IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 26 July 2022, accepted 4 August 2022, date of publication 8 August 2022, date of current version 16 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3197628

RESEARCH ARTICLE

Mobile Robot Path Planning Using a QAPF Learning Algorithm for Known and Unknown Environments

ULISES OROZCO-ROSAS[1], (Member, IEEE), KENIA PICOS[1], JUAN J. PANTRIGO[2], ANTONIO S. MONTEMAYOR[2], AND ALFREDO CUESTA-INFANTE[2]

[1]CETYS Universidad, Tijuana, Baja California 22210, Mexico
[2]Universidad Rey Juan Carlos, Móstoles, 28933 Madrid, Spain

Corresponding author: Ulises Orozco-Rosas (ulises.orozco@cetys.mx)

This work was supported in part by the Coordinación Institucional de Investigación and the Centro de Innovación y Diseño (CEID) of Centro de Enseñanza Técnica y Superior (CETYS Universidad); in part by the Consejo Nacional de Ciencia y Tecnología (CONACYT), Mexico; in part by the Spanish Government Research Funding under Grant RTI2018-098743-B-I00 (MICINN/FEDER) and Grant PID2021-128362OB-I00 (MICINN/FEDER); and in part by the Comunidad De Madrid Research Funding under Grant Y2018/EMT-5062.

ABSTRACT This paper presents the computation of feasible paths for mobile robots in known and unknown environments using a QAPF learning algorithm. Q-learning is a reinforcement learning algorithm that has increased in popularity in mobile robot path planning in recent times, due to its self-learning capability without requiring a priori model of the environment. However, Q-learning shows slow convergence to the optimal solution, notwithstanding such an advantage. To address this limitation, the concept of partially guided Q-learning is employed wherein, the artificial potential field (APF) method is utilized to improve the classical Q-learning approach. Therefore, the proposed QAPF learning algorithm for path planning can enhance learning speed and improve final performance using the combination of Q-learning and the APF method. Criteria used to measure planning effectiveness include path length, path smoothness, and learning time. Experiments demonstrate that the QAPF algorithm successfully achieves better learning values that outperform the classical Q-learning approach in all the test environments presented in terms of the criteria mentioned above in offline and online path planning modes. The QAPF learning algorithm reached an improvement of 18.83% in path length for the online mode, an improvement of 169.75% in path smoothness for the offline mode, and an improvement of 74.84% in training time over the classical approach.

INDEX TERMS Path planning, Q-learning, artificial potential field, reinforcement learning, mobile robots.

I. INTRODUCTION

The path planning problem is a fundamental issue in mobile robot navigation because of the need of having algorithms to convert high-level specifications of tasks from humans into low-level descriptions of how to move [1]. There are some good reasons to develop efficient planning algorithms. For example, the necessity to get machines that can solve some tasks that are difficult to solve for humans involves modeling planning problems, designing efficient algorithms, and developing robust implementations. Another reason is that planning algorithms have been successfully used in various industries and academic disciplines like robotics, manufacturing, and aerospace applications, among others [2].

The path planning problem consists in how the mobile robot (MR) determines to move in an environment to its predefined goal point without colliding with the obstacles presented in the environment. This concerns the computation of a collision-free path between the start point and the goal point [3]. Path planning varies according to different environments that the MR faces, such as a known environment, partially known environment, or unknown environment. Among the types of environments, the partially known environment is the most practical where some areas within the

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

environment have already been known before the navigation. Furthermore, path planning can be classified into static and dynamic, depending on the nature of the obstacles. In static path planning, the position and orientation of the obstacles are unchanged with time, while in dynamic path planning, the obstacles are free to move in the environment [4].

The path planning problem solution has been addressed from diverse methods [5]. One approach is reinforcement learning, due to it can be employed for MR path planning for known and unknown environments. Q-learning is a reinforcement learning algorithm that uses a scalar reinforcement signal or a reward to interact with a complex environment [6]. Q-learning maps situations to actions, to maximize a numerical reward through systematic learning. To obtain its own experience, the agent offers a trade-off between exploration and exploitation, so it not only has to exploit what it already knows through greater action in the current experience, but also must explore what action will work better in the future [7]. The mobile robot (MR) which is known as the agent receives a reward for collision-free action and receives a penalty when it collides with the obstacles [5].

In Q-learning, the reward concept is a key difference concerning both supervised and unsupervised solutions. The reward is less informative than it is in supervised learning, where the agent is given the correct actions to perform. Unfortunately, information regarding correct actions is not always available. However, the reward is more informative than unsupervised learning, where no explicit comments are made regarding performance [8].

Q-learning has been employed in mobile robot path planning due to its self-learning capability without requiring a priori model of the environment. Although, Q-learning presents slow convergence to the optimal solution, notwithstanding such an advantage [9]. To address this limitation, the concept of partially guided Q-learning through the APF weighting is employed wherein, the artificial potential field (APF) method is utilized to improve the classical Q-learning approach. The APF method is employed in path planning due to its effectiveness to provide smooth and safe paths. However, it presents disadvantages like the local minima problem, or the goal non-reachable with obstacles nearby, among others [10]. Therefore, the proposed QAPF learning algorithm for path planning can enhance learning speed and improve final performance using the combination of Q-learning and the APF method. In this manner, the QAPF learning algorithm overcomes or mitigates the disadvantages presented by both conventional methods.

The main contribution of this paper is the development of a new robust algorithm based on Q-learning and the APF method to solve path planning problems for mobile robots. The proposed QAPF learning algorithm presents short, smooth, and collision-free paths. The proposed QAPF learning algorithm is extensively validated against a path planning algorithm based on the classical Q-learning (CQL) approach. Both planning algorithms are tested in benchmark environments and compared concerning measures such as path length, path smoothness, and computation time. The contributions of this paper can be summarized as follows:

- A reinforcement learning-based algorithm combined with the artificial potential field method called QAPF learning algorithm is proposed to solve mobile robot path planning problems.
- The QAPF learning algorithm includes three operations: exploration, exploitation, and APF weighting to overcome the limitations presented by the classical Q-learning approach in path planning.
- A set of experiments and studies demonstrate that the proposed QAPF learning algorithm successfully solves diverse path planning problems arising from numerous complex environments that the mobile robot can face.

The organization of this paper is as follows. Section II summarizes the literature review and related work. Section III describes the path planning problem, the Q-learning algorithm, and the APF method. Section IV explains in detail the proposed QAPF learning algorithm. Section V presents the results of simulations that demonstrate the applicability of the proposal. Finally, the conclusions of this paper are drawn in Section VI.

II. RELATED WORK

In the literature, there are several proposals to address path planning problems using reinforcement learning [11], [12]. Some proposals based on reinforcement learning have been combined with other techniques to improve performance. In [13], the combination of reinforcement learning with an improved artificial potential field is presented to solve path planning problems. Another example, now combining reinforcement learning with a metaheuristic can be found in [14]. In that work, a reinforcement learning-based grey wolf optimizer (RLGWO) algorithm is presented to solve unmanned aerial vehicles (UAVs) path planning problems.

Furthermore, there are numerous proposals to address path planning problems using Q-learning. A few of them, which are worth mentioning, are described next. In [15], a modified Q-learning algorithm is presented to solve the path planning problem in product assembly. In that work, the proposed algorithm speeds up the convergence speed by adding a dynamic reward function, optimizes the initial Q table by introducing knowledge and experience through the case-based reasoning algorithm, and prevents entry into the trapped area through the obstacle avoiding method.

Another example is presented by Maoudj and Hentout [16] They propose an efficient Q-learning (EQL) algorithm to overcome the limitations of slow and weak convergence and ensure an optimal collision-free path in less possible time. In this regard, in that work a reward function is proposed to initialize the Q table and provide the robot with prior knowledge of the environment, followed by an efficient selection strategy proposal to accelerate the learning process through search space reduction while ensuring a rapid convergence toward an optimized solution.

In [17], a flexible Q-learning-based model is proposed to handle continuous space problems for decentralized multi-agent robot navigation in cluttered environments. In that research, an agent-level decentralized collision avoidance low-cost model is presented. Furthermore, a method to merge non-overlapping Q-learning features is employed to reduce its size significantly and make it possible to solve more complicated scenarios with the same memory size.

Bulut proposed an improved epsilon-greedy Q-learning (IEGQL) algorithm to enhance efficiency and productivity regarding path length and computational cost [18]. The IEGQL presents a reward function that ensures the environment's knowledge in advance for a mobile robot, and mathematical modeling is presented to provide the optimal selection besides ensuring a rapid convergence.

Some proposals have combined Q-learning with other algorithms to improve performance. An example is found in [19], where a combination of the Manhattan distance and Q-learning (CMD-QL) is presented to improve the convergence speed. In the CMD-QL, the Q table is firstly initialized with Manhattan distance to enhance the learning efficiency of the initial stage of Q-learning; secondly, the selection strategy of the ϵ-greedy action is improved to balance the exploration-exploitation relationship of the mobile agent's actions. The CMD-QL was tested under known, partially known, and unknown environments, respectively. The results revealed that the CMD-QL can converge to the optimal path faster than the classical Q-learning method.

An example, now combining Q-learning with a metaheuristic can be found in [20], Sadhu *et al.* proposed a modification of the Firefly Algorithm by including the Q-learning framework into it, called QFA. The QFA has been employed to plan the path of the robot arm end-effector such that it can reach a pre-assigned goal position by traversing the minimum possible distance while dodging the obstacles present in the environment.

Another work is presented by Low *et al.* [5]. They present an improved Q-learning based on the Flower Pollination Algorithm (FPA) for mobile robot path planning. Through the integration of the prior knowledge obtained from the FPA into the classical Q-learning, the initialization of the Q-values serves as a good exploration foundation to accelerate the learning process of the mobile robot.

There are some proposals employing deep Q-learning like in [21]. Where, Wu *et al.* presented a tailored design of state and action spaces and a dueling deep Q-network. That work consists of a deep reinforcement learning method for the autonomous navigation and obstacle avoidance of unmanned surface vehicles. In that work, the proposal outperforms related methods in the efficiency of exploration and the speed of convergence in static and dynamic environments.

III. PRELIMINARIES

In this section, we present the problem definition and the fundamentals of the proposed QAPF learning algorithm for mobile robot path planning. We start with a general definition of the path planning problem. Next, we explain in detail the Q-learning algorithm. Last, we conclude this section with the fundamentals of the artificial potential field method.

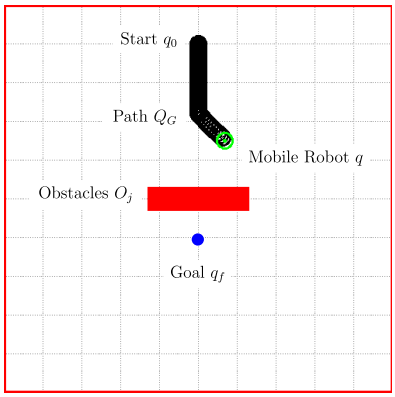


FIGURE 1. Path planning problem definition.

A. PATH PLANNING PROBLEM DEFINITION

Path planning is a problem that requires finding a continuous path, Q_G, between a given start point, q_0, and the goal point, q_f, for a particular system, subject to a variety of constraints [22]. Under this general definition, in this paper, we have defined the problem in a simplified form, which is described as follows.

The mobile robot (MR) environment is defined as a two-dimensional map that includes a set of obstacles O_j, $j = 1, 2, \ldots, n$, where n is the number of the obstacles in the environment. The instantaneous position of the MR is denoted by q, which is represented by its coordinates, physical radius, and the angle of orientation, i.e., $q(x, y, \phi, \theta)$. In this definition, we have assumed a circular occupancy area for the MR. The path planning goal is to find a feasible sequence Q_G of configurations that can drive the MR from a start point, q_0, to a goal point, q_f, in the x-y plane. Fig. 1 depicts the definition.

B. Q-LEARNING

Reinforcement learning, inspired by animal learning in psychology, learns optimal decision-making strategies from experience. It defines any decision maker as an agent and everything outside the agent as the environment. The agent aims to maximize the accumulated reward and obtains a reward value as a feedback signal for training through interaction with the environment [23]. Reinforcement learning is categorized into two groups: model-based methods and model-free methods.

For the model-based methods, always utilize a model of the environment to predict rewards for unseen state-action pairs. For the model-free methods, we can divide into two branches: policy-based methods and value-based methods. Specifically, the policy-based methods aim to generate a policy, in which the input is a state, and the output is an action.

These works apply deterministic policies, which generate an action directly. For value-based methods, the action with maximum Q-value over all the possible actions is selected as the best action [24]. Therefore, Q-learning is a value-based method. During the learning, the agent performs an action with the highest expected Q-values to estimate the optimal policy [14].

Q-learning [25] is model-free reinforcement learning, which combines theories, including the Bellman equations and Markov decision process (MDP), with temporal-difference (TD) learning. It is a way for agents to learn how to act optimally in controlled Markovian domains. Its form is one-step Q-learning, which is defined by Eq. (1).

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha(r + \gamma \max(Q(s', \forall a'))) \quad (1)$$

where $Q(s,a)$ estimates the action value after applying an action a in state s, α is the learning rate, γ is the discount factor, and r is the immediate reward received [26]. The Q-learning main components are as follows.

1) Agent: The learner that can interact with the environment via a set of sensors and actuators.
2) Environment: Everything that interacts with an agent, i.e., everything outside the agent.
3) Policy: A mapping from perceived states set, S, to the actions set, A.
4) Reward function: A mapping from state-action pairs to a scalar number.
5) Q-value: The total amount of reward an agent can expect to accumulate over the future, starting from that state.

Q-learning works by successively improving its evaluations of the quality of actions at particular states. Learning proceeds similarly to Sutton's method of temporal differences: an agent tries an action at a particular state and evaluates its consequences in terms of the immediate reward or penalty it receives and its estimate of the value of the state to which it is taken. By trying all actions in all states repeatedly, it learns which are best overall, judged by long-term discounted reward [25].

In Q-learning, the agent's experience consists of a sequence of distinct stages or episodes. In the nth episode, the agent:

1) Observes its current state s_t
2) Selects and performs an action a_t
3) Observes the subsequent state s_{t+1}
4) Receives an immediate payoff r_t

Then, the agent adjusts its Q values using a learning rate α, according to: note that this description assumes a look-up table representation for the $Q_{m\times n}(s_t, a_t)$.

A learning agent is composed of two fundamental parts, a learning element, and a performance element. The design of a learning element is dictated by what type of performance element is used, which functional component is to be learned, how that functional component is represented, and what kind of feedback is available [8].

C. ARTIFICIAL POTENTIAL FIELD

The main idea of the artificial potential field (APF) method [27] is to establish an attractive potential field force around the goal point, as well as to establish a repulsive potential force around obstacles [28]. By this idea, the APF method employs attractive and repulsive components to draw the MR to its goal while keeping it away from obstacles.

Therefore, the total artificial potential field, $U(q)$, includes two terms, the attractive potential function, $U_{att}(q)$, and the repulsive potential function, $U_{rep}(q)$. The total artificial potential field, $U(q)$, is then the sum of these two potential functions, as indicated in Eq. (2).

$$U(q) = U_{att}(q) + U_{rep}(q) \quad (2)$$

The attractive potential function is described by Eq. (3), where q represents the current MR position. The goal point is represented by q_f and k_{att} is a positive scalar-constant that represents the attractive proportional gain of the function.

$$U_{att}(q) = \frac{1}{2}k_{att}(q - q_f)^2 \quad (3)$$

The repulsive potential function is denoted by Eq. (4), where ρ_0 represents the limit distance of influence of the repulsive potential field and ρ is the shortest distance from the MR to the obstacle. The influence of the repulsive potential field, $U_{rep}(q)$, is presented in two cases. The first case is presented when the MR is under the influence of the obstacle, that is, if the distance from the robot to the obstacle, ρ, is less or equal to the limit distance of influence, ρ_0. Otherwise, in the second case, the repulsive potential field will be zero and the MR will be free of the influence of that obstacle. The selection of the distance ρ_0 depends on the MR maximum speed. The k_{rep} is a positive scalar-constant that represents the repulsive proportional gain of the function. Therefore, the repulsive potential function has a limited range of influence, and it prevents the movement of the MR from being affected by a distant obstacle [29].

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}(\frac{1}{\rho} - \frac{1}{\rho_0})^2, & \text{if } \rho \le \rho_0 \\ 0, & \text{if } \rho > \rho_0 \end{cases} \quad (4)$$

The artificial potential field method is widely employed in path planning for mobile robots due to its simplicity, mathematical elegance, and effectiveness in providing smooth and safe planning [10]. Therefore, in this paper, the artificial potential field method is utilized to improve the classical Q-learning approach. In that sense, the proposed QAPF learning algorithm for path planning can enhance learning speed and improve final performance using the combination of Q-learning and the APF method.

IV. QAPF LEARNING ALGORITHM FOR PATH PLANNING

To improve the disadvantage of slow convergence in the classical Q-learning, the QAPF learning algorithm is proposed in this work, which includes three operations: exploration, exploitation, and APF weighting.

The proposed QAPF learning algorithm combines Q-learning and the artificial potential field (APF) method to improve performance of the MR path planning. The objective of this section is to explain in detail the proposed QAPF learning algorithm. Therefore, firstly the classical Q-learning (CQL) pseudocode (Algorithm 1) is presented. Then, the proposed QAPF learning pseudocode (Algorithm 2) is explained. Lastly, the path generator pseudocode (Algorithm 3) is provided. Both the CQL algorithm and the QAPF learning algorithm employ Algorithm 3 to generate the path.

A. Q-LEARNING ALGORITHM

Algorithm 1 presents the CQL pseudocode for MR path planning. The CQL algorithm employs the next input parameters: the goal point, q_f, and the environment information, O_j. The objective of the CQL algorithm is to obtain the learning values $Q_{m\times n}$ to build the path, Q_G, through the Algorithm 3. The parameter that Algorithm 1 returns is the array $Q_{m\times n}$.

Algorithm 1 in line 1 initializes to zero the learning values $Q_{m\times n}$. From lines 2 to 11, there is the learning iterative process. The stop condition for the learning iterative process is presented when the maximum number of episodes, N_{ep}, is reached.

Algorithm 1 CQL

Input: goal point q_f and environment information O_j
Output: learning values $Q_{m\times n}$

```
1  initialize Q_{m×n}(s_t, a_t) ← {0}
2  for each episode do
3      set s_t ← a random state from the states set S
4      while s_t ≠ q_f and safe do
5          choose the best a_t in s_t by using Q_{m×n}
6          perform action a_t and receive reward r
7          find out the new state s_{t+1}
8          update Q_{m×n}(s_t, a_t) using Eq. 1
9          s_t ← s_{t+1}
10     end
11 end
12 return Q_{m×n}
```

In line 3, a random state is assigned to the current state, s_t. From lines 4 to 10, the explore-exploit process is performed. The stop condition is presented when the current state, s_t, is equal to the goal point, q_f, or the Boolean flag *safe* is False. The objective of the flag *safe* is to serve as a stop condition if a collision is presented with the obstacles or when the MR steps into the limits of the workspace.

In line 5, the best a_t in s_t is chosen using $Q_{m\times n}$. In line 6, the action, a_t, is performed and a reward is received. Next, in line 7, the new state, s_{t+1}, is computed. Then, in line 8, the learning value, $Q_{m\times n}(s_t, a_t)$, is updated using Eq. 1. Last, in line 9. The new state, s_{t+1}, is assigned to the current state s_t. In the end, the Algorithm 1 returns the resultant learning values $Q_{m\times n}$ to build the path, Q_G, through the Algorithm 3.

B. QAPF LEARNING ALGORITHM

Algorithm 2 presents the QAPF pseudocode for MR path planning. The QAPF learning algorithm employs the next input parameters: the goal point, q_f, that the MR must achieve and the environment information, which is composed of n rectangular obstacles in $O_j(x_j, y_j, l_j, w_j)$ format. The main objective of the proposed QAPF learning algorithm is to obtain the learning values $Q_{m\times n}$ to build the path, $Q_G = [q_0, q_1, \ldots, q_f]$, through the Algorithm 3, i.e., a collision-free path that will drive the MR to achieve the goal point with the minimum path length. The parameter that Algorithm 2 returns is the array $Q_{m\times n}$.

Algorithm 2 in line 1 initializes to zero the learning values $Q_{m\times n}$. From lines 2 to 21, there is the learning iterative process of the proposed QAPF algorithm to find the learning values $Q_{m\times n}$ to generate the path through the Algorithm 3. The stop condition for the learning iterative process is presented when the maximum number of episodes, N_{ep}, is reached.

Algorithm 2 QAPF

Input: goal point q_f and environment information O_j
Output: learning values $Q_{m\times n}$

```
1  initialize Q_{m×n}(s_t, a_t) ← {0}
2  for each episode do
3      set s_t ← a random state from the states set S
4      while s_t ≠ q_f and safe do
5          if ζ < uniform random number then
6              compute probability p_i using Eq. 5
7              compute APF weighting using Eq. 6
8              choose a_t in s_t by using APF weighting
9          else
10             if ζ < uniform random number then
11                 choose the best a_t in s_t by using Q_{m×n}
12             else
13                 choose random a_t in s_t
14             end
15         end
16         perform action a_t and receive reward r
17         find out the new state s_{t+1}
18         update Q_{m×n}(s_t, a_t) using Eq. 1
19         s_t ← s_{t+1}
20     end
21 end
22 return Q_{m×n}
```

In line 3, a random state is assigned to the current MR state, s_t. From lines 4 to 20, there is the decision iterative process to operate on artificial potential field (APF) weighting, exploitation, or exploration. The stop condition for the decision process is presented when the current state, s_t, is equal to the goal point, q_f, or the Boolean flag *safe* is False. The main objective of the flag *safe* is to serve as a stop condition if a collision is presented with the obstacles or when the MR steps into the limits of the workspace.

In line 5, a uniform random number is generated. If the uniform random number is greater than the decision rate, $\zeta \in (0, 1]$, the APF weighting procedure will be performed. Otherwise, the explore-exploit approach will be performed.

For the APF weighting, we are going to employ the Moore neighborhood, which is defined on a two-dimensional square lattice and is composed of a central cell, in this case, the current MR state s_t, and the eight cells that surround it.

The probabilities assigned to the neighbor cells of s_t are inversely proportional to their total artificial potential field. A neighbor cell with the lowest total artificial potential field has the greatest probability of being assigned to s_{t+1}, while the neighbor cell with the highest total artificial potential field has the lowest probability of being assigned to s_{t+1}. A random number determines which neighbor cell is selected.

Line 6 computes the probability p_i for $i = 1, \ldots, k$, where k is the number of neighbor cells. The probability p_i is defined by Eq. (5) that computes the inverse of the total artificial potential field, Eq. (2), for the state of each neighbor cell q_i.

$$p_i = \frac{1}{U(q_i)} \tag{5}$$

In line 7, the standard (unit) APF weighting function is computed. The APF weighting function $\sigma : \mathbb{R}^k \rightarrow (0, 1)^k$ is defined by Eq. (6) for $i = 1, \ldots, k$ and $\mathbf{p} = (p_1, \ldots, p_k) \in \mathbb{R}^k$.

$$\sigma(\mathbf{p})_i = \frac{p_i}{\sum_{j=1}^{k} p_j} \tag{6}$$

The cumulative probabilities obtained by the APF weighting function are used in choosing the action a_t in the state s_t (line 8). First, the probabilities contained by $\mathbf{p}$ are sorted. Then, a random number between zero and one is generated. Starting at the top of the list, the first neighbor cell with a cumulative probability that is greater than the random number is selected for the state s_{t+1}.

If the APF weighting procedure is not executed, the explore-exploit approach will be performed (line 9). In line 10, a uniform random number is generated. If the uniform random number is greater than the decision rate, ζ, the exploitation procedure will be performed. Otherwise, the exploration procedure is performed.

Exploitation highlights the direction of search to control the search within the neighborhood of the best solutions obtained by exploration. In that sense, in line 11, the best action, a_t, in the state, s_t, is chosen using the learning values, $Q_{m\times n}$.

If the exploitation procedure is not executed, the exploration approach will be performed (line 12). Exploration involves the process of determining different candidate solutions by randomly exploring the search space. In that way, in line 13, a random action, a_t, in the state, s_t, is executed.

Once the action, a_t, has been chosen through the APF weighting, exploitation, or exploration procedure. In line 16, the action, a_t, is performed and a reward is received. The agent's sole objective is to maximize the total reward that it receives over the long run. The reward signal thus defines what are the good and bad events for the agent [30].

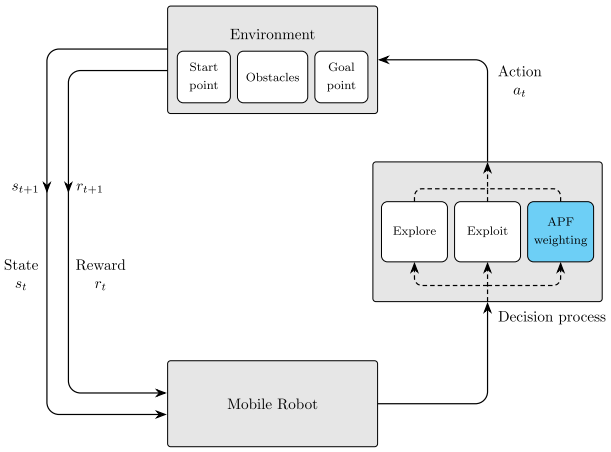


FIGURE 2. Generalized framework of the proposed QAPF learning algorithm.

Next, in line 17, the new state, s_{t+1}, is computed using Eq. 7. Each action, a_t, when applied from the current state, s_t, produces a new state, s_{t+1}, as specified by the state transition function, f, that is $f(s_t, a_t) = s_t + a_t$, in which $s_t \in S$ and $a_t \in A$.

$$s_{t+1} = f(s_t, a_t) \tag{7}$$

Then, in line 18. The learning value, $Q_{m\times n}(s_t, a_t)$, is updated using Eq. 1. Where, $Q_{m\times n}(s_t, a_t)$ estimates the action value after applying an action a_t in state s_t. Last, in line 19. The new state, s_{t+1}, is assigned to the current state s_t to continue with the iterative process until the goal point, q_f, is reached or an unsecured condition occurs.

In the end, the Algorithm 2 returns the resultant learning values $Q_{m\times n}$ to build the path Q_G through the Algorithm 3.

Fig. 2 summarizes the process described by Algorithm 2. The learner and decision maker is the mobile robot (MR). Everything outside the MR is considered as the environment. These interact continually, the MR selecting actions through the operations of exploration, exploitation, or APF weighting, and the environment responding to these actions and presenting new situations to the MR. The environment also gives rise to rewards, special numerical values that the MR seeks to maximize over time through its choice of actions [30].

The classical Q-learning approach employs random numbers or zeros to initialize the learning values $Q_{m\times n}$. In this work, we use zeros for both, the CQL algorithm and the QAPF learning algorithm. Therefore, the learning efficiency in the initial stage is low. Moreover, the balance between exploration and exploitation of actions is a fundamental factor affecting the convergence speed of the classical Q-learning approach. If the exploration rate is too high, it will cause some high-value behaviors not to be effectively used and affect the path planning; while the exploration rate is too low, it will make some high-value behaviors unable to be

effectively trained in a short time and slow down the learning convergence [19]. To overcome this problem, the proposed QAPF learning algorithm employs the concept of partially guided Q-learning through the APF weighting to virtualize the a priori environment into a total artificial potential field and compute the appropriate learning values, $Q_{m\times n}$. In consequence, to speed up the convergence.

The convergence is presented when the learning values $Q_{m\times n}$ remain unchanged or are under a certain bound set in advance. In that sense, the stop condition in this work for the learning process is presented when the maximum number of episodes, N_{ep}, is reached. Therefore, the convergence of the learning values $Q_{m\times n}$ means that the MR can plan a feasible path based on its experience [9]. However, the MR will not necessarily find the global shortest path. It will find a nearly optimal or optimal path in the best of cases. Consequently, the shortest path is defined on the basis of the experience the MR has.

C. PATH GENERATOR ALGORITHM

The MR uses Algorithm 3 to generate the path from the start point, q_0, to the goal point, q_f, under a known, partially known, or unknown environment. Algorithm 3 uses the learning values $Q_{m\times n}$ to generate the best path, Q_G, in the offline and online path planning modes. During the path generation, the environment is verified to see if new obstacles not considered at the beginning were added or dynamic obstacles have changed their position. If the environment change, the environment information, O_j, is updated.

Algorithm 3 Path Generator

Input: start point q_0, goal point q_f, environment information O_j, and learning values $Q_{m\times n}$
Output: path Q_G

1 $i \leftarrow 0$
2 set $s_t \leftarrow q_0$
3 **while** $s_t \neq q_f$ *and safe* **do**
4 $i \leftarrow i + 1$
5 verification of the environment
6 **if** *environment has changed* **then**
7 update environment information O_j
8 **end**
9 choose the best a_t in s_t by using $Q_{m\times n}$
10 perform action a_t
11 find out the new state s_{t+1}
12 $s_t \leftarrow s_{t+1}$
13 $q_i \leftarrow s_t$
14 **end**
15 $Q_G \leftarrow [q_0, q_1, \ldots, q_i]$
16 **return** Q_G

Algorithm 3 employs the input parameters: the start point, q_0, the goal point, q_f, the environment information, O_j, and the learning values $Q_{m\times n}$. The objective of the path generator algorithm is to build the path, Q_G, that is composed of objective points from the start point to the goal point. Therefore, the parameter that Algorithm 3 returns is the array $Q_G = [q_0, q_1, \ldots, q_f]$.

Algorithm 3 in line 1 initializes to zero an index, i, that will be employed by the objective points. In line 2, the start point, q_0, is assigned to the current state s_t. From lines 3 to 14, there is the path generator iterative process to build the path Q_G. The stop condition is presented when the current state, s_t, is equal to the goal point, q_f, or the Boolean flag *safe* is False.

In line 4, the index, i, is increased by one. Next, the verification of the environment is performed (line 5). If the environment has changed, then the environment information, O_j, is updated (line 7). In line 9, the best action, a_t, in the current state, s_t, is chosen using the learning values, $Q_{m\times n}$. Once the action, a_t, has been chosen, the action, a_t, is performed (line 10). Now, in line 11, the new state, s_{t+1}, is computed using Eq. 7. Last, in line 12. The new state, s_{t+1}, is assigned to the current state s_t, and the current state is assigned to the objective point q_i (line 13). In the end, the Algorithm 3 returns a collision-free path between the start point, q_0, and the goal point, q_f, if the flag *safe* remains True. And, an effective path, if the current state, s_t, is equal to the goal point, q_f in the end.

V. EXPERIMENTS AND RESULTS

In this section, we describe the experiments, and we present the results of a comparative study of the proposed QAPF learning algorithm versus the CQL algorithm in a set of ten test environments to evaluate the performance of both planning algorithms in terms of path length, path smoothness, learning time, and path planning time in offline mode for known environments and online mode for unknown or partially unknown environments.

A. TEST CONDITIONS

In all the experiments, we considered that the MR configuration is defined as $q(x, y, \phi, \theta)$, where x and y are the coordinate points, $\phi = 0.2$ is the physical radius of the MR in meters, and θ is the angle of orientation that is always oriented to the next coordinate point to visit. Hence, the initial point is centered over the starting point q_0 and oriented to the first coordinate to visit.

Table 1 presents the test environments configuration. The table contains the goal point, q_f, that the MR must attain and the test environment layout. Each test environment is configured by n rectangular obstacles $O_j(x_j, y_j, l_j, w_j)$, for $1 \leq j \leq n$, where the left-bottom vertex of each obstacle is placed at the coordinate points (x_j, y_j) and its length and width are indicated by l_j and w_j, respectively. These test environments were designed to evaluate the performance and accuracy of the QAPF learning algorithm. The benchmark maps presented in [31] inspired the test environments, and it has been labeled as Map01, Map02, ..., and Map10.

The test environments Map01 to Map10 cover well-known difficult path planning problems, e.g., path-following prediction problems, problematic areas to reach because the goal

TABLE 1. Test environments configuration with the goal point and obstacles information.

Test environment	Goal q_f (x_f, y_f)	Obstacles O_j (x_j, y_j, l_j, w_j)
Map01	(5.00, 4.00)	(3.70, 4.70, 2.60, 0.60)
Map02	(5.00, 0.50)	(0.10, 6.10, 1.80, 0.80), (5.10, 6.10, 4.80, 0.80), (5.10, 6.90, 0.80, 1.00), (0.10, 3.10, 1.80, 0.80), (3.10, 3.10, 2.80, 0.80), (3.10, 2.10, 0.80, 1.00), (8.10, 3.10, 1.80, 0.80)
Map03	(5.00, 4.00)	(3.20, 5.30, 0.60, 2.00), (3.20, 4.70, 3.60, 0.60), (6.20, 5.30, 0.60, 2.00)
Map04	(5.00, 1.00)	(3.20, 5.30, 0.60, 2.00), (3.20, 4.70, 3.60, 0.60), (6.20, 5.30, 0.60, 2.00), (1.20, 3.30, 0.60, 5.00), (1.20, 2.70, 7.60, 0.60), (8.20, 3.30, 0.60, 5.00)
Map05	(5.00, 1.00)	(1.80, 1.80, 0.40, 2.20), (1.80, 6.00, 0.40, 2.20), (7.80, 1.80, 0.40, 2.20), (7.80, 6.00, 0.40, 2.20), (3.60, 6.00, 0.40, 2.20), (3.60, 2.20, 0.40, 1.60), (6.00, 6.20, 0.40, 1.60), (6.00, 1.80, 0.40, 2.20), (3.00, 4.80, 4.00, 0.40), (4.00, 7.80, 0.60, 0.40), (4.00, 6.00, 0.60, 0.40), (4.00, 2.80, 0.60, 0.40), (5.40, 6.80, 0.60, 0.40), (5.40, 3.60, 0.60, 0.40), (5.40, 1.80, 0.60, 0.40)
Map06	(5.00, 9.00)	(2.35, 2.85, 0.30, 4.30), (7.35, 2.85, 0.30, 4.30), (2.65, 6.85, 4.70, 0.30), (2.65, 2.85, 2.00, 0.30), (5.35, 2.85, 2.00, 0.30)
Map07	(6.50, 3.50)	(4.85, 0.00, 0.30, 7.50), (1.85, 5.00, 0.30, 2.50), (7.85, 5.00, 0.30, 2.50), (1.85, 7.50, 6.30, 0.30)
Map08	(5.00, 1.00)	(0.00, 1.80, 7.20, 0.40), (0.00, 5.80, 7.20, 0.40), (2.80, 3.80, 7.20, 0.40), (2.80, 7.80, 7.20, 0.40)
Map09	(9.00, 5.00)	(4.85, 3.10, 1.80, 0.30), (6.35, 3.40, 0.30, 2.20), (4.85, 5.60, 1.80, 0.30), (3.35, 4.35, 1.80, 0.30), (3.35, 4.65, 0.30, 2.20), (3.35, 6.85, 1.80, 0.30), (4.85, 7.15, 0.30, 2.85), (4.85, 0.00, 0.30, 3.10)
Map10	(8.50, 5.00)	(2.70, 8.20, 0.60, 1.80), (2.70, 3.70, 0.60, 2.60), (2.70, 0.00, 0.60, 1.80), (5.70, 5.70, 0.60, 2.60), (5.70, 1.70, 0.60, 2.60)

is too close to an obstacle, and trap sites due to local minima, among other problems [32], [33]. The test environments described in Table 1 can be graphically observed in Fig. 3, where each test environment is configured with rectangular obstacles in red and a blue dot indicating the goal point q_f, as well as its corresponding APF surface aside. These environments present challenging problems for testing path-planning algorithms; thereby we used them in this work to evaluate the QAPF learning algorithm. These test environments represent just a sample of the types of environments that the MR can expect to find in typical real-world scenarios. All the test environments have a physical dimension of 10×10 meters and the input coordinates (x, y) are quantized into $161 \times 161 = 25,921$ states. Hence, each state in the grid has a physical separation of 0.0625 meters.

In this work, all the experiments were carried out on an Intel Core i9 CPU (3.60 GHz) with 16 GB of RAM running the Ubuntu Focal Fossa distribution of Linux with Python 3.7, and OpenCV 4.5. The experiments are composed of a learning phase and a testing phase. To make a fair comparison among the algorithms, we set the same parameters.

For the learning phase, we have the following remarks:

Remark 1: The reward function is defined by Eq. (8). Where the reward is $r = 100$ when the agent arrives at goal state q_f. The reward is $r = -1$ when the agent collides with obstacles or when it steps into the limits of the workspace, and then in both cases, the agent will be instantly reset to a valid random state. For other steps, the agent receives a reward $r = 0$.

$$r = \begin{cases} 100, & \text{if } s_t = q_f \text{ and } safe = True \\ -1, & \text{if } s_t \neq q_f \text{ and } safe = False \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

Remark 2: The learning rate, $\alpha \in (0, 1)$, regulates the range of the latest received data that will override the previous data. When $\alpha = 0$, the MR will learn nothing, whereas the MR will only read the latest received information when $\alpha = 1$ [5]. When the value of α is relatively small, Q-value in the function $Q(s, a)$ will eventually converge to the optimal Q-value, Q* after the MR has traveled all the states and considered all the possible actions in the given environment [25]. As such, the learning rate, α, is empirically set to 0.3 in this work.

Remark 3: The discount factor, $\gamma \in [0, 1)$, determines the type of reward that the MR receives. When $\gamma = 0$, the MR will only consider immediate reward, whereas the MR will consider future reward when γ approaches 1. In that sense, the discount factor, γ, is set to 0.8 in this work, following [5], [34].

Remark 4: The decision rate, $\zeta \in (0, 1]$, determines the process involved to choose the action a_t in the state s_t. These processes include the APF weighting procedure, and the explore-exploit approach explained in Section IV. The decision rate, ζ, is empirically set to 0.2 in this work.

Remark 5: The APF weighting employs the total artificial potential field in its process. Therefore, the attractive and repulsive proportional gains, $\{k_a, k_r \mid 0 < k_a, k_r < 10\}$ [29], are defined as follow. The attractive proportional gain, k_a, is empirically set to 0.25 as well the repulsive proportional gain, k_r, is empirically set to 0.60 in this work.

And for the testing phase, we have the following remarks:

Remark 6: The path length *dist* is defined as the sum of distances between the configuration states from the start point q_0 to the goal point q_f [35], and it is calculated by Eq. (9).

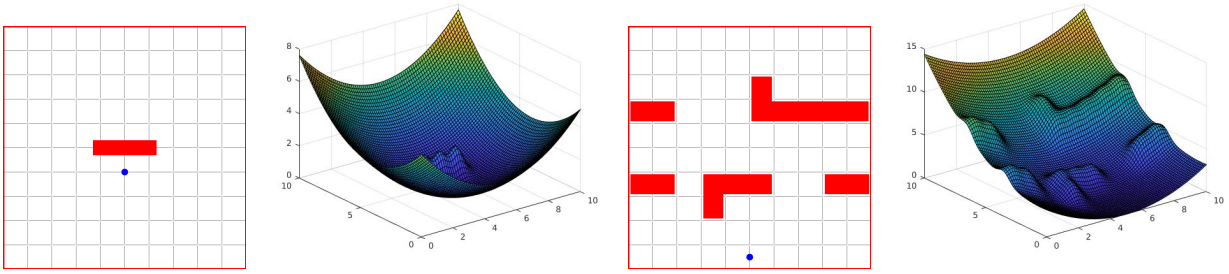
$$dist = \sum_{i=0}^{N_{conf}-1} L(i, i+1) \tag{9}$$

where, $L(i, i+1) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}$ is the distance between configuration states $i = (x_i, y_i)$ and $i+1 = (x_{i+1}, y_{i+1})$, the number of configuration states is N_{conf}.

Remark 7: Path smoothness, *smooth*, has as goal to measure how much snaky is the path, and it is calculated by Eq. (10).

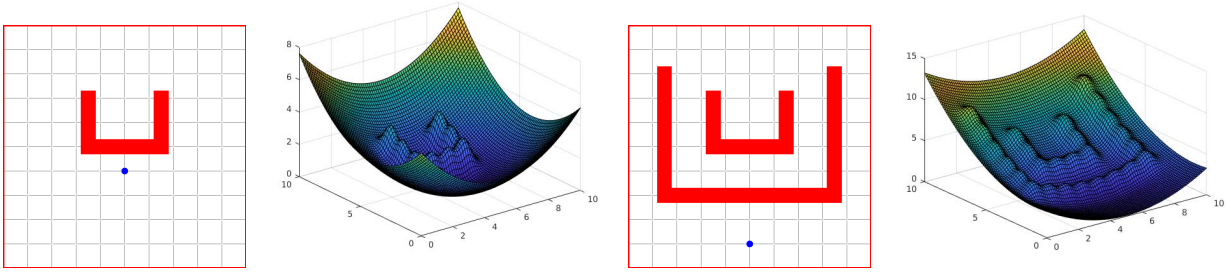
$$smooth = \frac{1}{N_{conf}} \sum_{i=0}^{N_{conf}-1} |\beta(i, i+1)| \tag{10}$$

where, $\beta(i, i+1)$ is the angle in every change of direction θ between configuration states $i = (x_i, y_i)$ and



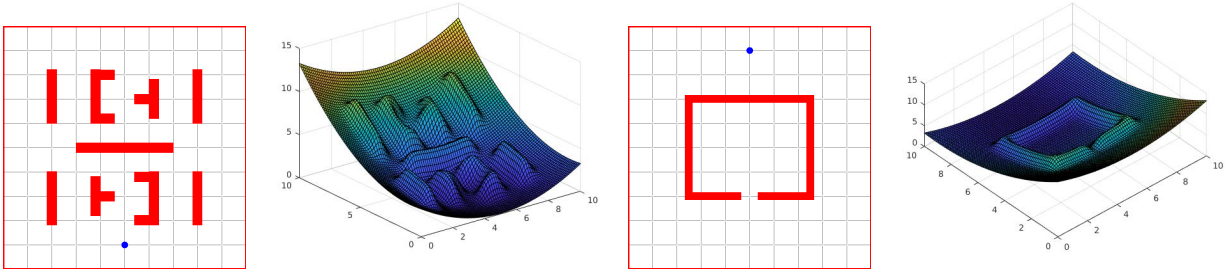
(a) Map01

(b) Map02



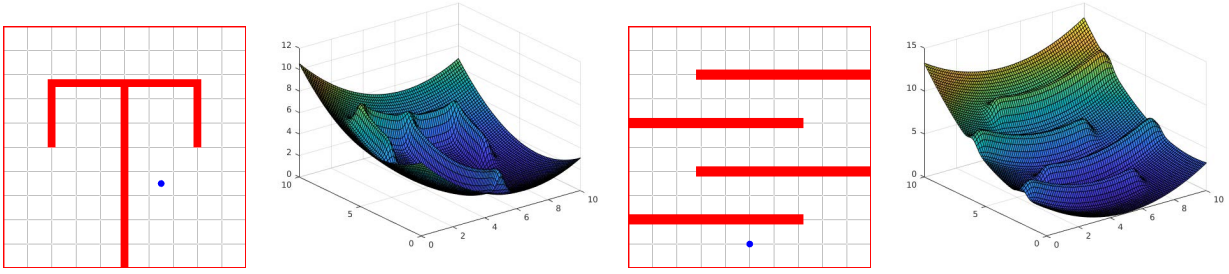
(c) Map03

(d) Map04



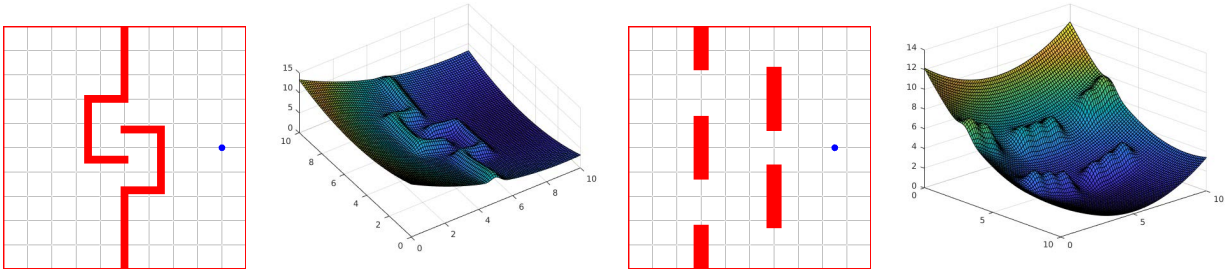
(e) Map05

(f) Map06



(g) Map07

(h) Map08



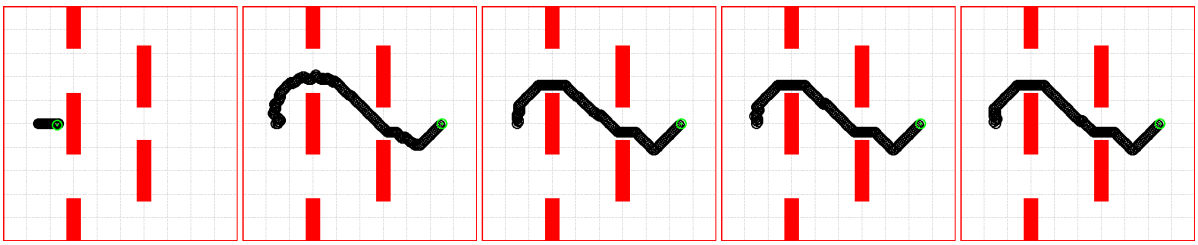
(i) Map09

(j) Map10

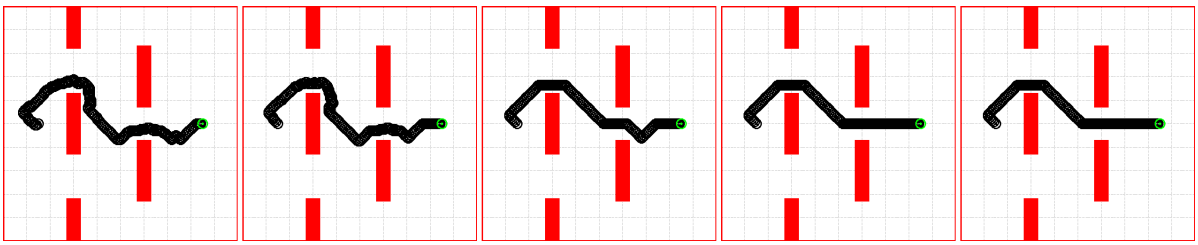
FIGURE 3. Test environments with rectangular obstacles in red and a blue dot indicating the goal point, each one is presented in an area of 10 × 10 and their correspondent APF surface aside.

TABLE 2. Results in terms of path length (distance in meters) for the learning phase considering a different number of episodes, the best result (lower is the best) for each test environment is in bold. For '- - -' the learning algorithm was unable of achieving a feasible result.

Test environment	Learning algorithm	Number of episodes					
		5×10^4	1×10^5	5×10^5	1×10^6	2×10^6	3×10^6
Map01	CQL	9.3817	8.9915	8.8236	8.6986	8.5433	8.0773
	QAPF	9.6013	8.5281	8.0255	**7.0418**	7.0418	7.0418
Map02	CQL	- - -	16.3163	16.2127	15.0234	14.1583	14.0548
	QAPF	- - -	16.7456	13.8074	13.5485	**13.4967**	13.4967
Map03	CQL	12.0762	11.3477	10.3628	10.2227	10.1709	10.1709
	QAPF	14.2770	10.0433	9.1354	**8.1516**	8.1516	8.1516
Map04	CQL	- - -	- - -	18.3921	17.3048	16.8906	16.4246
	QAPF	- - -	- - -	18.2216	17.9261	**15.8550**	15.8550
Map05	CQL	- - -	15.4816	13.1217	11.8425	11.7389	11.6354
	QAPF	- - -	17.6986	13.0333	11.6871	**11.0140**	11.0140
Map06	CQL	- - -	- - -	17.1269	17.0385	16.7948	16.7430
	QAPF	- - -	- - -	16.4020	15.8111	**15.5004**	15.5004
Map07	CQL	- - -	- - -	20.7001	19.0951	18.9397	18.9397
	QAPF	- - -	- - -	20.6520	**18.6809**	18.6809	18.6809
Map08	CQL	- - -	- - -	29.9051	29.3873	27.5903	27.0208
	QAPF	- - -	- - -	31.2649	29.4731	26.3477	**26.0888**
Map09	CQL	- - -	- - -	14.9475	14.5485	14.1494	14.1494
	QAPF	- - -	- - -	14.3477	14.0977	**13.6317**	13.6317
Map10	CQL	- - -	10.4789	9.9915	9.9915	9.9397	9.8880
	QAPF	12.1557	11.2567	9.5255	**9.0078**	9.0078	9.0078



(a) CQL algorithm



(b) QAPF learning algorithm

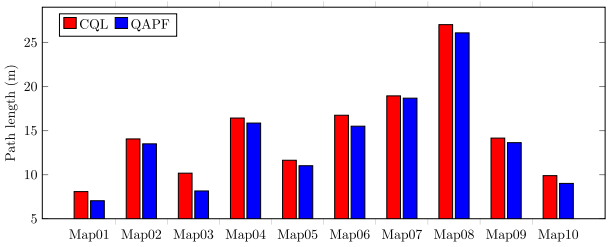
FIGURE 4. Learning results comparative. Each map shows the best path Q_G obtained by the CQL algorithm and the QAPF learning algorithm for 5×10^4, 1×10^5, 5×10^5, 1×10^6, and 2×10^6 episodes respectively from left to right.

$i + 1 = (x_{i+1}, y_{i+1})$. If the direction θ has no change $\beta(i, i+1) = 0$, otherwise, $\beta(i, i+1) = \arctan 2((y_{i+1} - y_i), (x_{i+1} - x_i))$.

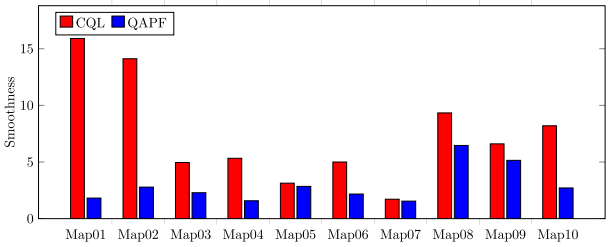
B. LEARNING PHASE RESULTS

To assess the learning performance of the proposed QAPF learning algorithm versus the CQL algorithm, Table 2 presents the results in terms of path length for a different number of episodes employed during the learning phase in each test environment. For the learning phase were employed from 5×10^4 to 3×10^6 episodes to reach the best results, after 3×10^6 episodes no better results or improvement is obtained in any of the test environments employed.

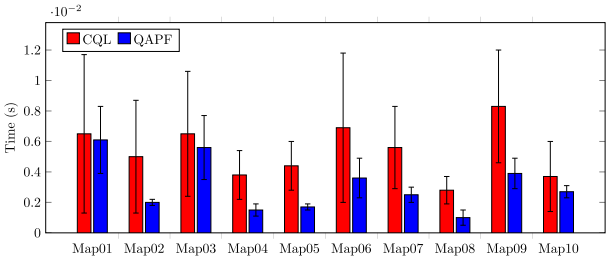
The Eq. (9) was employed to compute all the results in Table 2. The results show that the QAPF learning algorithm presents better performance in terms of path length in the learning phase over the CQL algorithm in all the test environments. For example, for the QAPF learning algorithm in Map10, the shortest path length is 9.0078 meters which is reached with 1×10^6 episodes of training. However, for the CQL algorithm at the same number of episodes the path length reached is 9.9915 meters. The difference between the results is 0.9837 meters, which represents an important advantage of the QAPF learning algorithm. The advantage of the QAPF learning algorithm can be observed in the results presented in Table 2. Therefore, it can be concluded that the QAPF learning algorithm outperforms the CQL algorithm in



(a) Path length in meters



(b) Path smoothness



(c) Training time in seconds (1×10^{-2}). The solid and error bars represent the mean and the standard deviation, respectively for a single episode

FIGURE 5. Best results (lower is the best) for the learning phase in a single mission in each test environment.

most of the stages of learning in terms of path length in the training phase.

Fig. 4 shows a visualization of the learning results presented in Table 2 for the CQL algorithm and the QAPF learning algorithm through the different number of episodes in Map10. It can be observed that for 5×10^4 episodes the CQL algorithm is unable to find a path, instead, the QAPF learning algorithm reaches a feasible solution with the same number of episodes. Also, it can be observed that each algorithm improves its solution through the increase in the number of episodes, finding that the QAPF learning algorithm presents better results for most of the stages of learning.

Fig. 5 shows an overall view of the best results presented in Table 2 for the learning phase in terms of path length, smoothness, and training time. In Fig. 5(a), it can be observed that the QAPF learning algorithm overcomes the CQL algorithm in all the test environments. To obtain the results presented in Fig. 5(b), the Eq. (10) was employed. The path smoothness results show a better performance for the QAPF learning algorithm over the CQL algorithm, whereas for the two first test environments the QAPF learning algorithm shows a wide advantage. Fig. 5(c) presents the training time results for a single episode. The mean and the standard deviation are lower for the QAPF learning algorithm in all the test environments. It can be concluded that the QAPF learning algorithm presents an efficient performance for the learning phase due to better results presented in all the test environments.

In summary, for the learning phase results that are shown in Fig. 5. Regarding path length, the CQL algorithm achieves an average of 14.7104 meters and the QAPF learning algorithm achieves an average of 13.8469 meters. Therefore, the difference is 0.8635 meters, reaching a 6.24% of improvement with the QAPF learning algorithm. Regarding path smoothness, the CQL algorithm achieves an average of 7.4297 units and the QAPF learning algorithm achieves an average of 2.9360 units. Therefore, the difference is 4.4937 units, yielding a 153.05% of improvement with the QAPF learning algorithm. Finally, regarding training time, the CQL algorithm achieves an average of 5.35 ms and the QAPF learning algorithm achieves an average of 3.06 ms for a single episode. Therefore, the difference is 2.29 ms, yielding a 74.84% of improvement with the QAPF learning algorithm.

C. TESTING PHASE: OFFLINE PATH PLANNING RESULTS

In offline path planning, the aim is to find a feasible collision-free path between the start point q_0 and goal point q_f, in a known static environment composed of obstacles. The environments described in Table 1 were used to test the QAPF learning algorithm in offline path planning mode. Fig. 6 shows the best path Q_G obtained in each test environment from Map01 to Map10. These paths have the shortest path length obtained by the QAPF learning algorithm as it is shown in Table 2, where the best results (shortest path) are in bold.

For a quantitative comparison between the proposed QAPF learning algorithm and the CQL algorithm in offline path planning mode, for each test environment, twenty different path planning problems were generated by modifying the start point at random. Fig. 7 shows an example of the offline path planning for twenty different random start points in Map10.

Fig. 8 shows the offline path planning results for twenty different missions (in which each one has a different start point) in terms of path length, path smoothness, and computation time for each test environment. In Fig. 8(a), it can be observed that the QAPF learning algorithm overcomes the CQL algorithm in all the test environments. Also, the path smoothness results (Fig. 8(b)) show a better performance for the QAPF learning algorithm over the CQL algorithm, in this case, the QAPF learning algorithm shows a wide advantage in most of the test environments. Fig. 8(c) presents the path planning computation time results for twenty different missions in each test environment. The mean and the standard deviation are slightly lower for the QAPF learning algorithm in all the test environments. It can be concluded that the QAPF learning algorithm presents an efficient performance for the testing phase in offline path planning due to better results presented in all the test environments.

In summary, for the testing phase in offline mode results, which are shown in Fig. 8. Regarding path length, the

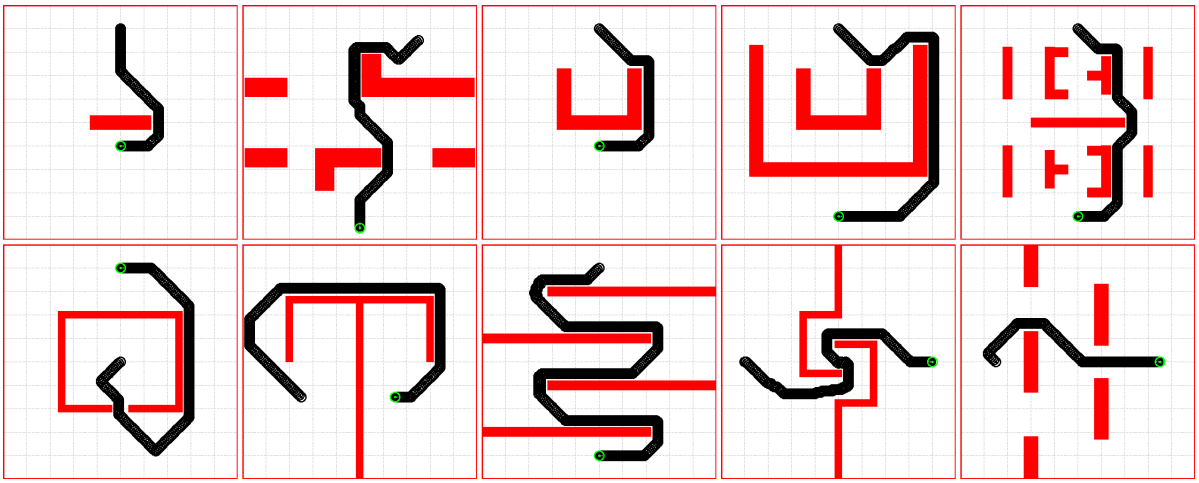


FIGURE 6. Path planning results for the different test environments. Each map shows the best path Q_G obtained by the QAPF learning algorithm in offline path planning mode.

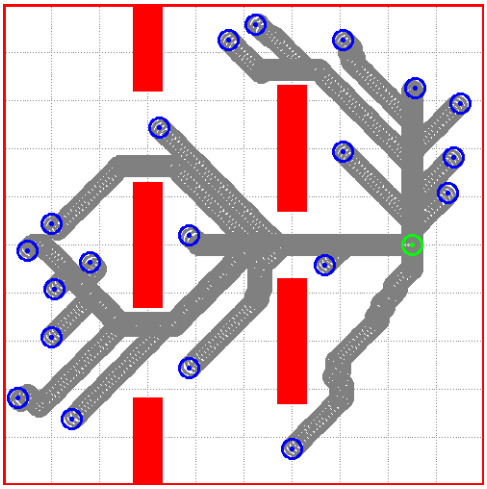
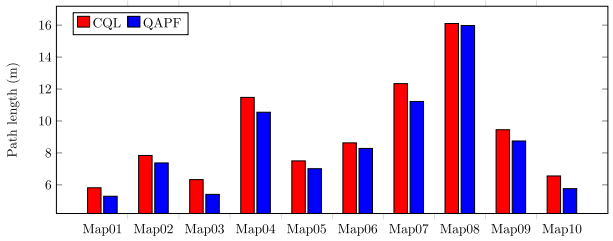
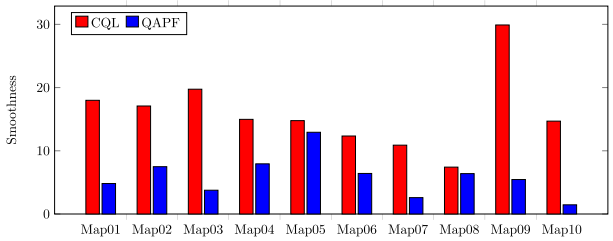


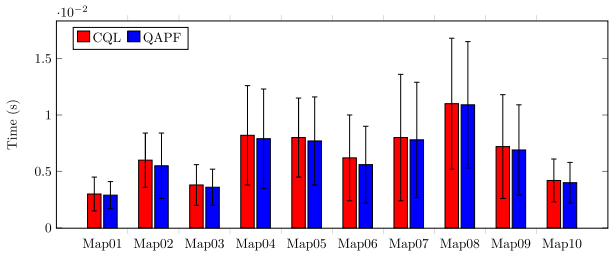
FIGURE 7. Offline path planning for twenty different random start points in Map10. The map shows the paths Q_G obtained by the QAPF learning algorithm, the different start points in blue, and the goal in green.



(a) Average path length in meters



(b) Average path smoothness



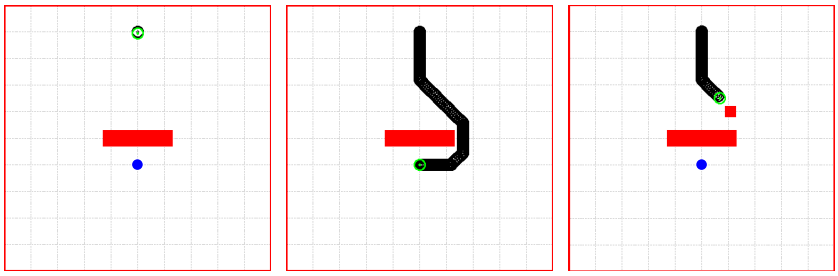
(c) Path planning computation time in seconds (1×10^{-2}). The solid and error bars represent the mean and the standard deviation, respectively in 20 separate sample missions for each test environment

FIGURE 8. Offline path planning results (lower is the best) for twenty different missions in each test environment.

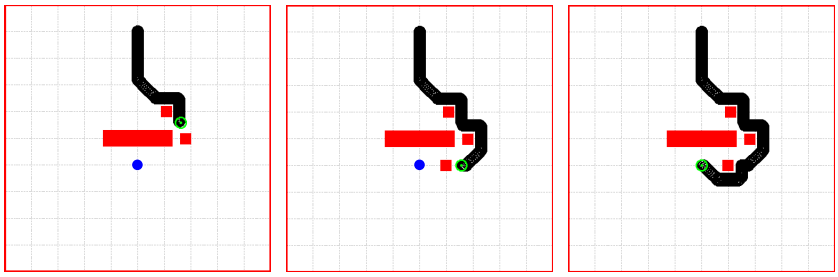
CQL algorithm achieves an average of 9.2027 meters and the QAPF learning algorithm achieves an average of 8.5599 meters. Therefore, the difference is 0.6428 meters, reaching a 7.51% of improvement with the QAPF learning algorithm. Regarding path smoothness, the CQL algorithm achieves an average of 15.9806 units and the QAPF learning algorithm achieves an average of 5.9242 units. Therefore, the difference is 10.0564 units, yielding a 169.75% of improvement with the QAPF learning algorithm. Finally, regarding path planning computation time, the CQL algorithm achieves an average of 6.56 ms and the QAPF learning algorithm achieves an average of 6.28 ms. Therefore, the difference is 0.28 ms, reaching a 4.46% of improvement with the QAPF learning algorithm.

D. TESTING PHASE: ONLINE PATH PLANNING RESULTS

In real-world missions, the MR commonly faces unknown or partially known environments. These environment conditions require that the MR can respond and take decisions, a task that is called online path planning. Fig. 9 exemplifies the online path planning experiment considering three new static obstacles. For this experiment, the test environment Map01 is employed, see Fig. 9(a). First, the path planning is performed

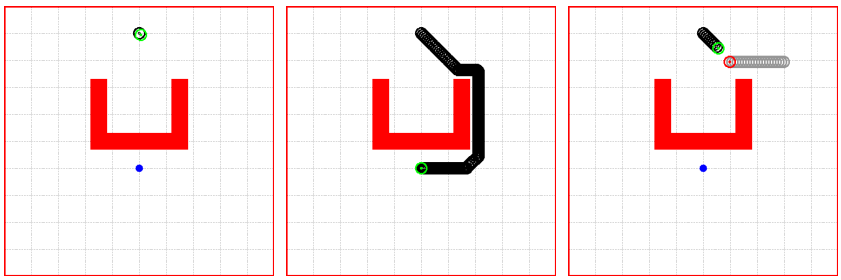


(a) Map01 (b) Offline path planning (c) Online path planning, first unknown obstacle

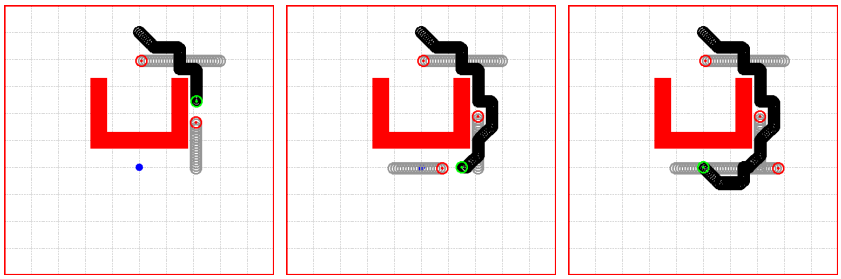


(d) Online path planning, second unknown obstacle (e) Online path planning, third unknown obstacle (f) Online path planning successfully completed

FIGURE 9. Online path planning.



(a) Map03 (b) Offline path planning (c) Online path planning, first dynamic obstacle



(d) Online path planning, second dynamic obstacle (e) Online path planning, third dynamic obstacle (f) Online path planning successfully completed

FIGURE 10. Online path planning with dynamic obstacles.

in offline mode because we know the environment information described by the original test environment Map01, see Fig. 9(b). The minimum path length found to reach the target position is 7.0418 meters, using the best results, as described in Table 2. At position (6.0800, 6.0000), a new static obstacle is added to change the environment configuration. After a while, when the MR has moved 2.8472 meters, it reaches the position (5.6875, 6.5000). The MR senses the new obstacle; it calculates the obstacle position to update the environment layout map, as shown in Fig. 9(c). The MR path planning algorithm based on the QAPF learning algorithm now has a different environment layout. Therefore, it is necessary to update the path and continue with the movement to reach the goal point.

Next, at position (6.8000, 4.9800), a second new static obstacle is added to change the environment configuration.

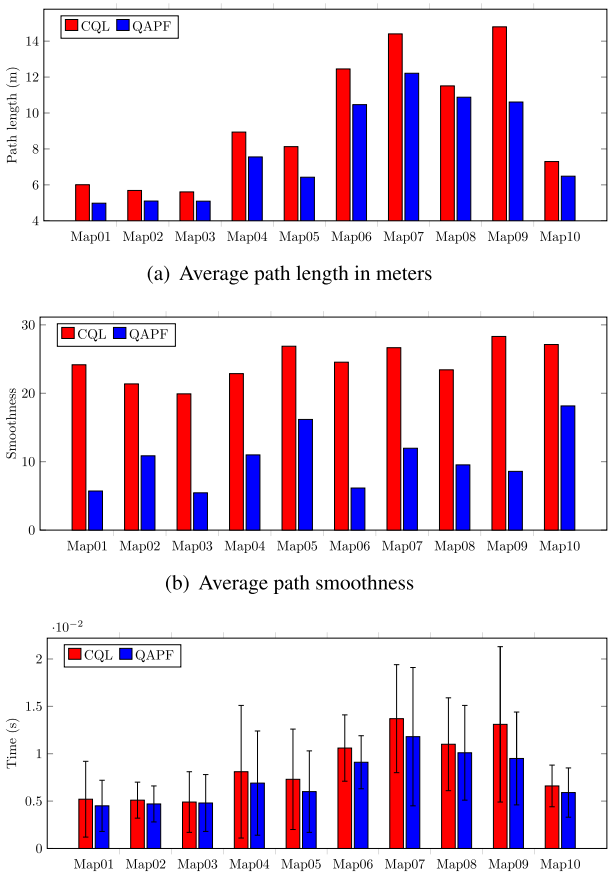
After a while, when the MR has moved an additional distance of 1.7393 meters, it touches the position (6.6250, 5.5625). The MR senses the second new obstacle; it calculates the obstacle position to update the environment layout, as shown in Fig. 9(d). Then, at position (5.9800, 4.0000), a third new static obstacle is added to change the environment configuration. After a while, when the MR has moved an additional distance of 2.6339 meters, it touches the position (6.5625, 4.0000). The MR senses the third new obstacle; it calculates the obstacle position to update the environment layout, as shown in Fig. 9(e). Finally, the MR follows the new path to reach the goal point. The complete path to accomplish the goal point is shown in Fig. 9(f). The total path length from the original start point to the goal point is $2.8472+1.7393+2.6339+2.5955 = 9.8159$ meters.

Now, we are going to present an experiment where the QAPF learning algorithm will be employed for online path planning dealing with new dynamic obstacles. This experiment looks at the case of new dynamic obstacles that can be persons or other mobile robots moving through the environment. Contrarily to the case of the unknown static obstacles, here the new obstacle will not remain static, it will be moving with a defined trajectory that is unknown to the MR.

For this experiment, we are going to use the test environment Map03, see Table 1. Fig. 10 shows the online path planning experiment considering unknown dynamic obstacles. First, the path planning is performed in offline mode because we have the environment information described by the original test environment Map03, see Fig. 10(b). The minimum path length found to achieve the goal point is 8.1516 meters, using the QAPF learning algorithm, see Table 2. Now the MR navigation can start. After a while, when the MR has traveled 0.7955 meters, it reaches the position (5.5625, 8.4375). The MR senses a new obstacle (first mobile robot in red), which is located at (5.9800, 7.9400) at that moment, the MR calculates the obstacle position to update the environment layout map, as shown in Fig. 10(c).

The MR now has a different environment layout. Consequently, it is necessary to update the path to achieve the goal point. After a while, when the MR has traveled 3.4893 meters, it reaches the position (7.1250, 6.4375). The MR senses the second new dynamic obstacle that is located at (7.1000, 5.7600) at that moment. The MR calculates the obstacle position to update the environment layout, as shown in Fig. 10(d). Then, when the MR has traveled 4.2722 meters, it reaches the position (6.5000, 4.0000). The MR senses a third new dynamic obstacle, located at (5.8800, 3.9600) at that moment. The MR calculates the obstacle position to update the environment layout, as shown in Fig. 10(e). Finally, the MR follows the new path to reach the target position; the complete path is shown in Fig. 10(f). The total path length from the original start point to the goal point is $0.7955+3.4893+4.2722+1.8258 = 10.3828$ m.

Fig. 11 shows the online path planning results in terms of path length, path smoothness, and computation time for twenty different missions in each test environment.



(a) Average path length in meters

(b) Average path smoothness

(c) Path planning computation time in seconds (1×10^{-2}). The solid and error bars represent the mean and the standard deviation, respectively in 20 separate sample missions for each test environment

FIGURE 11. Online path planning results (lower is the best) for twenty different missions in each test environment.

In Fig. 11(a), it can be observed that the QAPF learning algorithm overcomes the CQL algorithm in all the test environments. Besides, the path smoothness results (Fig. 11(b)) show a better performance for the QAPF learning algorithm over the CQL algorithm, in this case, the QAPF learning algorithm shows a wide advantage in most of the test environments. Fig. 11(c) presents the path planning computation time results for twenty different missions in each test environment. The mean and the standard deviation are slightly lower for the QAPF learning algorithm in all the test environments. It can be concluded that the QAPF learning algorithm presents an efficient performance for the testing phase in online path planning due to better results presented in all the test environments.

In summary, for the testing phase in online mode results, which are shown in Fig. 11. Regarding path length, the CQL algorithm achieves an average of 9.4827 meters and the QAPF learning algorithm achieves an average of 7.9800 meters. Therefore, the difference is 1.5027 meters, reaching an 18.83% of improvement with the QAPF learning algorithm. Regarding path smoothness, the CQL algorithm achieves an average of 24.5259 units and the QAPF learning algorithm achieves an average of 10.3578 units. Therefore,

the difference is 14.1681 units, reaching 136.79% of improvement with the QAPF learning algorithm. Finally, regarding path planning computation time, the CQL algorithm achieves an average of 8.56 ms and the QAPF learning algorithm achieves an average of 7.33 ms. Therefore, the difference is 1.23 ms, reaching a 16.78% of improvement with the QAPF learning algorithm.

VI. CONCLUSION

In this work, the proposed QAPF learning algorithm successfully solves path planning problems in offline and online modes for known and unknown environments. The combination of the Q-learning approach and the APF method can get over the drawback of the classical Q-learning approach, such as slow learning speed, time consumption, and impossible learning in known and unknown environments. All the simulation results demonstrate that the proposed QAPF learning algorithm can enhance learning speed and improve path planning in terms of path length and path smoothness.

For the training phase, the QAPF learning algorithm reached an improvement of 6.24% in path length, 153.05% in path smoothness, and 74.84% in training time over the classical approach. For the testing phase in offline mode, the QAPF learning algorithm reached an improvement of 7.51% in path length, 169.75% in path smoothness, and 4.46% in path planning computation time over the classical approach. And, for the testing phase in online mode, the QAPF learning algorithm reached an improvement of 18.83% in path length, 136.79% in path smoothness, and 16.78% in path planning computation time over the classical approach. The effects of the APF weighting in the results of the QAPF learning algorithm are beneficial over the classical Q-learning approach due to the advantages presented by the APF method such as the effectiveness in providing smooth and safe planning.

We employed different test scenarios to evaluate the QAPF learning algorithm. The results obtained in the different known and partially unknown environments show that the proposed QAPF learning algorithm achieves the three requirements efficiently to solve the path planning problem: safety, length, and smoothness, which makes the QAPF learning algorithm appropriate to find competitive results for MR navigation in complex and real scenarios.

The path planning results demonstrate that the QAPF learning algorithm produces better solutions in all the test environments. Furthermore, the path planning results are achieved with a lower number of episodes in the learning phase when the QAPF learning algorithm is employed. Another advantage of the proposed QAPF learning algorithm is low variability in training time, making it highly reliable for MR path planning.

In this regard, the QAPF learning algorithm could be useful for many applications in MRs for local and global path planning, including industrial and domestic MRs, self-driving cars, exploration vehicles, unmanned aerial vehicles, and autonomous underwater vehicles.

There are several possible directions to extend this work in the future. Firstly, it could be motivating to focus on multi-agent (multi MR) path planning in complicated dynamic and uncertain situations. Secondly, other types of reinforcement learning can be considered as Deep Q-Networks, through the combination of reinforcement learning and deep neural networks it can be possible to solve a wide range of path planning problems. Lastly, the QAPF learning algorithm can be extended to work in a 3D space that could be useful for many applications for gathering information (i.e., drones), disaster relief, and exploration.

REFERENCES

[1] S. M. La Valle, *Planning Algorithms*. New York, NY, USA: Cambridge Univ. Press, 2006.
[2] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.*, vol. 30, pp. 319–328, May 2015.
[3] A. A. Saadi, A. Soukane, Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "UAV path planning using optimization approaches: A survey," *Arch. Comput. Methods Eng.*, vol. 2022, pp. 1–52, Apr. 2022.
[4] O. Montiel, U. Orozco-Rosas, and R. Sepúlveda, "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Syst. Appl.*, vol. 42, no. 12, pp. 5177–5191, 2015.
[5] E. S. Low, P. Ong, and K. C. Cheah, "Solving the optimal path planning of a mobile robot using improved Q-learning," *Robot. Auto. Syst.*, vol. 115, pp. 143–161, May 2019.
[6] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Auto. Robots*, vol. 45, no. 1, pp. 51–76, Jan. 2021.
[7] Y.-H. Wang, T.-H.-S. Li, and C.-J. Lin, "Backward Q-learning: The combination of Sarsa algorithm and Q-learning," *Eng. Appl. Artif. Intell.*, vol. 26, no. 9, pp. 2184–2193, Oct. 2013.
[8] M. Zolfpour-Arokhlo, A. Selamat, S. Z. M. Hashim, and H. Afkhami, "Modeling of route planning system based on Q value-based dynamic programming with multi-agent reinforcement learning algorithms," *Eng. Appl. Artif. Intell.*, vol. 29, pp. 163–177, Mar. 2014.
[9] M. Zhao, H. Lu, S. Yang, and F. Guo, "The experience-memory Q-learning algorithm for robot path planning in unknown environment," *IEEE Access*, vol. 8, pp. 47824–47844, 2020.
[10] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.
[11] Z. Cui and Y. Wang, "UAV path planning based on multi-layer reinforcement learning technique," *IEEE Access*, vol. 9, pp. 59486–59497, 2021.
[12] D. L. Cruz and W. Yu, "Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning," *Neurocomputing*, vol. 233, pp. 34–42, Apr. 2017.
[13] Q. Yao, Z. Zheng, L. Qi, H. Yuan, X. Guo, M. Zhao, Z. Liu, and T. Yang, "Path planning method with improved artificial potential field—A reinforcement learning perspective," *IEEE Access*, vol. 8, pp. 135513–135523, 2020.
[14] C. Qu, W. Gai, M. Zhong, and J. Zhang, "A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning," *Appl. Soft Comput.*, vol. 89, Apr. 2020, Art. no. 106099.
[15] X. Guo, G. Peng, and Y. Meng, "A modified Q-learning algorithm for robot path planning in a digital twin assembly system," *Int. J. Adv. Manuf. Technol.*, vol. 119, nos. 5–6, pp. 3951–3961, Mar. 2022.
[16] A. Maoudj and A. Hentout, "Optimal path planning approach based on Q-learning algorithm for mobile robots," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106796.
[17] V. B. Ajabshir, M. S. Guzel, and E. Bostanci, "A low-cost Q-learning-based approach to handle continuous space problems for decentralized multi-agent robot navigation in cluttered environments," *IEEE Access*, vol. 10, pp. 35287–35301, 2022.
[18] V. Bulut, "Optimal path planning method based on epsilon-greedy Q-learning algorithm," *J. Brazilian Soc. Mech. Sci. Eng.*, vol. 44, no. 3, p. 106, Mar. 2022.

[19] T. Gao, C. Li, G. Liu, N. Guo, D. Wang, and Y. Li, "Hybrid path planning algorithm of the mobile agent based on Q-learning," *Autom. Control Comput. Sci.*, vol. 56, no. 2, pp. 130–142, Apr. 2022.
[20] A. K. Sadhu, A. Konar, T. Bhattacharjee, and S. Das, "Synergism of firefly algorithm and Q-learning for robot arm path planning," *Swarm Evol. Comput.*, vol. 43, pp. 50–68, Dec. 2018.
[21] X. Wu, H. Chen, C. Chen, M. Zhong, S. Xie, Y. Guo, and H. Fujita, "The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method," *Knowl.-Based Syst.*, vol. 196, May 2020, Art. no. 105201.
[22] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Pseudo-bacterial potential field based path planner for autonomous mobile robot navigation," *Int. J. Adv. Robot. Syst.*, vol. 12, no. 7, p. 81, 2015.
[23] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, Oct. 2021.
[24] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, and X. He, "State representation modeling for deep reinforcement learning based recommendation," *Knowl.-Based Syst.*, vol. 205, Oct. 2020, Art. no. 106170.
[25] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 279–292, 1992.
[26] A. K. Sadhu and A. Konar, "An efficient computing of correlated equilibrium for cooperative *Q*-learning-based multi-robot planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 8, pp. 2779–2794, Aug. 2020.
[27] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Mar. 1985, pp. 500–505.
[28] O. Montiel, R. Sepúlveda, and U. Orozco-Rosas, "Optimal path planning generation for mobile robots using parallel evolutionary artificial potential field," *J. Intell. Robot. Syst.*, vol. 79, no. 2, pp. 237–257, Aug. 2015.
[29] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial potential field," *Appl. Soft Comput. J.*, vol. 77, pp. 236–251, Apr. 2019.
[30] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.
[31] *Motion Planning Maps, Intelligent and Mobile Robotics Group, Department of Cybernetics, Czech Technical University in Prague*. Accessed: Jun. 27, 2022. [Online]. Available: http://imr.ciirc.cvut.cz/planning/maps.xml
[32] F. Bayat, S. Najafinia, and M. Aliyari, "Mobile robots path planning: Electrostatic potential field approach," *Expert Syst. Appl.*, vol. 100, pp. 68–78, Jun. 2018.
[33] B. Wang, S. Li, J. Guo, and Q. Chen, "Car-like mobile robot path planning in rough terrain using multi-objective particle swarm optimization algorithm," *Neurocomputing*, vol. 282, pp. 42–51, Mar. 2018.
[34] L. Khriji, F. Touati, K. Benhmed, and A. Al-Yahmedi, "Mobile robot navigation based on Q-learning technique," *Int. J. Adv. Robot. Syst.*, vol. 8, no. 1, p. 4, Mar. 2011.
[35] X. Wang, G. Zhang, J. Zhao, H. Rong, F. Ipate, and R. Lefticaru, "A modified membrane-inspired algorithm based on particle swarm optimization for mobile robot path planning," *Int. J. Comput. Commun. Control*, vol. 10, no. 5, pp. 732–745, Oct. 2015.

ULISES OROZCO-ROSAS (Member, IEEE) received the M.S. and Ph.D. degrees in digital systems from the Instituto Politécnico Nacional, Mexico, in 2014 and 2017, respectively. He has held a postdoctoral position. From 2017 to 2018, he was appointed at the Department of Computer Science, Universidad Rey Juan Carlos, Spain. He is currently an Associate Professor with the School of Engineering, CETYS Universidad, and a member of the National System of Researchers at CONACYT. His research interests include machine learning, computational intelligence, parallel and heterogeneous computing, autonomous vehicles, and mobile robots.

KENIA PICOS received the Ph.D. degree from the Instituto Politécnico Nacional, Centro de Investigación y Desarrollo de Tecnología Digital, in 2017. She is currently a full-time Professor with the School of Engineering, CETYS Universidad, Tijuana, Mexico. She is a member of the National System of Researchers (Sistema Nacional de Investigadores) at CONACYT. Her current research interests include computer vision, object recognition, three-dimensional object tracking, pose estimation, and parallel computing with graphics processing units.

JUAN J. PANTRIGO received the M.S. degree in fundamental physics from Universidad de Extremadura, in 1998, and the Ph.D. degree from Universidad Rey Juan Carlos, in 2005. He is currently an Associate Professor with Universidad Rey Juan Carlos and a member of the CAPO Research Group, Department of Computer Science. His research interests include high-dimensional space-state tracking problems, computer vision, metaheuristic optimization, machine learning, and hybrid approaches.

ANTONIO S. MONTEMAYOR received the M.S. degree in physics from Universidad Autónoma de Madrid and the Ph.D. degree in computer science from Universidad Rey Juan Carlos, in 2006. He is currently an Associate Professor with Universidad Rey Juan Carlos and a PI of the CAPO Research Group, Department of Computer Science and Statistics, Madrid, Spain. His research interests include computer vision, artificial intelligence, and GPU computing.

ALFREDO CUESTA-INFANTE received the M.S. degree in physics from Universidad Complutense de Madrid and the Ph.D. degree in computer science from Universidad Nacional de Educación a Distancia. He is currently an Associate Professor with Universidad Rey Juan Carlos and a member of the CAPO Research Group, Department of Computer Science. His research interests include reinforcement and imitation learning, computer vision, and synthetic data generation.