ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

A Route Planning Method for Intelligent Ships Based on the AS-DDQN Algorithm

Haoran Zuo [1, a], Jianhui Cui [1, b] *,Jiyong Li [2, c], Liu Fei [3, d], and Zhongxian Zhu [1, e]

[1] Maritime College, Tianjin University of technology, Tianjin, 300384, China;

[2] Tianjin Navigation Instrument Research Institute, Tianjin, 300131, China;

[3] China Merchants Viking Cruises Limited, Shenzhen, 518068, China.

[a] 18835792225@163.com, [b] jianhuicui@email.tjut.edu.cn, [c] lijiyong@tjnavi.com

[d] andyliufei1860@hotmail.com, [e] zzx19861018@163.com

Abstract: Addressing the issues in traditional DQN algorithms for intelligent vessel path planning—such as routes overly close to obstacles, frequent sharp turns, slow convergence, overestimation of Q-values, and network instability— this paper proposes an intelligent ship path planning method (AS-DDQN) based on the DDQN algorithm. It combines Angle Searching with DDQN by pre-setting search angles to define search regions, thereby optimizing travel paths, reducing redundant exploration, and accelerating convergence. Additionally, supplementary reward functions are introduced to the traditional reward structure. These include rudder angle deviation penalties, safe distance penalties for the vessel's track, time penalties, and distance-based propulsion rewards. This ensures the vessel maintains a safe distance from obstacles while minimizing steering maneuvers. Finally, simulations were conducted in two distinct maritime environments near Zhoushan with varying complexity levels. Experimental results demonstrate that compared to the DDQN algorithm, the AS-DDQN algorithm significantly accelerates algorithm convergence. Moreover, across different maritime environments, the final planned routes achieved notable effectiveness in ensuring navigation safety and economic efficiency, better meeting the practical navigation requirements of vessels.

Keywords: deep Reinforcement Learning; search Angle; AS-DDQN Algorithm; reward Function; route Planning

1. Introduction

As the process of global economic integration continues to deepen, maritime transportation has become the backbone of international trade. Consequently, ship route planning has gradually emerged as a hot topic in the industry. Route planning essentially involves finding a collision-free, feasible, and efficient trajectory[1] from the starting point to the destination under specified constraints and optimization objectives. Currently, common route planning algorithms for intelligent vessels can be categorized into traditional search-based algorithms, sampling and randomized algorithms, bio-inspired algorithms, and artificial intelligence algorithms.

For traditional search algorithms (such as A*[2], Dijkstra's algorithm, etc.), Liang et al.[3] proposed a jump-node search strategy as an enhancement to A*. This acceleration technique for grid maps employs “symmetry pruning” and “leap-based” expansion to significantly reduce the number of searched nodes and improve pathfinding efficiency. However, since the starting point lacks a parent direction, it necessitates attempting jumps in all eight directions, resulting in residual redundant nodes. Sunita et al.[4]proposed a dynamic shortest path algorithm based on Dijkstra's algorithm. By introducing a rollback-capable priority queue, it enables Dijkstra's algorithm to “return to the past” and undo old relaxation operations while incrementally re-relaxing affected sections when edge weights dynamically change. This achieves efficient dynamic shortest path maintenance and effectively reduces its runtime. Second, for sampling and randomization-based algorithms (e.g., RRT[5]), Huang Yifan et al.[6] proposed the RRT-Connect algorithm. Its core idea is to significantly accelerate path search speed from start to goal by incorporating bidirectional growth and “connection” operations on top of RRT, thereby hastening algorithm convergence. However, the

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40
final path trajectories still exhibit numerous abrupt turns. Among intelligent bionic algorithms, these include Genetic Algorithms (GA)[7] and Particle Swarm Optimization (PSO). Huang Xinlin et al.[8] proposed an improved genetic algorithm based on Gaussian matrix mutation. Within the classical genetic algorithm (GA) framework, the "Gaussian matrix mutation" operator was introduced to enhance population diversity and local search capabilities. Compared to traditional genetic algorithms, while maintaining a solution accuracy difference within 1%, its average convergence speed was significantly improved.

Artificial intelligence algorithms, including deep learning and reinforcement learning, play roles such as perception, decision-making, planning, and interaction across various scenarios. They are increasingly becoming a research hotspot in the field of intelligent ship path planning. Addressing the overestimation issues arising from bootstrapping and maximizing true value in traditional DQN algorithms, Hasselt et al.[9] proposed a double Q-learning (DQL) algorithm. Building upon the concept of target networks, this approach decomposes the calculation of the temporal discount (TD) objective into two steps: selection and evaluation. This effectively mitigates the overestimation caused by bootstrapping and maximization. Xie Tian et al.[10] proposed an improved path planning algorithm based on a Dueling Deep Double Q-Network (R-D3QN). This method integrates Dueling Networks with Double Q-Learning, separating action selection and value evaluation through a dual-network design to further reduce Q-value overestimation. However, the reward function design remains overly monotonous, resulting in path planning diagrams that still exhibit issues such as excessive proximity to obstacles and numerous abrupt turns. Li Lingyu et al.[11] established an intelligent vessel navigation strategy model based on deep Q-learning and artificial potential fields for vessel navigation. By designing the action and state spaces for deep reinforcement learning, their modifications effectively shortened the navigation path to the target point compared to other algorithms. Liu et al.[12] defined a state and action space based on the Deep Double Q-Network (DDQN) model framework, embedding a neural network model. Their optimized algorithm significantly enhanced the agent's collision avoidance capabilities.

Building upon the research foundation established in previous literature, this paper employs a Double Deep Q-Network (DDQN) architecture based on deep reinforcement learning algorithms. To address the monotonous reward function inherent in traditional DQN algorithms, an additional reward function is designed to reduce redundant turning points and sharp corners in the planned paths of intelligent vessels. Finally, within the ε–greedy framework, the Angle Search Strategy replaces the entirely random "exploration" action with an angle-optimized action. By defining a search angle to determine the search domain, this approach reduces the computational load and learning time of the initial DDQN algorithm, accelerates convergence, and smooths the path while avoiding local optima. This significantly enhances the intelligent vessel's exploration capabilities.

2. Algorithm Overview

2.1 Q-learning Algorithm

Q-learning is a model-free reinforcement learning algorithm based on Temporal Difference (TD) theory. Proposed by Watkins[13] in 1989, its core principle involves iteratively updating the value of $Q(s,a)$ to gradually approximate the optimal action–value function $Q^*(s,a)$.Its update formula is based on the Bellman optimality equation, directly optimizing the optimal policy without relying on the action selection of the current policy. The update formula is:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (1)$$

In the equation: s_t represents the current state; a_t denotes the current action; r_t signifies the reward obtained from environmental feedback after the agent executes the action in the current state; s_{t+1} indicates the state at the next time step; α is the learning rate ($0<\alpha<1$), used to control the update step size; γ is the discount factor ($0\leqslant\gamma<1$), used to balance the importance of current

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

rewards and future rewards; $\max_a Q(s_{t+1}, a)$ denotes the maximum Q-value for all possible actions a in the next state s_{t+1}, representing the value of the optimal action.

The Q-learning algorithm is also a type of TD algorithm that stores actions and rewards in a table, known as the Q-table. Consequently, it becomes evident that as the number of actions increases and the environment grows more complex, the Q-table will expand indefinitely, making it impossible to construct. Therefore, it cannot handle coherent and relatively intricate problems.

2.2 DQN Algorithm

In addressing many problems, states are not discrete, allowing the Q-function to replace the Q-table. The Q-function can still be denoted as $Q(s, a)$. Thus, for any state, its corresponding Q-value can be computed directly through the function. Building upon this concept, Mnih et al.[14] first proposed the Deep Q-Network (DQN) algorithm in 2013. This approach replaces the traditional Q-value table with a neural network. If constructed sufficiently large, the neural network can theoretically approximate any function, effectively serving as the Q-function itself. The weights w represent the parameters of this neural network, allowing the Q-function to be expressed as $Q(s, a; w)$. This enables reinforcement learning tasks in high-dimensional state spaces. Furthermore, the DQN algorithm incorporates two core components: the experience replay mechanism and the target network. Experience replay involves dividing the agent's trajectory into quadruples (s_t, a_t, r_t, s_{t+1}), which are stored in an experience replay buffer. The buffer size must be manually specified. Only when the buffer contains sufficient quadruples does experience replay begin to update the DQN. This mechanism integrates past and current experiences, breaking sequence dependencies while enhancing training stability. During training, the DQN algorithm replicates the network into two components: an evaluation network and a target network. Simply put, the evaluation network makes decisions while the target network calculates target values. Only the evaluation network's parameters are updated during each training iteration. After a specified time step, these updated parameters are then copied to the target network, further stabilizing the training process. The principle diagram of DQN is shown in Fig 1. When training the DQN algorithm, the weight w is a critical parameter. The update of weight w can be divided into the following steps:

(1) Perform forward propagation on the DQN to obtain the Q-value:

$$\hat{q}_t = Q(s_t, a_t; w_{now}) \tag{2}$$

$$\hat{q}_{t+1} = \max_{a \in \mathcal{A}} Q(s_{t+1}, a; w_{now}). \tag{3}$$

(2) Calculate the TD objective and TD error:

$$\hat{y}_t = r_t + \gamma \cdot \hat{q}_{t+1} \tag{4}$$

$$\delta_t = q_t - \hat{y}_t. \tag{5}$$

(3) Perform backpropagation on the DQN to compute the gradient of L with respect to w:

$$\nabla_w L(w) = (\hat{q}_t - \hat{y}_t) \cdot \nabla_w Q(s_t, a_t; w). \tag{6}$$

(4) Update the parameters of DQN using the gradient descent formula:

$$w \leftarrow w_{now} - \alpha \cdot \delta_t \cdot \nabla_w Q(s_t, a_t; w). \tag{7}$$

The loss function is defined as: $L(w) = \frac{1}{2}[Q(s_t, a_t; w) - \hat{y}_t]^2$, where $\delta_t = (\hat{q}_t - \hat{y}_t)$ represents the TD error. $\nabla_w Q(s, a; w) \triangleq \frac{\partial Q(s,a;w)}{\partial w}$ denotes the gradient of the function value $Q(s, a; w)$ with respect to parameter w, $\hat{y}_t$ represents the target value computed by the target network, and w_{now} denotes the weight at the time of update.

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

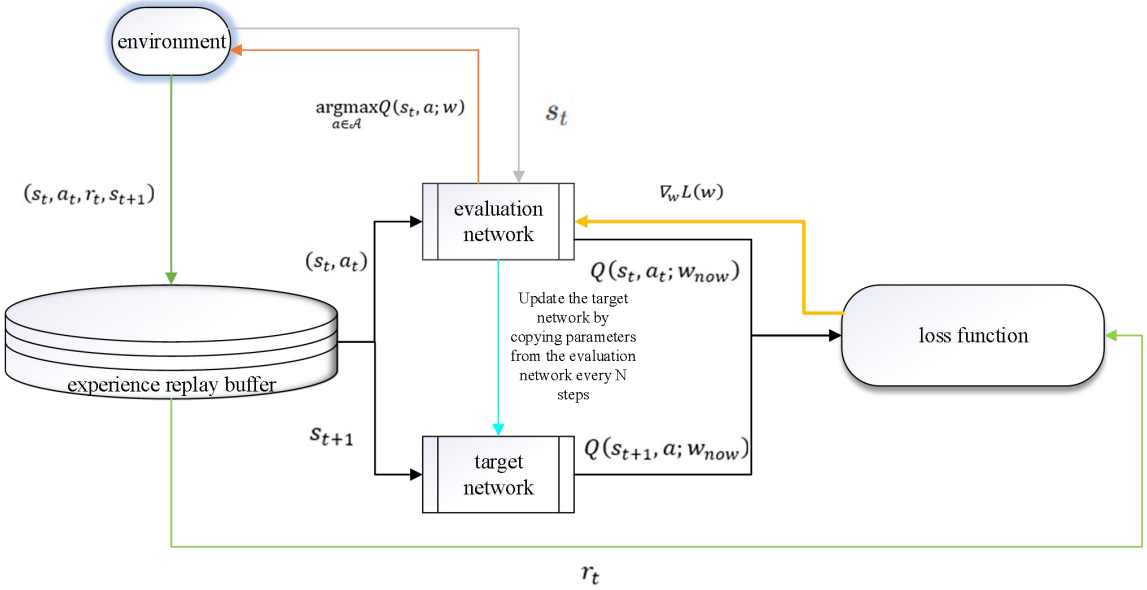


Fig. 1 DQN Algorithm Principle Structure Diagram

2.3 DDQN Algorithm

The reason for DQN's overestimation lies not in inherent flaws within the DQN model itself, but rather in shortcomings of the Q-learning algorithm. These can be summarized in two aspects: First, bootstrapping causes bias propagation, specifically the spread of bias from a single (s,a) pair to more pairs. Second, maximizing $\max_{a\in\mathcal{A}} Q(s_{t+1},a;w)$ leads to overestimation of the TD objective. The previously mentioned dual Q-learning algorithm[15] builds upon the target network with further refinements.

The core idea is to decompose the maximization in TD objectives into two steps: selection and evaluation. 1) For selection, the previous DQN network is still used: $a^* = \underset{a\in\mathcal{A}}{\text{argmax}} Q(s_{t+1},a;w)$; 2) During evaluation, the target network computes $\tilde{y}_t = r_t + Q(s_{t+1},a^*;w^-)$. Thus, dual Q-learning separates action selection from policy evaluation into two distinct steps, effectively mitigating overestimation caused by bootstrapping and maximization. After gradient descent updates the DQN parameters, the target network parameters undergo weighted average updates according to the following formula:

$$w^- \leftarrow \tau \cdot w + (1-\tau)\cdot w^-. \tag{8}$$

The initial w and w^- in the equation represent the current parameters of the DQN and target network, respectively, while $\tau \in (0,1)$ is a hyperparameter requiring manual adjustment.

3. As-Ddqn-Based Intelligent Ship Path Planning

To address the common issues in traditional DQN algorithms for intelligent ship path planning—such as routes overly close to obstacles, frequent sharp turns, slow convergence speed, overestimation of Q-values, and network instability— this paper introduces new penalty functions—rudder angle deviation, safe track distance, time, and voyage propulsion reward—on top of the original reward mechanism. DDQN is employed for planning to mitigate overestimation caused by maximization and enhance navigation safety. Subsequently, the AS-DDQN algorithm is designed. By defining the search domain, it reduces the number of grid points traversed by the intelligent vessel during path planning, accelerating algorithm convergence and enabling the vessel to swiftly find a smooth, safe path. Finally, simulations across navigation scenarios of varying complexity validate the feasibility of the enhanced reward-penalty function design and the AS-DDQN algorithm.

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

3.1 Angle Search Strategy

In intelligent vessel path planning, the angular search strategy[16] involves pre-setting a search angle φ along the line connecting the starting point and the target point. Simultaneously, the angle α is defined as the angle formed between the line connecting the vessel's current position to the center of each of the adjacent eight grid cells and the line connecting the vessel's current position to the center of the target grid cell. Grids where φ exceeds α and are neither boundaries nor obstacles become the next feasible grid for progression, thereby defining the search area. An optimal route is then sought within this area, as illustrated in Fig 2. Typically, the target point's position relative to the vessel's position exhibits four distinct relationships, as shown in Fig 3, corresponding to symbols 1, 2, 3, and 4.

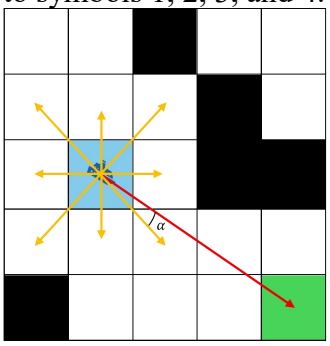


Fig. 2 Schematic Diagram of the Alpha Angle

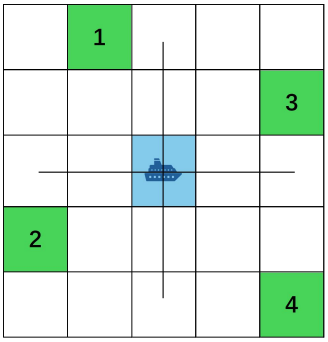


Fig. 3 Common Positional Relationship Diagram

When selecting the search angle, an excessively large angle may result in a large number of iteratively selectable grid cells, significantly increasing computational load. Conversely, an excessively small angle may lead to situations where only one grid cell around the vessel can be selected per iteration, or where no iteratively selectable grid cells can be identified at all, ultimately causing local deadlock. This paper comprehensively considers the above factors when selecting the search angle, striving to limit the number of grids covered by the search angle to no more than three. Therefore, the search angle φ is defined as 45°, with this search region designated as the navigable area. Regions outside this navigable area are defined as non-navigable zones.

In actual navigation scenarios, a vessel may occasionally encounter situations where no viable grid cells exist within the designated search area, as illustrated in Fig 4. Within the search angle φ range, both grid cells are obstructed, preventing vessel passage. Therefore, this paper proposes the following solution: Among the grid nodes adjacent to the vessel's navigation node, select the grid corresponding to the minimum heading angle $\beta(\beta > \varphi)$ formed by the direction vector from these grid nodes to the vessel's position and the line connecting the vessel's position to the target point, as illustrated in Fig 5.

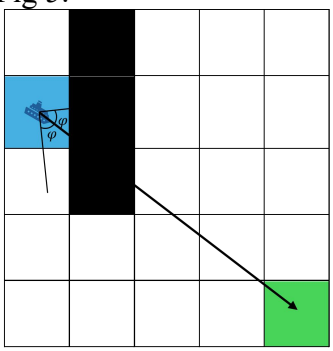


Fig. 4 No forward-moving grid cells within the search area

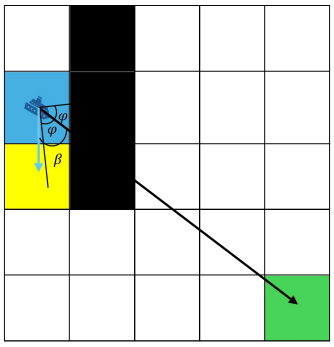


Fig. 5 Resolution

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

3.2 State And Action Space Design

The state space denotes the set of all possible states within the environment. Let the current position of the intelligent vessel be (x_i, y_i) and the target position be (x_a, y_a). This paper represents the state space using the coordinate difference between the two positions: $(x_i - x_a, y_i - y_a)$.

The action space defines the complete set of operations that an intelligent vessel can perform within a given environment. When an intelligent vessel is in state s_i, its available paths are the eight adjacent grid points. Therefore, this paper discretizes actions by dividing the vessel's movement directions into eight predefined courses: forward, backward, left, right, left-forward, left-backward, right-forward, and right-backward relative to the current grid node. These form the action set A, as expressed in the following equation.

$$A = \{\text{forward, backward, left, right, left forward, left backward, right forward, right backward}\}$$

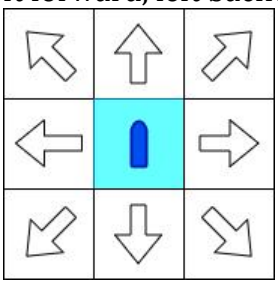


Fig. 6 Schematic diagram of ship action space

3.3 Design And Training Process Of The As-Ddqn Algorithm

Action decision-making in path planning is crucial for intelligent vessels, serving as a key factor influencing their exploration efficiency. The DDQN algorithm often employs a dynamic $\varepsilon-$ greedy exploration strategy for action selection, setting an ε value ($0.01 < \varepsilon < 1$) to indicate that the intelligent vessel has a certain probability ε of choosing a random action. This approach exhibits high randomness during early training phases, gradually diminishing with increasing training iterations until reaching the minimum ε value. The remaining $1-\varepsilon$ probability selects the action corresponding to the maximum Q-value.

This exploration strategy maintains a degree of randomness throughout the training process of intelligent vessels, effectively preventing them from getting stuck in local optima. However, due to this randomness, when ε is large in the early training stages, intelligent vessels may sometimes engage in excessive ineffective exploration, leading to prolonged algorithm convergence times.

To enhance exploration efficiency and reduce algorithm convergence time, this paper introduces the Angle Search (AS) strategy into the original $\varepsilon-$greedy exploration policy, resulting in the AS-DDQN algorithm.

During the early phase of the AS-DDQN algorithm, the intelligent vessel adopts a random policy within the specified angular search region to accumulate samples and learn experience. In the later training phase, after the neural network training is complete, the vessel adopts the policy with the maximum Q-value with probability $1 - min\varepsilon$, selecting the action corresponding to the maximum Q-value as follows:

$$a_t^{(k)} = \begin{cases} \underset{a_t \in A}{max}\left[Q\left(s_t^{(k)}, a_t; w\right)\right] & ,\varepsilon \\ \text{random}_{a_t \in A} & ,1-\varepsilon \end{cases} \tag{9}$$

As training progresses, ε gradually decays to a minimum value, as shown in the following equation:

$$\varepsilon = \begin{cases} \varepsilon - \varepsilon_{\text{down}} & ,\varepsilon > \varepsilon_{\min} \\ \varepsilon_{\min} & ,\varepsilon = \varepsilon_{\min} \end{cases} \tag{10}$$

The training structure of AS-DDQN is shown in Fig 7. After system initialization, the model enters a global training loop. Each training cycle encompasses a complete task execution and learning process: First, the neural network generates an action policy based on the current observed

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40
state. AS-DDQN then determines whether navigation is feasible before executing environmental interactions. Subsequently, it collects the new state and immediate reward signal to feed into AS-DDQN. Knowledge accumulation is achieved through a triple learning mechanism: learning parameter tuning, experience replay storage, and neural network weight updates. During training, all states, actions, and corresponding rewards from AS-DDQN are stored in the experience replay pool. Once the experience count reaches a preset threshold, the AS-DDQN algorithm samples batches from the replay pool. It calculates the loss value using the loss function and updates the network parameters via gradient descent with the learning rate, continuing until the entire training task is completed.

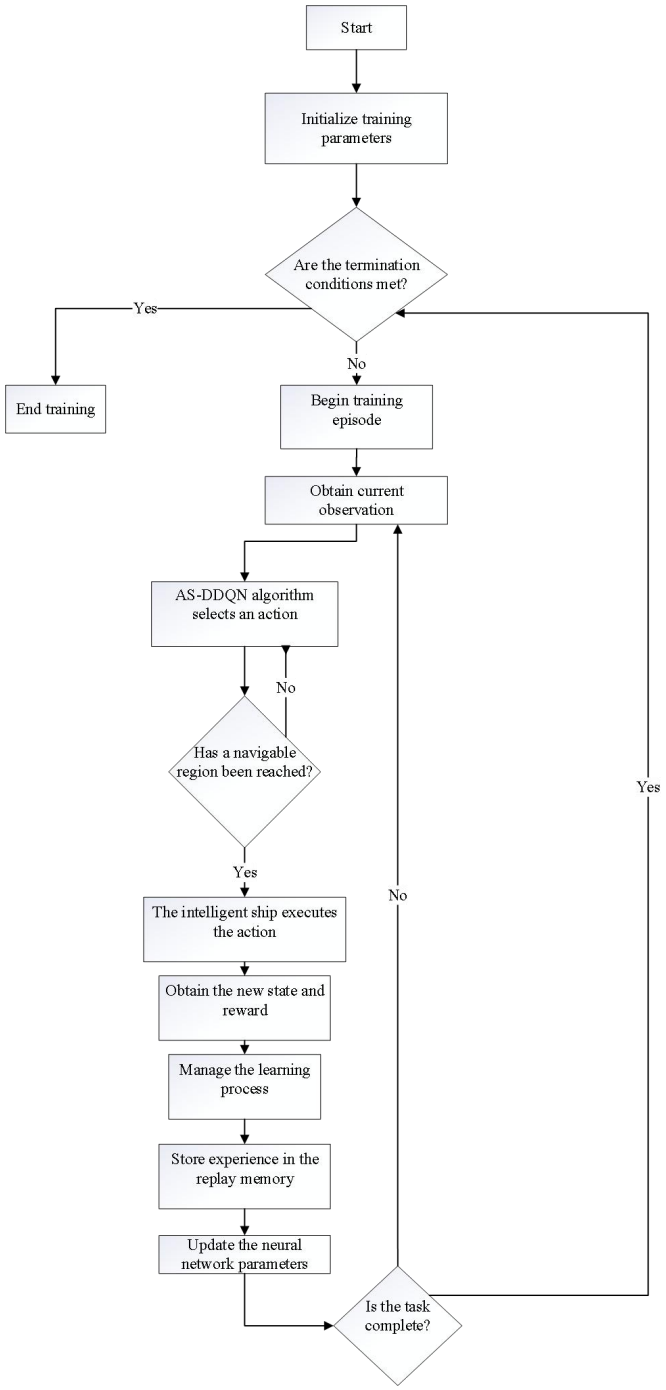


Fig. 7 Training flowchart

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

3.4 Reward Function Design

In intelligent vessel path planning, the design of reward functions must fully consider factors such as vessel navigation economy and heading deviation angle. However, the traditional reward function of the DDQN algorithm only includes collision penalties and arrival rewards at target points, as shown in the following equation. Therefore, this paper proposes four additional reward functions: rudder angle reward, safe track distance reward, navigation time reward, and voyage propulsion reward.

$$\begin{cases} R_{ar} = 100 \\ R_{co} = -30 \end{cases} \tag{11}$$

In the formula: R_{ar} denotes the reward for reaching the target point; R_{co} denotes the penalty incurred from collisions.

（1）Progress Reward

This function is designed to enable smart vessels to reach their destination points more quickly, with the reward value increasing as the vessel approaches the target point.

$$R_{\text{progress}} = \sqrt{(x_i - x_a)^2 + (y_i - y_a)^2} - \sqrt{(x_{i+1} - x_a)^2 + (y_{i+1} - y_a)^2} \tag{12}$$

In the formula: (x_{i+1}, y_{i+1}) represents the position of the intelligent vessel after executing an action.

（2）Obstacle Proximity Reward

To ensure vessel safety during navigation, vessels should maintain maximum distance from obstacles to minimize collision risks, with penalties increasing as proximity to obstacles grows. In actual navigation, the safe distance between vessels and obstacles is typically at least twice the vessel's length. For simplification, this paper adopts k as a fixed constant.

$$R_{\text{proximity}} = -\frac{k}{d+\epsilon}, k = 1.0 \tag{13}$$

In the formula: d represents the Euclidean distance between the vessel and the nearest obstacle; the introduction of the infinitesimal constant ϵ is to prevent division-by-zero errors.

（3）Time Penalty

To reduce ineffective exploration by intelligent vessels, a small fixed penalty is applied for each step taken.

$$R_{\text{ti}} = -0.01 \tag{14}$$

（4）Rudder Angle Reward

To minimize yaw deviations during intelligent vessel navigation, encourage vessels to maintain straight-line courses whenever possible outside of a few extreme environmental conditions. This approach enhances path smoothness and reduces overall voyage distance.

$$R_{smooth} = \begin{cases} +0.1 & \text{if} \Delta\theta_1 \le 45^\circ \wedge \Delta\theta_2 \le 45^\circ \\ -0.12 & \text{if} \Delta\theta_1 \ge 90^\circ \wedge \Delta\theta_2 \ge 90^\circ \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

In the formula: $\theta_{t-2}, \theta_{t-1}, \theta_t$ represent the heading angles at three consecutive time steps; $\Delta\theta_1 = \min(|\theta_{t-1} - \theta_{t-2}|, 360^\circ - |\theta_{t-1} - \theta_{t-2}|)$, $\Delta\theta_2 = \min(|\theta_t - \theta_{t-1}|, 360^\circ - |\theta_t - \theta_{t-1}|)$ represents the corresponding angular difference.

Therefore, the overall reward function for the intelligent vessel is:

$$R = R_{ar} + R_{co} + R_{\text{progress}} + R_{\text{proximity}} + R_{\text{ti}} + R_{smooth} \tag{16}$$

In intelligent vessel path planning, collisions represent a severe adverse event, hence collision penalties are set at a high value. Second, this paper introduces the R_{progress} and $R_{\text{proximity}}$ reward functions to incentivize intelligent vessels to reach their destination points faster and more safely. Simultaneously, the economic efficiency of vessel navigation is also crucial. Therefore, the R_{ti} and R_{smooth} functions are designed to effectively reduce unnecessary exploration and yawing by intelligent vessels.

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

4. Simulation Experiment

To validate the feasibility of the additional reward function designed for the DDQN algorithm and the AS-DDQN algorithm in this paper, simulations were conducted for intelligent ship path planning. The primary metric was convergence rounds, with secondary metrics being the minimum obstacle distance and the number of turning points. The traditional DDQN algorithm with the additional reward function was compared with the AS-DDQN algorithm. Table 1 presents the hyperparameter settings for both DDQN and AS-DDQN algorithms. Additionally, this paper selected a simple and a complex sea area near Zhoushan, modeling them into 80×80 grid maps as shown in Figures 8 and 9. Green indicates the starting point, red the endpoint. After grid conversion, each grid corresponds to a real-world area measuring 82.5793m in length and 80.4818m in width. Black areas represent obstacles such as islands and landmasses, indicating non-navigable zones. The algorithms were executed on a Windows 11 operating system with a 3.40GHz CPU and 16GB RAM.

Table 1. DDQN and AS-DDQN Hyperparameter Settings

Parameters	Value
Learning rate (α)	0.0005
Discount factor (γ)	0.91
Mini-batch size	128
Maximum training episodes	1500
Target network update frequency	100
Initial ε	1
Minimum ε	0.01

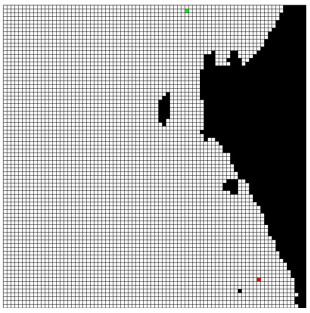


Fig. 8 Simple maritime gridded map

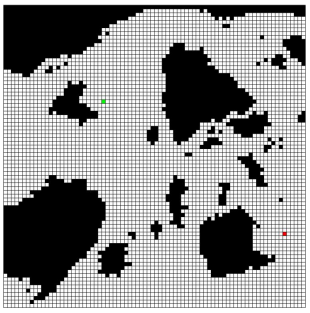
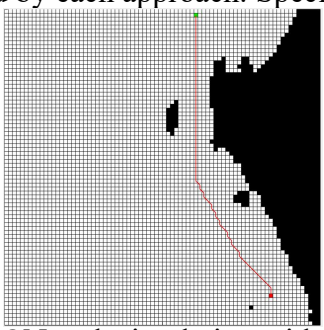


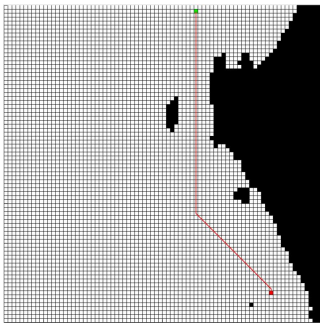
Fig. 9 Complex maritime gridded map

4.1 Comparison and Analysis of Simulation Results in Simple Sea Areas

The neural network parameters and reward values are identical across both methods to highlight the advantages of the improved AS-DDQN algorithm. The figure below shows path simulations of both methods in a simple maritime environment, with the red lines representing the optimal paths converged by each approach. Specific metrics are detailed in Table 2.



(a) DDQN path simulation with additional reward



(b) AS-DDQN path simulation with additional reward

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

Table 2. Comparison of Simulation Results in Simple Sea Conditions

Algorithm	Number of episodes until convergence	Minimum distance/m	Total number of turning points
DDQN algorithm with an additional reward	Approximately 750	247.738	18
AS-DDQN algorithm with an additional reward	Approximately 310	247.738	2

As shown in the final path map Fig 10 and Table 2, both algorithms incorporating additional reward functions maintained a safe distance from obstacles without producing sharp turns. However, the path planned by the AS-DDQN algorithm exhibited greater smoothness due to its predefined search domain, which reduced the number of grid points traversed by the intelligent vessel during path planning. Further analysis of Table 2 reveals that the number of path turning points under AS-DDQN planning decreased by 88.89% compared to DDQN.

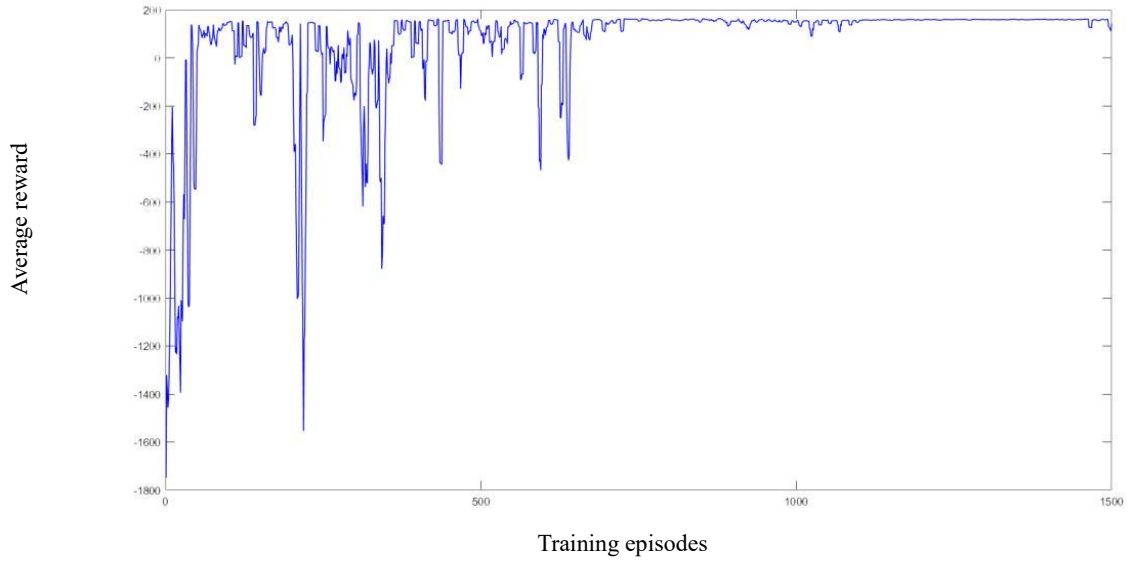


Fig. 11 Variation in average reward values of the DDQN algorithm in simple marine environments

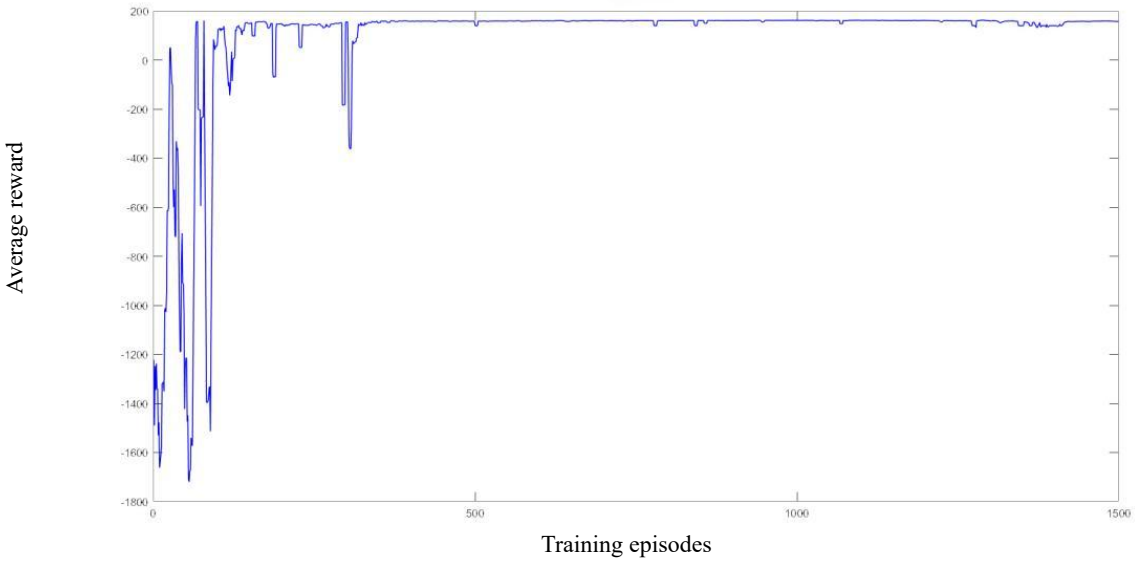
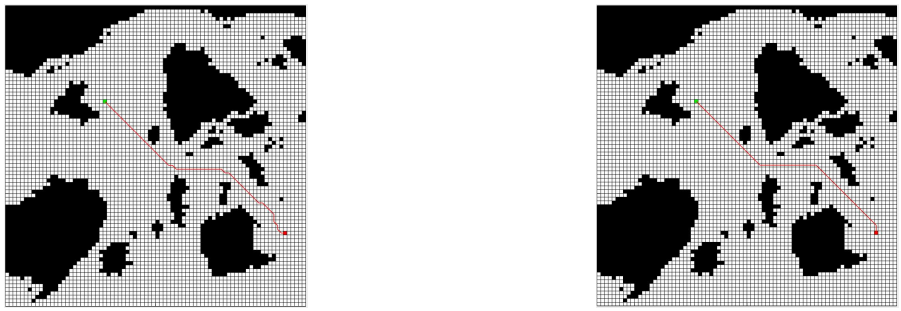


Fig. 12 Variation in average reward values of the AS-DDQN algorithm in simple marine environments

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

Analysis of Figures 11 and 12 reveals that in a simple maritime environment, the DDQN begins to converge around 750 epochs, while the AS-DDQN converges around 310 epochs. In contrast, when both algorithms incorporate the same additional reward function, the improved AS-DDQN algorithm achieves a 58.67% faster convergence rate in the identical environment.

4.2 Comparison and Analysis of Simulation Results in Complex Sea Areas



(a) DDQN path simulation with additional reward (b) AS-DDQN path simulation with additional reward

Fig. 13 Path Planning Simulation of Two Algorithms in a Complex Marine Environment

Table 3. Comparison of Simulation Results in Complex Sea Conditions

Algorithm	Number of episodes until convergence	Minimum distance/m	Total number of turning points
DDQN algorithm with an additional reward	Approximately 1000	115.311	13
AS-DDQN algorithm with an additional reward	Approximately 520	115.311	3

Based on the path planning performance of the two algorithms in simple sea environments, as shown in Fig 13, under complex sea conditions, both algorithms maintained a safe distance from obstacles when an additional reward function was introduced, and neither exhibited sharp turns. The paths planned by the AS-DDQN algorithm were smoother, with fewer turning points. Further analysis of the Table 3 reveals that the total number of turning points in the AS-DDQN algorithm decreased by 76.92% compared to the DDQN algorithm.

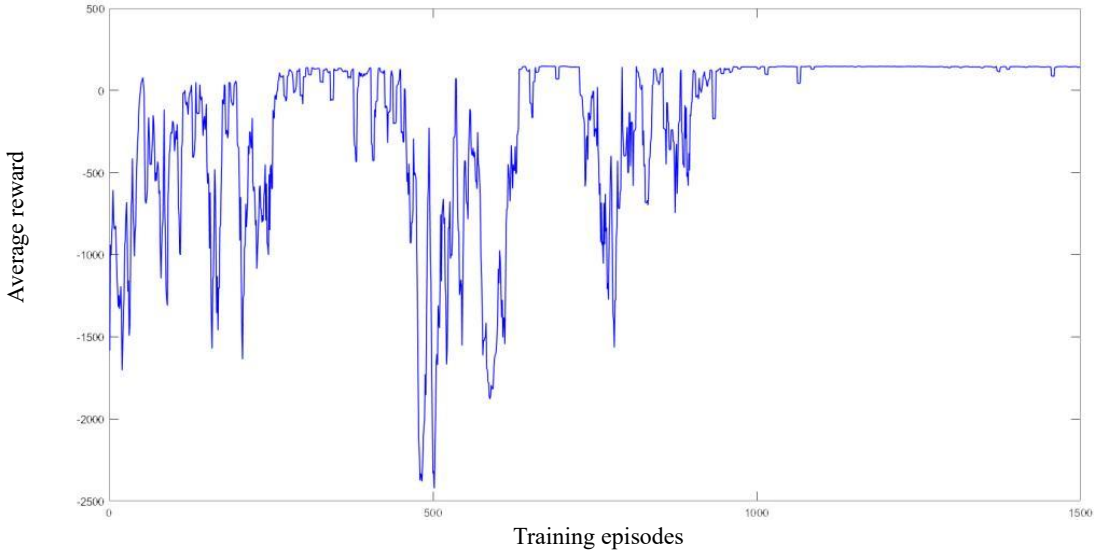


Fig. 14 Variation in average reward values of the DDQN algorithm in complex marine environments

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40

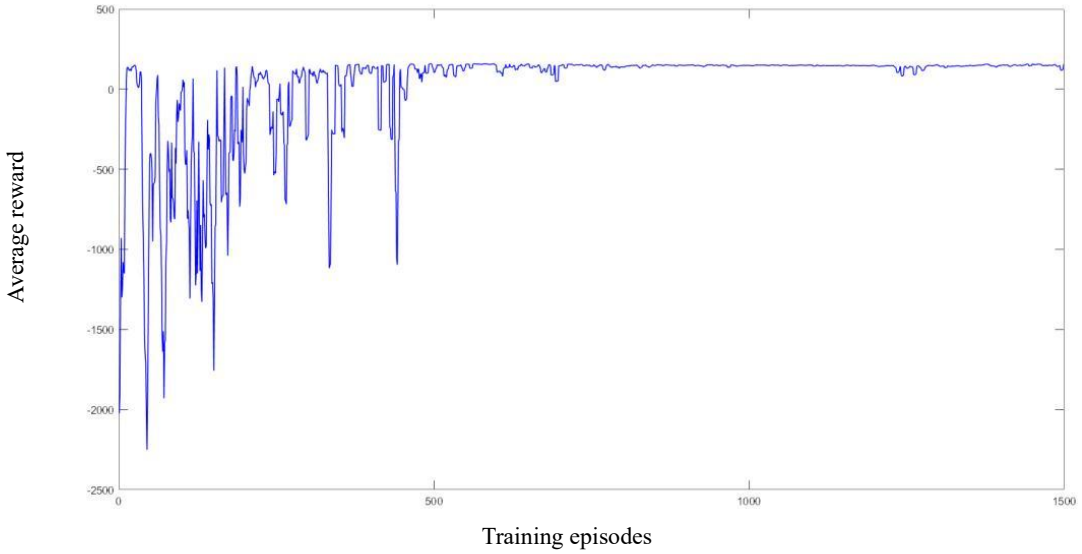


Fig. 15 Variation in average reward values of the AS-DDQN algorithm in complex marine environments

Comparing Figures 14 and 15 reveals that in complex maritime environments, as environmental complexity increases, the DDQN algorithm begins to converge around 1000 epochs, while the AS-DDQN algorithm converges around 520 epochs. This represents a 48% improvement in convergence speed compared to the DDQN algorithm. It is evident that the proposed AS-DDQN algorithm maintains robust stability even in more complex environments.

1. Conclusion

To address the overestimation issues arising from bootstrapping and overfitting in deep Q-learning, this paper adopts a dual deep Q-learning (DDQN) algorithm as its foundation. It specifically targets problems encountered in traditional DDQN algorithms for intelligent vessel path planning, such as routes overly close to obstacles and frequent abrupt steering maneuvers. This paper designs additional reward functions, such as a rudder angle reward function, on top of the traditional reward function. The final simulation path map demonstrates that the designed reward functions effectively prevent the aforementioned issues encountered by intelligent vessels during path planning, enabling the trained vessel to reach the target point more economically and safely. Additionally, addressing the slow convergence and lengthy training time of DDQN, this paper proposes the AS-DDQN algorithm. By incorporating an angle search strategy into the original exploration policy and pre-defining the search domain through a search angle parameter, it reduces ineffective exploration during training. Simulation results demonstrate that the proposed algorithm significantly accelerates convergence while maintaining exploration efficiency, offering valuable insights for intelligent vessel path planning.

References

[1] Zirui Deng, Jianhui Cui, Zhongxian Zhu, Yangwen Dan, Huawei Xie, Yingjun Hu, Adaptive trajectory tracking control of underactuated ships using parameter prediction based neural network, Ocean Engineering, Volume 329, 2025, 121188, ISSN 0029-8018.

[2] Zhai Yongjian, Cui Jianhui, Meng Fanbin, et al. Ship Path Planning Based on Sparse A* Algorithm (English) [J].Journal of Marine Science and Application,2025,24(01):238-248.

[3] XIN Yu, LIANG Huawei, DU Mingbo, MEI Tao, WANG Zhiling, JIANG Ruhai. An Improved A* Algorithm for Searching Infinite Neighbourhoods[J]. ROBOT, 2014, 36(5): 627-633.

ISSN: 3007-9640 DOI: https://doi.org/10.70114/acmsr.2025.5.1.P40
[4] Garg D. Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue[J]. Journal of King Saud University-Computer and Information Sciences, 2021, 33(3): 364-373.

[5] Liu Zhaoqi, Cui Jianhui, Meng Fanbin, et al. Research on Intelligent Ship Route Planning Algorithm Based on Adaptive Step Size Informed-RRT* Algorithm (English) [J].Journal of Marine Science and Application,2025,24(04):829-839.

[6] Huang Yifan, Hu Likun, Xue Wenchao. Mobile Robot Path Planning Based on an Improved RRT-Connect Algorithm [J]. Computer Engineering, 2021, 47(08): 22-28.

[7] Tu J, Yang S X. Genetic algorithm based path planning for a mobile robot[C]//2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422). IEEE, 2003, 1: 1221-1226.

[8] Eberhart R, Kennedy J. A new optimizer using particle swarm theory[C]//MHS'95. Proceedings of the sixth international symposium on micro machine and human science. Ieee, 1995: 39-43.

[9] Van Hasselt H, Guez A, Silver D. Deep reinforcement learning with double q-learning[C]//Proceedings of the AAAI conference on artificial intelligence. 2016, 30(1).

[10] Xie Tian, Zhou Yi, Qiu Yufeng. Path Planning Algorithm for Mobile Robots Based on an Improved Deep Q-Network [J]. Computer Systems Applications, 2025, 34(07): 37-47.

[11] Li Lingyu. Research on Intelligent Ship Navigation Strategies Based on Deep Reinforcement Learning [D]. Jimei University, 2022.

[12] YU Y,LIU Y F,WANG J C, et al. Obstacle avoidance method based on double DQN for agricultural robots[J]. Computers and Electronics in Agriculture, 2023, 204: 107546.

[13] Watkins C J C H. Learning from delayed rewards[J]. 1989.

[14] Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning[J]. arXiv preprint arXiv:1312.5602, 2013.

[15] VAN HASSELT H, GUEZ A, SILVER D. Deep reinforcement learning with double q-learning[C]//SCHUURMANS D, WELLMAN M P. 30th AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA, 2016. Palo Alto, California, USA: AAAI Press, 2016

[16] Wang, Yaru; Yao, Dexin; Liu, Zengli; et al. A Path Planning Method for Mobile Robots Based on Angle Search. Journal of System Simulation, 2024, 36(07): 1643-1654.