$\mathcal{R} = \sum_{(i,t)}^{(N,T)} r_i(s_t, a_t) \tag{7}$$

- **Mean weighted idleness** μ: For an alternative evaluation metric of the Patrolling Problem solutions, the mean weighted idleness across the navigable cells is computed. This is:

$$\mu = \frac{1}{M}\sum_{k}^{M} \mathcal{W}_k \times \mathcal{I}_k \quad \text{where } k \in \mathit{Visitables} \tag{8}$$

B. PROPOSED APPROACHES COMPARISON

For the hyperparametrization of the network, some values were tested based on previous published researches [4]: the *learning rate* α is fixed to 1e-4, as a higher value did not guarantee convergence in some cases. The ϵ-decay rate is chosen to be 4e-5 (In 24.000 episodes ϵ decreases from 0.99 to 0.01, i.e., almost full explotative behavior). With higher values, both networks (DDQL and Dueling) tend to explore too little for good performance. Finally, the net size (total number of convolutional filters and dense layer sizes) was selected based on the successful results in [4]. The number of convolutional filters were expanded (from 8 to 16) to enable the network to process multiple agents and the kernel size is reduced (from 5×5 to 3×3) to conform the new resolution. The final hyperparameters used are:

The trained policies of both approaches will be compared with the Independent Deep Q-Learning (IDQL) equivalent and with a non-reactive and trivial solution, such as the lawn mower trajectory (LMT) [40]. The IDQL will use the same hyperparameters for training and the same network of the DDQL as a heuristic decision for comparison since both have been proven valid for the independent learning case. Every agent will optimize its own network independently based on its perception of the state: an independent agent state is formed by a RGB image of the scenario (similarly to the proposed approach), but only its own position is distinguished from the others agents' positions (see Fig. 10). In the LMT, the strategy tends to cover the maximum possible area in a cyclic movement. From one agent to four agents (the whole fleet), the LMT approach will share the same sized lake

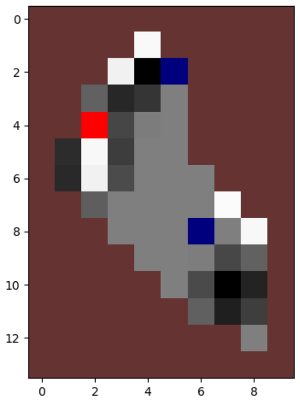

FIGURE 10. RGB state for an independent agent in IDQL.

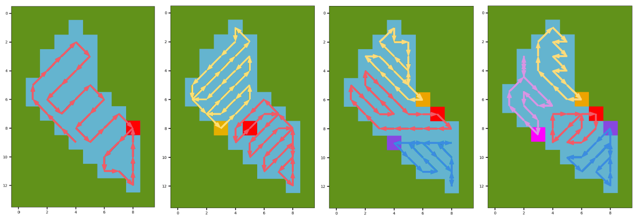

FIGURE 11. Lawn mower trajectories for one, two, three and four ASVs.

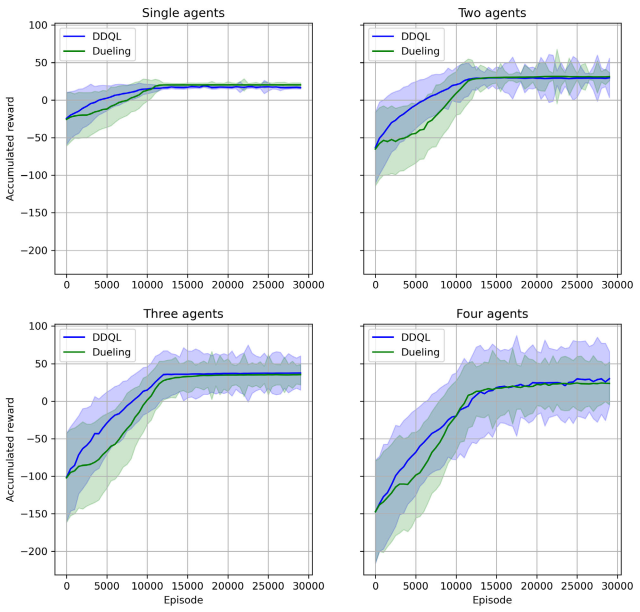

FIGURE 12. Result of the 30.000 episodes training from one to four agents for DDQL and Dueling.

areas for a homogeneous covering, ignoring any importance criteria. Fig. 11 presents the cyclic lawn mower trajectories. Those trajectories were selected arbitrarily, with the only criteria of starting in a deploy cell and covering the same number of cells as the other agents.

1) TRAINING RESULTS

In Fig. 12 are presented the results for different numbers of agents. These results show that both DDQL and

TABLE 2. Hyperparameters selected.

Parameter	Value
α	1E-4
Batch Size	32
Mem. Size	10.000
ϵ-decay	8.25E-5
minimum ϵ	0.05
γ	0.95
Steps	30
Epochs	3000

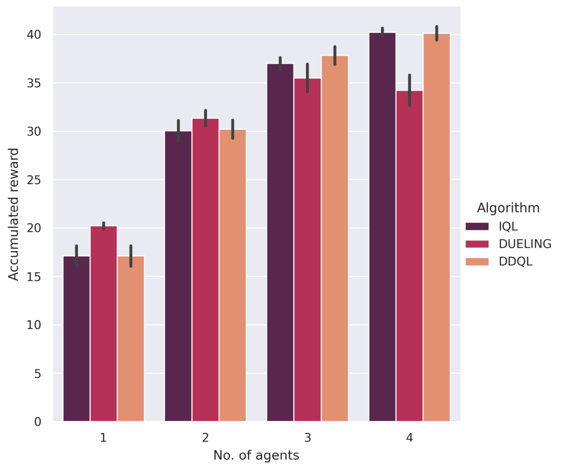

FIGURE 13. Average rewards of 100 simulations with the trained networks.

Dueling-DQL algorithms can successfully train the agents to pursue the maximization of the total reward. It is understandable that the more agents are involved in the problem, the higher the variation that is observed in the reward because of the exponential growth of the state-space domain with the number N of agents $\mathcal{S} \times |A|^N$. Also, with the *epsilon*-greedy policy, there is always a possibility of a random movement (5% according to Table 2). Notice that, in the four agents scenario, a random movement or a failed movement which is stated in the assumptions as a possibility, could cause a very high penalization because of the high probability of multiple collisions.

For this analysis, it is also necessary to observe that this environment has a limitation in the maximum reward possible since every cell decreases its interest with every visit. In other words, the total interest available for reward will be the result of visiting each cell five times with a 20% less reward every time, and as a result, the maximum available reward is not proportional to the number of agents.

Nevertheless, the algorithms show that the agents learn a policy based on collision-avoidance and revisiting high-interest cells: from an average reward of -100 at the beginning of the training with a fully random policy in the three-agent case, to a 35 average reward with the Dueling. Both DDQL and Dueling-DQL approaches have similar average rewards. With the Dueling architecture, the average reward from one episode to another for one and two agents is slightly better (a 11.5% and 5.5% better respectively). This suggests that the Dueling architecture is more sample efficient [39], because of the good estimation of V(s). It also converges in a more robust policy for a lower dimension of the problem but its convergence is worse when the dimension increases (see Table 3 for the complete result metrics). Although the two proposed algorithms achieve solid solutions, due to the high dimensionality of the problem, more episodes of learning are needed to achieve an optimal solution. Nonetheless, for a fair efficiency and performance comparison only 30,000 episodes were simulated.

2) COMPARISON WITH OTHER APPROACHES

For the comparison, both DDQL and Dueling policies are changed to be full greedy ($\epsilon = 0$) to measure their best decisions. Table 3 contains the results for the different number of agents and different approaches. Note that for the single agent case, IDQL and DDQL have the same results because in this case they are actually the same algorithm.

It is remarkable how the three Deep Reinforcement Learning approaches have a much better performance than the classic lawn mower trajectory. The DDQL improves in a 12%, 20%, 15%, and 13% with the best solution for every number of agents respectively. The Dueling Network improves in a 24%, 21%, 12%, and 11%, whereas the improvement in the IDQL case is 12%, 15%, 12%, and 12% respectively. It is easily understandable that, when the lawn mower trajectory is specified, there is no criterion related to the inherent interest of the lake cells. The Reinforcement Learning approaches take the trajectory calculation into account, resulting in better minimization of the mean weighted idleness across the scenario.

Comparing now between Deep RL methods, the proposed DDQL approach shows a slight improvement with respect to the maximum reward achieved by the other algorithms (an average improvement in every multiagent scenario of the 6% and 2% respectively for IDQL and Dueling). IDQL has proved to also be a suitable algorithm for the multiagent case. In this particular scenario, the inferred policy for the individuals has also shown robustness in avoiding collisions and illegal movements. Nevertheless, the DDQL approach achieves better record trajectories, although the averages are very similar compared to IDQL (see Figs. 13 and 14 for the two aforementioned metrics with the trained policies in each case). If we observe the particular episode of the best-achieved solution for every agent, it is possible to visualize the effect of DDQL algorithm in the speed of importance-weighted coverage (see Fig. 15). As the reward and redundancy criteria determined by the reward function (6) is the same no matter the size of the fleet, the marginal improvement of a new agent decreases. The improvement of reward in the best DDQL trajectories from 1 to 2 agents is about 78%.

TABLE 3. Comparative table with the rewards and average weighted idleness of the Ypacaraí Lake.

Num. of agents		R_{max}	$\bar{R}$	σ_R	μ_{min}	μ
N = 1	DDQL	18.47	17.01	1.06	12.61	13.15
	Dueling	20.6	20.21	0.31	12.59	12.79
	IDQL	18.47	17.01	1.06	12.61	13.15
	LMT	16.49	-	-	13.54	-
N = 2	DDQL	33.02	30.23	0.94	7.65	8.84
	Dueling	33.23	31.35	0.81	7.56	8.41
	IDQL	31.59	30.05	1.1	7.66	8.46
	LMT	27.38	-	-	8.76	-
N = 3	DDQL	39.13	37.84	0.92	5.25	6.02
	Dueling	38.03	35.51	1.40	5.68	6.55
	IDQL	38.01	37.01	0.53	5.21	5.79
	LMT	33.89	-	-	7.03	-
N = 4	DDQL	41.66	40.21	0.70	4.01	4.83
	Dueling	38.12	35.02	1.57	4.88	6.58
	IDQL	41.04	40.09	0.46	4.14	4.60
	LMT	36.66	-	-	6.44	-

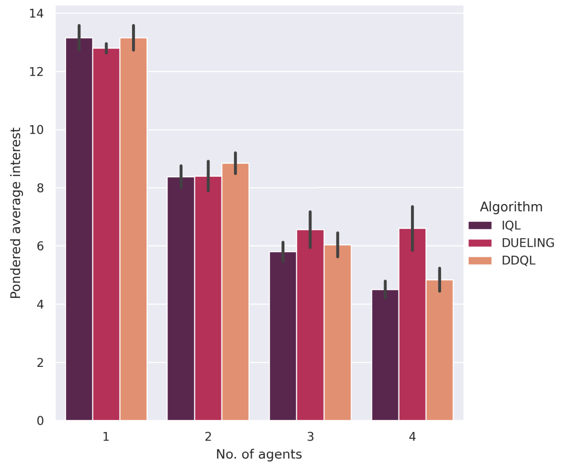

FIGURE 14. Average weighted idleness of 100 simulations with the trained networks.

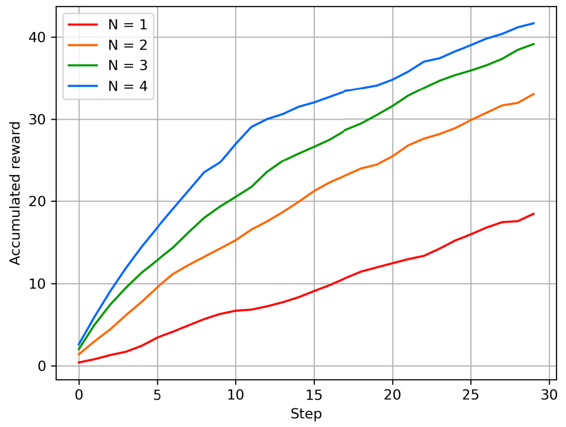

FIGURE 15. Accumulated reward with DDQL for each fleet size. The higher the number of agents, the more reward is achieved. As the available reward is limited to 5 visits, the increment of the possible maximum reward is not proportional to N.

Increasing from 2 agents to 3 improves the reward by 19%. Increasing from 3 agents to 4 returns only a 7% improvement. This figure (Fig. 15) allows a pre-planification on the desired coverage given a number of steps and a number of agents. If the mission time is lower, more agents could be chosen and vice versa.

Fig. 16 shows the resulting idle maps of the optimized trajectories learned by each number of agents in the best case. The optimized policies force the agents to share the interest space. In the single agent case, as the number of steps is insufficient to cover the whole map, the ASV focuses on the most important areas and visits the maximums at least once (occasionally twice). In the multiagent experiments, the ASVs first visit the nearest maximum-interest areas and proceed to explore the other ones, tending to avoid very long travels to revisit those important zones once the idleness goes up. These results indicate a well-coordinated and cooperative behavior, since the complete map is explored and the zones are explored proportionally to their interests. proportionally to its interest.

C. TRAINING PERFORMANCE

While it is true that the IDQL algorithm achieves similar results to the proposed DDQL and Dueling Algorithm, it faces a scalability problem relative to the number of optimization steps needed for convergence. On the one hand, the IDQL approach provides every agent with an individual convolutional neural network so, during training, every agent must train its own mini-batch and take as much optimization

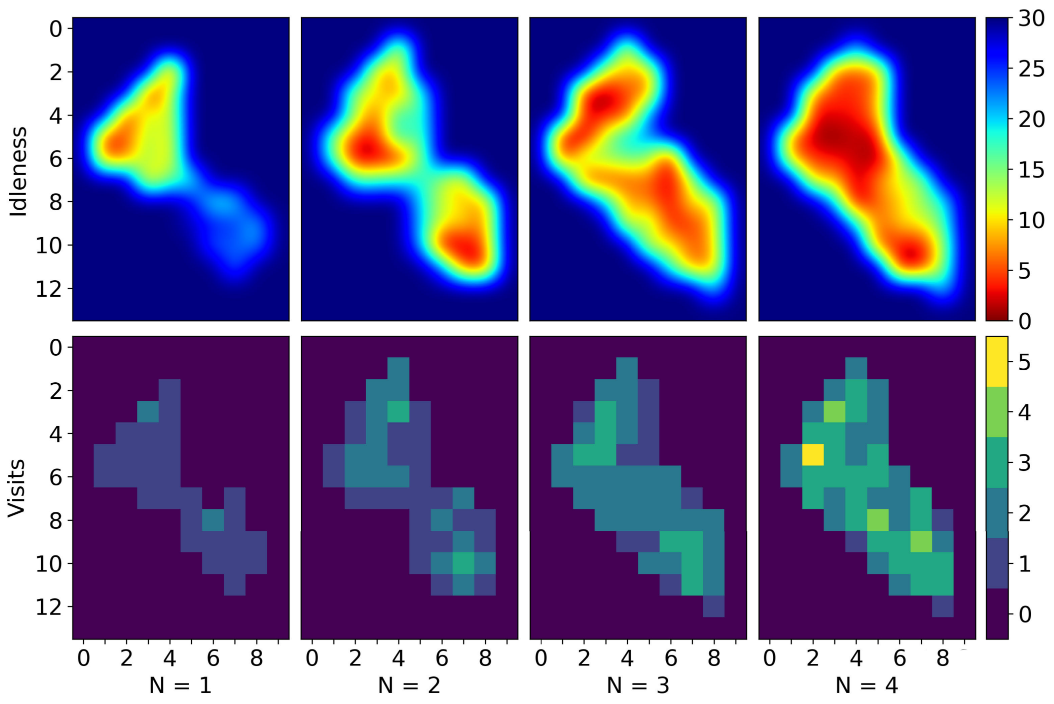

FIGURE 16. Map of idleness (up) and frequency of visits (down) in the DDQL best trajectories for every number of agents. Note that, as the trajectories are optimized, the interesting zones receive much more attention and visits. With N = 3 and N = 4, as there is not sufficient interest for all agents, the complete coverage of the map is achieved.

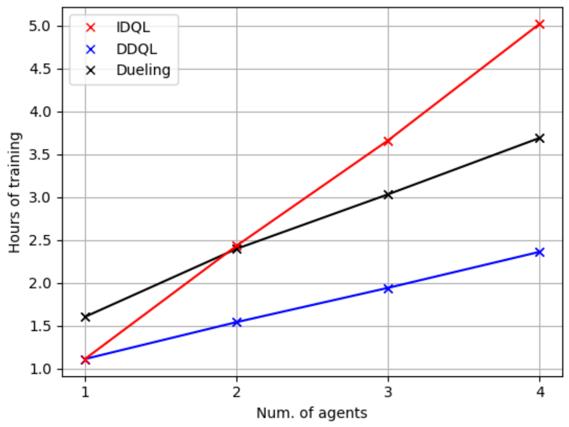

FIGURE 17. Time of training for the three DRL approaches. Note that the growth of the learning time is almost three times higher in the IDQL approach.

steps as the number of agents. On the other hand, the decoupled approach allows for more efficient learning by sharing a bigger part of the network. This will reduce the time of training which is in this application an important metric to consider. The need of retraining the network for a transfer learning process or because the importance map changes has to be feasible in terms of scalability. The much higher times for training do not compensate for the nearly identical performance of the other approaches. Fig. 17 shows the hours needed to train the scenarios previously mentioned with the computer workstation available for the agents learning. It can be observed that the proposed DDQL approach is three times faster than the independent approach.

Regarding the Dueling architecture, it has been observed that the operations related to the Advantage function computation slows considerably the network optimization resulting in a not-so-fast training. Besides that, this algorithm is also two times faster than the IDQL and is a candidate for train with more episodes. In the end, the DDQL approach result in a suitable, efficient and robust methodology for the Ypacaraí non-homogeneous patrolling case and more importantly, dimensionally feasible for standard computation resources.

D. BATTERY SIMULATION

Considering the battery, the agents have to face the fact that, occasionally, the battery will fail. This makes the policy more difficult to optimize, because the required cooperation changes when a ASV disappears in order to sample the important zones. When an agent is disabled, it will remain static, it will not accumulate any further reward and this area will be considered an obstacle for the other agents. To test the proposed algorithm for this task, a typical three agent scenario is simulated with different initial battery levels as stated in Subsection III-B. Only a modification to the previous algorithm is done: the agents colour in the state changes to pale green when its battery is 0 to infer its status (see Fig. 18)

In Fig. 19 is illustrated the result of an experiment with the batteries enabled. The agents start with battery levels

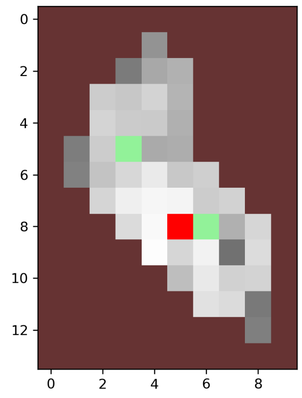

FIGURE 18. State with two agents without battery (pale green) and an active agent (red).

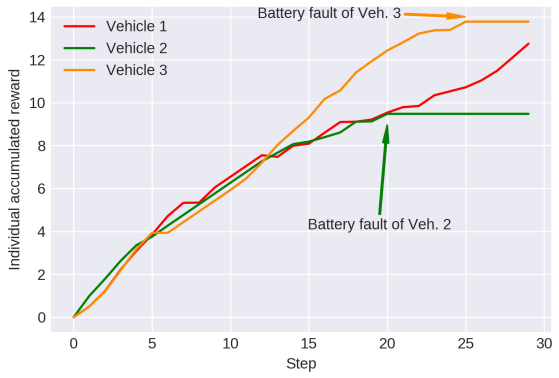

FIGURE 19. Accumulated rewards for every ASV with the batteries enabled. When vehicles 2 and 3 stop, vehicle 1 keep travelling through interesting areas avoiding collisions.

of 100%, 85% and 70%, which means the battery will fault in step 30, 25 and 20, respectively. It can be seen how the vehicle 1, after the faults of the other ones, continuously increases its reward. It has been observed, since the disabled agents does not move across the map anymore, the remain agent tends to move to the new uncovered zones.

This suggests that the proposed algorithm is able to work with a variable number of agents and is fairly robust to agents faults, a very desirable in this particular application. The average reward for 100 episodes in this new battery-simulated scenario with three agents is similar but slightly lower (obviously because of the battery faults) than the previous experiment. The results are compared in table 4. As the DDQL is trained with the new battery situation, it overcomes the results of a DDQL non trained for this situation and achieves better maximum and average rewards than the non-trained algorithm.

VI. IMPLEMENTATION OVERVIEW

In the real case scenario, a fleet of three ASVs will be available for the patrolling task. Every agent knows its position thanks to a differential GNSS system[5] which provides

[5]https://emlid.com/reachrs/

TABLE 4. Results after 100 experiments: (Situation A) Batteries enabled with DDQL trained for this situation, (Situation B) Batteries enabled with DDQL not trained for this situation, (Situation C) Batteries disables with DDQL. The R_{loss} is calculated as the relative difference between every enabled-battery case and the ideal non-battery situation.

	Situation A	Situation B	Situation C
R_{max}	36.35	35.45	39.13
$\bar{R}$	34.46	33.51	37.84
$\bar{R}_{loss}(\%)$	-9	-12	0
μ_{min}	7.51	8.31	5.25
$\bar{\mu}$	8.69	9.27	6.02

sufficient accuracy to control the agent with a 10 cm error in position. The ASVs communicate with a central beacon in one of the landing points with a LoRa WAN communication technology.[6] For the processing of the state and the computing of the desired action for every agent, they connect to a server deployed in a Amazon Web Service (AWS) using a 4G module. This server receives continuously the position of the fleet and synthesises the state. The server can be requested from every agent the next optimal position and it will respond with the computed Q-values using the trained network (see Fig. 20 for a connection scheme). The water quality sensor data is also sent to the base station via the web server.

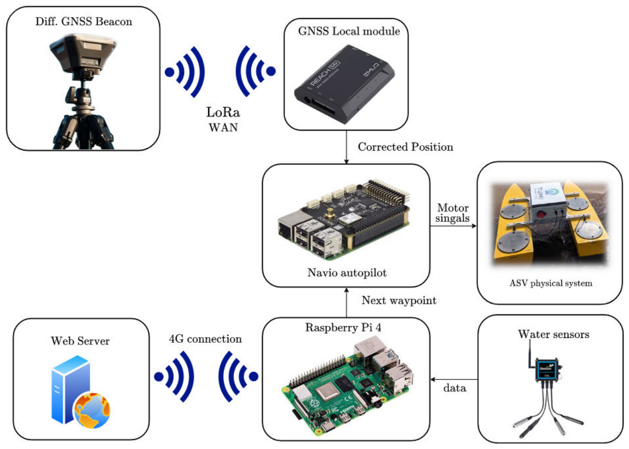

FIGURE 20. Implementation diagram for every agent. This diagram is the same for every agent available all connected to the same Web Server through 4G.

Once the next action is selected for every agent, the Raspberry Pi 4 sends the next waypoint to the Navio autopilot.[7] The autopilot is in charge of generate a local path from the actual position to the next reference position by interpolating and acting on the motors. The Raspberry Pi 4 is also used for the different sensors: LIDAR, Camera, water quality sensors…The LIDAR and Camera will serve the purpose of avoiding minor obstacles for the local path planning (buoys, etc). In case a movement to the desired cell is not possible

[6]https://lora-alliance.org/
[7]https://navio2.emlid.com/

(restricted access due to possible constructions works or similar), the network will provide also the Q-values for a sub-optimal selection (the second highest Q-value).

A. DISCUSSION OF THE RESULTS

Given the previous results, it is possible to discuss the following considerations:

1) The Deep Reinforcement Learning proposed approaches are able to learn the dynamics of the environment and solve the Patrolling Problem with higher rewards than the LMT. The learning process results in policies that overcome the LMT solutions on average rewards for every fleet size by a 15% with DDQL and 17% with Dueling. This shows not only the DRL approach is able to optimize the performance in this problem but also to adapt to different scenario dynamics.
2) Our proposed approach is more efficient than the compared IDQL because it takes advantage of the agents experiences being equivalent. As the agents experiences can be shared, the individual layers can use only one memory replay based on the common experiences. This results in a training 3 times faster in the DDQL case and 2 times faster in the Dueling case respect to the IDQL method. The trajectories of IDQL, in spite of being also acceptable in terms of the metrics, are slightly worse with $N = 1$ and $N = 2$ respect the Dueling architecture (by 18% and 5% on average for each case). For $N = 3$ and $N = 4$, the proposed DDQL algorithm overcomes the IDQL on average by a 3% and 2% respectively.
3) The slightly improvement respect to IDQL is achieved with N less gradient steps as our approach centralize the learning. In an online learning or in an retraining, our approach is very convenient because of the learning efficiency.
4) In the enabled-battery situation, the DDQL approach has also proven to be convenient for inferring a good policy in case one or more ASVs are unavailable for the patrolling task. After the training, the new policy is able to adapt the previous behavior in order to cover the available areas reducing the reward loss on average by a 33% respect the non-trained case.

VII. CONCLUSION

Deep Reinforcement Learning algorithms are powerful methodologies for the resolution of stochastic and high-complex scenarios. The Ypacaraí Lake monitoring task has been proven challenging because the enormous dimensions and the need of an effective and non-homogeneous coverage. When addressing the multiagent case, the high number of different possible paths makes the problem unfeasible for the conventional methodologies and DRL brings the possibility to deal with the scenario only with an RGB representation of it. The Value Iteration methods like Double Deep Q-Learning and Dueling Deep Learning has been proven very efficient in the resolution of such problems: there is no need of a previous model of the environment since it can adapt robustly to different dynamics and interactions.

The multiagent case with these methods is approached by a centralized convolutional neural network to extract the features for the agents to choose their actions. As every agent has its own independent neural network in parallel and because they are equivalents in their actions, the proposed architecture achieves higher rewards in most of the cases respect to the Independent Q-Learning counterpart. Besides, it has been demonstrated that the proposed Dueling architecture achieves better results in the single agent case. Furthermore, the proposed DDQL architecture is 3 times faster in learning as it uses a common experience replay and takes less optimization steps and the proposed Dueling architecture achieves better results with less agents

The achieved results overcomes the common approaches for path planning such as lawn mower trajectories since the optimized policy learned effectively to move to the high idle zones with high importance. This behavior can be also achieved when the battery is simulated: after training, the agents can generalize the policy and overcome a situation in which several agents fault. The results show an improvement of the achieved reward respect to the agents not trained for this kind of situation.

For future works, it is planned to explore other Reinforcement Learning methodologies as Policy Iteration to address a particular case of the multiagent exploration task: the partially observable scenario. The Partially Observable Markov Decision Process will be studied using recurrent networks like Long Short Term Memory (LSTM) for the temporal dependencies in the state inferring. In this regard, a further analysis of wireless connectivity fault-tolerant methods will be interesting, as the proposed method only focused in battery faults and not in communication errors. This is a particular and interesting case because the assumption of an always available connection can be violated in a real implementation. Thus, mechanisms for estimating the movement of other agents based on the previously mentioned RNN could serve this purpose.

Other interesting direction is the use of Bayesian Neural Networks not to minimize the idleness of the zones but also to include uncertainties in the optimization process, like possible obstacles or non-modeled algae-blooms. With this approach, it is considered also interesting to enhance the speed of learning by developing a model-based methodology. As the scenario can be modeled easily with a sufficiently accurate simulator, a further research with this kind of methods could be very interesting. Finally, it will be studied how the different resolutions of the state domain affects the performance by conducting a comparison between Deep Reinforcement Learning and Genetic Algorithms.

ACKNOWLEDGMENT

The authors would like to thank Inko Bovenzi and Jaime Jiménez for the assistance in the correction of this article.

REFERENCES

[1] I. P. López *et al.*, "Technical report: 12th sampling campaign from Ypakarai lake," Multidisciplinary Technol. Res. Center (CEMIT), Nat. Univ. Asuncion (UNA), San Lorenzo, Paraguay, Tech. Rep. 12, 2016.

[2] M. Arzamendia, D. Gregor, D. G. Reina, and S. L. Toral, "An evolutionary approach to constrained path planning of an autonomous surface vehicle for maximizing the covered area of Ypacarai lake," *Soft Comput.*, vol. 23, no. 5, pp. 1723–1734, Mar. 2019.

[3] J. Sánchez-García, J. M. García-Campos, M. Arzamendia, D. G. Reina, S. L. Toral, and D. Gregor, "A survey on unmanned aerial and aquatic vehicle multi-hop networks: Wireless communications, evaluation tools and applications," *Comput. Commun.*, vol. 119, pp. 43–65, Apr. 2018.

[4] S. Y. Luis, D. G. Reina, and S. L. T. Marin, "A deep reinforcement learning approach for the patrolling problem of water resources through autonomous surface vehicles: The Ypacarai lake case," *IEEE Access*, vol. 8, pp. 204076–204093, 2020.

[5] R. S. Sutton and A. G. Barto, *Introduction to Reinforcement Learning*, 1st ed. Cambridge, MA, USA: MIT Press, 1998.

[6] J. Xiao, G. Wang, Y. Zhang, and L. Cheng, "A distributed multi-agent dynamic area coverage algorithm based on reinforcement learning," *IEEE Access*, vol. 8, pp. 33511–33521, 2020.

[7] P. Hernandez-Leal, B. Kartal, and E. Taylor, "A survey and critique of multiagent deep reinforcement learning," *Auto. Agents Multi-Agent Syst.*, vol. 33, pp. 750–797, Oct. 2019.

[8] R. Bellman, *Dynamic Programming. Rand Corporation Research Study*. Princeton, NJ, USA: Princeton Univ. Press, 1957.

[9] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[10] M. M. Drugan, "Reinforcement learning versus evolutionary computation: A survey on hybrid algorithms," *Swarm Evol. Comput.*, vol. 44, pp. 228–246, Feb. 2019.

[11] M. Arzamendia, I. Espartza, D. G. Reina, S. L. Toral, and D. Gregor, "Comparison of Eulerian and Hamiltonian circuits for evolutionary-based path planning of an autonomous surface vehicle for monitoring Ypacarai lake," *J. Ambient Intell. Humanized Comput.*, vol. 10, no. 4, pp. 1495–1507, Apr. 2019.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A Graves, M. Riedmiller, A. k. Fidjeland, G. Ostrovski, and S. Petersen, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] K. Lin, R. Zhao, Z. Xu, and J. Zhou, "Efficient collaborative multi-agent deep reinforcement learning for large-scale fleet management," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 1–13.

[14] Y. Wang, B. Han, T. N. Wang, H. Dong, and C. Zhang, "Off-policy multi-agent decomposed policy gradients," 2020, pp. 1–20, *arXiv:2007.12322*. [Online]. Available: https://arxiv.org/abs/2007.12322

[15] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6380–6391.

[16] J. Foerster, N. Nardell, G. Farquhar, T. Afouras, P. H. S. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn. (ICML)*, vol. 3, 2017, pp. 1879–1888.

[17] X. Zhou, P. Wu, H. Zhang, W. Guo, and Y. Liu, "Learn to navigate: Cooperative path planning for unmanned surface vehicles using deep reinforcement learning," *IEEE Access*, vol. 7, pp. 165262–165278, 2019.

[18] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-UAV target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264–146272, 2019.

[19] C. Chen, J. Jiang, N. Lv, and S. Li, "An intelligent path planning scheme of autonomous vehicles platoon using deep reinforcement learning on network edge," *IEEE Access*, vol. 8, pp. 99059–99069, 2020.

[20] J. Fraga, J. Sousa, G. Cabrita, P. Coimbra, and L. Marques, *Squirtle: An ASV for Inland Water Environmental Monitoring*. Madrid, Spain: Springer, 2014, pp. 33–39.

[21] E. Rimon, I. Kamon, and J. F. Canny, "Local and global planning in sensor based navigation of mobile robots," in *Robotics Research*, Y. Shirai and S. Hirose, Eds. London, U.K.: Springer, 1998, pp. 112–123.

[22] F. Peralta, M. Arzamendia, D. Gregor, D. G. Reina, and S. Toral, "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The Ypacarai lake case-study," *Sensors*, vol. 20, no. 5, p. 1488, Mar. 2020.

[23] F. P. Samaniego, D. G. Reina, S. L. T. Marin, M. Arzamendia, and D. O. Gregor, "A Bayesian optimization approach for water resources monitoring through an autonomous surface vehicle: The Ypacarai lake case study," *IEEE Access*, vol. 9, pp. 9163–9179, 2021.

[24] V. Francois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," 2018, *arXiv:1811.12560*. [Online]. Available: http://arxiv.org/abs/1811.12560

[25] Q. Zhang, J. Lin, Q. Sha, B. He, and G. Li, "Deep interactive reinforcement learning for path following of autonomous underwater vehicle," *IEEE Access*, vol. 8, pp. 24258–24268, 2020.

[26] X. Zhang, C. Wang, Y. Liu, and X. Chen, "Decision-making for the autonomous navigation of maritime autonomous surface ships based on scene division and deep reinforcement learning," *Sensors*, vol. 19, no. 18, p. 4055, Sep. 2019.

[27] M. S. Imtiaz and J. Wang, "A multiagent reinforcement learning control approach to environment exploration," in *Proc. SoutheastCon*, Mar. 2017, pp. 1–4.

[28] W. Shi, J. Li, H. Wu, C. Zhou, N. Cheng, and X. Shen, "Drone-cell trajectory planning and resource allocation for highly mobile networks: A hierarchical DRL approach," *IEEE Internet Things J.*, early access, Aug. 28, 2020, doi:10.1109/JIOT.2020.3020067.

[29] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PLoS ONE*, vol. 12, no. 4, Apr. 2017, Art. no. e0172395.

[30] T. Chu, S. Qu, and J. Wang, "Large-scale multi-agent reinforcement learning using image-based state representation," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 7592–7597.

[31] L. Matignon, G. J. Laurent, and N. L. Fort-Piat, "Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems," in *Knowledge Engineering Review*, vol. 27. 2012.

[32] H. M. Schwartz, *Multi-Agent Machine Learning: A Reinforcement Approach*, vol. 9781118362. Hoboken, NJ, USA: Wiley, 2014.

[33] C. Piciarelli and G. L. Foresti, "Drone patrolling with reinforcement learning," in *Proc. 13th Int. Conf. Distrib. Smart Cameras*, Sep. 2019, pp. 1–6.

[34] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV coverage path planning under varying power constraints using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, 2020. [Online]. Available: https://arxiv.org/abs/2003.02609

[35] L. Lv, S. Zhang, D. Ding, and Y. Wang, "Path planning via an improved DQN-based learning policy," *IEEE Access*, vol. 7, pp. 67319–67330, 2019.

[36] A. Krishna Lakshmanan, R. Elara Mohan, B. Ramalingam, A. Vu Le, P. Veerajagadeshwar, K. Tiwari, and M. Ilyas, "Complete coverage path planning using reinforcement learning for tetromino based cleaning and maintenance robot," *Autom. Construct.*, vol. 112, Apr. 2020, Art. no. 103078.

[37] M. Wooldridge *et al.*, *An Introduction to Multiagent Systems*, 2nd ed. Hoboken, NJ, USA: Wiley, 2013.

[38] Y. Cheva, "Theoretical analysis of the multi-agent patrolling problem," in *Proc. IEEE/WIC/ACM Int. Conf. Intell. Agent Technol.*, Jul. 2004, pp. 302–308.

[39] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. D. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proc. 33rd Int. Conf. Mach. Learn. (ICML)*, 2016, vol. 4, no. 9, pp. 2939–2947.

[40] C. H. Chung, K. C. Wang, K. T. Liu, Y. T. Wu, C. C. Lin, and C. Y. Chang, "Path planning algorithm for robotic lawnmower using RTK-GPS localization," in *Proc. Int. Symp. Community-Centric Syst. (CcS)*, Sep. 2020, pp. 1–4.

[41] G. Giardini and T. Kalmár-Nagy, "Genetic algorithm for combinatorial path planning: The subtour problem," *Math. Problems Eng.*, vol. 2011, pp. 1–31, May 2011.

[42] E. Meyer, H. Robinson, A. Rasheed, and O. San, "Taming an autonomous surface vehicle for path following and collision avoidance using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 41466–41481, 2020.

[43] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," 2017, *arXiv:1703.03864*. [Online]. Available: https://arxiv.org/abs/1703.03864

[44] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

SAMUEL YANES LUIS was born in Tenerife, Spain, in 1997. He received the M.S. degree in electronics and robotics engineering from the University of Seville, Spain, in 2019. He is currently pursuing the Ph.D. degree with the Department of Electronic Engineering, University of Seville. His research interests include reinforcement learning and machine learning for robotics applications, autonomous vehicles control, and artificial vision.

DANIEL GUTIÉRREZ REINA received the B.E. degree (Hons.) in electronic engineering, the M.S. degree in electronics and telecommunications, and the Ph.D. degree (Hons.) in electronic engineering from the University of Seville, Seville, Spain, in 2009, 2011, and 2015, respectively. He worked as an Assistant Professor with Loyola University from October 2018 to April 2019. He has been a Visitor Researcher with Liverpool John Moores University, U.K., the Free University of Berlin, Germany, the Colorado School of Mines, USA, and Leeds Beckett University, U.K. He is currently working as an Assistant Professor with the Electronic Engineering Department, University of Seville. He has published about 40 articles in JCR journals with impact factor. His current research interests include the application of meta-heuristic algorithms to solve wireless multi-hop network optimization problems, such as MANETs, VANETs, DTNs, and FANETs. He is a part of the editorial board of several journals, such as *International Journal of Distributed Sensor Networks* (SAGE), *Electronics* (MDPI), *Future Internet* (MDPI), and *Wireless Communications and Mobile Computing* (Hindawi), organizing numerous SI for these journals.

SERGIO L. TORAL MARÍN was born in Rabat, Morocco, in 1972. He received the M.S. and Ph.D. degrees in electrical and electronic engineering from the University of Seville, Spain, in 1995 and 1999, respectively. He is currently a Full Professor with the Department of Electronic Engineering, US. He is actually an author or coauthor of 95 articles in major international peer-reviewed journals (with JCR impact factor) and of over 100 articles in well-established international conferences and workshops. His research interests include ad hoc networks and their routing protocols, flying ad hoc networks, deployment of unmanned vehicles, and intelligent transportation systems.