Journal of Marine Science and Application (2026) 25: 575-586
https://doi.org/10.1007/s11804-025-00697-2

RESEARCH ARTICLE

Path Planning for Unmanned Surface Vehicles in Dynamic Environments Based on Artificial Potential Field and Global Guided Reinforcement Learning

Shanqiang Li[1] and Chaoxi Li[1]

Received: 12 December 2024 / Accepted: 01 April 2025

Abstract

For unmanned surface vehicles (USVs), how to find an effective, feasible path that substantially improves mission success rates and time efficiency in dynamic marine environments is a critical issue. To address the path planning problem for USVs using deep reinforcement learning (DRL) in dynamic ocean environments, an improved algorithm based on Deep Q-Networks (DQN) is proposed, which is called Fast Guided Deep Q-Network Algorithm (FG-DQN). This algorithm combines DQN with the artificial potential field (APF) method and uses the A* algorithm to initialize a guiding path in a global static environment and to provide prior knowledge for the USVs. Additionally, the configuration of the reward function using APF and the guiding path effectively reduces the frequency of random movements during the early exploration phase of the DQN algorithm, which accelerates convergence, improves the computational efficiency of path planning, and increases path safety. Finally, the performance of the presented algorithm is validated through experiments in a 2D environment. Compared with traditional reinforcement learning methods such as Q-learning and Sarsa, as well as the original DQN algorithm, FG-DQN is more effective for USV path planning.

Keywords Deep reinforcement learning; Path planning; Unmanned surface vehicles; Fast guided deep Q-Network algorithm

1 Introduction

With the continuous advancement of artificial intelligence technology, research in the field of unmanned vehicles, drones, and unmanned surface vehicles (USVs) has developed rapidly and become more intelligent (Cheng et al., 2021). While research and applications for unmanned vehicles and drones are fairly mature, the complexity and uncertainty of marine environments have restricted the study of USVs. However, as the demand for work and exploration in marine environments continues to grow, USV technology will become increasingly important. USVs are extensively used in fields such as marine exploration, search and rescue, and environmental monitoring (Hong et al., 2020; Zhang et al., 2023a; Gaugue et al., 2019). To accomplish tasks in marine environments, the path planning strategies employed by unmanned vessels play a critical role.

Path planning requires calculating feasible paths from a starting coordinate to a target coordinate on a map based on relevant optimization criteria (Zeng et al., 2016). At present, many traditional path planning algorithms can implement path planning in static environments, such as the A* algorithm (Dechter and Pearl, 1985), which expands from the starting coordinate to the surrounding grids while performing heuristic calculations until the target path is found. This method is simple and suitable for path planning strategies, but it relies heavily on accurate environmental models. When the algorithm fails in a dynamic environment, replanning is required, and repeated replanning can lead to inefficiencies in path planning calculations and poorer success rates. Other methods from different fields, such as genetic algorithms (Cobb et al., 1993; Li et al., 2023), particle swarm optimization (Eberhart and Kennedy, 1995; Hu et al., 2023), and ant colony optimization (Colorni et al., 1991; He et al., 2022), are frequently used in path plan-

Article Highlights

- Deep reinforcement learning methods are used to enable unmanned surface vehicles (USVs) to plan paths in dynamic maritime environments.
- The A* algorithm is employed to generate a guiding path in static nautical charts, direct the USV's movement along the global path, reduce randomness during the early exploration phase of DQN, and decrease the computation time for path planning.
- This approach is combined with the artificial potential field (APF) method to decrease the possibility of collisions for the USV, while the DQN algorithm is used to alleviate the issue of local optima that can occur from using the APF method.

✉ Shanqiang Li
lishanqiang@gmail.com

[1] School of Computer Science and Mathematics, Fujian University of Technology, Fuzhou 350118, China

ning strategies. Given the need for autonomous learning capabilities in current path planning, reinforcement learning algorithms have been introduced as a new solution, such as the Q-learning method (Watkins and Dayan, 1992), which has exhibited satisfactory results in path planning. Reinforcement learning methods are an important category of machine learning, where agents interact with the environment by taking a series of actions to acquire corresponding environmental observations and rewards and offer strong adaptability and learning capabilities. However, traditional reinforcement learning methods struggle to maintain competent generalization in dynamic environments, and the large state space for complex problems leads to dimensionality.

The marine environment is characterized by complexity and uncertainty, which are chiefly reflected in the following aspects: 1) Dynamic Environment: Weather and ocean conditions are greatly variable. Meteorological factors such as wind speed and visibility, as well as oceanographic factors such as currents and tides, remarkably increase the difficulty of navigation. 2) Obstacles: This aspect includes static obstacles such as reefs, islands, and port structures, as well as dynamic obstacles such as other vessels, large marine animals, and floating debris. 3) Communication and Navigation: Remote control operations may suffer from communication delays, whereas navigation signals can be subject to interference that leads to positioning errors. 4) Mission Complexity: USVs are often compelled to perform multiple tasks simultaneously, such as monitoring, search and rescue, and exploration. 5) Uncertainty: Sensor inaccuracies can weaken environmental perception, and the inherent difficulty in creating fully accurate environmental models further complicates prediction and decision making.

As environmental complexity and uncertainty continue to increase, the requirements for USV path planning also rise (Cai et al., 2020; Mac et al., 2016). Thus, researchers have started to employ deep reinforcement learning (DRL) methods to complete path planning in partially unknown dynamic environments (Zhai et al., 2021). DRL does not rely on map data models and only requires the final target to improve path planning results through interaction and training with the environment. Consequently, it has obtained satisfactory results in many applications, but it still faces challenges, such as high randomness in exploration under sparse environmental rewards, which leads to inefficient training processes (Yang et al., 2022).

To address the environmental uncertainties caused by static and dynamic obstacles in marine environments and to overcome the shortcomings of existing methods, this paper improves the existing Deep Q Network (DQN) method by proposing the Fast Guided Deep Q-Network Algorithm (FG-DQN) that combines the A* algorithm and the artificial potential field (APF) method. FG-DQN leverages the advantages of traditional path planning and DRL-based path planning. Specifically, FG-DQN employs traditional path planning algorithms to generate an optimal path in a static environment, which is referred to as a guiding path. When the USV moves in a dynamic environment, it follows the rewards from the APF and guiding path to explore paths and avoid obstacles, thereby reducing exploration randomness, accelerating convergence, and improving path planning efficiency. FG-DQN benefits from deep neural networks and effectively addresses the dimensionality faced by traditional reinforcement learning methods.

The results of this paper are summarized as follows: 1) A 2D marine environment is created and gridified. 2) The guiding path generated by the A* algorithm in a static environment is combined with the current dynamic environmental rewards to decrease randomness during the early exploration phase, increase algorithm convergence speed, and lower calculation time. 3) Integration with the APF method facilitates the utilization of DRL to address the issue of local optima that prevents reaching the target point while combining static and dynamic rewards, enables the USV to plan paths autonomously, and avoids obstacles in the simulated 2D environment. 4) Compared with other traditional reinforcement learning algorithms, FG-DQN can accomplish path planning tasks considering path length, obstacle collision avoidance, and path generation time.

The remaining sections of this paper are as follows: Section 2 introduces the current research status in the areas of path planning, reinforcement learning, and DRL. Section 3 presents the definitions used in our research and the specifics of the proposed FG-DQN algorithm. Section 4 compares the experimental results of the proposed method with those of traditional reinforcement learning methods in a 2D simulated marine environment. Section 5 discusses the summary and conclusions.

2 Related work

2.1 Traditional path planning algorithm

Recently, with the increasing demand for work and exploration in marine environments, the path planning and obstacle avoidance problems of USVs have become a popular research topic, and more scholars are constantly proposing new ideas and algorithms to solve these problems. Traditional path planning methods, such as the A* algorithm, are widely used in static environments. The A* algorithm maintains an open list and a closed list and utilizes a heuristic function to estimate costs. The A* algorithm iteratively selects the optimal node for expansion until the shortest path from the start point to the goal point is found. However, in dynamic environments, the performance of the A* algorithm is greatly affected due to the inability to

predict and avoid unknown obstacles. Yu et al. (2021) improved the existing global path planning methods based on the A* and APF methods and addressed the limitations of path planning being independent of the USV control phase. They also established the effectiveness of the algorithm in different environments through simulation. However, the path length of the algorithm is not optimal, and dynamic obstacles in the marine environment have not been considered. To improve the autonomy of the system, increase fault-tolerant resilience, and solve the problems of low payload capacity and short endurance of USV, Sang et al. (2021) proposed a new deterministic algorithm called Multi Sub Objective Artificial Potential Field Method, but this algorithm does not consider dynamic obstacles in marine environments.

2.2 Traditional reinforcement learning algorithm

With the development of artificial intelligence, reinforcement learning has also been applied as a solution to path planning problems. As one of the classic reinforcement learning methods, the Q-learning algorithm is strongly adaptable to the environment, and reinforcement learning methods do not require any human knowledge as a priori knowledge (Watkins and Dayan, 1992). The Q-learning algorithm enables the agent to learn an optimal policy through continuous trial and error in the environment, updates the Q-value table iteratively, and eventually finds the best path from the start point to the goal point. However, the Q-learning algorithm has disadvantages such as long learning time and low exploration efficiency, so many studies have improved classical reinforcement learning methods. Hao et al. (2023) proposed a dynamic fast Q-learning (DFQL) algorithm for the path planning problem of unmanned underwater vehicles in some known marine environments. The DFQL algorithm combines the Q-learning algorithm with the APF method. The performance of the DFQL algorithm was verified in different environments, but its calculation time is extremely long, and its convergence is slow. Zhang et al. (2023b) introduced a new reward function derived from the predator–prey model into the traditional Q-learning algorithm to overcome the problem of becoming stuck in local optima. Simulation and experimental verification exhibited that the algorithm outperforms traditional Q-learning in terms of repetition rate and turn count, but it does not consider dynamic obstacles.

2.3 Deep reinforcement learning algorithm

To address the problems of unsuitability to complex environments and poor generalization ability in reinforcement learning, DRL is proposed by combining deep neural networks with reinforcement learning. The DQN (Mnih, 2013) algorithm combines the Q-learning algorithm and deep neural networks, which solves the problem that the Q-learning algorithm is not suitable for complex problems. The DQN algorithm processes actions as input data to the neural network, which then outputs the corresponding Q-values. This approach removes the need for the Q-learning algorithm to store Q-values in a Q-table and thereby enables effectively addressing dimensionality in complex environments. The introduction of the DQN algorithm has demonstrated the potential of DRL algorithms in path planning applications. Presently, numerous studies have investigated the use of DQN-based algorithms in the field of USV path planning. Zhai et al. (2021) applied the DQN algorithm to the path planning problem of unmanned underwater vehicles (UUVs), considering that DRL possesses perception, decision making, and strong learning capabilities to improve the autonomous navigation capability of USVs. The approach was validated on a simulation platform, but the algorithm suffers from slow convergence during the initial training phase. Huang et al. (2021) proposed a double-DQN algorithm to optimize obstacle avoidance and path planning. This algorithm decouples action selection and action evaluation for target Q-values and uses the current network to index the action value corresponding to the maximum Q-value. The selected action value is then fed into the target network to compute the target Q-value. The simulation results demonstrate the superior performance of the double-DQN algorithm in path planning compared with the standard DQN algorithm. However, the algorithm remains unstable during the initial training phase and has issues similar to those of the traditional DQN algorithm. Liu et al. (2024) introduced a maritime search and rescue path planning method for USVs based on the DQN algorithm and enabled USVs to perform search and rescue tasks efficiently and rationally. First, an improved task allocation algorithm is used to divide the search area and to ensure that each task region has a priority and avoids overlap. Second, the algorithm designs a probability-weighted reward function and trains the USV to obtain an optimal search path. Finally, based on the simulation results, the search path generated by this algorithm prioritizes high-probability regions and improves search and rescue efficiency while comprehensively considering obstacle avoidance and collision prevention. However, the algorithm relies on probability estimation, which may lead to potential collisions with obstacles. Additionally, in dynamic environments, the success probability may need to be recalculated, which results in poor real-time performance. Therefore, this paper selects the DQN method as the framework to address the problems of reinforcement learning methods, proposes an FG-DQN algorithm to reduce the convergence time to the optimal solution, and generates feasible paths that are highly safe and short in length in environments with dynamic obstacles, which can solve the problems of high initial exploration randomness and long calculation time in DRL algorithms.

3 Method

The decision model of the USV in this article is based on DRL, whose underlying mathematical model is based on the Markov decision process (MDP). This section analyzes MDP and the traditional DQN algorithm and proposes the FG-DQN algorithm.

3.1 Markov decision process

Reinforcement learning is a type of unsupervised learning algorithm proposed by Minsky (1954). Reinforcement learning is achieved through an agent–environment interface, which includes elements such as the current state of the environment (s), the next state of the environment (s'), actions (a), environmental rewards (R), and strategies (Π). The goal of reinforcement learning is to acquire the state action value function of the state through interaction with the environment, continuously optimize its behavior strategy, adjust its behavior, and finally determine the best decision strategy. Intelligent agents are not informed of the correct action during the action, and they must discover which actions can bring more rewards to maximize the rewards. The learning of reinforcement learning is a continuous trial-and-error process between the agent and the environment, so reinforcement learning can also be considered a dynamic interactive process (Tamang et al., 2021), as shown in Figure 1.

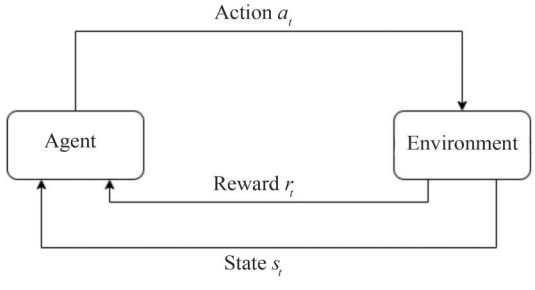

Figure 1 Reinforcement learning principle

When an intelligent agent interacts with the environment, performing different actions results in different states. The state transition model of the environment can be seen as a simplified probability model that represents the probability of the agent transitioning from the current state s to the next state s' after taking action a. Assuming that the state transition model conforms to Markov properties, the environment transition model can be represented by Equation (1):

$$p_{ss'}^{a} = E\left(S_{t+1} = s' \middle| S_t = s, A_t = a\right) \tag{1}$$

Strategy Π can be represented by Equation (2):

$$\pi(a|s) = P\left(A_t = a \middle| S_t = s\right) \tag{2}$$

The state value function is the cumulative expected return from the current state s to the final state, which can be expressed by the Bellman formula shown in Equation (3):

$$\begin{aligned} v_\pi(s) &= E_\pi\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \middle| S_t = s\right) = \\ &E_\pi\left(R_{t+1} + \gamma v\left(S_{t+1}\right) \middle| S_t = s\right) \end{aligned} \tag{3}$$

The action value function is the cumulative expected return acquired by continuing to interact with the environment according to strategy Π after completing action a from the current state s. The action value function can also be expressed by the Bellman formula shown in Equation (4):

$$q_\pi(s,a) = E_\pi\left(R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots \middle| S_t = s, A_t = a\right) \tag{4}$$

Similar to the state value function, the state action function can be expressed by the Bellman formula shown in Equation (5):

$$q_\pi(s,a) = E_\pi\left(R_{t+1} + \gamma q\left(S_{t+1}, A_{t+1}\right) \middle| S_t = s, A_t = a\right) \tag{5}$$

where R_{t+1} is the environmental reward of state s_{t+1}.

Equations (3), (4), and (5) are further simplified as Equation (6):

$$\begin{aligned} v_\pi(s) &= \sum_{a \in A} \pi(a|s) q_\pi(s,a) = \\ &\sum_{a \in A} \pi(a|s)\left(R_s^a + \gamma \sum_{s'} P_{ss'}^a v_\pi(s')\right) \end{aligned} \tag{6}$$

where R_s^a is the environmental reward obtained after selecting action a in state s.

In reinforcement learning, calculating the state value function determines the expected return value in the current state. The state value function is the expected return value of an action in the current state. To maximize the state value function, it is found after selecting the optimal action a in the current state.

3.2 DQN algorithm

DRL combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning. The advantage of DRL lies in using deep neural networks to learn the low-dimensional features of high-dimensional states through dimensionality reduction when interacting with the environment. DQN further proposes an experience replay mechanism and adopts a structure of two networks based on DRL methods.

3.2.1 Experience replay

The underlying mathematical model of reinforcement learning is the Markov Decision Model (MDP), which requires caching a batch of quadruples (s, a, r, s') into the experience pool for training. Because each action executed transitions to the next state and receives corresponding rewards, the quadruples generated by each action during each training cycle are directly placed into the experience pool.

Due to the continuous transition of states, the quadruples are correlated, which can lead to overfitting when training without an empirical replay mechanism due to the lack of independence between training samples. Experience replay overcomes the correlation of experience samples and accelerates training by randomly selecting a batch size of quadruples from the experience pool as the training set.

3.2.2 Target network

DQN uses two networks: the current network and the target network. The algorithm structure is presented in Figure 2. The action value function of the current network is represented as $Q(s, a)$, and the action value function of the target network is represented as $Q(s', a')$. The calculation formulas are simplified by Equation (4) and represented by Equations (7) and (8):

$$Q_t(s, a) = r + \gamma \max_a Q_{t+1}(s, a) \tag{7}$$

$$Q_t(s', a') = r + \gamma \max_{a'} Q_{t+1}(s', a') \tag{8}$$

The current network calculates the Q-value and iteratively updates the Q-value for policy selection. The target network determines the Q-value of the next state. $Q(s, a)$ is the approximation of $Q(s', a')$ by the current network under the current network parameters. During training, $Q(s, a)$ is made closer to $Q(s', a')$ by updating the network parameters. The MSE loss function is used during training and can be represented by Equation (9):

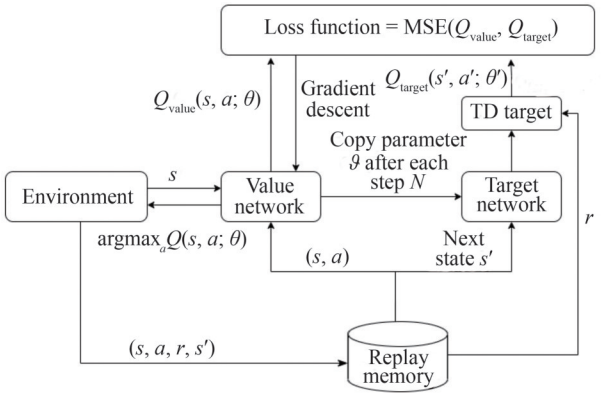

Figure 2 Flowchart of DQN

$$L(\theta) = E\left[\left(r + \gamma \max_{a'} Q(s', a'|\theta) - Q(s, a|\theta)\right)^2\right] \tag{9}$$

Network parameter θ is updated through stochastic gradient descent, and the formula can be represented by Equation (10):

$$\nabla L(\theta) = E\left[\left(r + \gamma \max_{a'} Q(s', a'|\theta) - Q(s, a|\theta)\nabla Q(s, a|\theta)\right)\right] \tag{10}$$

Combining the current network with the target network reduces the update frequency of the target network, provides a more stable target to update the main network, reduces the volatility of learning, stabilizes learning, and prevents overfitting.

3.3 Improvement of DQN algorithm

3.3.1 Global guidance path

The main function of the global guidance path is to generate and provide long-term effective global information quickly through a global static algorithm in a known static environment to assist in the path planning and movement of the USV. To address the issue of high randomness in the early stages of DQN training, a guiding path is generated. This goal is achieved through a reward function that provides rewards without requiring the USV to follow the current global guidance strictly. When encountering movable unknown obstacles, the reward function still moves through the DQN algorithm's own perception and decision-making ability, which promotes the convergence of the USV algorithm.

A global static guidance path is generated before the start of each training round. At each step of training, the reward function provides rewards through different situations: 1) When the USV moves on a blank unit guided by the guidance path, a small positive reward r_1 is generated. 2) When the USV collides with a static or dynamic obstacle, it receives a large negative reward r_2. 3) When the USV collides while traveling on the guidance path, it receives a negative reward with a value of $r_1 + r_2$. The reward function can be defined as Equation (11):

$$R_G(t) = \begin{cases} r_1 & p_{usv}(t+1) \in G \\ r_2 & p_{usv}(t+1) \in O_s + O_d \\ r_1 + r_2 & p_{usv}(t+1) \in G \cup (O_s \cup O_d) \end{cases} \tag{11}$$

where $p_{usv}(t+1)$ is the position of the USV after taking action a_t step t, and $R_G(t)$ is the guidance reward at step t.

3.3.2 Artificial potential field

The APF (Khatib, 1986) method introduces potential

field forces from mechanics and assumes the presence of virtual force fields in the environment that influence movement. The APF method has the advantages of simple design ideas, low computational complexity, and the ability to adjust according to actual changes. Improving DQN using the APF method can accelerate algorithm convergence and address the issue of paths generated by the DQN algorithm being extremely close to obstacles during training. Thus, the algorithm is improved using APF to generate corresponding rewards for attraction and repulsion in the environment. The current position of the USV is used as the origin, attractive and repulsive forces are generated, and the subsequent resultant force vector is compared with the next action of the USV: 1) If the resultant force vector corresponds to the same vector component sign as the USV action vector, a positive reward is obtained. 2) When the sign of the corresponding vector component of the resultant force vector differs from that of the USV action vector, a negative reward is obtained. The calculation for the APF method is presented in Equations (12)–(15):

$$U_a(s_t) = \frac{1}{2} k_a \rho_g^2(s_t) \tag{12}$$

$$U_r(s_t) = \begin{cases} \frac{1}{2} k_r \left(\frac{1}{\rho_{\text{ob}}(s_t)} - \frac{1}{\rho_o} \right)^2 & \rho_{\text{ob}}(s_t) \leqslant \rho_o \\ 0 & \rho_{\text{ob}}(s_t) > \rho_o \end{cases} \tag{13}$$

$$U_n(s_t) = U_r(s_t) + U_a(s_t) \tag{14}$$

$$U(s_t) = \left| \frac{U_{\max} - U_{n1}(s_t)}{U_{\max}} \right| \tag{15}$$

where $U_a(s_t)$ is the gravitational field that generates gravity in state s_t, $U_r(s_t)$ is the gravitational field that generates gravity in state s_t, $U_n(s_t)$ is the total potential energy of state s_t, $\rho_g(s_t)$ is the Euclidean distance from state s_t to the center of the target position, $\rho_{\text{ob}}(s_t)$ is the Euclidean distance from state s_t to the center of the obstacle, ρ_o is the obstacle influencing factor, K_a and K_r are proportionality factors, $U(s_t)$ is the potential energy of state s_t, and $U_{\max}$ is the highest potential energy of state s_t. The reward function is expressed as Equation (16):

$$R_{\text{APF}}(t) = n_1 \times r_4 + n_2 \times r_5 \tag{16}$$

where n_1 is the number of vector components with the same symbol, and n_2 is the number of vector components with different symbols. The enhanced algorithm is presented above as Algorithm 1.

Algorithm 1 FG-DQN algorithm

Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function Q with weights $\theta^- = \theta$
for episode = 1, M **do**
  Initialize state space s_0
  Initialize a global guidance path P_{final}
  for t = 1, T **do**
    With probability ε select a random action a_t
    otherwise select $a_t = \text{argmax}_a Q(s_t, a; \theta)$
    Execute action a_t
    emulator and observe reward r_t^o
    Compare with P_{final} and the APF vector, obtain guidance reward r_t^G and APF reward r_t^{APF}
    Add reward r_t^o, r_t^G, r_t^{APF} in proportion and get reward r_t and next state s_{t+1}
    Set $s_{t+1} = s_t$
    Store transition (s_t, a_t, r_t, s_{t+1}) in D
    Sample random minibatch of transitions (s_t, a_t, r_t, s_{t+1}) from D
    Set $y = \begin{cases} r_j & \text{if episode terminates at step } j+1 \\ r_j + \gamma \max_{a'} Q(s', a'; \theta^-) & \text{otherwise} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(s, a; \theta))^2$
    Update the network parameter by action-value function approximation $\theta = \theta + \Delta\theta$
    Every C steps reset $\theta^- = \theta$
  end for
end for

4 Experiment design and simulation results

4.1 Experiment design

4.1.1 Simulated environment

To verify the proposed algorithm, a map coordinate system is designed, and the simulation environment is divided into a grid of 100 × 100 squares, each with a side length of 2r. In this map, free navigable areas are represented by white grids, whereas static obstacles (such as shoals or islands) are denoted as 100 × 100 blue squares. Dynamic obstacles (small vessels or floating debris) are represented as 5 × 5 yellow squares. Figure 3(a) presents the current marine environment, where the blue areas indicate obstacles detected by sensors. When converting the current marine environment into a grid map, any static obstacle smaller than 10 × 10 is approximated as 10 × 10, as illustrated in Figure 3(b). In Figure 3(c), the red squares indicate the starting and ending points of the USV, with the starting coordinates at (10,

0) and the ending coordinates at (90, 100). The schematic of the USV and the definition of safety distance are presented in Figure 3(d), where x_1, y_1 represents the map coordinate system, and x_2, y_2 represents the USV body coordinate system, with the centroid of the body coordinate system as the origin. To avoid collisions during path planning, the safety distance for the USV is defined as a circle with a radius of r. Based on this safety distance definition, the USV is projected onto the map and treated as a 10 × 10 square with a side length of $2r$. This paper evaluates and analyzes two different grid maps in Figure 4 to determine the effectiveness and feasibility of the proposed algorithm.

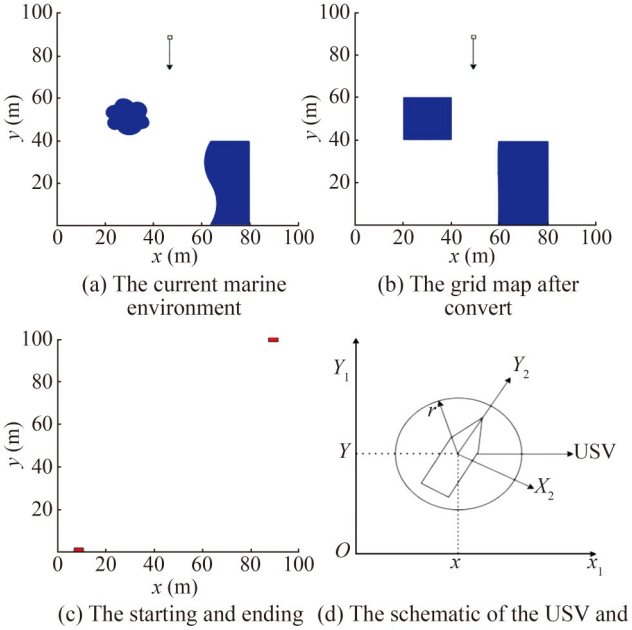

Figure 3 USV path planning environment and problem description

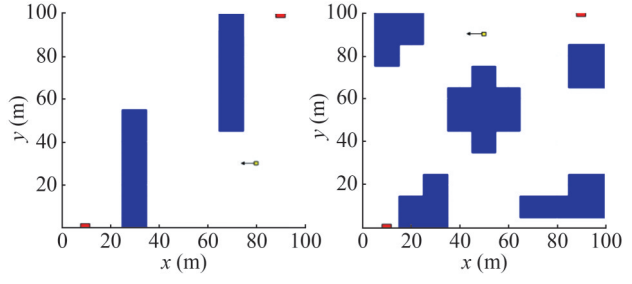

Figure 4 Two testing environments for USV

4.1.2 Action space design

Action space is the collection of all possible actions of the USV in the current environment. To simplify the complexity and difficulty of experimental simulation for algorithm training, as well as to fit the grid map environment model, this paper discretizes the action space of USV and defines it as the set of discrete actions in eight navigation directions, as presented in Equations (17) and (18), where d is set as 10 in our experiment. Each action causes a change in the current state of the USV, as demonstrated in Figure 5.

$$A = \left[a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8\right] \tag{17}$$

$$A = \begin{cases} a_1 = \text{up} \\ a_2 = \text{down} \\ a_3 = \text{left} \\ a_4 = \text{right} \\ a_5 = \text{up to the left} \\ a_6 = \text{up to the right} \\ a_7 = \text{down to the left} \\ a_8 = \text{down to the right} \end{cases} \tag{18}$$

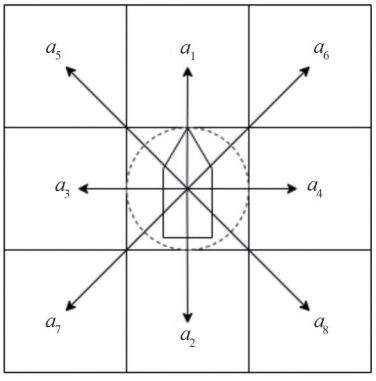

Figure 5 Action space

4.1.3 Reward function

The reward and punishment function is employed to determine the value of behavior; USV interacts with the environment, obtains rewards through the reward and punishment function, and determines the strategy of action through the reward function (Jiang et al., 2019). The DRL methods aim to find the optimal path from the starting point to the target point by maximizing the cumulative reward value of the USV.

USV has four states: 1) Collision occurs with obstacles or map boundaries. 2) No collision occurs, and the action belongs to [a_1, a_2, a_3, a_4]. 3) No collision occurs, and the action belongs to [a_5, a_6, a_7, a_8]. 4) The target point is reached. The first state should be avoided; thus, a large negative reward is given when the USV is in this state. The second and third states are in the process of searching for the path but have not reached the target point, so a small basic negative reward is given. However, the third state moves further than the second state, so the negative reward given is slightly greater than the second state. Based on the basic negative reward, the improved reward in this paper is added to the second and third states according to a specific parameter ratio. The fourth state is the tar-

get state of the USV, so a large positive reward is given when it is reached. The reward function is represented by Equation (19):

$$R = \begin{cases} -20 & S \in S_1 \\ -1 + \alpha \times R_{\mathrm{APF}} + \beta \times R_G & S \in S_2 \\ -1.5 + \alpha \times R_{\mathrm{APF}} + \beta \times R_G & S \in S_3 \\ 20 & S \in S_4 \end{cases} \tag{19}$$

When the USV collides with obstacles or the map boundary, it receives a reward of −20. If the USV moves vertically or horizontally without colliding, it receives a base reward of −1 and additional rewards based on the attractive potential field (APF) and global rewards, weighted by factors α and β. For diagonal movement without collision, the USV receives a reward of −1.5, along with the same APF and global rewards. Finally, when the USV successfully reaches the target point, it earns a reward of 20.

4.1.4 Parameter configuration

The experimental simulation platform is configured with Windows 11, an i7-12650h processor, 16 GB of RAM, an NVIDIA GeForce RTX 3060 graphics card, and Python version 3.7. The parameter settings used in the experiment are listed in Table 1.

During training, parameter selection aims to balance between stability and performance. Actions transition from extensive random exploration of the environment in the initial phase to leveraging learned strategies in the later phase, and the exploration rate gradually decreases as training progresses. A higher initial exploration rate facilitates thorough environmental exploration in the early stages, whereas a lower final exploration rate guarantees a focus on exploitation in the later stages. An extremely low initial exploration rate may result in insufficient early exploration. An overly high final exploration rate may lead to unstable strategies in the later phases. A higher discount factor allows the algorithm to prioritize future rewards, whereas a lower discount factor may result in short-sighted strategies. Thus, the discount factor is set to 0.99. The learning rate controls the magnitude of updates, and a smaller learning rate helps avoid instability caused by excessively rapid updates. An extremely high learning rate may lead to training instability, whereas an overly low learning rate may result in slow convergence. The total number of episodes required for algorithm convergence is determined experimentally. For the four algorithms, DQN and FG-DQN converge within 200 episodes, whereas the traditional reinforcement learning algorithms fail to converge even after 500 episodes. Thus, the training episodes are uniformly set to 200 to facilitate the comparison of the results of the algorithms under the same training conditions. In the improved FG-DQN algorithm, a lower proportion of the APF reward ensures that the USV maintains a safe distance from obstacles while avoiding navigation failure in environments with dense obstacles. A higher proportion of the global guiding reward provides the USV adequate guidance during movement and reduces the computational time caused by extensive random exploration in the initial phase. An extremely small global guiding reward may lead to excessive random exploration. Through comparative experiments with different reward configurations, the optimal performance metrics—such as computation time, path length, and smoothness—are achieved when $\alpha = 0.6$ and $\beta = 0.1$.

4.2 Simulation results

Two testing environments are established for obstacles of different densities, and dynamic obstacles are set on the USV's moving route.

4.2.1 Simulation and analysis of a small number of obstacles

The starting point of the USV is (10, 0), and the target point is (90, 100). The path planning diagram and step change diagram of the experimental results are illustrated in Figures 6 and 7, respectively.

The simulation path planning diagrams of the four algorithms under a small number of obstacles are shown in Figure 6, where Figure 6(a) is Q-learning, Figure 6(b) is Sarsa, Figure 6(c) is original DQN, and Figure 6(d) is FG-DQN. Comparing the step count changes of the four algorithms in Figure 7 reveals that Q-learning and Sarsa algorithms do not converge in environments with dynamic obstacles. DQN achieves convergence stability after 180 rounds of iterative training, whereas the improved FG-DQN already converges and stabilizes after 125 rounds.

Table 1 Parameter settings in the experiment

Algorithms	Parameter selection
Q-learning	Learning rate = 0.01, $\gamma = 0.99$, $\varepsilon_{\mathrm{initial}} = 0.2$, $\varepsilon_{\mathrm{final}} = 0.001$, number of explore episodes = 200
Sarsa	Learning rate = 0.01, $\gamma = 0.99$, $\varepsilon_{\mathrm{initial}} = 0.2$, $\varepsilon_{\mathrm{final}} = 0.001$, number of explore episodes = 200
DQN	Learning rate = 0.001, $\alpha = 0.6$, $\beta = 0.1$, $\gamma = 0.99$, $\varepsilon_{\mathrm{initial}} = 0.6$, $\varepsilon_{\mathrm{final}} = 0.1$, replay buffer size = 100 000, batch size = 128, number of explore episodes = 200
FG-DQN	Learning rate = 0.001, $\alpha = 0.6$, $\beta = 0.1$, $\gamma = 0.99$, $\varepsilon_{\mathrm{initial}} = 0.6$, $\varepsilon_{\mathrm{final}} = 0.1$, replay buffer size = 100 000, batch size = 128, number of explore episodes = 200

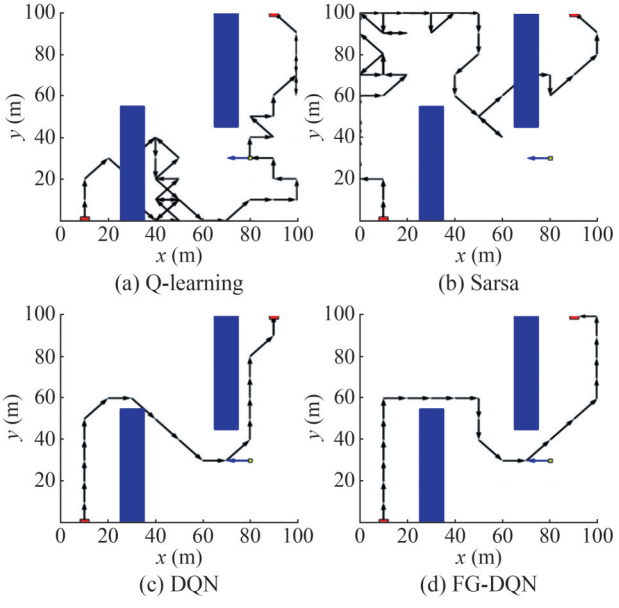

Figure 6 Simulation path planning diagrams of four algorithms under a small number of obstacles

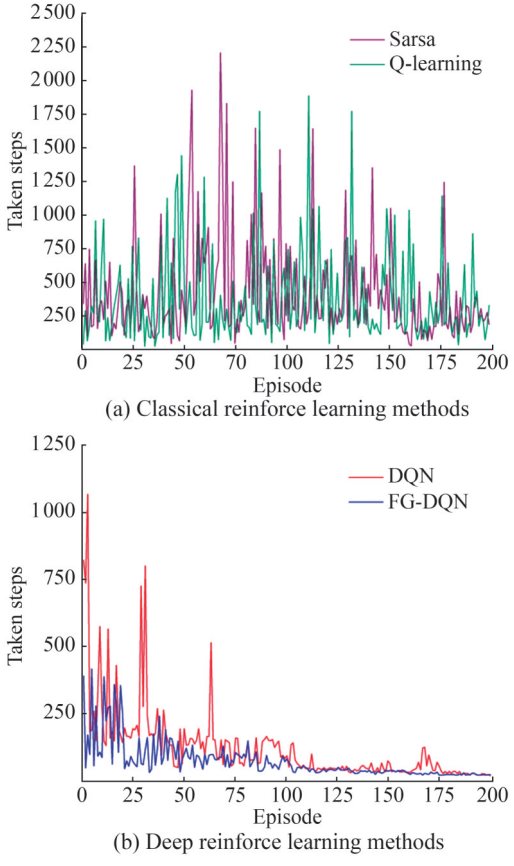

Figure 7 Step count variation diagram of four algorithms under a small number of obstacles

In environments with dynamic obstacles and a moderately low density of obstacles, the Q-learning and SARSA algorithms experience an increase in state space as training progresses, which leads to dimensionality and prevents training from converging. This limitation results in substantial increases in computation time, episode steps, and the number of turns. By contrast, the DQN and FG-DQN algorithms can achieve convergence in such dynamic environments by leveraging deep neural networks and demonstrate remarkable advantages over traditional reinforcement learning algorithms in terms of performance metrics. Compared with the standard DQN algorithm, FG-DQN incorporates an APF reward and ensures that the generated paths maintain a safe distance from obstacles. Additionally, the global guiding reward reduces the randomness of initial exploration and enables convergence while decreasing the computational time required for training and accelerating convergence. FG-DQN exhibits higher computational efficiency in such environments.

Table 2 presents that the Q-learning and Sarsa algorithms encounter collisions in environments with dynamic obstacles, whereas DQN and FG-DQN successfully reach the target point while avoiding obstacles. FG-DQN maintains a certain distance from obstacles to prevent being extremely close, which results in an overall path that is 7% longer than that of the DQN algorithm. However, the difference in path length is minimal, and the path generated by the improved algorithm is safer, with 12% fewer turns. Moreover, the improved algorithm converges more quickly and reduces the time consumed by 18%.

Table 2 Comparison of four algorithms in a simulation environment with a small number of obstacles

Statistics	Q-learning	Sarsa	DQN	FG-DQN
Path steps	51	58	18	22
Time	453.83	485.32	305.34	248.61
Inflection point	35	37	8	7

4.2.2 Simulation and analysis of a large number of obstacles

In this simulation environment, the starting point of the USV is (10, 0), and the target point is (90, 100). The path planning diagram and step change diagram of the experimental results are presented in Figures 8 and 9, respectively.

The simulation path planning diagrams of the four algorithms under a large number of obstacles are presented in Figure 8, where Figure 8(a) is Q-learning, Figure 8(b) is Sarsa, Figure 8(c) is original DQN, and Figure 8(d) is FG-DQN. Comparing the step count changes of the four algorithms in Figure 9 reveals that Q-learning and Sarsa still collide in environments with dynamic obstacles, which leads to algorithm nonconvergence. DQN achieves convergence stability after 100 rounds of iterative training, whereas the improved FG-DQN already acquires convergence stability after 75 rounds.

Table 3 indicates that the Q-learning and Sarsa algorithms do not converge, whereas DQN and FG-DQN suc-

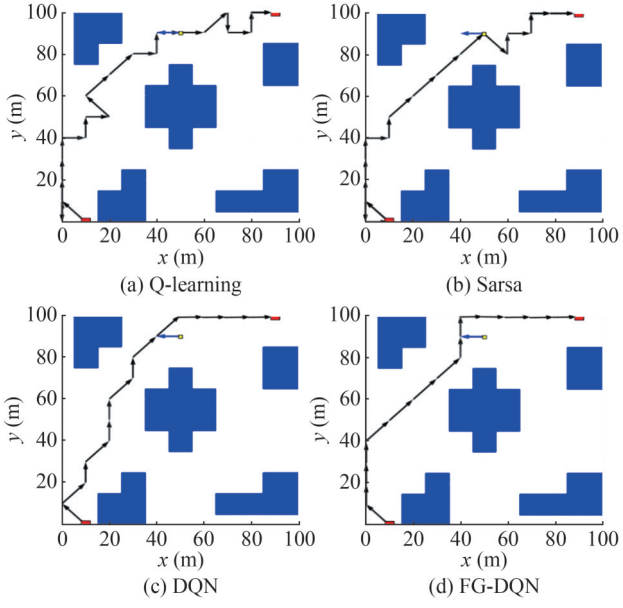

Figure 8 Simulation path planning diagrams of four algorithms under a large number of obstacles

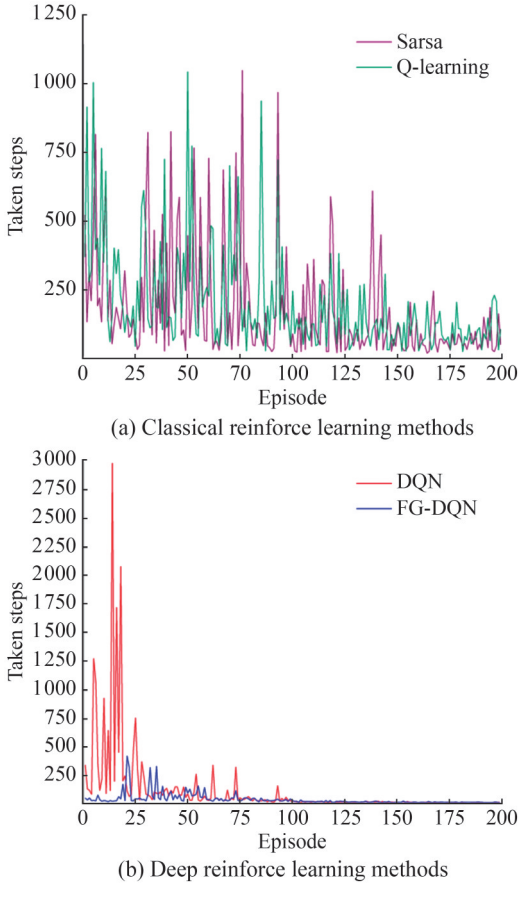

Figure 9 Step count variation diagram of four algorithms under a large number of obstacles

cessfully reach the target point while avoiding obstacles. FG-DQN uses 7% more steps than DQN in path planning

Table 3 Comparison of four algorithms in a simulation environment with a large number of obstacles

Statistics	Q-learning	Sarsa	DQN	FG-DQN
Path steps	22	19	14	15
Time	329.25	311.46	213.12	175.61
Inflection point	15	12	8	4

but is safer, with 50% fewer turns, which results in better path smoothness. Moreover, the time consumed by the algorithm is reduced by 17%.

In environments with dynamic obstacles and a high density of obstacles, the paths generated by Q-learning and SARSA often collide with dynamic obstacles, which results in a failure to converge and a substantial increase in computation time. These algorithms continuously attempt to adjust their strategies, but due to the dynamic nature of the environment, they struggle to find stable policies, which leads to wasted computational resources. By contrast, DQN and FG-DQN can generate the shortest collision-free paths under the same dynamic conditions. However, while the standard DQN algorithm requires 100 training iterations to achieve convergence stability, FG-DQN incorporates an APF reward and ensures safer paths when intersecting with dynamic obstacles. Additionally, the global guiding reward contributes to smoother paths and further decreases computation time. Consequently, FG-DQN achieves convergence in only 75 training iterations and demonstrates faster convergence and smoother paths compared with the DQN algorithm. FG-DQN exhibits stronger adaptability and higher computational efficiency in dynamic environments.

Comparative analysis of the two sets of experiments reveals that the proposed FG-DQN in this paper is effectively adaptable to different map environments. For the first grid map, the DRL algorithms exhibit substantial advantages over traditional reinforcement learning algorithms. Although the difference in training steps between DQN and FG-DQN is not pronounced due to the sparse distribution of obstacles, FG-DQN demonstrates considerable superiority in terms of average steps, computation time, and the number of turns. For the second grid map, the DRL algorithms again outperform traditional methods by effectively avoiding dynamic obstacles. However, due to the higher density and dispersion of obstacles in this environment, DQN requires more exploration steps in the initial phase. Consequently, FG-DQN not only excels in overall performance metrics but also considerably reduces the number of initial exploration steps. The analysis of these two experiments reveals the clear advantages of FG-DQN over traditional reinforcement learning algorithms and the standard DQN algorithm in terms of performance metrics and further validates the effectiveness, feasibility, and superiority of the proposed algorithm.

5 Conclusion

To solve the path planning problem of USVs in dynamic marine environments based on DRL, FG-DQN is proposed. FG-DQN utilizes the A* algorithm to initialize a guiding path in a global static environment, provide prior knowledge to the USV, and address excessive exploration rates during the initial phase, and thus avoid blind exploration. FG-DQN also integrates with the APF method, enhances the directional guidance of initial exploration and improves obstacle avoidance while ensuring a safe distance from obstacles. The experimental results confirm that the improved algorithm effectively reduces the frequency of random movements during the initial exploration phase of DQN through the configuration of rewards based on APF and the guiding path, accelerates convergence, and improves computational efficiency. The resulting paths are safer, less prone to collide with obstacles, and computed in less time compared with other approaches. Compared with traditional reinforcement learning algorithms and the original DQN algorithm, the proposed improved algorithm demonstrates better path planning efficiency and practicality for USVs. For our future research, we plan to integrate mean field game strategies into the path planning of large number of USVs; see (Imanuvilov et al., 2023; Liu and Zhang, 2025; Liu and Lo, 2025; Ding et al., 2025; Liu et al., 2023) for some related recent breakthroughs.

Funding Supported by the Science Research Foundation for Introduced Talents, Fujian Province of China under Grant Nos. GY-Z21215, GY-Z21216.

Competing interest The authors have no competing interests to declare that are relevant to the content of this article.

References

Cheng C, Sha Q, He B, Li G (2021) Path planning and obstacle avoidance for AUV: A review. Ocean Engineering 235: 109355. https://doi.org/10.1016/j.oceaneng.2021.109355

Cai K, Wang C, Cheng J, De Silva CW, Meng MQH (2020) Mobile robot path planning in dynamic environments: A survey. Instrumentation 6(2): 90-100. https://doi.org/10.48550/arXiv.2006.14195

Cobb HG, Grefenstette JJ (1993) Genetic algorithms for tracking changing environments. Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, USA, 523-530

Colorni A, Dorigo M, Maniezzo V (1991) Distributed optimization by ant colonies. Proceedings of the First European Conference on Artificial Life, Vol. 142, Paris, France, 134-142

Dechter R, Pearl J (1985) Generalized best-first search strategies and the optimality of A*. Journal of the ACM (JACM) 32(3): 505-536. https://doi.org/10.1145/3828.3830

Ding MH, Liu H, Zheng GH (2025) Determining a stationary mean field game system from full/partial boundary measurement. SIAM Journal on Mathematical Analysis 57(1): 661-681. https://doi.org/10.1137/23M1594327

Eberhart R, Kennedy J (1995) A new optimizer using particle swarm theory. In MHS'95, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39-43. https://doi.org/10.1109/MHS.1995.494215

Gaugue A, Menard M, Migot E, Bourcier P, Gaschet C (2019) Development of an aquatic USV with high communication capability for environmental surveillance. OCEANS 2019-Marseille, Marseille, France, 1-8. https://doi.org/10.1109/OCEANSE.2019.8867256

Hong SM, Nam KS, Ryu JD, Lee DG, Ha KN (2020) Development and field test of unmanned marine vehicle (USV/UUV) with cable. IEEE Access 8: 193347-193355. https://doi.org/10.1109/ACCESS.2020.3032961

Huang Z, Lin H, Zhang G (2021) The USV path planning based on an improved DQN algorithm. 2021 International Conference on Networking, Communications and Information Technology (NetCIT), Beijing, China, 162-166. https://doi.org/10.1109/NetCIT53626.2021.00037

Hao B, Du H, Yan Z (2023) A path planning approach for unmanned surface vehicles based on dynamic and fast Q-learning. Ocean Engineering 270: 113632. https://doi.org/10.1016/j.oceaneng.2022.113632

Hu X, Wu H, Sun Q, Liu J (2023) Robot time optimal trajectory planning based on improved simplified particle swarm optimization algorithm. IEEE Access 11: 44496-44508. https://doi.org/10.1109/ACCESS.2023.3277035

He K, Fei Y, Teng X, Chu X, Ma Z (2022) Optimal path planning for underwater robots based on improved ant colony algorithm. 2022 IEEE International Conference on Unmanned Systems (ICUS), Beijing, China, 1118-1123. https://doi.org/10.1109/ICUS56496.2022.10009089

Imanuvilov O, Hongyu L, Yamamoto M (2023) Unique continuation for a mean field game system. Applied Mathematics Letters 145: 108757. https://doi.org/10.1016/j.aml.2023.108757

Jiang L, Huang H, Ding Z (2019) Path planning for intelligent robots based on deep Q-learning with experience replay and heuristic knowledge. IEEE/CAA Journal of Automatica Sinica 7(4): 1179-1189. https://doi.org/10.1109/JAS.2019.1911749

Khatib O (1986) Real-time obstacle avoidance for manipulators and mobile robots. The International Journal of Robotics Research 5(1): 90-98. https://doi.org/10.1177/027836498600500106

Li Y, Zhao J, Chen Z, Xiong G, Liu S (2023) A robot path planning method based on improved genetic algorithm and improved dynamic window approach. Sustainability 15(5): 4656. https://doi.org/10.3390/su15054656

Liu H, Lo CW (2025) Determining state space anomalies in mean field games. Nonlinearity 38(2): 025010. https://doi.org/10.1088/1361-6544/ada67d

Liu H, Mou C, Zhang S (2023) Inverse problems for mean field games. Inverse Problems 39(8): 085003. https://doi.org/10.1088/1361-6420/acdd90

Liu H, Zhang S (2025) Simultaneously recovering running cost and Hamiltonian in mean field games system. arXiv preprint arXiv: 2303.13096. https://doi.org/10.48550/arXiv.2303.13096

Liu L, Shan Q, Xu Q (2024) USVs path planning for maritime search and rescue based on POS-DQN: Probability of success-deep Q-Network. Journal of Marine Science and Engineering 12(7): 1158. https://doi.org/10.3390/jmse12071158

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing Atari with deep reinforcement learning. arXiv preprint arXiv: 1312.5602. https://doi.org/10.48550/

arXiv.1312.5602

Minsky ML (1954) Theory of neural-analog reinforcement systems and its application to the brain-model problem. PhD thesis, Princeton University, Princeton, USA

Mac TT, Copot C, Tran DT, De Keyser R (2016) Heuristic approaches in robot path planning: A survey. Robotics and Autonomous Systems 86: 13-28. https://doi.org/10.1016/j.robot.2016.08.001

Sang H, You Y, Sun X, Zhou Y, Liu F (2021) The hybrid path planning algorithm based on improved A* and artificial potential field for unmanned surface vehicle formations. Ocean Engineering 223: 108709. https://doi.org/10.1016/j.oceaneng.2021.108709

Tamang J, Nkapkop JDD, Ijaz MF, Prasad PK, Tsafack N, Saha A, Kengne J, Son Y (2021) Dynamical properties of ion-acoustic waves in space plasma and its application to image encryption. IEEE Access 9: 18762-18782. https://doi. org/10.1109/ACCESS.2021.3053867

Watkins CJ, Dayan P (1992) Q-learning. Machine Learning 8: 279-292. https://doi.org/10.1007/BF00992698

Yang J, Huo J, Xi M, He J, Li Z, Song HH (2022) A time-saving path planning scheme for autonomous underwater vehicles with complex underwater conditions. IEEE Internet of Things Journal 10(2): 1001-1013. https://doi.org/10.1109/JIOT.2022.3191814

Yu K, Liang XF, Li MZ, Chen Z, Yao YL, Li X, Zhao ZX, Teng Y (2021) USV path planning method with velocity variation and global optimisation based on AIS service platform. Ocean Engineering 236: 109560. https://doi.org/10.1016/j.oceaneng.2021.109560

Zhang H, Huang Y, Qin H, Geng Z (2023a) USV search mission planning methodology for lost target rescue on sea. Electronics 12(22): 4584. https://doi.org/10.3390/electronics12224584

Zhai H, Wang W, Zhang W, Li Q (2021) Path planning algorithms for USVs via deep reinforcement learning. 2021 China Automation Congress (CAC), Beijing, China, 4281-4286. https://doi.org/10.1109/CAC53057.2021.9727909

Zhang M, Cai W, Pang L (2023b) Predator-prey reward based Q-learning coverage path planning for mobile robot. IEEE Access 11: 29673-29683. https://doi.org/10.1109/ACCESS.2023.3262236

Zeng Z, Sammut K, Lian L, He F, Lammas A, Tang Y (2016) A comparison of optimization techniques for AUV path planning in environments with ocean currents. Robotics and Autonomous Systems 82: 61-72. https://doi.org/10.1016/j.robot.2016.04.002