Received: March 27, 2024. Revised: May 10, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Path Planning Improvement Using a Modified Q-learning Algorithm Based on Artificial Potential Field

Firdos N. Irzoqe[1]* **Firas A. Raheem[1]** **Ahmed R. Nasser[1]**

[1]*Control and Systems Engineering Department, University of Technology, Iraq*
* Corresponding author's Email: cse.22.23@grad.uotechnology.edu.iq

Abstract: Over recent years, the demand for mobile automated navigation has grown significantly. Path planning has emerged as an important and exciting field of research in many disciplines, intending to create shorter and smoother paths through the use of several types of algorithms. This work presents a comparative analysis between the artificial potential field (APF) method and the hybrid approach that combines Q-learning with APF (QL-APF). Subsequently, these methodologies are juxtaposed with a proposed approach, modified Q-learning with APF (MQL-APF), which introduces modifications to QL-APF by incorporating a dynamic reward function with a static reward function. The proposed approach has been empirically proven effective, and it is capable of generating safe and efficient paths even in complex environments. The MQL-APF approach achieved improvements in terms of path length of approximately 67.25% when compared with the APF method. Furthermore, the average enhancement percentage is approximately 14.68% compared to the QL-APF method. Additionally, the MQL-APF method achieved an improvement of 50.15%, 7.21%, and 24.63% for QL, MQL, and QL-APF, respectively.

Keywords: Path planning, Artificial potential field, Reinforcement learning, Q-learning.

1. Introduction

In the last few years, the need for mobile robot navigation has increasingly emerged [1]. They are now being used in many sectors, such as rescue, medicine, agriculture, space, the military, education, and more.

One of the core issues in robotics is path planning, which can be defined as a suitable path from an initial point to reach a specific target point while avoiding any obstructions present in a given environment. Several factors must be considered, including avoiding obstacles, determining the shortest path, using the least amount of time, less energy, and achieving path smoothness. There are two approaches to planning: offline planning, which assumes static obstacles and completely known surroundings, and online planning, which concentrates on handling dynamic obstacles and partially known environments [2]. A mobile robot's path planning depends on the different environments it encounters, which might include known, partially known, and unknown surroundings. In addition, depending on the kinds of obstacles, path planning may be further classified into two categories: static and dynamic. In static path design obstacles maintain fixed locations and orientations throughout time, whereas obstacles in dynamic path planning are free to move about the environment [3].

The problem of determining optimal path planning has been the focus of much research for a long time. The process of finding the optimal and possible path for the robot or agent with the speed and accuracy possible has become the main concern of many researchers [4]. Several algorithms and methods for path planning have been presented, such as higher-ranking heuristic algorithms like rapidly random-exploring tree (RRT) [5], Dijkstra's algorithm [6], the artificial potential field method [7], group evolutionary algorithms such as genetic evolution [8], and particle swarm algorithms [9]. Fig. 1 provides a summary of popular path-planning techniques.

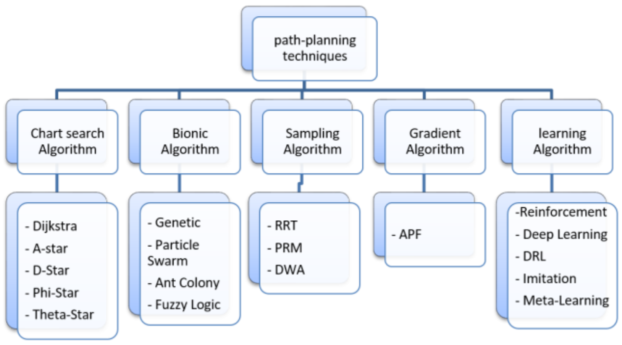


Figure. 1 Summary of popular path planning techniques

One of the common techniques for resolving path planning issues is reinforcement learning (RL). Reinforcement learning is a subfield of machine learning (ML) that deals with the issue of automatically making the best decision over time. The RL approach allows the agent to learn acceptable behaviour from its surroundings. One of the advantages of this technique is that it allows the agent to adjust its policy based on the rewards or penalties it receives. Given their power, RL algorithms have now made significant progress in path planning for mobile robots [10].

The Q-learning algorithm (QL) is one of the most popular techniques for reinforcement learning. Because this algorithm, which was inspired by behaviourism psychology, does not require any prior information, knowledge about the environment is acquired by the agent based on the rewards it receives for completing various tasks. After several rounds, the agent may finally obtain a convergent Q table, which serves as a guide for determining how to maximize the cumulative reward [11].

The APF approach is commonly used by robots for path planning. The APF hypothesis suggests that the robot is influenced by a combined force of repulsive and attractive forces. The strength of the repulsive force is inversely related to the robot's distance from the obstacle. The robot is unaffected if the distance is larger than the influence area. The attractive force draws the robot to the goal, acting on its surroundings simultaneously. The combined force components create a complete field of force, allowing the robot to travel towards the objective while avoiding potential collisions [12].

In this work, a comparative analysis between the APF method and the hybrid approach that combines QL-APF. Subsequently, these methodologies are juxtaposed with a proposed approach, termed MQL-APF, which introduces modifications to QL-APF by incorporating a dynamic reward function with a static reward function. Our objective is to comprehensively examine each method, elucidate their underlying mechanisms, and assess their respective merits and limitations.

The remainder of this essay is structured as follows: Section 2 reviews the related work literature. In Section 3, the theoretical framework is described and the definition of the three approaches is given. Simulation and discuss the results are described in Section 4. Finally, conclusions are given in Section 5.

2. Related work

Numerous approaches to solving path planning issues utilizing various algorithms sometimes alone, sometimes in conjunction with other algorithms. A mixing technique for modified robot path planning is provided in [13], by fusing the techniques of artificial potential field (APF) and probabilistic roadmap (PRM), the attractive potential field is utilized to improve the position of the nodes and improve the roadmap's creation, which can be challenging to implement due to parameters settings difficulty. The author in [14] presented an approach for local and global path planning of unmanned aerial vehicles in dynamic situations called the enhanced ant colony optimization artificial potential field (ACO-APF) algorithm. However, this combination faces challenges in balancing exploration and exploitation, affecting convergence speed and final solution quality. Ahmed S. et al. [15] offer a hybrid strategy to address conventional APF local minimum problems that combines the global optimization powers of the modified APF algorithm with the real-time flexibility of the A-Star path-planning technique. Balancing local optimization capabilities with global optimization strengths can be challenging, and the hybrid strategy may not always achieve the desired balance.

Several ideas based on reinforcement learning have been mixed with other methods to enhance performance. In [16] proposes a collision-free navigation solution by combining reactive navigation and Q-learning, despite potential computational overhead due to real-time decision-making and updating Q-values. Moreover, this combination of real-time decision-making, learning, and updating Q-values may increase computational overhead. The algorithm proposed in [17] introduces an improved Q-learning method that combines Dyna Q-learning with a feature matrix to accelerate learning in unfamiliar environments. It also introduces an adaptive artificial potential field method to navigate the mobile robot, but this may introduce complexity in implementation and parameter tuning. Another example is presented by Low, E. S. et al. [18] who provide an enhanced Q-learning method for mobile robot path planning, incorporating the flower pollination algorithm (FPA) to initialize the Q-table, thereby addressing the delayed convergence issue of traditional Q-learning and potentially enabling alternative path exploration. Initializing the Q-table using the (FPA) might introduce a bias towards certain regions of the state space, potentially limiting the exploration of alternative paths or solutions. Combines the APF approach with Q-learning in [19], employing the APF weighting function for explore-exploit tactics. It optimizes total rewards for path planning by updating learning values based on rewards from action execution. However, this could add complexity and call for careful design and tuning for effective optimization.

The APF approach is generally used in lots of studies for robotic path planning, and it's far successful in easy environments and gives excellent consequences. However, it has weaknesses together with local minima, sharp edges, carefully spaced impediments, and dead ends. [20]. Additionally, when using the classical Q-learning (QL) method within the path planning set of rules, there are now and again troubles with a prolonged studying period, bad exploration performance, and slow convergence velocity [21].

Overall, the study intends to overcome the obstacles to autonomous navigation by enhancing the paths of mobile robots. The work objectives are to improve path planning and pave the way for extra-superior and powerful mobile robotic structures by modifying the QL-APF method.

3. Theoretical framework

This section describes the three types of algorithms: firstly, the APF algorithm, then combining the QL with the APF, and finally, presenting a modified QL-APF method by integrating the dynamic reward function with a static reward function. The notations used in this work are listed in Table 1.

Table 1. List of notations

Notations	Description
u_{total}	Total potential field
u_{att}	Attractive potential field
u_{rep}	Repulsive potential field
f_{total}	Total force
k_a	Attraction coefficient
k_r	Repulsive coefficient
p_0	Repulsion range
q_x, q_y	Coordinates of the position q
q_{xg}, q_{yg}	Coordinates of the target q_g
q_{ox}, q_{oy}	Coordinates of the obstacle position
s_t	State
s_{t+1}	Next state
a_t	Action
a_{t+1}	Next action
Α	learning rate
R	Reward
Γ	Discount factor
DR	Dynamic reward
TR	Total reward
D_i	Distance between any point and target
D_j	Distance between any point and obstacles
N_{state}	Number of states
Θ	Angle in each direction change

3.1 Artificial potential field path planning

One popular method for path planning and obstacle avoidance in autonomous mobile robotics is the artificial potential field. *Khatib* originally proposed the concept of an artificial potential field [22], which is a local collision avoidance strategy that may be used when a robot can perceive its surroundings while it is performing a motion. In this scenario, the robot knows nothing about the surroundings beforehand. Robots are thought of as particles that go from a high-potential point to a destination via low-potentials in the field of robot path planning. The two fields that make up this are the repulsive potential field and the attractive potential field. The robot is drawn towards the goal location by the attractive potential generated by the goal. The robot moves away from obstacles to prevent collisions due to the repulsive potential generated by identified impediments inside the effective zone.

The robot is guided towards the objective position by the combination of both potential fields when it is submerged in the potential field, as provided by Eq. (1). The purpose of combining these two forces is to steer the robot along a safer, collision-free course [23].

$$u_{total}(q) = u_{att}(q) + u_{rep}(q) \tag{1}$$

The two potential field functions' negative gradients are used to calculate the total force $f_{otal}\,(q)$ for each point in the environment, as provided by Eq. (2).

$$f_{total}(q) = \; -\nabla u_{total}(q) \tag{2}$$

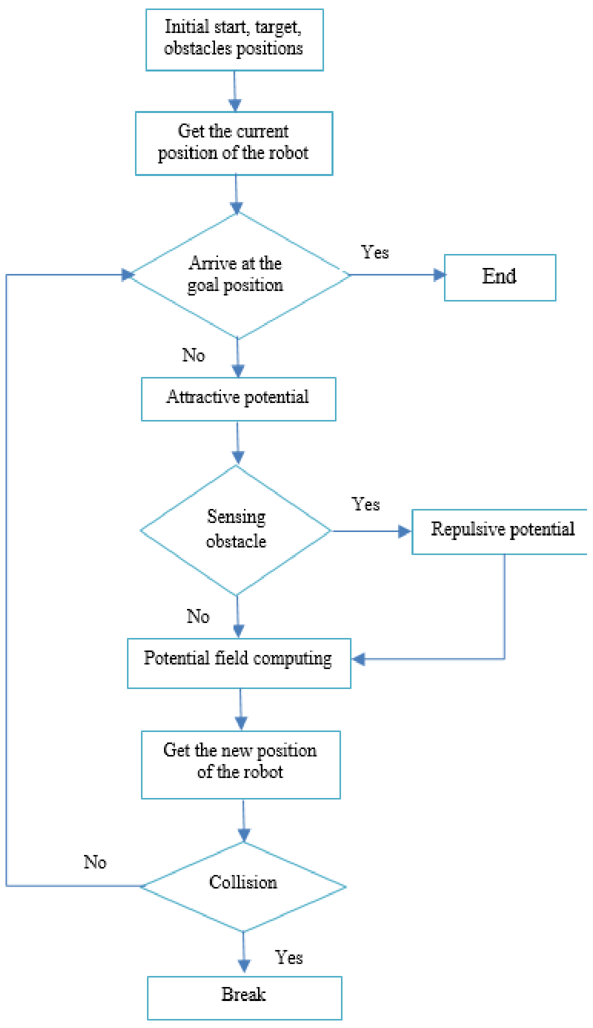

Initial start, target, obstacles positions
Get the current position of the robot
Arrive at the goal position
Yes
End
No
Attractive potential
Sensing obstacle
Yes
Repulsive potential
No
Potential field computing
Get the new position of the robot
No
Collision
Yes
Break

Figure. 2 Flowchart of the APF method

Fig. 2 presents the flowchart of the artificial potential field for robot path planning. The APF method employs the next input parameters: the start point q, the goal point q_g, and some of the obstacles Oj.

In robot path planning, the target creates an attractive potential field that pulls the robot towards it, with the strength of the attraction increasing as the robot gets farther from the target. The negative gradient of the target's potential field determines this attraction. A quadratic equation characterizing its intensity at a generic place is the definition of the attraction force formula and the attractive potential field formula [24]. They are provided by Eq. (3) and Eq. (4) sequentially:

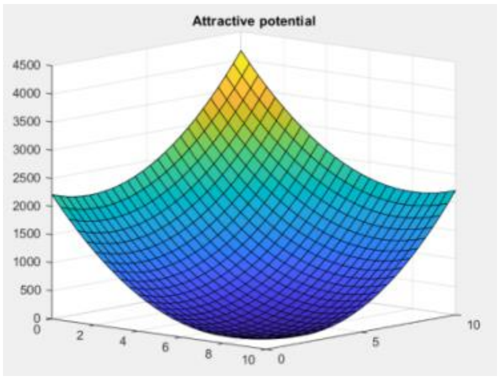
$$f_{att} = -\nabla u_{att} \tag{3}$$

$$u_{att}(q) = \frac{1}{2} k_a \, d^2(q, qg) \tag{4}$$

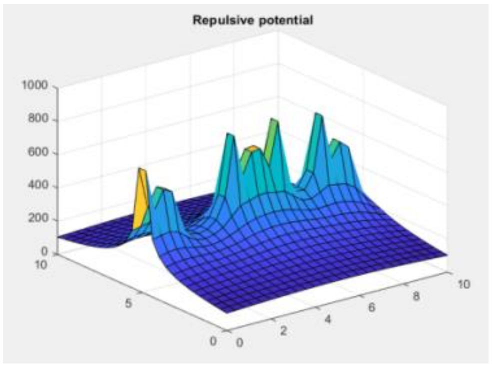
Where the $d^2(q, q_g)$ is the Euclidian distance between the current point of the robot and the target point and is given by Eq. (5).

$$d(q, qg) = \sqrt{(q_x - q_{gx})^2 + (q_y - q_{gy})^2} \tag{5}$$

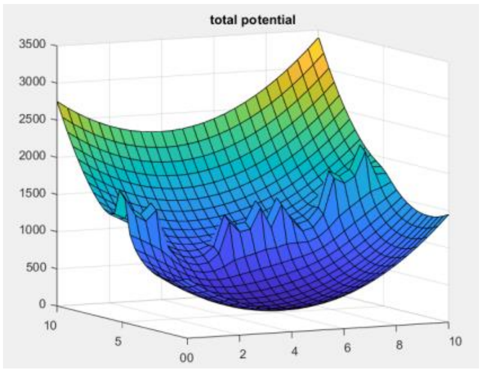
The repulsive potential field created by obstacles has a repulsion range. As the distance between the robot and the obstacle decreases, the repulsive potential energy increases, causing the robot to be pushed back.



(a)



(b)



(c)

Figure. 3 An example of an attractive, repulsive, and total potential field: (a)Attractive potential, (b)Repulsive potential, and (c)Total potential

When the distance exceeds the repulsion range, the repulsive field becomes zero, and the robot is not pushed back. The repulsive force is the negative slope of the repulsive force field [24]. The repulsive potential field and repulsive force are given by Eq. (6) and Eq. (7) sequentially:

$$f_{rep} = -\nabla u_{rep} \tag{6}$$

$$u_{rep}(q) = \begin{cases} \frac{1}{2}\, k_r \left(\frac{1}{p(q)} - \frac{1}{p_0}\right) & if\ p(q) \leq p_0 \\ 0 & if\ p(q) > p_0 \end{cases} \tag{7}$$

Where the *p(q)* is the Euclidian distance between the current position of the robot and the obstacle position and is given by Eq. (8).

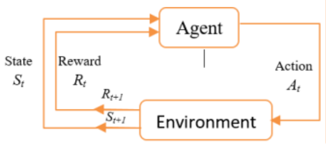
$$p(q) = \sqrt{(q_x - q_{ox})^2 + (q_y - q_{oy})^2} \tag{8}$$

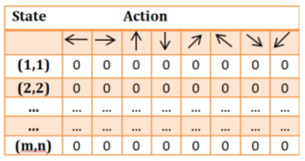
Fig. 3 shows an example of the classical artificial potential field in a 10*10 environment that contains some obstacles, a start point, and a goal point.

Where Fig. 3.a represents the attractive potential field generated by the goal point. Fig. 3.b shows the repulsive potential field generated by obstacles, while Fig. 3.c represents the combination of two potential fields to generate the total potential field.

3.2 Reinforcement learning path planning

Is a psychological technique that learns the best ways to make decisions through experimentation and exploration to gain experience. According to RL, the environment is everything outside of the agent, and any decision-maker is an agent. By interacting with them, the agent learns optimal actions based on rewards and punishments as a feedback signal for the update table. The Markov Decision Process (MDP) is the traditional mathematical framework that allows reinforcement learning to take place. A tuple *(S, A, R, P)* is the mathematical definition of a discrete-time stochastic process, often known as a Markov decision process. RL involves agents, observation, reward, action, and the environment [25]. When it comes to robot path planning, the robot acts as an agent, while its surroundings serve as an observation. The state *S* represents the coordinates of the environment. Action *A* is selected by the agent based on the table in Fig. 2, where each side represents one of the eight directions surrounding the agent. The step reward *R* is the reward the agent receives for progressing from one state to the next. *p* is a probability transition function that determines the probability of reaching a new state from the current state when action is taken [26], clarified by Eq. (9):





State	Action							
	←	→	↑	↓	↗	↖	↘	↙
(1,1)	0	0	0	0	0	0	0	0
(2,2)	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
(m,n)	0	0	0	0	0	0	0	0

Figure. 4 Q-learning model and the Q table's structure [11]

$$p(s, \acute{s}) = \mathrm{p}(s_{t+1} = \acute{s} | s_t = s,\ a_t = a) \tag{9}$$

Fig. 4 depicts the Q-learning model and the Q-table's structure. Each state has states and actions, with being the starting value.

Q-learning is model-free reinforcement learning that integrates theories of the Markov decision process (MDP) and the Bellman equations, to teach agents how to behave optimally in controlled Markovian environments, which is defined by Eq. (10).

$$Q(s_{t+1}, a_{t+1}) = Q(s,t) + \alpha[(R + \gamma \max Q(s_{t+1}, a_{t+1}) - Q(s,a))] \tag{10}$$

Where *Q(s$_{t+1}$,a$_{t+1}$)* estimates the action value after applying an action a in state s, max *Q(s$_{t+1}$,a$_{t+1}$)* is the maximum expected reward [27].

Selecting the learning rate parameter α carefully is crucial to ensure optimal learning rates in the environment. Close to zero results in slow learning and inefficiency in navigation tasks, while close to 1 leads to quick learning and nonoptimal paths. The discount factor γ influences future actions and rewards. Close to zero or 1 results in short-sighted planning, while values above 1 exclude optimal paths due to infinite future rewards, ensuring optimal local paths [28]. The choice of action is made using the ε −greedy method. The random action is chosen with a constant probability of $0 \leq \varepsilon \leq 1$. The agent chooses a random action if the random number is smaller than $\varepsilon -$ greedy. If not, the agent chooses the action with the highest estimated reward [29]. The reward R is the constant value given to the agent during moving through the environment; it is a large constant value when the agent reaches the goal; a negative constant penalty when the agent collides with the obstacles, and a 0 reward in any other place.

3.2.1. The QL-APF method

The QL-APF for robot path planning is shown in Algorithm 1. The following input parameters are used by this method: the environment information, which is made up of n circular obstacles in OJ, the target point (qg), and the start point (q); Getting the learning values Qm*n to construct the path that is QG=[q0, q1, …, qg], a collision-free path that will drive to the goal point without halting in local minima—is the primary purpose of the QL-APF method. The learning values Qm*n are initialized to zero in the first step. The QL-APF algorithm's iterative learning process, which finds the learning values Qm*n to construct the path, is shown in the rest of the steps from the algorithm. As soon as the maximum number of episodes is reached, the learning iterative process terminates [19].

Algorithm I: QL-APF

```
Input: a great environment that contains a start position, target position, the positions of the obstacles, learning rate, discount factor, the attractive gain coefficient, and the repulsive gain coefficient,
Output: learning value $Q_{m*n}$
initialize $Q_{m*n}(s,a) \leftarrow (all\ state\ is\ zero)$
For each episode do
set $st \rightarrow at$ random state from the states
    while $st \neq qg$
  For (i) in nearest neighbour
      $compute\ total\ force\ (F_{total})$
    descending sorting value
     $\sigma =$ max $(sorting\ value)$ by eq. 11
     $if\ \sigma > \ \rho\ (random\ unmber)\ then$
      choose $a_t$ by using $total\ force$
     $else$
        choose the best $a_t\ from\ \ Q_{m*n}$
 End
End
compute reward by eq. 12
update $Q_{m*n}(s,a)$ by eq. 10
$s_t = \ s_{t+1}$
 end
end
return $Q_{m*n}$
```

For each point, the agent observes eight connected neighbouring cells and then computes the force for each cell. The likelihood of a neighbour cell being allocated to a new state is highest for the cell with the lowest ftotal, while the neighbouring cell with the highest ftotal has the lowest likelihood of being assigned to a new state.

The action and state are selected using the cumulative probability produced by the total force function. Eq. (11) defines the cumulative probability (σ): for i=1, 2, …., n and f =1, 2, …, fn.

$$\sigma(i) = \frac{f(i)}{\sum_{i=1}^{n} f(i)} \tag{11}$$

The probabilities that are included in σ are first descending sorted. Next, a random number ρ between 0 and 1 is produced. When choosing a new state, the first neighbour cell at the top of the list whose cumulative probability is greater than the random number is selected. If the random number exceeds the decision rate, the ftotal process will be conducted, and the decision will be made with low force. If not, the best action will be chosen using the explore-exploit method by the learning value Qm*n.

The agent's sole objective is to increase the total rewards it’s long-term. The static reward signal identifies the highest fixed positive value given to the agent when the next state of the agent reaches the target point and a negative value (penalty) when the agent collides with obstacles; otherwise, the zero-reward value is in any other state. These are calculated by Eq. (12).

$$Reward = \begin{cases} 100 & s_t = qg \\ -1 & collides\ with\ the\ obstacle \\ 0 & otherwise \end{cases} \tag{12}$$

After that, the action is completed, a reward is received, and the Q(st+1, at+1) table is updated by Eq. (10). Lastly, the current state st is given the new state st+1 and so on until the target point qg is reached or an insecure circumstance occurs. Finally, it returns the learning values that are obtained, Qm*n to construct the path QG. The decision process's stop condition is indicated when the target point and the current state are equal [19].

3.2.2. Proposed method (MQL-APF)

In the proposed MQL-APF method, algorithm 1 is applied, but with a change in the calculation of the reward function. It was proposed to modify the QL-APF method (MQL-APF) by integrating a dynamic reward function with the static reward function to improve the path and get the smallest path in the same environment with the same number of episodes. This combination allows the system to learn from experience and make decisions based on a dynamic reward function.

Following an action by the agent, the environment generates feedback information relevant to the action that is utilized to assess the action's effectiveness. The reward function's design has a significant impact on the agent path planning model's training and learning process, as well as how effective and efficient learning. The efficacy and safety of agent behavior decision-making, which plays an important role in results, may be evaluated using the reward function.[30].

Received: March 27, 2024. Revised: May 10, 2024.

This work changed the reward feature and advised a dynamic reward function design method to address the issues. The primary hints made in this work are four reward functions: (1) positive value while the agent is close to the goal; (2) less positive value when the agent is moderately close to the goal; and (3) negative value while the agent is near the obstacles; (4) less negatives have less fee when the agent is moderately near the obstacles. Eq. (13) and Eq. (14) are employed to sequentially compute the dynamic reward (DR) and the cumulative overall reward (TR).

$$DR = \begin{cases} 10 & Di < 2 \quad\quad close\ to\ q_g \\ 5 & Di < 5 \quad\quad moderately\ close\ to\ q_g \\ -0.8 & Dj < 1 \quad\quad close\ to\ O_j \\ -0.05 & Dj < 3 \quad\quad moderately\ close\ to\ O_j \\ 0 & \quad\quad otherwise \end{cases} \tag{13}$$

$$TR = \begin{cases} 100 & if\ \ Di = 0 \quad\quad s_t = q_g \\ -2 & if\ \ Dj = 0 \quad\quad s_t = O_j \\ DR & \quad\quad otherwise \end{cases} \tag{14}$$

At every point where the agent is located, the distance *Di*(q_i,q_g) between the agent's location and the target location is calculated. If the distance *Di*(q_i,q_g) is less than 2 meters, the reward is 10, and if the distance *Di*(q_i,q_g) is less than 5 meters, the reward is 5. Additionally, the distance *Dj*(q_i,o_j) between the agent's location and the locations of the obstacles is calculated. If the *Dj*(q_i,o_j) is less than 1 meter, the reward is -0.8, and if the *Dj*(q_i,o_j) is less than 3 meters, the reward is -0.05. Otherwise, the reward is zero. Eq. (15) and Eq. (16) are employed to sequentially compute *Di* and *Dj*, and these equations are utilized to generate the dynamic reward function.

For $i = 1,2,\ \dots,\ n$ and for $j = 1,2,\ \dots,\ Oj$ then

$$Di(q_i, q_g) = \sqrt{(q_{ix} - q_{gx})^2 + (q_{iy} - q_{gy})^2} \tag{15}$$

$$Dj(q_i, o_j) = \sqrt{(q_{ix} - q_{ox})^2 + (q_{iy} - q_{oy})^2} \tag{16}$$

The process is summed up in Fig. 5, where the agent makes decisions.

The agent interacts with the environment to choose action via the operations of exploration or APF force. The environment reacts to the agent's activities by presenting new states and producing rewards. Additionally, the environment produces dynamic rewards, which the agent attempts to optimize over time by selecting the best actions.

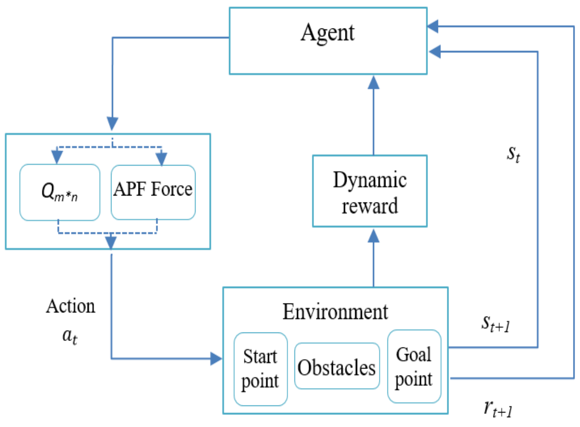


Figure. 5 Structure of the proposed MQL-APF method

Moreover, Eq. (17) is applied to provide a smooth path.

$$smooth\ path = \frac{1}{N_{state}} \sum_{i=0}^{N_{state}-1} \left| \beta(i, i+1) \right| \tag{17}$$

Where the angle in each direction change (*θ*) between states is denoted by *β(i,i+1)*. The angle *β* is 0 when the direction *θ* is unchanged. Otherwise, can be calculated by Eq. (18).

$$\beta(i, i+1) = \arctan 2\left((y_{i+1} - y_i), (x_{i+1} - x_i)\right) \tag{18}$$

4. Simulation results

The simulation results are explained in this part, along with the findings of a comparison study conducted of the APF versus QL-APF and then with the proposed approach, the MQL-APF, by combining static and dynamic reward functions in six test situations to assess how well the three approaches performed in terms of learning time, path smoothness, and length. These approaches are designed on an Intel Core ™ i7-8650 CPU with 16 GB of RAM in Python 3.11.5.

The agent makes use of discretized maps of the surroundings, measuring 10 by 10 and 15 by 15. The parameters attractive coefficient, repulsive coefficient, repulsion range, learning rate, discount factor, and number of episodes are the constants used in all types of test environments and are shown in Table 2. These values were chosen by trial and error. In all test environments, the target point is shown by a green star, with a pink x representing the start point, while the red circles represent obstacles, and the path is shown in blue.

Table 2. Constant parameters are used in all test environments

Parameter	Value
K_a	2
K_r	0.6
P_0	4
α	0.3
γ	0.8
No. of episodes 10*10	$1*10^3$
No. of episodes 15*15	$3*10^3$

To assess the capability of the proposed MQL-APF method versus the APF algorithm and the QL-APF method. The results pertaining to the APF, the QL-APF method, and the MQL-APF method with dynamic reward function are presented in Parts 1, 2, and 3 subsequently of this section. The fourth part compares the proposed MQL-APF method and similar techniques to explore the impact of adding a dynamic reward function on path improvement.

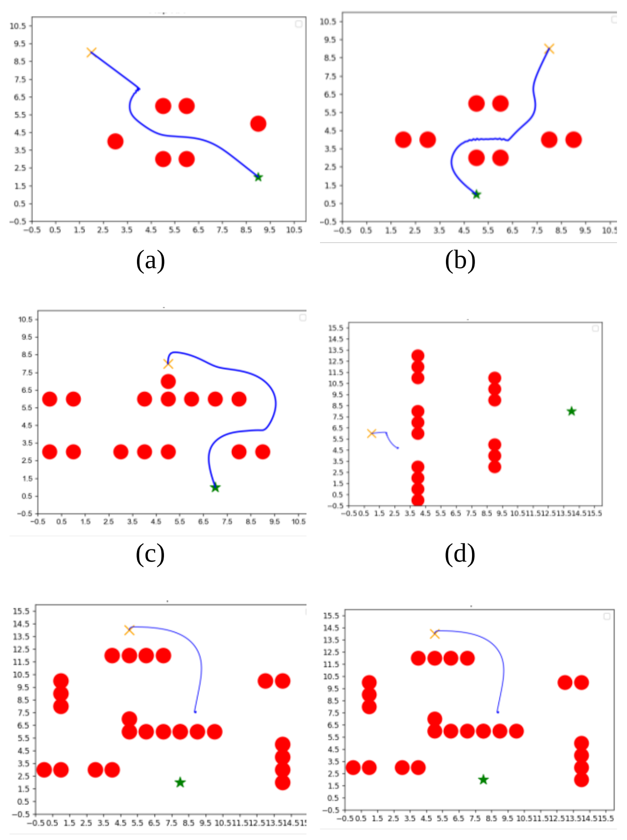


Figure. 6 Simulation results of the APF method: (a) the first environment, (b) The second environment, (c) The third environment, (d) The fourth environment, (e) The fifth environment, and (f) the sixth environment

4.1 Simulation results of the APF method

In this part, the APF method is implemented across various test environments, with Fig. 6 depicting the simulation outcomes. Where Fig. 6.A shows the path planning results within the first test environment, wherein the robot effectively navigates around obstacles, achieving a path length of 11.2384 meters. Subsequently, in the second environment (Fig. 6. B), the path length distance is 10.8959 m. While in the third test environment, the robot takes a long path to reach the target, 12.8699 meters long, as shown in Fig. 6. C. In larger and more complex environments, when using the APF algorithm, the robot fails to reach the target point, becoming at a local minimum, as can be observed in Fig.6.D, E, and F sequentially.

4.2 Simulation results of the QL-APF method

The algorithm of the QL-APF is applied in the six same test environments. The results of the analysis are presented in this section. Fig. 7 shows all types of environments and how this algorithm was able to

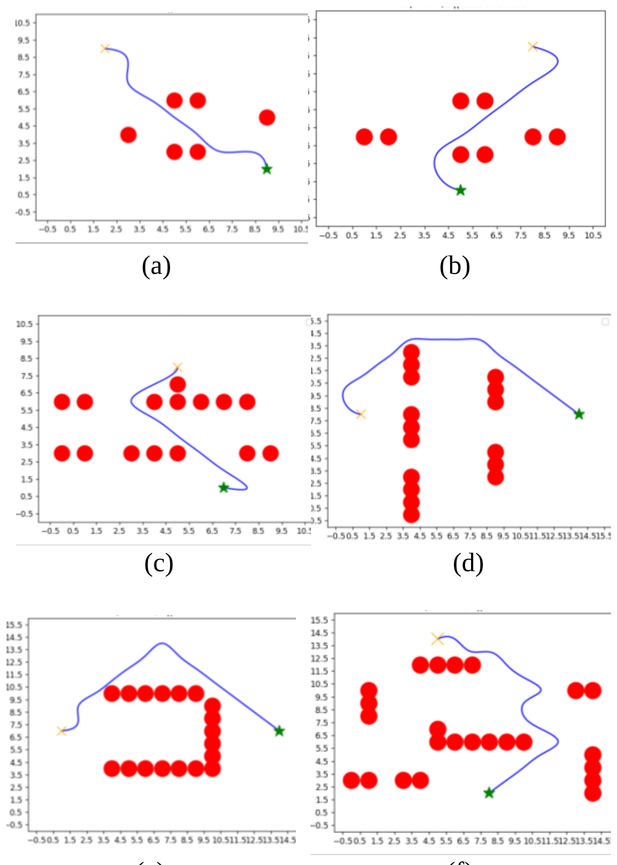


Figure. 7 Simulation results of the QL-APF method: (a) the first environment, (b) The second environment, (c) The third environment, (d) The fourth environment, (e) The fifth environment, and (f) the sixth environment

reach the goal in all testing environments. For example, in the first environment (Fig. 7. A), the agent reaches the target point with a path length of 6.2831 m, which is different by 4.9553m from the path drawn by the APF method in the same test environment. The path length in the second environment is 10.8594 m, which is equal to the path drawn by the APF method in the same test environment, it can be seen in Fig. 7.B. Fig. 7. C shows the third environment, where the path length is 9.4247 m and the path is enhanced by 26.8% from the path described by the APF method. In large and more complex environments, the agent reaches the goal by a path length of 9.6235 m, 10.9955 m, and 18.0641 m in environments 4, 5 and 6, respectively. The paths are presented in Figs. 7.D, E, and F.

4.3 Simulation results of the proposed MQL-APF method.

This part showcases the simulation results for the proposed method, namely MQL-APF, which integrates the dynamic reward function with the static reward function. The capability of the modified method demonstrated across identical in the same test environments, as illustrated in Fig. 8. Comparative analysis reveals that the agent achieves the target point via shorter paths compared to alternative methods. For example, in Fig. 8. A, the path length differs by 5.7407 m from that generated by the APF method and by 0.7854 m from the QL-APF within the same test environment. Notably, Fig. 8. B depicts a 14.91% enhancement in path length relative to other methods. In the third environment, the agent reaches the target point with a path length of 8.0685 m, as depicted in Fig. 8.C., another example is In Fig. 8.D, where the path length differs by 1.7696 from the path drawn by the QL-APF method, whereas the APF method fails to reach the goal within the same test environment. Fig. 8. E presents the fifth test environment with a path length of 10.2101 m. Finally, Fig. 8. F demonstrates a 21.74% improvement in path length compared to the QL-APF method within the sixth test environment.

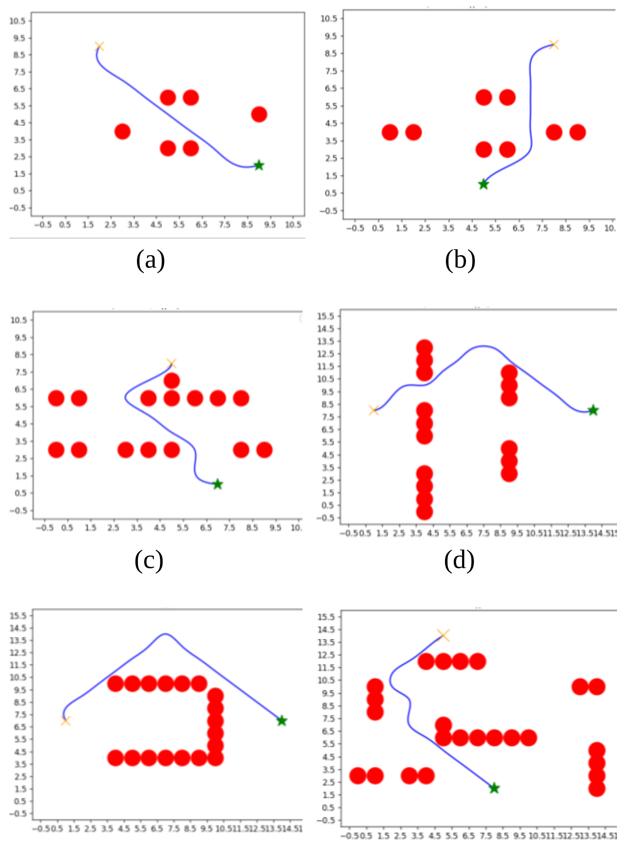

(a) (b) (c) (d) (e) (f)

Figure. 8 Simulation results of the MQL-APF method: (a) the first environment, (b) The second environment, (c) The third environment, (d) The fourth environment, (e) The fifth environment, and (f) the sixth environment

Table 3. Path length in meters in all test environments

	Path length of APF	Path length of QL-APF	Path length of MQL-APF
Env1	11.2384	6.2831	5.4977
Env2	10.8959	10.8594	9.2426
Env3	12.8699	9.4247	8.0685
Env4	No path	9.6235	7.8539
Env5	No path	10.9955	10.2101
Env6	No path	18.0641	14.1371

Table 4. The enhancement percentage in path length

	APF vs QL-APF	**APF vs MQL-APF**	**QL-APF vs MQL-APF**
Env. 1	43.96%	51.1 %	12.51%
Env. 2	0.33 %	15.1 %	14.91%
Env. 3	27.00 %	37.3 %	14.38%
Env. 4	100 %	100 %	18.38%
Env. 5	100 %	100 %	7.14%
Env. 6	100 %	100 %	21.74%

The path length was calculated in all test environments, where the paths planning by the proposed method were the shortest and best paths, as shown in Table. 3 the path length in all test environments.

The enhancement percentage in path length can be observed in Table 4, which shows the improved path when applying the proposed method, MQL-APF. In our case, the average enhancement percentage is approximately 61.88% between the APF method and the OL-APF method. Furthermore, the average enhancement percentage is approximately 67.25% between the APF method and the MOL-APF method. Finally, the average enhancement percentage is approximately 14.68% between the QL-APF method and the MOL-APF method. This means that, on average, the MQL-APF method produces paths that are about 67.25% shorter than those generated by the APF method and approximately 14.68% shorter than those generated by the QL-APF method. This improvement can be crucial in various applications that require path planning in their work.

4.4 Comparison with similar techniques

For further verification of the effectiveness of the proposed MQL-APF method in this work, it was compared with QL [31], modified QL (MQL) [32], and QL-APF [19], methods used in papers marked next to each.

As shown in Fig. 9, MQL-APF has achieved the shortest path length among all the tested algorithms. The red path represents the MQL-APF method in the two types of environments, while the blue path represents the other method.

Table. 5 shows the path length comparison between the three methods and our proposed method in the two test environments.

In another scenario, it applies the four methods mentioned above in the same environment to show the changes in path length. Fig. 10 presents the path planning for the four methods mentioned above, where the blue path represents the QL method, the black path represents the MQL, the green path represents the QL-APF, and the red path represents the prpposed MQL-APF method which shows a clear improvement in terms of path length. These lengths can be seen in Table 6, where the enhancement percentage in the path is approximately 50.15%, 7.21%, and 24.63% for QL, MQL, and QL-APF, respectively, compared to our proposed method.

In summary, the proposed MQL-APF method proved its superiority and achieved favourable results across all test environments. In contrast, the APF algorithm succeeded in simpler design environments

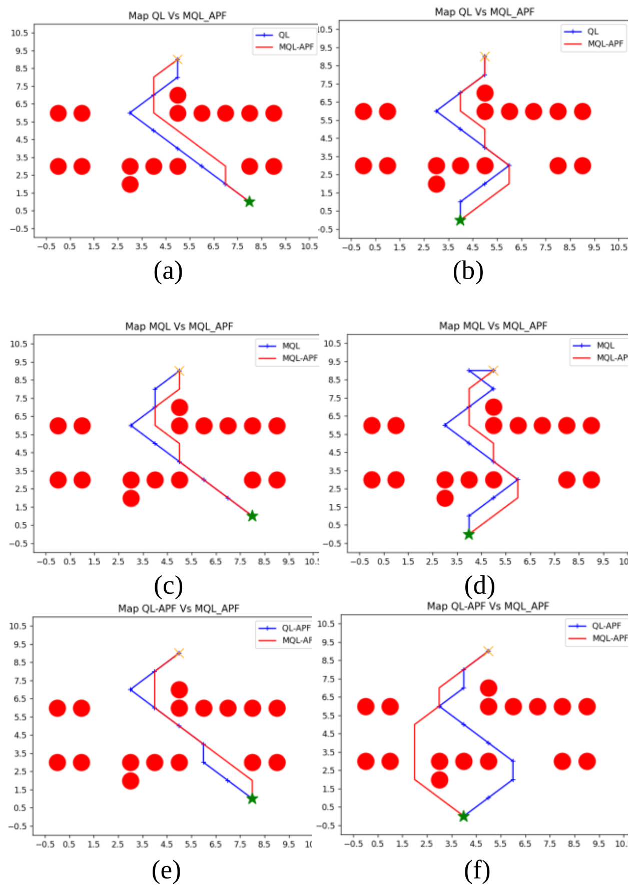


Figure. 9 Path comparison in two test environments: (a) Env.1 CQL Vs. MQL-APF, (b) Env.2 CQL Vs. MQL-APF, (c) Env.1 MQL Vs. MQL-APF, (d) Env.2 MQL Vs. MQL-APF, (e) Env.1 QL-APF Vs. MQL-APF, and (f) Env.2 QL-APF Vs. MQL-APF

Table 5. Path length comparison in meter

Environment	Method	Path length
a	QL	10.9984
	MQL-APF	10.0710
b	QL	11.6839
	MQL-APF	11.0710
c	MQL	10.7964
	MQL-APF	10.0710
d	MQL	12.3137
	MQL-APF	11.0710
e	QL-APF	10.9784
	MQL-APF	10.0710
f	QL-APF	11.8994
	MQL-APF	11.0710

Table 6. Path length comparison

CQL	MQL	QL-APF	MQL-APF
17.2426	12.3137	14.3137	11.4852

Received: March 27, 2024. Revised: May 10, 2024.

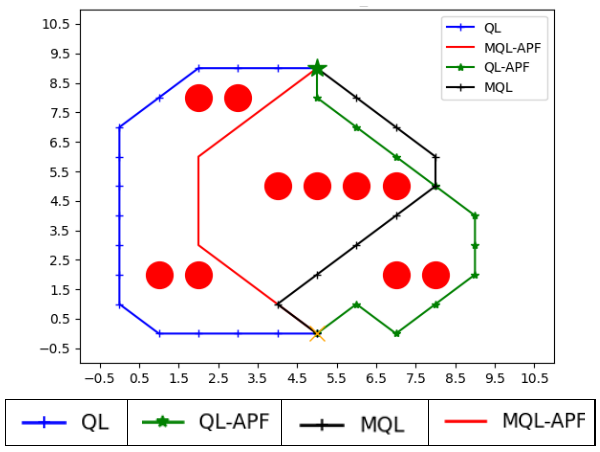


Figure. 10 Path comparison of CQL, MQL, QL-APF and MQL-APF

but failed in more complex ones. Meanwhile, the QL-APF method consistently reached the goal in all test environments, albeit with longer paths. Moreover, the proposed method succeeds in reaching the goal point on a smaller path compared to the QL, MQL, and QL-APF methods.

5. Conclusions

Efficient path planning stands as a crucial necessity across numerous domains. The limitations of the conventional APF and classical QL tactics are triumphed over through the combination (QL-APF) of the two processes. Moreover, the proposed amendment of the combined method (MQL-APF) by involving of concerning dynamic reward with static reward yielded enhanced results in all simulated environments and improved path planning in phrases of path length and smoothness. The results underscore its efficacy as an algorithm capable of calculating safe and dynamically efficient paths even in complicated environments. Notably, it outperforms both the APF method and the QL-APF method, attaining a reduction in path length approximately of about 67.25% as compared to APF and approximately 14.68% in comparison to QL-APF on common. Moreover, our proposed method has demonstrated its superiority as compared with similar strategies, with an improvement rate of approximately 50.15%, 7.21%, and 24.63% for QL, MQL, and QL-APF, respectively.

Regarding the enhancement above, the MQL-APF algorithm presents a multitude of potential applications in path planning, such as self-driving automobiles, exploration vehicles, unmanned aerial vehicles, and autonomous underwater vehicles

This work may expand in several ways in the future, such as by working in environments that contain known and unknown moving obstacles; additionally, the potential for employing multiple agents within complex environments warrants further exploration.

Conflicts of Interest

The authors advertise they have no competing interests. This study was carried out on its own dime, without assistance from outside sponsors.

Author Contributions

Conceptualization, methodology, software, validation, Firdos N Irzoqe; formal analysis, investigation, Firas A. Raheem and Ahmed R. Nasser; resources, data curation, writing—original draft preparation, writing—review and editing, Firdos N. Irzoqe; visualization, project administration, supervision, Firas A. Raheem and Ahmed R. Nasser.

References

[1] A. N. A. Rafai, N. Adzhar, and N. I. Jaini, "A review on path planning and obstacle avoidance algorithms for autonomous mobile robots", *Journal of Robotics*, Vol. 2022, 2022.

[2] Z. E. Kanoon, A. S. Al-Araji, and M. N. Abdullah, "Enhancement of Cell Decomposition Path-Planning Algorithm for Autonomous Mobile Robot Based on an Intelligent Hybrid Optimization Method.", *International Journal of Intelligent Engineering & Systems*, Vol. 15, No. 3, 2022, doi: 10.22266/ijies2022.0630.14.

[3] 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.

[4] M. S. Abed, O. F. Lutfy, and Q. F. Al-Doori, "A Review on Path Planning Algorithms for Mobile Robots", *Engineering and Technology Journal*, Vol. 39, No. 5A, pp. 804–820, 2021, doi: 10.30684/etj.v39i5A.1941.

[5] B. Li and B. Chen, "An adaptive rapidly-exploring random tree", *IEEE/CAA Journal of Automatica Sinica*, Vol. 9, No. 2, pp. 283–294, 2021.

[6] D.-D. Zhu and J.-Q. Sun, "A new algorithm based on Dijkstra for vehicle path planning considering intersection attribute", *IEEE Access*, Vol. 9, pp. 19761–19775, 2021.

[7] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu, "Improved artificial potential field method

applied for AUV path planning,” *Math Probl Eng*, Vol. 2020, pp. 1–21, 2020.

[8] Y. Pan, Y. Yang, and W. Li, “A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV”, *IEEE Access*, Vol. 9, pp. 7994–8005, 2021.

[9] S. Sahu and B. B. Choudhury, “PSO based path planning of a six-axis industrial robot”, In: *Proc. of Computational Intelligence in Data Mining: Proceedings of the International Conference on ICCIDM 2018*, pp. 213–220, 2020.

[10] S. Zhou, X. Liu, Y. Xu, and J. Guo, “A deep Q-network (DQN) based path planning method for mobile robots”, In: *Proc. of 2018 IEEE International Conference on Information and Automation (ICIA)*, pp. 366–371, 2018.

[11] 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.

[12] F. A. Raheem and M. M. Badr, “Smoothed artificial potential field (apf) via points path planning algorithm based on pso”, *AL-yarmouk Journal*, Vol. 11, No. 1, 2019.

[13] F. Raheem and M. Abdulkareem, “DEVELOPMENT OF A* ALGORITHM FOR ROBOT PATH PLANNING BASED ON MODIFIED PROBABILISTIC ROADMAP AND ARTIFICIAL POTENTIAL FIELD”, *Journal of Engineering Science and Technology*, Vol. 15, pp. 3034–3054, 2020.

[14] Y. Chen, G. Bai, Y. Zhan, X. Hu, and J. Liu, “Path planning and obstacle avoiding of the USV based on improved ACO-APF hybrid algorithm with adaptive early-warning”, *IEEE Access*, Vol. 9, pp. 40728–40742, 2021.

[15] A. S. Abdel-Rahman, S. Zahran, B. E. Elnaghi, and S. F. Nafea, “Enhanced Hybrid Path Planning Algorithm Based on Apf and A-Star”, *The International Archives of the Photogrammetry*, Remote Sensing and Spatial Information Sciences, Vol. 48, pp. 867–873, 2023.

[16] J. Zhang, “AI based Algorithms of Path Planning, Navigation and Control for Mobile Ground Robots and UAVs”, *arXiv preprint arXiv:2110.00910*, 2021.

[17] M. Ataollahi and M. Farrokhi, “Online path planning of cooperative mobile robots in unknown environments using improved Q-Learning and adaptive artificial potential field”, *The Journal of Engineering*, Vol. 2023, No. 2, p. e12231, 2023.

[18] E. S. Low, P. Ong, and K. C. Cheah, “Solving the optimal path planning of a mobile robot using improved Q-learning”, *Rob Auton Syst*, Vol. 115, pp. 143–161, 2019.

[19] U. Orozco-Rosas, K. Picos, J. J. Pantrigo, A. S. Montemayor, and A. Cuesta-Infante, “Mobile robot path planning using a QAPF learning algorithm for known and unknown environments”, *IEEE Access*, Vol. 10, pp. 84648–84663, 2022.

[20] F. A. Raheem and M. M. Badr, “Development of Modified path planning algorithm using artificial potential field (APF) based on PSO for factors optimization”, *American Scientific Research Journal for Engineering*, Technology, and Sciences (ASRJETS), Vol. 37, No. 1, pp. 316–328, 2017.

[21] Y. Wang, C. Lu, P. Wu, and X. Zhang, “Path planning for unmanned surface vehicle based on improved Q-Learning algorithm”, *Ocean Engineering*, Vol. 292, p. 116510, 2024.

[22] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots”, In: *Proc. of 1985 IEEE international conference on robotics and automation*, IEEE, pp. 500–505, 1985.

[23] T. Weerakoon, “Artificial potential field and feature extraction method for mobile robot path planning in structured environments”, *Kyushu Institute of Technology*, 2016.

[24] F. A. Raheem, S. M. Raafat, and S. M. Mahdi, “Robot Path-Planning Research Applications in Static and Dynamic Environments”, *Earth Systems Protection and Sustainability*, Vol. 1, pp. 291–325, 2022, doi: 10.1007/978-3-030-85829-2_12.

[25] K. Zhu and T. Zhang, “Deep reinforcement learning based mobile robot navigation: A review”, *Tsinghua Sci Technol*, Vol. 26, No. 5, pp. 674–691, 2021.

[26] Q. Yao et al., “Path planning method with improved artificial potential field—a reinforcement learning perspective”, *IEEE Access*, Vol. 8, pp. 135513–135523, 2020.

[27] E. S. Low, P. Ong, and K. C. Cheah, “Solving the optimal path planning of a mobile robot using improved Q-learning”, *Rob Auton Syst*, Vol. 115, pp. 143–161, 2019.

[28] P. Chintala, R. Dornberger, and T. Hanne, “Robotic path planning by q learning and a performance comparison with classical path finding algorithms”, *International Journal of Mechanical Engineering and Robotics Research*, Vol. 11, No. 6, pp. 373–378, 2022.

[29] D. Elsayed, E. Nasr, A. E. D. Ghazali, and M. Gheith, “PGAQK: An Adaptive QoS-aware

Received: March 27, 2024. Revised: May 10, 2024.

Web Service Composition Approach", *International Journal of Intelligent Engineering and Systems*, Vol. 11, No. 4, pp. 231–240, 2018.

[30] S. Guo, X. Zhang, Y. Du, Y. Zheng, and Z. Cao, "Path planning of coastal ships based on optimized DQN reward function", *J Mar Sci Eng*, Vol. 9, No. 2, p. 210, 2021.

[31] V. Martovytskyi and O. Ivaniuk, "Approach to building a global mobile agent way based on Q-learning", *СУЧАСНИЙ СТАН НАУКОВИХ ДОСЛІДЖЕНЬ ТА ТЕХНОЛОГІЙ В ПРОМИСЛОВОСТІ*, No. 3 (13), pp. 43–51, 2020.

[32] X. Guo, G. Peng, and Y. Meng, "A modified Q-learning algorithm for robot path planning in a digital twin assembly system", *The International Journal of Advanced Manufacturing Technology*, Vol. 119, No. 5, pp. 3951–3961, 2022.