Hindawi
Journal of Advanced Transportation
Volume 2021, Article ID 7765130, 23 pages
https://doi.org/10.1155/2021/7765130

Research Article

An Optimized Path Planning Method for Coastal Ships Based on Improved DDPG and DP

Yiquan Du,[1] **Xiuguo Zhang**,[1] **Zhiying Cao**,[1] **Shaobo Wang**,[2] **Jiacheng Liang**,[1] **Fengge Zhang**,[1] **and Jiawei Tang**[1]

[1]*School of Information Science and Technology, Dalian Maritime University, Dalian 116026, China*
[2]*School of Navigation, Dalian Maritime University, Dalian 116026, China*

Correspondence should be addressed to Xiuguo Zhang; zhangxg@dlmu.edu.cn and Zhiying Cao; czysophy@dlmu.edu.cn

Received 10 May 2021; Revised 21 July 2021; Accepted 22 September 2021; Published 13 October 2021

Academic Editor: Umar Iqbal

Deep Reinforcement Learning (DRL) is widely used in path planning with its powerful neural network fitting ability and learning ability. However, existing DRL-based methods use discrete action space and do not consider the impact of historical state information, resulting in the algorithm not being able to learn the optimal strategy to plan the path, and the planned path has arcs or too many corners, which does not meet the actual sailing requirements of the ship. In this paper, an optimized path planning method for coastal ships based on improved Deep Deterministic Policy Gradient (DDPG) and Douglas–Peucker (DP) algorithm is proposed. Firstly, Long Short-Term Memory (LSTM) is used to improve the network structure of DDPG, which uses the historical state information to approximate the current environmental state information, so that the predicted action is more accurate. On the other hand, the traditional reward function of DDPG may lead to low learning efficiency and convergence speed of the model. Hence, this paper improves the reward principle of traditional DDPG through the mainline reward function and auxiliary reward function, which not only helps to plan a better path for ship but also improves the convergence speed of the model. Secondly, aiming at the problem that too many turning points exist in the above-planned path which may increase the navigation risk, an improved DP algorithm is proposed to further optimize the planned path to make the final path more safe and economical. Finally, simulation experiments are carried out to verify the proposed method from the aspects of plan planning effect and convergence trend. Results show that the proposed method can plan safe and economic navigation paths and has good stability and convergence.

1. Introduction

With the development of economic globalization, trade between countries is getting closer. Ships have been an important means of transportation in international trade and national transportation due to their characteristics of large transportation capacity, low energy consumption, small transportation cost, and green environment protection, which has a pivotal position in economic development [1, 2]. The development of economy has put forward requirements for maritime intelligent transportation, and ship automation is the most basic and urgent part of the solution. In the study of ship automation, path planning is one of the most important parts [3–5].

The coastal waters are different from narrow waters and open waters. The narrow waters have coastline restrictions and are relatively narrow, mainly referring to straits and rivers. Ships sailing in open waters are not restricted by coastlines, but dynamic obstacles such as icebergs will appear. There are proven obstacles in coastal waters, and there will be no dynamic obstacles and reefs like icebergs, but the shore-based information obstacles (temporary obstacle areas) such as ship-wreck area, restricted navigation area, and military exercise area will appear. The path planning of coastal ships is mainly to avoid proven obstacles and shore-based information obstacles and plan a safe and effective path for ships [6, 7]. The problem of avoiding ships belongs to the field of collision avoidance, and there are special rules for collision avoidance [8], so this paper mainly avoids proven obstacles and shore-based information obstacles. For the coastal ship path planning problem, many methods have been proposed by domestic and foreign scholars, mainly including traditional path planning methods, bionic intelligence algorithms, and machine learning-related algorithms. Traditional path planning methods require complete environmental information as a priori knowledge, but it is quite difficult to obtain the surrounding environmental information in the unknown marine environment. Bionic intelligence algorithms regard the path planning problem as an optimal path problem, using the path distance as a constraint and the collision hazard and range loss as objective functions [9]. However, this kind of method is particularly prone to local optimality, and the solution is computationally intensive and requires very high system performance.

In recent years, DRL has a good performance in the field of path planning. DRL obtains the state from the environment by interacting with the unknown environment and provides training samples for neural network. At the same time, it can use the strong fitting ability of neural network to complete the task better [10, 11]. At present, most DRL-based path planning methods use algorithms based on discrete action space, such as Q-learning and Deep Q-learning (DQN) [12–15]. In these algorithms, the optional actions of ships are limited, which may make it impossible to learn an optimal path planning strategy. Some scholars also use DDPG or A3C algorithm to establish path planning models in continuous action space [16, 17]. However, this kind of research partly depends on the grid environment, and grid partition strategy directly affects the planning path. Moreover, the reward function is defined as follows: while the ship performing an action, it will be given a fixed positive value if its next state is closer to the target point; otherwise, it will be given a fixed negative value. This will lead to the slow convergence of the algorithm, and the planned path does not conform to the ship navigation specification. In addition, the influence of the historical state on the current state is not considered, which also lowers the learning efficiency and convergence speed of the model.

This paper focuses on the ship autonomous path planning in continuous action space and continuous environment and adopts the DDPG algorithm for path planning. Meanwhile, in view of the poor processing capability of time-series data in the fully connected layer of DDPG, LSTM, which has better timing capability, is added to improve the approximation accuracy and the data utilization rate by approximating current environment state information with historical state information. Meanwhile, the mainline reward function and auxiliary reward function are established to optimize the strategy of DDPG and guide the ship to the target point by avoiding obstacles. Because of the size of the ship, the planned path is required to be as straight as possible, with few corners and avoiding passing through complex obstacles, which is the biggest difference from the path planning of unmanned vehicle and robot paths. Therefore, this paper proposes an improved Douglas–Peucker (DP) algorithm to optimize the planned path so as to make it more in line with the actual navigation requirements of ships.

In summary, the main contributions of this paper are as follows:

(1) The network structure of the DDPG is improved. In the traditional DDPG network structure, each layer of the network is a fully connected layer network, and only the current status data is obtained each time, which is ignoring the historical status data. For this reason, the method in this paper not only obtains the current state data of the ship but also obtains the final training data input of the historical observation state as a collection of current data and historical state data. To better learn the relationship between the historical state and the current state, this paper changes the first layer of the DDPG network to the LSTM that processes more time-series data, so that the LSTM allows the DDPG to predict better actions.

(2) A ship path planning method based on the above-improved DDPG and reward function optimization is proposed. Aiming at the problems of low data utilization and slow convergence speed of most unmanned ship path planning based on deep reinforcement learning, this paper optimizes the traditional reward function and designs the mainline and auxiliary reward function. The mainline reward function is used to guide the ship to reach the target point and complete the path planning task. Meanwhile, the auxiliary function gives reasonable punishment in the process of path planning, so as to avoid obstacles. The ship path planning is realized by improved DDPG and optimized reward function.

(3) A path optimization method based on the improved Douglas-Peucker (DP) algorithm is proposed. Because of the excessive turning points in the planned path, which increases the risk of the ship during navigation and is not economical, this paper proposes an improved DP algorithm to compare the path. Optimization and removal of redundant turning points make the planned path safer and more economical and more in line with the actual sailing requirements of the ship.

The rest of this paper is organized as follows. Section 2 reviews the related works. Section 3 presents the ship path planning model based on optimized DDPG. Section 4 recommends path optimization based on an improved DP algorithm. Section 5 introduces simulation experiments and result analysis. Conclusion and future work are presented in Section 6.

2. Related Research

Currently, the main fields involved in path planning contain robotics [18, 19], unmanned vehicles [20, 21], and ships. For ship path planning, researchers have proposed many methods, which are mainly classified into traditional algorithms, bionic intelligence algorithms, and machine learning-related algorithms.

The traditional algorithms mainly include the speed barrier method, A*, Artificial Potential Field (APF), and Rapidly Exploring Random Tree (RRT). The A* algorithm divides the area to be searched into square lattices, checks its adjacent squares from the starting point, then expands around to find the target, and finally finds the path with the least movement cost from the feasible lattices. For example, Gao et al. [22] proposed a global path planning method for surface unmanned ships based on an improved A* algorithm. The method finds the global optimal solution in a larger range by expanding the search region of the original A* to 24 and 48 neighborhoods in the established raster map. The disadvantage of this method is that it depends on the design of the raster map, and the size and number of raster spacing can directly affect the computational speed and accuracy of the algorithm. Chen et al. [23] proposed a hybrid method for global path planning of autonomous surface ships at sea, which considers the collision risk, the proximity of the path to obstacles, and the speed constraint that the ship may reach while generating a ship navigation path. Experimental results show that the method can find the optimal path considering collision risk and obstacle distance. Fan et al. [24] added a distance correction factor and positive hexagonal guidance to the repulsive potential field function to address the problems of unreachable targets and the existence of local minimum in the traditional AFP method, while the relative velocity method for motion target detection and obstacle avoidance was proposed for dynamic environments. Experiments show that the method can be used in both static and dynamic environments. However, when there is an equal repulsive and attractive force or when the repulsive force at the target point is large, the ship stalls and falls into a local optimum. Wang et al. [25] combined the ship domain model with the artificial potential field method to plan the path by judging the motion characteristics of the obstacle, taking into account the speed and heading of ship. Experiments show that the method can plan out a path, but it is not practical. Xiang et al. [26] proposed an improved two-way RRT algorithm for local path planning of unmanned ships. The algorithm solves the problem of more course twists and turns of the original RRT planning by adding corner constraints to the randomly set nodes of the original RRT [27] and setting a step length strategy. The method can plan out a path quickly, but it may not be the optimal one.

Bionic intelligent algorithms mainly include genetic algorithm, particle swarm algorithm, and ant colony algorithm. Wei et al. [28] proposed an improved genetic algorithm to establish the motion model of the underwater robot and design a reasonable crossover and variation adaptive probability model. Experiments proved that the algorithm effectively improved the convergence of the genetic algorithm. Jiang et al. [29] proposed an improved adaptive genetic simulated annealing algorithm, which improved the initial population generation strategy based on the traditional genetic algorithm and introduced an improved adaptive operator with a simulated annealing strategy in the mutation operation. The experimental results show that the algorithm may avoid falling into the local optimum and convergence fast. Ding et al. [30] modeled the navigation environment information extracted from the electronic chart as the experimental data needed and adopted a particle swarm algorithm for unmanned ship path planning with path distance as the constrain. The simulation experimental results show that the algorithm works. Lazarowska et al. [31] converted the ship path planning problem into an optimization problem and used the ant colony algorithm to solve the optimal path with collision hazard and range loss as the objective function. Experiments show that the method can plan out a ship navigation path, but it is computationally intensive and requires very high system performance. Xie et al. [32] proposed an improved Beetle Antenna Search (BAS) based algorithm for the underdriven surface ship path planning problem. Simulation experiments show that the planned path is quite good, but the algorithm is mainly used for offline decision-making.

Machine learning algorithms mainly refer to deep learning algorithms, reinforcement learning algorithms, and DRL algorithms. DRL generates training samples by interacting with the environment and guides itself to learn a strategy to complete the task based on these samples [33]. The purpose of deep reinforcement learning is to maximize the cumulative rewards that an agent receives during training and to learn the optimal strategy [34].

Many scholars have applied DRL to ship path planning. For example, Zhou et al. [12] proposed a DQN-based collaborative path planning for unmanned ships by defining thirteen actions and letting the ship choose the optimal action in the current state among them. The action space of this method is discrete and cannot be applied to a continuous action environment. Chen et al. [13] used the Q-learning algorithm for unmanned ship path planning and maneuvering. Using the trained model, ships could find the correct path and navigation strategy by themselves. A comparison with existing methods shows that the method is more effective in self-learning and continuous optimization and is closer to human operation. However, the method is prone to be latitude disasters under complex navigation conditions. Bhopale et al. [14] used an improved Q-learning algorithm for underwater vehicle obstacle avoidance, which forced the underwater robot to leave the unsafe area instead of attempting new or random actions when the underwater vehicle detected an obstacle, reducing the number of collisions. Experiments show that the method allows the underwater vehicle to avoid the detected obstacles. However, the method relies on Q-tables and has a poor fitting ability. Zhang et al. [35] proposed a behavior-based hazard avoidance decision model for unmanned ships based on the Sarsa algorithm and argued that it is feasible for the Sarsa algorithm to enhance the hazard avoidance of unmanned ships. However, the method only demonstrated the feasibility at the theoretical level and was not tested in an experimental environment and no experimental results were given. Zhang et al. [15] proposed an autonomous navigation decision model based on hierarchical deep reinforcement learning. The model consists of two main layers: a scene segmentation layer and an autonomous navigation decision layer. The method uses the environment model, ship motion space, reward function, and search strategy to learn the environment state in quantized subscenes and train the navigation strategy. Experimental results show that the improved DRL algorithm can effectively enhance navigation safety and collision avoidance ability, but the planned path still has some unnecessary waypoints, which increase the ship navigation overhead. Guo et al. [16] performed ship path planning by combining DDPG and artificial potential field, but it mainly focused on local collision avoidance and did not consider the influence of the historical state on the current state. Cao et al. [17] proposed an A3C-based target search algorithm for underwater robots, which enables the underwater robots through a designed asynchronous dominance evaluation network structure to learn from their experience and generate a search strategy and uses DRL and dual-stream Q-learning algorithms for underwater robot navigation to further optimize the search path. It is shown that the agent can avoid obstacles and reach the search target point, but this method depends on the grid environment and the action space is discrete and it cannot be applied to the continuous action environment.

The feature, advantages, and disadvantages of the above-mentioned ship path planning methods are compared, as shown in Table 1.

To solve these problems, this paper proposes a ship path planning model based on LSTM and DDPG. The model uses the historical state information to approximate the current environmental state information and constructs the mainline reward function and auxiliary reward function to optimize the action selection strategy of DDPG to guide the ship to avoid obstacles and reach the target point. An improved DP algorithm is designed to compress and optimize the planned path, which makes it safer and more economical.

3. Ship Path Planning Based on Improved DDPG

A ship path planning model is trained by ship's actual navigation environment information, so the actual navigation environment information should be processed firstly, and then the structure, state space, action space, and reward function of the model should be designed.

3.1. Coastal Ship Path Planning Framework. To improve the safety and practicability of coastal ship path planning, this paper proposes research methods related to path planning. By processing environmental information to complete marine environment modeling, the coastal ship path planning model is constructed according to safety and economic indicators. The coastal ship path planning framework is shown in Figure 1.

As can be seen from Figure 1, the framework includes two parts: marine environment modeling and path planning of coastal ships. First, the marine environmental information is processed and the experimental environment is constructed using quantitative environmental data. At the same time, the ship state space is designed to analyze actual navigation characteristics. Secondly, according to the design of the relevant algorithm based on the combination of the DDPG algorithm and the path planning method, the path planning model based on the improved DDPG is obtained by improving the network structure of the DDPG algorithm and optimizing the reward function. In response to the actual navigation requirements of the ship, this paper proposes an improved DP algorithm to optimize the planned path, so that the final path is safer and more economical. In this section, we mainly introduce the modeling of the marine environment and the path planning model based on the improved DDPG. Path optimization based on the improved DP algorithm will be introduced in Section 4.

3.2. Ship Actual Navigation Information Processing. When processing the actual navigation environment information of the ship, first the Mercator transformation method is used to convert the longitude and latitude of the obstacle and the ship's starting and target point into coordinates in the Cartesian coordinate system. Then the smallest area enclosing the starting point and target point is regarded as the target environment, and coordinates of obstacles are scaled to the target environment. Finally, each obstacle is replaced with an expanded circumscribed circle to prevent the planned path from being too close to the obstacle and increase the navigation risk of the ship.

Visibility Graph is adopted to obtain the circumcircle of an obstacle. Assume that the number of vertices of an obstacle is n. A polygon is obtained by drawing a line between every two adjacent vertices. Then the circumscribed circle of the polygon is created. Since obstacles are not regular polygons in the actual environment, if the center of the polygon is defined as the center of the circle, the obstacle might not be enclosed within it. In this paper, the center of gravity of the polygon is regarded as the center of the circle, and the longest one of $n \cdot (n-1)/2$ line segments generated by connecting any two vertices of the obstacle is regarded as the diameter of the circle.

3.3. Structure Design of the Path Planning Model Based on Improved DDPG. DDPG can randomly select actions in the continuous action space according to the learned strategy. It is a deterministic policy algorithm that can output only a certain action according to a given state. Compared with DQN, DDPG samples fewer data and executes more efficiently. Therefore, this paper uses the DDPG algorithm for ship path planning.

The path planning of the ship needs to obtain the current ship state information and then calculate the data of the hidden layer network by formula (1). Finally, the hidden layer network predicts an action according to the learned strategy. The ship performs this action and the reward function evaluates the state after performing the action and gives a reward or punishment. At the same time, rewards and penalties in turn affect the parameter update of the network. Through such a cycle of learning and updating, the model learns a good strategy for path planning.

Table 1: Comparison of the existing ship path planning methods.

Category	Algorithm	Feature	Advantage	Disadvantage
Traditional algorithm	A*	Complete path planning by searching for feasible nodes in the grid environment	Strong versatility	The planned path is relatively close to the obstacle
	APF	Guide the ship to complete path planning through attraction and repulsion	Applicable to static and dynamic environments	The planned path has a curve that does not apply to ships
	RRT	By randomly sampling from the feasible area of the environment until reaching the target point	High speed	The planned path is not guaranteed to be the shortest, and there are many turning points
Bionic intelligence algorithm	Genetic algorithm	The search for a suitable path is carried out mainly through step-by-step iterations of selection, crossover, and mutation operations by genetic algorithms	High scalability and robustness	Slow convergence speed, easy to fall into local optimum
	Ant colony algorithm	Guiding ant colonies towards optimal paths by mimicking the pheromones released by ants during foraging	Better overall, suitable for small environments	The convergence speed is slow, and it is easy to fall into the local optimum. When the environment is large, the algorithm execution time will increase accordingly
	Particle swarm optimization algorithm	Through the interplay of each particle's self-awareness and the social experience of all particles, iterative path planning is eventually achieved	Few parameters, fast convergence	Easily trapped in a local optimum
DRL algorithm	Q-learning	Through the establishment of the Q table, the state, action, and reward are saved in the Q table, and the path is planned by finding the action with the largest reward in a similar state from the Q table each time	Suitable for small environments and limited action options	Depends on Q table, poor fitting ability
	DQN	Convert the Q table in Q-learning into a neural network, and use the neural network to fit the relationship between state, action, and reward, so as to maximize the reward for each predicted action and get the optimal path	There is no need to look up the action, but to predict the action directly. After the model is trained, it can be directly migrated to a similar scene	The available actions are limited
	DDPG	The selection of predictive actions by neural networks employs a slow update mechanism to prevent overfitting	There is no restriction on the number of actions, and the trained model can be directly transferred to similar scenarios	Only focus on the current state during training, without considering the historical state information
	Method of this paper	Consider historical state information, process historical state information and current state information through LSTM, and design primary and secondary reward functions to speed up the convergence of the algorithm	There is no limit on the number of actions. Historical state information is taken into account, and data utilization is increased. The trained model can be directly migrated to similar scenarios	

3.3.1. Improvement of DDPG Algorithm

(1) Improvement of the DDPG Structure. DDPG is a DRL algorithm based on the AC framework, which includes a policy network, called an Actor network, and an evaluation network, called a Critic network. The Actor network is used to map states to a specific action, and the Critic network is used to estimate the action. The network of DDPG is structured as the main network and target network. The main network, including the main Actor and Critic network, is used to yield and evaluate actions and update network parameters. The target network includes the target Actor and

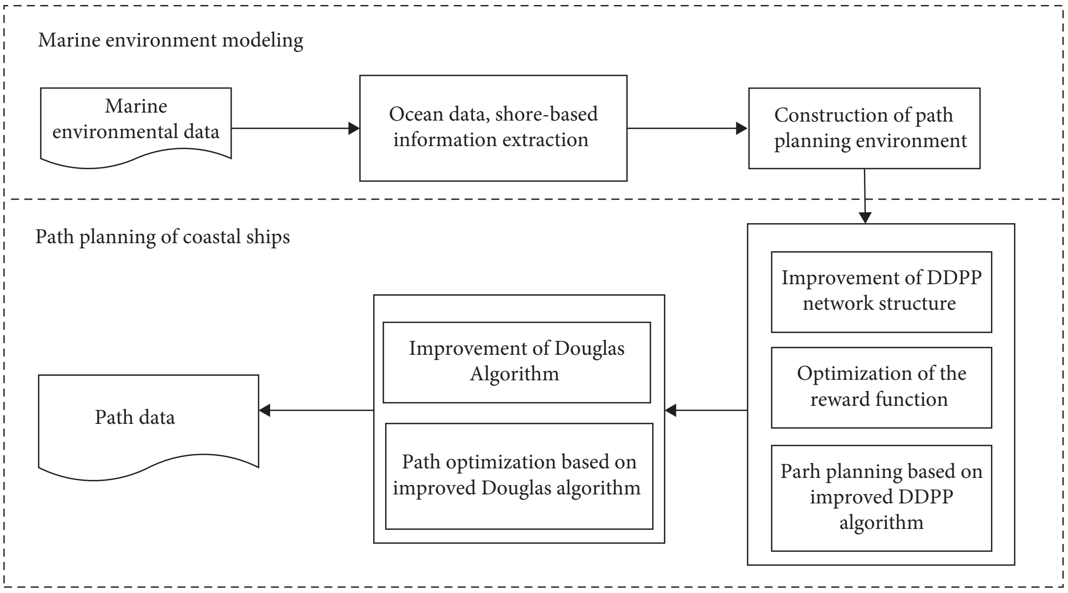

Marine environment modeling
Marine environmental data
Ocean data, shore-based information extraction
Construction of path planning environment
Path planning of coastal ships
Improvement of DDPP network structure
Optimization of the reward function
Parh planning based on improved DDPP algorithm
Improvement of Douglas Algorithm
Path optimization based on improved Douglas algorithm
Path data

Figure 1: Coastal ship path planning framework.

Critic network, which is used to update the value function and select the optimal action according to the next state. The target network does not carry out online training and updating of network parameters. The target and main network have the same neural network structure and initialization parameters. A soft update method is used to update the target network parameters, greatly improving the stability of learning.

In the process of path planning, the ship preferentially avoids the nearest obstacle in the current state, because the ship can only observe part of the environment in the current state, which increases the difficulty of the algorithm to predict the action [36, 37].

In the traditional DDPG network structure, each layer of the network is a fully connected layer network, and only the current status data is obtained each time, ignoring the historical status data. The DRL algorithm learns the strategy to complete the task by interacting with the environment. If the data obtained in this way is not fully utilized, the learned strategy may not be optimal, and the most effective action may not be predicted. For this reason, this paper chooses Long Short-Term Memory (LSTM) as the first layer network of the Actor and Critic networks in DDPG. By integrating current state information and historical state information, the integrated data is used to calculate the next input to the layer network, so that the action predicted by the algorithm is more consistent with the current state [38, 39]. The improved DDPG network structure is shown in Figure 2.

The LSTM layer in the Actor network is used to receive the ship status information s_t at the current moment t in the simulation environment and integrate it with the historical status information $h_{t-1,}$ and the integrated information is recorded as h_t. At the same time, h_t is used to calculate the input o_t of the next layer of the network through the forget gate and information enhancement gate of LSTM, and finally, the integrated data h_t is saved through the output gate

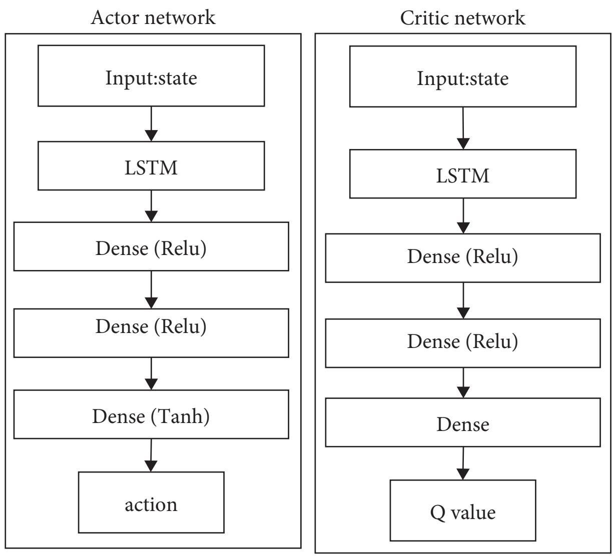


Figure 2: Improved DDPG network structure.

for the next calculation. The calculation of o_t is shown in formula (1), where f is determined by LSTM and ω is the network parameter of LSTM, and then enter o_t into the hidden layer composed of a 3-layer fully connected layer network. This can help DRL learn the optimal strategy faster. The number of neurons in each hidden layer is 256, and the ReLU activation function performs nonlinear processing on the output node of each hidden layer. Among them, in a neural network, when the number of neurons is too small, the network cannot have the necessary learning ability and information processing ability. On the contrary, if there are too many neurons, it will not only greatly increase the complexity of the network structure but also make the network easier to fall into a local minimum during the learning process, and the learning speed of the network will become very slow. This paper comprehensively refers to the selection of the number of neurons in literature [15] and literature [16] and finally determines the number of neurons in this paper. The DDPG algorithm predicts the best execution action in the current state according to formula (2), where μ is the hidden layer network parameter and π is the learned strategy. At the same time, in the last layer of the network, the Tanh activation function is used to limit the network single output action value between $[-1, 1]$, and finally, the network output action is converted into action in the continuous action range. Critic network and Actor network have the same network structure. The evaluation value of the output action of the Critic network is the final network result, which is called the Q value. When outputting the final action value Q, the activation function is not used to ensure that the output result of the network is a certain action value, which is used to evaluate the output action of the Actor network.

At each timestep t, the LSTM layer in the Actor network is used to receive the ship status information s_t and form the integrated state information o_t:

$$o_t = f(h_t; \omega), \tag{1}$$

where f is a transition function determined by LSTM, ω is the network parameter of the LSTM, and $h_t = s_{t-T}, s_{t-T+1}, \ldots, s_t$ is the historical observations from $t-T$ to t. Inputting o_t instead of h_t into the hidden layer composed of a 3-layer fully connected layer network can help DDPG learn the optimal strategy faster. The number of neurons in each hidden layer is 256, and the ReLU activation function performs nonlinear processing on the output node of each hidden layer. At the same time, the Tanh activation function is used to limit the network output action value at between $[-1, 1]$ in the last layer of the network:

$$a_t = \pi(o_t; \mu), \tag{2}$$

where μ indicates parameters of the hidden layer. The Critic network that has the same network structure as the Actor network is used to evaluate the output action of the Actor network. Its result is called the Q value which does not need to be processed by the activation function in the final output.

(2) Optimized Design of Reward Function. In deep reinforcement learning, the reward function plays an important role in evaluating the effectiveness of behavior decision and the safety of obstacle avoidance. The goal of deep reinforcement learning is to obtain the most rewarding search strategy in the process of ship navigation. When designing the reward function, the ship should avoid the obstacles safely and quickly to reach the target point. At present, most of the reward functions used in the path planning of unmanned ship based on deep reinforcement learning are fixed positive reward values when the next state of the ship is closer to the target point after the ship performs the action. Otherwise, given a fixed negative reward value, the use of this reward function will lead to slow convergence speed of the deep reinforcement learning algorithm. And the planned path does not meet the ship navigation specifications [40]. Based on the traditional reward function, this paper optimizes and designs the mainline reward function and auxiliary reward function. The mainline reward function is used to guide the ship to reach the target point and complete the path planning task. At the same time, the path planning task is further decomposed into subobjectives, and auxiliary reward functions are designed, respectively, so as to guide the agent to seek advantages and avoid disadvantages and improve the probability of mainline events. In order to ensure the core status and attractiveness of the mainline reward function, the absolute value of the auxiliary reward function is set relatively small to avoid affecting the guiding role of the mainline reward function. In this paper, the auxiliary reward function is divided into two parts: the first part is the punishment near the obstacle, the second part is the reward near the target point. The punishment near the obstacle is mainly used to help the ship learn to avoid the obstacle strategy, and the reward near the target point is used to help the ship move towards the target point quickly:

(1) Mainline reward function: the primary reward function is divided into two parts. One part is mainly used to guide the ship to move towards the target point in the process of ship navigation to complete the path planning task. The other part is to give the ship a larger final reward to encourage the ship to reach the target point. For the first part, in order to make the ship move to the target point as much as possible, the reward function set in this paper is shown in the following formula:

$$\text{reward} = -\left(d_{\text{goal}} + \alpha\left(\frac{\kappa}{\kappa + \min(d_{\text{obs}})}\right)^{\sigma}\right), \tag{3}$$

where d_{goal} is the distance between the ship and the target point and $\min(d_{\text{obs}})$ is the distance between the ship and the nearest obstacle, κ is the adjustment factor, which is used to adjust the impact of the nearest obstacle on the reward in general, and σ is the index coefficient, which has the same effect with κ. The value of σ and κ is $[1, 10]$. For the second part, in order to guide the ship to reach the target point and ensure the core status and attractiveness of the mainline reward function, this paper selects $\text{reward} = 10$ as the final reward to reach the target point.

(2) Auxiliary reward function: auxiliary reward function is divided into punishment near obstacles and reward near target point. Its main function is to assist the mainline reward function to let ships learn the strategy of avoiding obstacles and reaching target point quickly.

The penalty near the obstacle refers to the fact that when the ship has not met the obstacle near the obstacle (hereinafter referred to as the dangerous area), in order to help the ship leave the area quickly, the penalty should be increased near the obstacle. The penalty value is inversely proportional to the distance between the ship and the obstacle. Similarly, in order to avoid falling into the local optimum, the punishment in the dangerous area should not be too intensive, and there is a certain gap between the punishment in the dangerous area and that in the obstacle area. The specific punishment value is calculated by

$$\begin{cases} \min(d\_\text{obs}) > \alpha \text{ and } \min(d\_\text{obs}) \leq \beta, \text{reward} = -3, \\ \min(d\_\text{obs}) > \beta \text{ and } \min(d\_\text{obs}) \leq \delta, \text{reward} = -1.5, \end{cases} \tag{4}$$

where $\min(d\_\text{obs})$ is the minimum distance between the ship and the obstacle, α, β, and δ are thresholds standing for diverse distance ranges to the obstacle, and different penalties are given in the range of α, β, and δ.

The reward near the target point mainly refers to that when the ship has not reached the target point near the target point (hereinafter referred to as the reward domain), in order to help the ship reach the target point quickly, different rewards are given according to the distance between the ship and the target point to speed up the convergence speed of the model. At the same time, in order to prevent the ship from falling into the local optimum, the rewards in the reward domain should not be too dense. There should be a gap with the reward of reaching the target point. The specific calculation method of reward value is shown in

$$\begin{cases} d_{\text{goal}} > \partial \text{ and } d_{\text{goal}} \leq l, \text{reward} = 1.5, \\ d_{\text{goal}} > l \text{ and } d_{\text{goal}} \leq \zeta, \text{reward} = 1, \end{cases} \tag{5}$$

where ∂, l, and ζ are thresholds representing diverse distance ranges to the target point, d_{goal} is the distance between the ship and the target point, and different rewards are given in the range of ∂, l, and ζ, respectively.

3.3.2. Structure of Path Planning. Figure 3 shows the ship path planning model based on improved DDPG. The model mainly includes improved DDPG algorithm and environment model (ship's actual environment information, action controller, and path optimization). During the path planning process, the model first processes the ship's actual navigation information environment status information according to the description in Section 3.1, which is denoted as s_t. Meanwhile, it is entered into the Actor network and the Critic network. Then, the optimal ship action strategy is output, which satisfies the ship's maximum cumulative return during the learning process, by randomly extracting data from the experience buffer pool for repeated training and learning. Action controller module in the environment executes the generated action, calculates the reward value of the ship's execution of the action according to the reward function, and stores the current ship state, the executed action, the return value of the executed action, and the ship state at the next moment after the action is executed in the replay buffer. The improved DDPG uses the status and return value to estimate the value of current actions and constantly adjusts its value function so that its output action is more in line with the current ship's actual sailing status. Finally, the planned path is further optimized through the path optimization module, which makes the optimized path safer and more economical. During the training process, the Actor network in the algorithm uses Actor Optimizer, uses deterministic policy gradients to update network parameters, and continuously corrects the action strategies which is generated. Critic network uses Critic Optimizer to train network parameters by minimizing the loss function and evaluate the action strategy in terms of action value.

In the process of updating the network parameters, first, obtain some experience from the replay buffer D, and then, obtain the target return value y_i through the target network, and its calculation is shown in

$$y_i = r_\text{i} + \gamma Q'\left(s_{i+1}, \mu'\left(h_{i+1} \mid Q^{\mu'}\right)\theta^{Q'}\right). \tag{6}$$

Then, update the main Critic network according to the target return value y_i , and input s_i and a_i into the main Critic network to obtain the actual value Q and the policy gradient $\nabla_{\theta^\mu} J$. Then calculate the error of the main Critic network according to the error equation, and update the network by minimizing the error. The error equation is shown in formula (7). At the same time, the main Actor network is updated according to the policy gradient $\nabla_{\theta^\mu} J$, and the calculation of $\nabla_{\theta^\mu} J$ is shown in

$$L = \frac{1}{N} \sum_i \left(s_i - Q\left(s_i, a_i \middle| \theta^Q\right)\right)^2, \tag{7}$$

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_{ii} \nabla_a Q\left(s, a \middle| \theta^Q\right)\Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu\left(h \middle| \theta^\mu\right)\Big|_{s_i}. \tag{8}$$

Finally, the target network parameters are updated. The target network will not directly copy the parameters of the main network but will update the parameters in a soft update mode; that is, the parameters are only updated a little bit each time. The update is shown in formula (9):

$$\begin{cases} \theta^{Q'} = \tau\theta^Q + (1-\tau)\theta^{Q'}, \\ \theta^{\mu'} = \tau\theta^\mu + (1-\tau)\theta^{\mu'}. \end{cases} \tag{9}$$

Among them, θ^Q and θ^μ are the parameters of the main Actor network and the main Critic network, $\theta^{Q'}$ and $\theta^{\mu'}$ are the parameters of the target Actor network and the target Critic network, and $\tau \ll 1$ is the update coefficient.

3.3.3. State Space. This paper mainly studies the path planning of coastal ships. In the planning process, the ship first needs to read and quantify the environmental information in the nautical chart. At the same time, the ship will receive shore-based information (shore-based information mainly includes ship-wreck area, restricted navigation area, and military exercise area). These pieces of information are sent by special departments. We will also read shore-based information and quantify it and convert it into obstacle environmental data in the experimental environment. When planning a path, the strategy of avoiding obstacles closest to the ship at the current moment is adopted. At each moment, the ship navigation status information provided by the

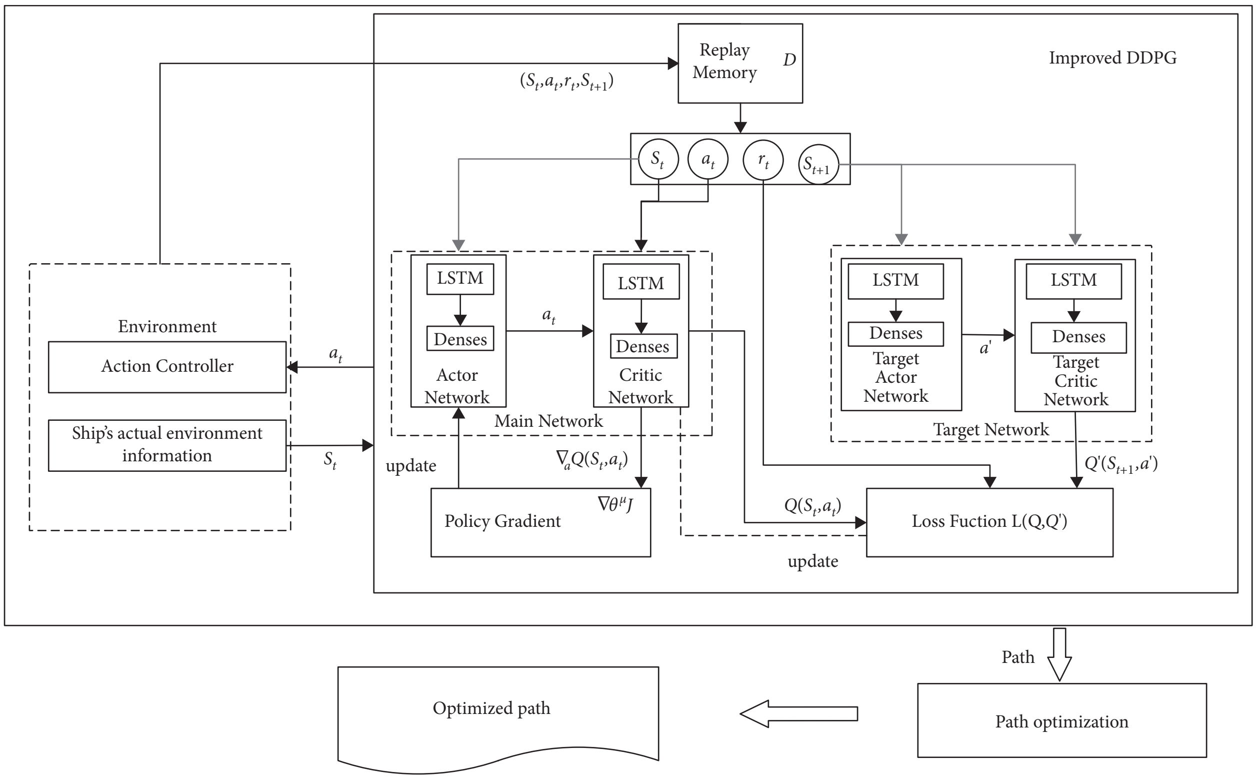


FIGURE 3: Ship path planning model based on improved DDPG.

experimental environment mainly includes the distance between the ship and the nearest obstacle, the distance between the ship and the target point, and the speed and azimuth of the ship.

Figure 4 shows the ship state information diagram in the experimental environment at time t. A diamond represents the ship S_{ship} and its current position is denoted as $(x_{\text{ship}}, y_{\text{ship}})$; a pentagram represents the position of the next waypoint, namely, the target point S_{goal}, and its position is recorded as $(x_{\text{goal}}, y_{\text{goal}})$; a circle represents an obstacle $S_{\text{obscale}} = (x_{\text{obs}}, y_{\text{obs}})$. In the environment, the north direction is the direction of axis Y; the east direction is the direction of axis X; v_{ship_p} denotes the speed of the ship; (v_x, v_y) represents the component of v_{ship} on the X, Y coordinate axis separately, which can be calculated by formula (10); φ_v is the speed azimuth of the ship; ϕ is the angle between the target point and the ship speed; φ_{obs} is the relative azimuth between the ship speed and the obstacle; d_{goal} is the distance between the ship and the target point; and d_{obscale} is the distance between the ship and the obstacle:

$$\begin{cases} v_x = v_{\text{ship}} \sin \varphi_v, \\ v_y = v_{\text{ship}} \cos \varphi_v. \end{cases} \tag{10}$$

The experimental environment is based on the position of obstacles and the ship; the ship navigation information fusion module calculates the distance between each obstacle and the ship at the current time, selects the nearest obstacle $(x_{\text{obs}}, y_{\text{obs}})$, and calculates $(dx_{\text{obs}}, dy_{\text{obs}})$ according to formula (11), which represents the projections of the distance between the ship and the obstacle at the moment t in X-axis and Y-axis directions:

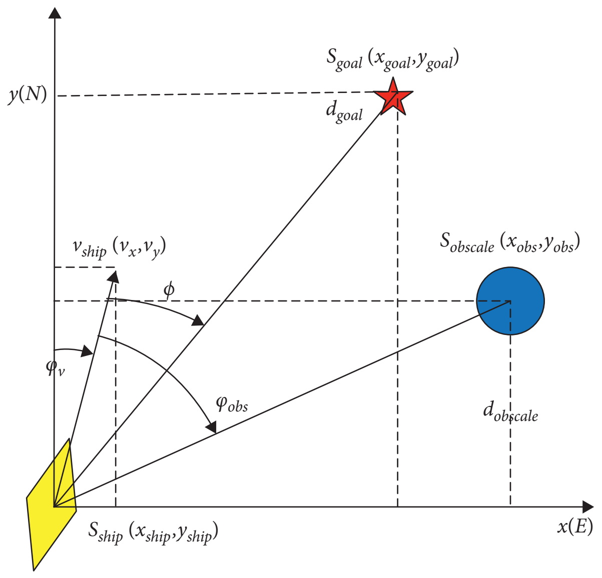

$S_{goal}\,(x_{goal},y_{goal})$
$y(N)$
d_{goal}
$v_{ship}\,(v_x,v_y)$
$S_{obscale}\,(x_{obs},y_{obs})$
ϕ
φ_v
φ_{obs}
$d_{obscale}$
$S_{ship}\,(x_{ship},y_{ship})$
$x(E)$

FIGURE 4: Information the ship sensed at time t.

$$\begin{cases} dx_{\text{obs}} = x_{\text{obs}} - x_{\text{ship}}, \\ dy_{\text{obs}} = y_{\text{obs}} - y_{\text{ship}}. \end{cases} \tag{11}$$

The state information at time t is defined as $s_t = [v_x, v_y, p_x, p_y, dx_{\text{obs}}, dy_{\text{obs}}, \varphi_v, \phi]$, where p_x and p_y are the projections of the distance between the ship's position and the target point at the moment t in X-axis and Y-axis directions, and they are calculated as shown in

$$\begin{cases} p_x = x_{\text{goal}} - x_{\text{ship}}, \\ p_y = y_{\text{goal}} - y_{\text{ship}}. \end{cases} \tag{12}$$

3.3.4. Action Space Design. During the ship's navigation, the pilot ensures the safety of the ship in the complex navigation area by changing the course and speed. In ship path planning based on improved DDPG, the ship motion is controlled by action which is predicted by the algorithm according to the ship state. At the same time, in order to let the agent try more new actions to explore better action strategies and avoid falling into the local optimum, random noise is introduced in the training process, and the decision-making process of the action is changed from determinism to a random process. The value a_t of the action is sampled from the random process. The action selection process is shown in Figure 5.

In the learning process, there is a trade-off between exploration and exploitation. On one hand, the agent needs to choose as many different behaviors as possible to find the optimal strategy, which can be called exploration. On the other hand, the agent will consider the behavior with the largest Q value to obtain huge returns, which can be called exploitation. Exploration is very important for learning and only through exploration can the optimal strategy be determined. However, too much exploration will reduce the performance of ship path planning and affect the learning efficiency. Because the $\varepsilon -$ greedy strategy can prevent the system from falling into the local optimal, the algorithm adopts it to complete the action selection of ship path planning. In the $\varepsilon -$ greedy strategy, a certain probability of random change is increased in the process of behavior selection. In the current state, the agent will randomly select the action with probability ε to ensure that all state spaces can be explored and choose the action $a_{\max}$ with the largest current Q value with probability $1 - \varepsilon$ to make the best use of the knowledge learned.

In this paper, the ship's action space mainly includes action control strategies and action exploration strategies. The action control strategy mainly adopts Actor network to predict the action of the ship. The action exploration strategy is to add random noise to the output action when designing the neural network structure to encourage the ship to try more new actions.

(1) Action Control Strategy. In our path planning model, Actions yield by the Actor network includes the ship's speed increment d_v, which is used to control the changes of the magnitude of speed, and heading increment d_α, which is used to control the speed direction change, and the ship's movement is controlled by the speed and heading together. The Tanh activation function is used to ensure that the output value of the neural network is between -1 and 1. The update formulas for heading, speed, and ship position in this paper are shown in formulas (13)–(15), where α_t is the

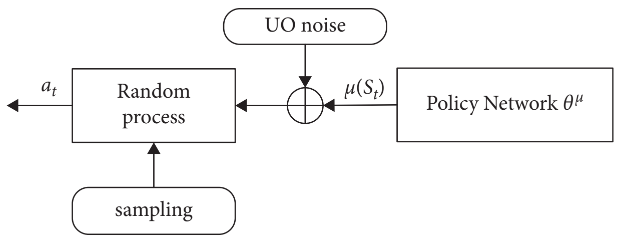


Figure 5: Improved DDPG action selection process.

current heading, α_{t-1} is the previous heading, M_α is the maximum heading increment that can be selected, M_v is the maximum speed increment, $(x_{\text{ship}_t}, y_{\text{ship}_t})$ is the current ship position, $(x_{\text{ship}_t}, y_{\text{ship}_t})$ is the ship position at the previous time, and dt is the update time step.

$$\alpha_t = \frac{\left(\alpha_{t-1} + \text{Tanh}\left(d_\alpha\right) M_\alpha\right)}{\pi}, \tag{13}$$

$$v_t = \left(v_{t-1} + \text{Tanh}\left(d_v\right) M_v\right), \tag{14}$$

$$\begin{cases} x_{\text{ship}_t} = x_{\text{ship}_t} + v_t \cdot \cos\left(\alpha_t\right) \cdot dt, \\ y_{\text{ship}_t} = x_{\text{ship}_t} + v_t \cdot \sin\left(\alpha_t\right) \cdot dt. \end{cases} \tag{15}$$

At the same time, in order to prevent the ship's speed from increasing endlessly, the maximum axis speed $V_{\max}$ is set. The maximum axis speed refers to the maximum speed value in a single direction along the X-axis or the Y-axis. When the maximum value of ship speed v_{ship} on the X axis or the Y-axis is reached, no further increase is made to limit the ship speed.

(2) Action Exploration Strategy. In terms of action exploration strategies, the value of Action in algorithms such as DQN is discrete, but the value of Action in DDPG is continuous. Action exploration in the continuous control space enables the unmanned ship to explore well and find better actions. DDPG constructs an exploration policy μ' by adding random noise from a noise process И to the actor policy.

$$\mu'\left(s_t\right) = \mu\left(s_t\right) + \text{И}. \tag{16}$$

This paper adopts the Ornstein-Uhlenbeck (OU) noise mentioned in the literature [27], which is generated by the OU process and is suitable for continuous space. The OU process is time-dependent, and the exploration of the environment is more efficient. Therefore, adding OU noise to the action policy in the DDPG algorithm can accelerate the training speed and improve the exploration efficiency.

4. Path Optimization Based on Improved DP Algorithm

Since the path planning model based on the improved DDPG algorithm ultimately retains the random exploratory nature of the ships to fully explore the environmental information, the path planned by this algorithm has many unnecessary turning points. In order to reduce the operational risk during the actual navigation, it is necessary to compress these inappropriate turning points. An improved DP algorithm is proposed to optimize the path planned.

The trajectory data compression algorithms are mainly divided into two categories: one is the nonlinear trajectory fitting algorithm to smooth the trajectory [41], and the optimized trajectory handled with this type of method is more consistent with the actual robot motion trajectory, but it is not suitable for ships. Another category is the segmented linearization of the motion trajectory algorithm [42], which optimizes the trajectory as segmented straight lines, and the path optimized is in the shape of polyline. Due to the fact that the ship's path in the actual navigation process is composed of several waypoints and its navigation path is a polyline, the second type of algorithm is more suitable for the path optimization of the ship. Among many linear compression algorithms, the DP algorithm [43] is the most representative and widely used algorithm.

The DP is an algorithm that approximates the curve as a series of discrete points, connects a straight line fictitiously between the first and last points, and optimizes the curve as a polyline according to the distance between every point and the line. The basic idea of the algorithm is to initialize a trade-off threshold, connect the first and last points of the curve to a straight line, and compute the distance between every intermediate point and the straight line. Then find the maximum distance dmax, and compare dmax with the trade-off threshold: if dmax < threshold, all intermediate points on the curve are discarded; if dmax ≥ threshold, the curve is divided into two parts by this point, and the above process is repeated for these two parts of the curve until all the points are processed.

In nautical practice, when ships sail along the path, more turning points will cause additional resource consumption. Therefore, it is necessary to ensure that turning points on the optimized path are as few as possible. Although the traditional DP algorithm is simple and relatively efficient, the threshold value directly determines the number of turning points in the optimized path. There will be the following problems: if the threshold is too large, the optimized path may pass through the obstacle; if the threshold is too small, there may still be some turning points to be removed. Therefore, it is necessary to improve the DP algorithm.

4.1. Improvement of DP Algorithm. The basic idea of the improvement is to remove the superfluous turning points in the path to the maximum extent and to ensure that the path can avoid all obstacles. The specific approach is to reoptimize the path obtained by the traditional DP algorithm to reduce the unnecessary turning points.

First of all, the last track point of the path is set as the current point. For each point starting from the first track point to the prior point of the current point in the path, a segment is drawn between it and the current point successively; if there are no obstacles on the line, the path will be updated by removing all track points between these two points on the path. Then the second-to-last point on the path is set as the current point, and repeat the operation when the last track point is the current point. Repeat until the second point is set as the current point. Finally, the optimized path can be obtained by successively connecting the remaining track points on the path to form a broken line.

For example, there is a path optimized by the traditional DP algorithm; as shown in Figure 6(a), there are five track points on the path, named A, B, C, D, and E. While the last track point E is set as the current point, as shown in Figure 6(b), line segment AE is drawn firstly and there is an obstacle on it, so points B, C, and D remain; then line segment BE is drawn and there is an obstacle on it too, so points C and D remain, but as line segment CE is drawn, there are no obstacles on CE, so point D is removed from the path, as shown in Figure 6(c). Point B is removed while C is set as the current point.

Compared with the path optimized by the tradition DP algorithm, as shown in Figure 6(a), the turning points in the path optimized by the improved DP algorithm, as shown in Figure 6(c), are fewer, and there is only one turning point, which greatly improves the economic benefits of the ship during navigation.

4.2. Path Optimization Algorithm Based on Improved DP. Using the improved DDPG algorithm, a trajectory curve containing a set of points can be obtained. These points are the input of the improved DP algorithm used to optimize the planned path. The steps are as follows:

Step 1. Set the value D as the threshold for whether to delete a track point. The line segment between the starting point P_{start} and the ending point P_{end} of the trajectory curve is taken as the chord l of the curve. Traverse all other points on the curve, calculate the distance from each point to l, and find the point $P_{\max}$ farthest from l and the distance $d_{\max}$ from $P_{\max}$ to l.

Step 2. Compare the size of $d_{\max}$ with the threshold D; if $d_{\max} < D$, then take the line segment as the approximation of the trajectory curve and go to Step 4.

Step 3. If $d_{\max} \geq D$, divide the curve into segments $P_{\text{start}}P_{\max}$ and $P_{\text{end}}P_{\max}$, reset the start and endpoints for each section, and go to Step 1.

Step 4. Store all segmentation points and P_{start} and P_{end} in a bidirectional circular linked list denoted as P. Define the left and right pointers pointing to P_{start} and P_{end}, respectively.

Step 5. If *l*eft and *right* are pointed to two adjacent points, go to Step 7.

Step 6. If there is no obstacle on the line segment between the point pointed by *l*eft and the point pointed by *right*, remove all the dividing points in the middle of these two points, and go to Step 7; otherwise, let *l*eft point to the next dividing point and go to Step 5.

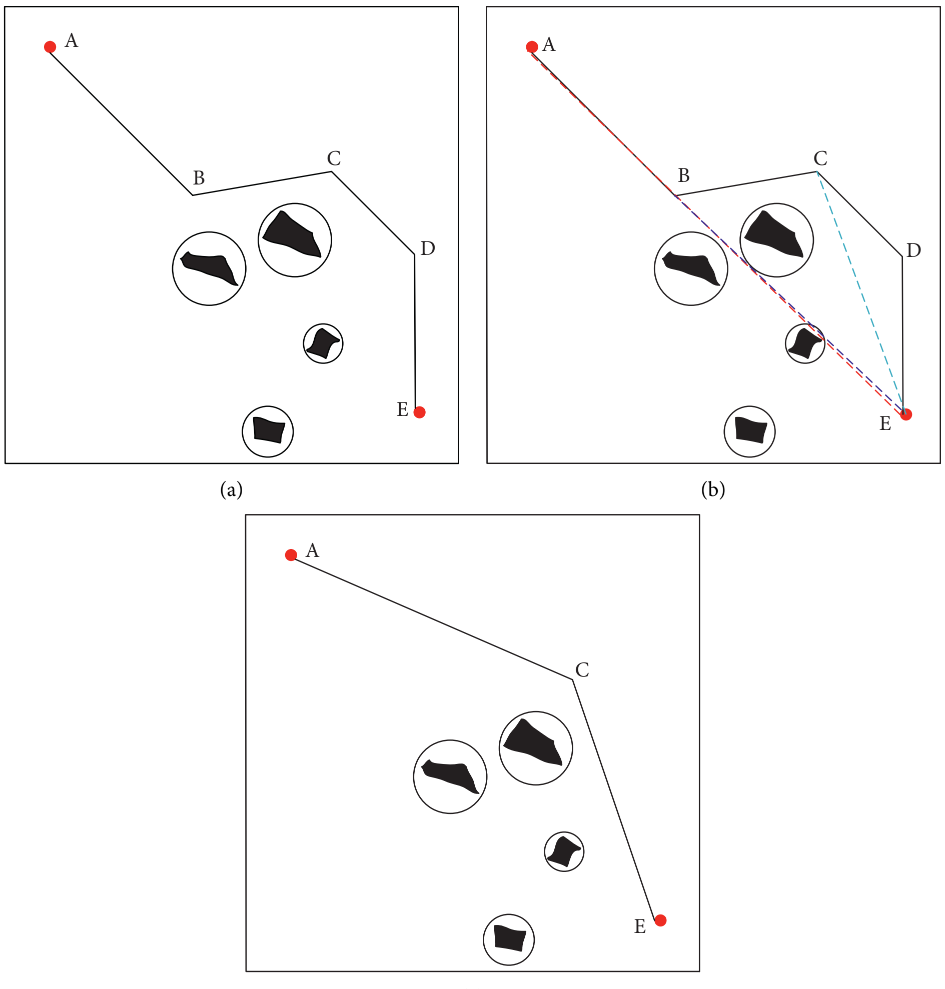


Figure 6: Improved DP algorithm idea diagram. (a) Path optimized by DP algorithm. (b) Optimization thinking chart of point E. (c) Improved DP algorithm optimized path.

Step 7. Let the *right* pointer move one step forward and pointer *left* point to P_{start}. If pointer *right* reaches P_{start}, the optimization is complete, and go to Step 8; otherwise, go to Step 5.

Step 8. Connect the remaining points in the bidirectional circular linked list P in turn to form a fold line, which is the optimized path of the original trajectory.

5. Experimental Verification and Result Analysis

In this section, simulation experiments are conducted to verify the reliability and effectiveness of the method proposed in this paper. It mainly includes verification in the simulation experimental environment and comparison with other algorithms. It is assumed that the sea area of the ship sails is an open sea area with no coastline but only buoys and other obstacles. In the actual environment, the number of proven obstacles and shore-based information in coastal waters will be very small, so this paper sets the corresponding number of obstacles according to the actual situation for experimental comparison.

5.1. Environment Construction and Parameter Setting. Python, Gym, and Matlab are used to construct an environment for algorithm verification. Gym is a python package used to develop and compare reinforcement learning algorithms. It allows researchers to customize training scenarios completing tasks and visualizing the process of completing tasks. At the same time, it also provides many existing scenarios for researchers to verify the effectiveness of the algorithm. Matlab is a mathematical science and technology application software. It provides researchers with a high-tech computing environment for scientific calculation, visualization, and interactive programming. It is widely used in data analysis, deep learning, robotics, and control systems.

Obtain map data information through electronic chart platform. This paper selects the real sea environment as the environment space of model training. Read the obstacles in the nautical chart, the ship's starting point and target point, and other data firstly. Then use the method in Section 3.1 to process and use Python and Gym to show the processed environment. During the process of the ship's actual voyage information, 1 pixel (px) corresponds to 0.1 nautical miles. The size of the experimental environment is set for $600px \cdot 600px$, which is equivalent to 60 nm $*$ 60 nm in the actual chart. Figure 7 shows the experimental environment after the actual environment has been processed. The green point is the ship's starting point with coordinates [540, 540], the red point is the ship's target point with coordinates [60, 60], and the yellow is the obstacle part. The color change around the obstacle part represents the change in the height of the obstacle, and the dark blue part is the navigable area of the ship.

During the training, the environment interacts with the improved DDPG algorithm to plan the ship's sailing path. When there is no obstacle in the environment or it is far away from the obstacle, the algorithm will choose the action that preferentially approaches the target point. When it is close to an obstacle, the algorithm will prioritize actions to avoid the obstacle and move towards the target point while avoiding it. The path planning does not end until it reaches the target point. In this process, the algorithm continuously interacts with the environment to improve the ability of action decision-making. The parameter settings of the proposed model in the training process are shown in Table 2. Because the DRL algorithm takes a long time to train the model, we generally refer to the selection of the original algorithm and the parameter settings of better papers in the same research field for the selection of model parameters. Among the model parameters in Table 2, the parameter action space is set by this paper. According to the predicted action, the model will be mapped to a specific action through the formula in Section 3.3.4. The parameter ReLU is the activation function used in this paper. We refer to the selection of the literature [9] and literature [29] for the two parameters update factor τ and explore decay rate.

5.2. Model Validation. In order to verify the effectiveness of this method, this section will compare the two parts of model verification and experimental comparison. The model verification compares the improved DDPG algorithm in this paper with the traditional DDPG algorithm and the improved DP algorithm with the traditional DP algorithm. The experimental comparison mainly compares the path planned by the method proposed in this paper with the traditional DDPG algorithm, A* algorithm, RRT* algorithm, RRT algorithm, APF algorithm, and BUG2 algorithm.

5.2.1. Improved DDPG Algorithm Validation. In path planning, while detecting the ship reaches the target point or collides with an obstacle, the current episode ends and the next episode starts. For safety, if the ship is 1 nautical mile away from the obstacle, it is considered that a collision has been detected. Figure 8 shows the path planned by the model under different iteration times. Since there are many turns in the path planned by the algorithm in the initial exploration stage, the 3D environment is not easy to display, so this

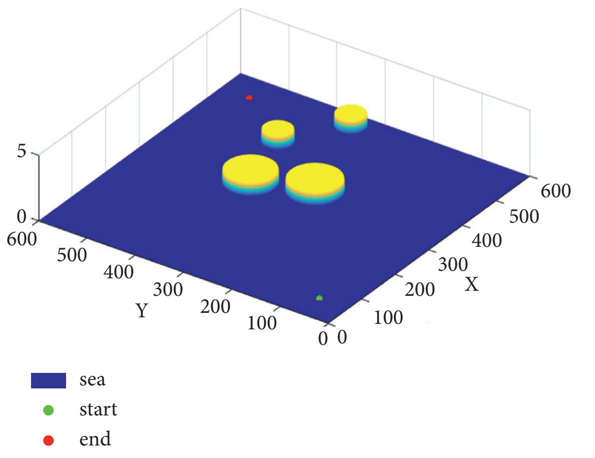


Figure 7: Experimental environment.

section uses the 2D environment to show the path planned by the model under different iteration times. Among them, the black polygons represent obstacles, and the circle around each obstacle is the enclosing circle. The red circle in the upper left corner represents the target point, the green circle in the lower right corner is the starting point, and the blank area in the middle is the navigable area.

As shown in Figure 8(a), in the initial 200 iterations, the algorithm is in the initial exploration stage, and the strategy learned is not optimal. Although the task of avoiding obstacles to reach the target point is completed, there are many exploratory actions at the target point and near the obstacle, leading to many turning in the planned path. Figure 8(b) shows the planned path at 600 iterations. It can be seen from the figure that there are few broken lines near the obstacle, and the broken lines near the target point are also reduced, which indicates that the algorithm has learned the strategy to avoid obstacles, but the strategy is not very stable due to the small number of iterations. The planned path at 800 iterations is shown in Figure 8(c). By continuous "exploration," the collision phenomenon is gradually reduced, and the planned path is guaranteed from the safety aspect. Although the path is still highly volatile, the redundant path points are reduced considerably. The final planned path is shown in Figure 8(d). It can be seen that the ship has successfully avoided all obstacles to reach the target point. The planned paths gradually tend to be stable, but there is still fluctuation, which is due to the retention of the random exploration rate of the action space. The reward score of the algorithm during the training process is shown in Figure 9.

It can be seen from Figure 9 that since there is no strategy at the beginning of the iteration and the model is in the exploratory stage, the episode reward is very low. From around 200 episodes, the reward value begins to rise, but it is still in the stage of exploring learning strategies. With the continuous optimization of the model learning strategy, the episode reward is on the rise. From around 700 episodes, it can be seen that the reward starts to be stable and the model has basically converged, but there are still poor rewards in some episodes. This is because the algorithm is still exploratory during the training process, and there is still a probability to choose some bad actions.

Table 2: Parameter settings.

Name	Value	Description
Action space	[−0.001, 0.001]	Optional action of the ship
Learning rate α	0.001	Learning rate of neural network
Decay factor γ	0.99	Decay factor of cumulative reward
Update factor τ	0.001	Discount factor for model
Explore decay rate	0.995	Exploration decay rate of action
Experience replay memory D	50000	Storing historical experience data
Sample size $D_{\min}$	256	Size of extracted empirical data
Episode	1000	Number of training rounds
Activation function	ReLU	Neuron activation function

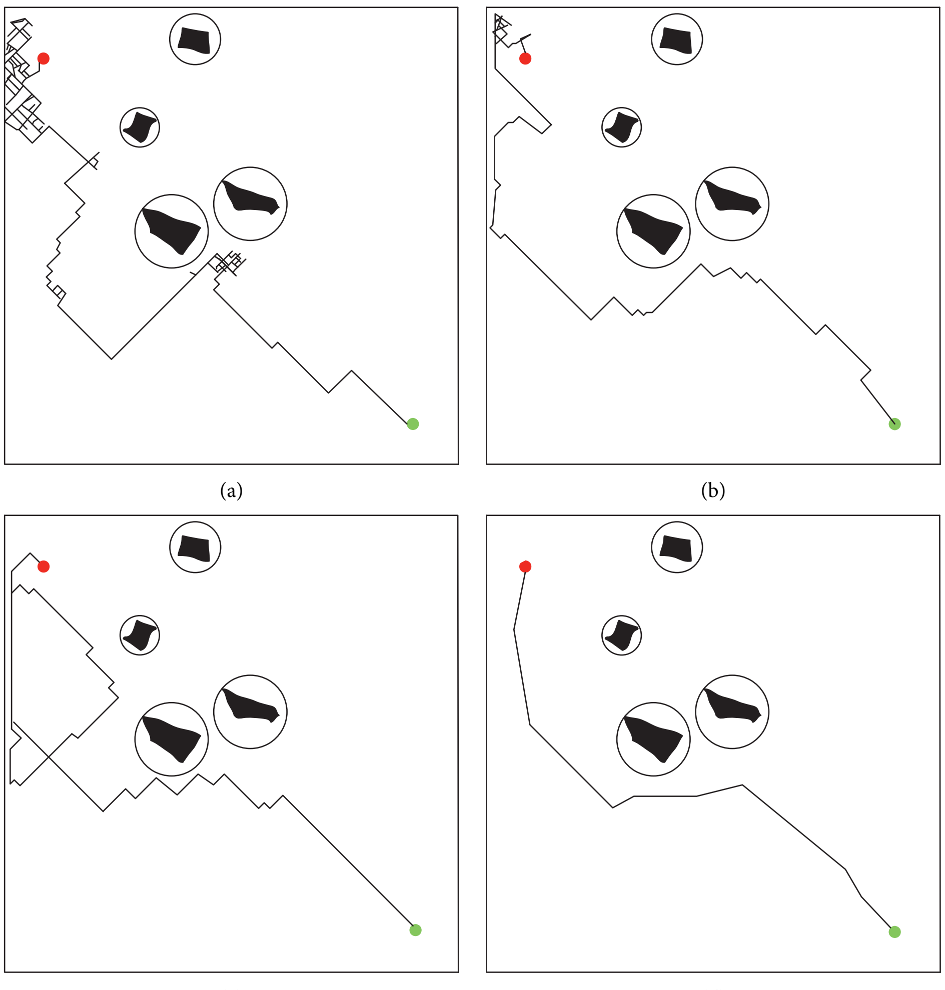


Figure 8: Planned path effects under different iteration times. Planned path at (a) 200 iterations, (b) 600 iterations, (c) 800 iterations, and (d) 1000 iterations.

The number of training episodes and corresponding steps of the DDPG algorithm combined with LSTM (denoted as LSTM + DDPG) proposed in this paper and the traditional DDPG algorithm (denoted as DDPG) is shown in Figure 10. The abscissa in the figure represents the number of training rounds, and the ordinate recommends the number of steps required from the starting point to the target point for each iteration. The blue line indicates the iteration trend of the DDPG algorithm combined with LSTM, while the red line represents the iterative trend of the traditional DDPG algorithm. Figure 10 clearly and intuitively shows the convergence speed and training effect of the two algorithms during training. It can be seen that the number of round steps of the LSTM + DDPG starts to decrease around the 100th round, showing a trend of gradual convergence, while the DDGP starts to converge around the 200th round. At the same time, it can be seen that the number of round steps of the DDPG algorithm is about 180, while the number of steps

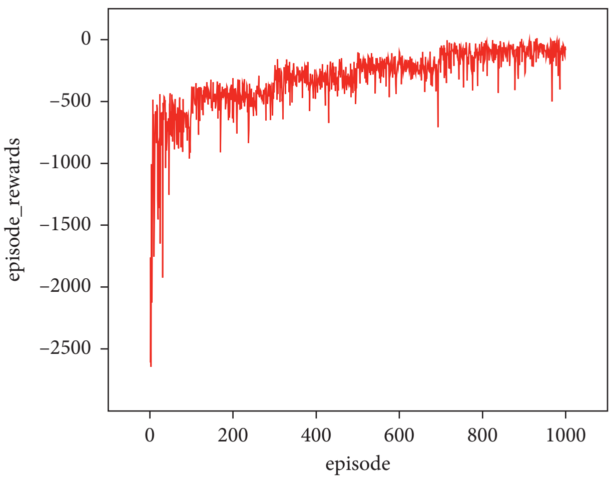


FIGURE 9: Reward score graph.

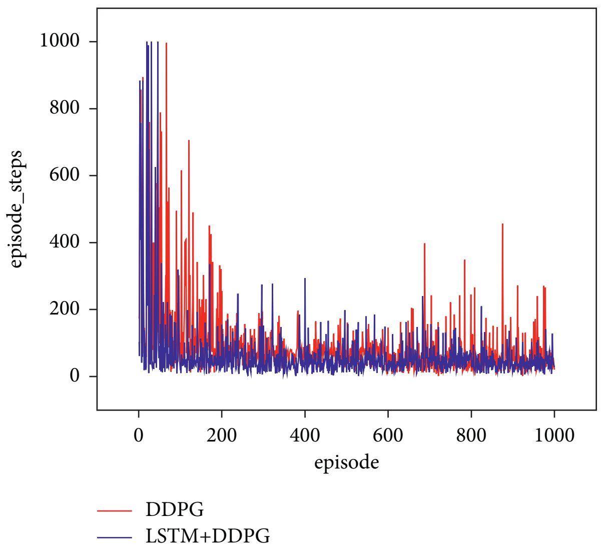


FIGURE 10: Comparison results of iterative convergence trends.

of the LSTM + DDPG is about 150, which indicates that the path planned by the DDPG algorithm has more redundancy. In the rounds between 700 and 1000, the frequency of fluctuations in the number of steps of the LSTM + DDPG is very small, but the number of steps of the DDPG algorithm fluctuates greatly. It shows that the LSTM + DDPG is better than DDPG in terms of stability.

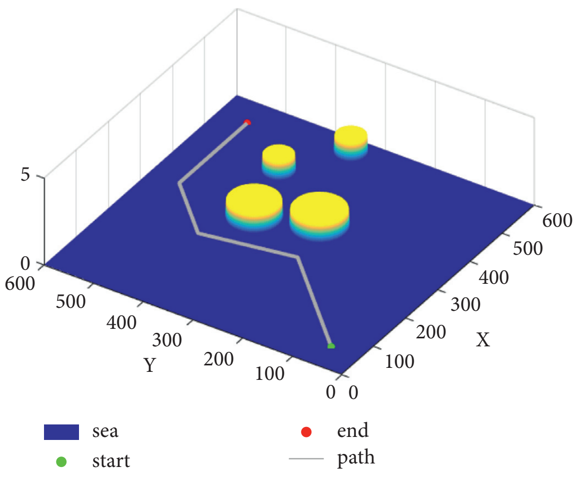
5.2.2. Improved DP Algorithm Verification. Since the turning points of the ship's navigation path during actual sailing should be as few as possible, the turning points that can be combined in the planned path should be handled further. The DP algorithm is a relatively efficient path optimization algorithm so far, but its threshold value is not easy to determine, and the optimized curve may have too many turning points. The DP algorithm is improved in this paper. Figure 11 shows the path optimized by the DP algorithm and the path optimized by the improved DP algorithm. The green circle in the lower right corner is the starting point, the red circle in the upper left corner is the target point, the yellow is the obstacle part, the color change around the obstacle part represents the height change of the obstacle, the dark blue part is the navigable area of the ship, and the white line is the planned path.

Figure 11(a) shows the path optimized by the traditional DP algorithm. Compared with the path without optimization (see Figure 8(d)), it is much smoother overall, and the number of turning points is much less. The number of turning points is reduced to 3, but it can be seen from the figure that some turning points can still be optimized. Figure 11(b) shows the final path optimized by the improved DP algorithm. The path is very smooth overall and there is no redundant turning point. Compared with the path without optimization, the number of turning points on the path is reduced by 7. Compared with the path optimized by the traditional DP algorithm, the number of turning points is reduced to a minimum, with only one turning point. In terms of the path length, the path length before optimization is 101.597n miles, and the path length optimized by the traditional DP algorithm is 80.863n miles, which is 20.734n miles less than that without optimization. The path length optimized by the improved DP algorithm is 73.1170n miles, which is 28.48n miles less than the path without optimization and 7.746n miles less than the path optimized by the traditional DP algorithm. From both the number of turning points and the length, it shows that the path optimized by the improved DP algorithm is more economical and safer.

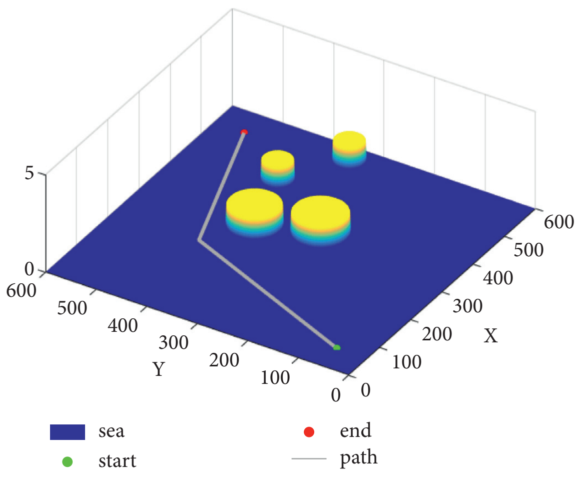
5.3. Experimental Comparison. Figure 12 shows the path planned by the proposed LSTM + DDPG algorithm, DDPG algorithm, RRT algorithm, RRT* algorithm, APF method, A* algorithm, and BUG2 algorithm in the same marine environment.

Among them, the parameter setting of the LSTM + DDPG algorithm is shown in Table 3.

The path planned by the model proposed in this paper (Figure 12(a)) has no redundant turning points, and the path meets the actual sailing requirements of the ship and is highly maneuverable. The path planned by the DDPG algorithm (Figure 12(b)) has fewer turning points and shorter distances. However, the path passes through multiple obstacles, which increases the risk of navigation. The path planning based on the RRT algorithm (Figure 12(c)) is more suitable for obstacles, but there are more path turning points. At the same time, the path passes through two relatively close obstacles, which increases the risk of ship driving and is not suitable for actual navigation regulations. The path planned based on the RRT* algorithm (Figure 12(d)) is more in line with the path of the ship sailing. But compared with the path planned by the model proposed in this paper, there is one more turning point, and the overall path length is longer than the path planned by the model proposed in this paper. The path planned by the APF algorithm (Figure 12(e)) has a curvature, which does not conform to the actual navigation path of the ship as a whole, and the planned path passes through two relatively close obstacles, which increases

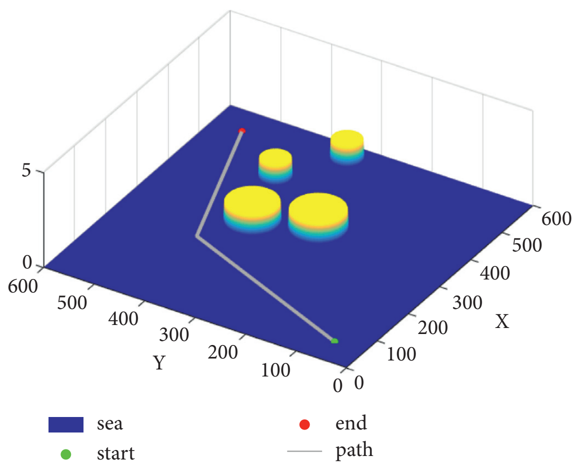


(a)

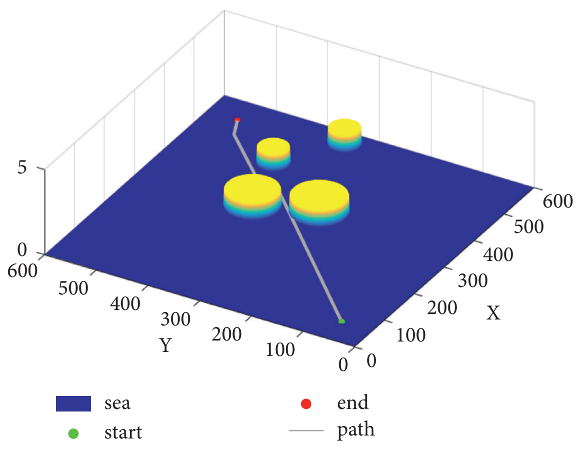


(b)

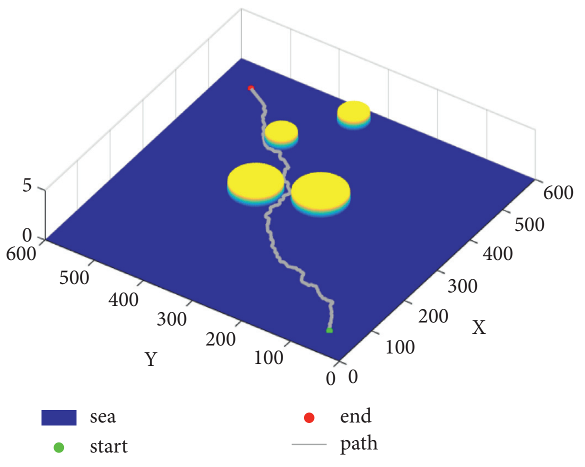
FIGURE 11: Comparison of the path optimized by the DP algorithm and the path optimized by the improved DP algorithm. (a) Path optimized by DP algorithm. (b) Path optimized by improved DP algorithm.



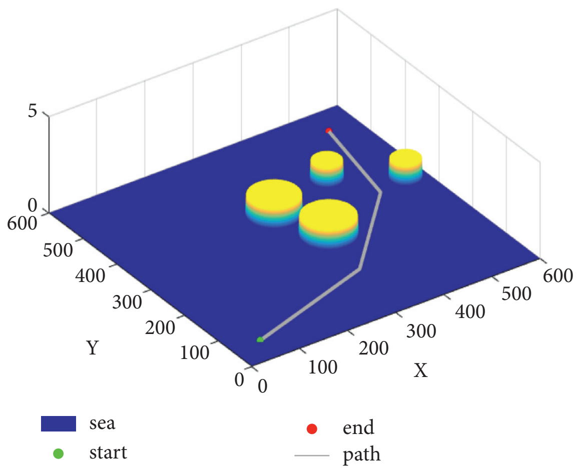
(a)



(b)

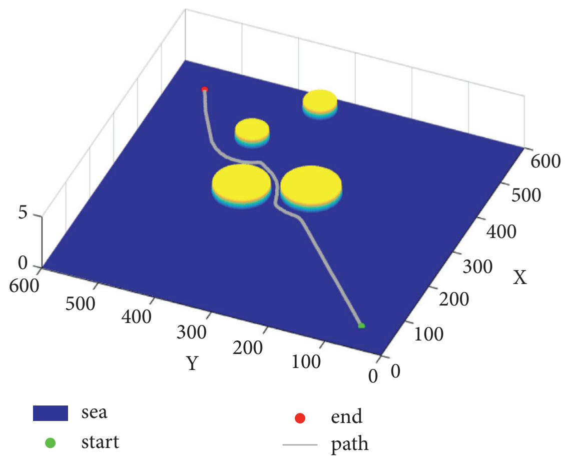


(c)

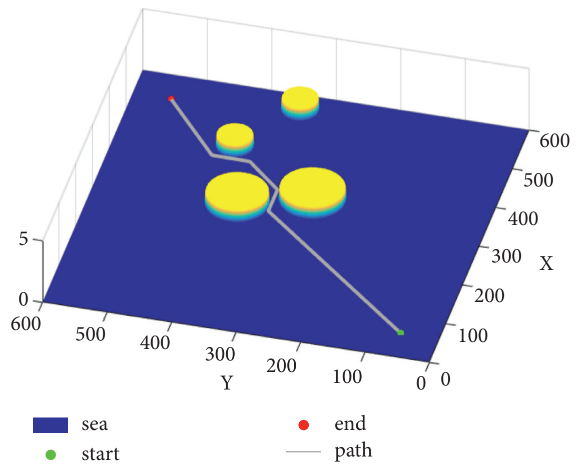


(d)

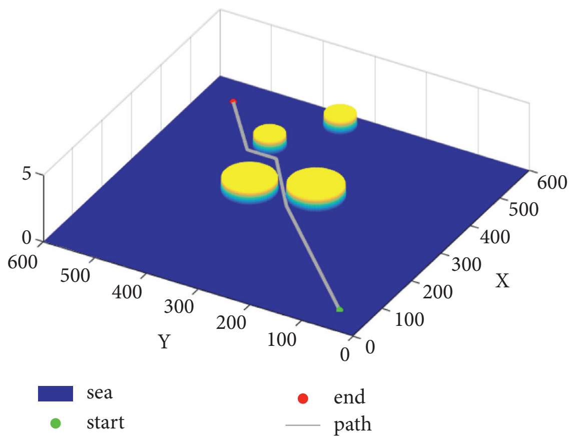
FIGURE 12: Continued.



(e)



(f)



(g)

FIGURE 12: Paths planned by different algorithms. (a) Path planned by LSTM + DDPG algorithm. (b) Path planned by DDPG algorithm. (c) Path planned by RRT algorithm. (d) Path planned by RRT* algorithm. (e) Path planned by APF algorithm. (f) A* algorithm-planned path. (g) Path planned by BUG2 algorithm.

TABLE 3: Parameter settings of the comparison algorithm.

Algorithm	Parameter settings
LSTM + DDPG	The parameter settings are shown in Table 2
DDPG	The parameter settings are shown in Table 2
A*	8 neighborhood search
RRT*	Step size: 10, sampling rate: 0.1, search radius: 20, and number of iterations: 10000
RRT	Step size: 5, sampling rate: 0.05, and number of iterations: 10000
APF	Attraction coefficient: 1.0, repulsion coefficient: 1000.0, step length: 2, iteration number: 5000, and obstacle influence radius: 3
BUG2	The distance D from the ship's current position to the target point; the distance F from the ship to the first visible obstacle

the risk of ship driving. Figure 12(f) shows the path planned by the A* algorithm. The A* algorithm walks in a straight line when there is no obstacle and walks next to the obstacle when it encounters an obstacle. It can be seen that the path planned by the A* algorithm has no arc, and it passes between two obstacles that are relatively close and do not have practical operability. The BUG2 algorithm will go around obstacles when planning the path (Figure 12(g)), and the path will stick to the obstacles. We can see that the whole path planned with the BUG2 algorithm has no arc, and there are relatively few turning points. However, the planned path will also pass between two close obstacles, which is not practical.

Figure 12(a) is planned by the method in this paper. It can be seen that the planned path has no redundant turning points, and the path meets the actual sailing requirements of the ship and is highly maneuverable. Compared with the DDPG algorithm, although the length of the method in this

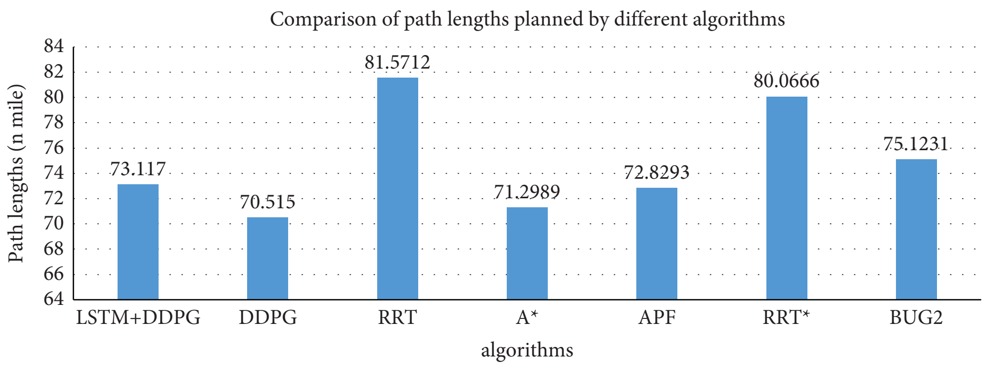


Figure 13: Path lengths planned by different algorithms.

paper is longer, the planned path does not pass through the multiple obstacles, which ensures the safety of the ship during navigation. Compared with the RRT algorithm, the number of turning points in the path planned by the method in this paper is much smaller than that of the RRT algorithm, and the turning points in the path increase the risk of the ship during navigation and reduce the economic benefits of the ship. In the same way, the path planned by the APF method has arcs; during the actual navigation of the ship, too many arcs of the path will greatly reduce the economic benefits of the ship. Compared with the method in this paper, the BUG2 algorithm, and the A* algorithm, the planned path passes through the position closer to the obstacle, which increases the risk of the actual navigation of the ship. Compared with the method in this paper, the RRT* algorithm has a larger number of path turning points than the path planned in this paper, so the economic benefits are lower.

In order to further compare the different paths planned by different algorithms, this paper further compares the length of the planned path and the number of corners. Figure 13 and 14 represent the comparison of the length of the path planned by different algorithms and the comparison of turning points number of the paths planned by different algorithms.

From Figures 13 and 14, the used DDPG algorithm, A* algorithm, APF algorithm, the algorithm proposed in this paper, and the path planned by the BUG2 algorithm are sorted according to the length from small to large, and their values correspond to 70.515n miles, 71.2989n miles, 72.8292n miles, 73.117n. miles, and 75.1231n miles. It can be seen that the path lengths planned by these algorithms are relatively close, and the difference is not very large. The longest path planned by the RRT algorithm is 81.5172n miles, followed by the RRT* algorithm at 80.066n miles. Compared with the previous algorithms, the length of the path planned by these two algorithms is relatively long. From the comparison of the number of corners, the number of turning points in the path planned by the algorithm proposed in this paper is the least of one, followed by RRT*, BUG2 algorithm, DDPG, and A* algorithm. Furthermore, the path planned by the APF algorithm and RRT algorithm with radian, leading to the number of turning points, cannot be calculated, so the number of corners of the APF algorithm and the RRT algorithm in Figure 14 is marked as n. Through these comparisons, we can see that, compared with the above algorithms, the planned path is more practical, economical, and safer.

It can also be seen from Figures 13 and 14 that the path length planned by the method in this paper is 73.117 n miles, and the number of turning points is one. Although in terms of length, the method in this paper is longer than the DDPG, A*, and APF algorithms, the safety of the path planned by these three algorithms is insufficient, and the actual ship operability is poor. In terms of the number of turning points, the path planned by this method has fewer turning points than the path planned by the above six methods. On the whole, the path planned by the method in this paper has stronger safety, higher economic benefits, and stronger actual ship operability.

At the same time, this paper further compares the time of path planning with the above algorithms, as shown in Figure 15. It can be seen from the figure that the shortest time for the RRT algorithm to plan a path is 0.6432s. This is because the RRT algorithm randomly selects points for path planning and ends as soon as the target point is reached, without considering issues such as path length and the number of turning points. The algorithm in this paper takes 0.7845s, the DDPG algorithm takes 0.9553s, the A* algorithm takes 1.324s, the RRT* algorithm takes 1.0234s, and the APF algorithm takes 2.341s. The longest time for the BUG2 method is 4.3256s; this is because the BUG2 algorithm will walk around the obstacle when it encounters an obstacle and then determine which direction to plan from, so it takes a long time.

To further illustrate the generality of the algorithm in this paper, the algorithm is verified in different environments. Figure 16 shows two environments with different levels of complexity. The obstacles in environment 1 are the same in size, and the obstacles in environment 2 are different in size. Both environments use the algorithm in this paper for path planning and path optimization, and the effects are shown in Figure 17.

As seen from Figure 17, the path planned by this paper algorithm in environment 1 has only one waypoint, and environment 2 is relatively complicated, so there are two waypoints in the planned path. There is no case that the paths planned by the two environments are close to or

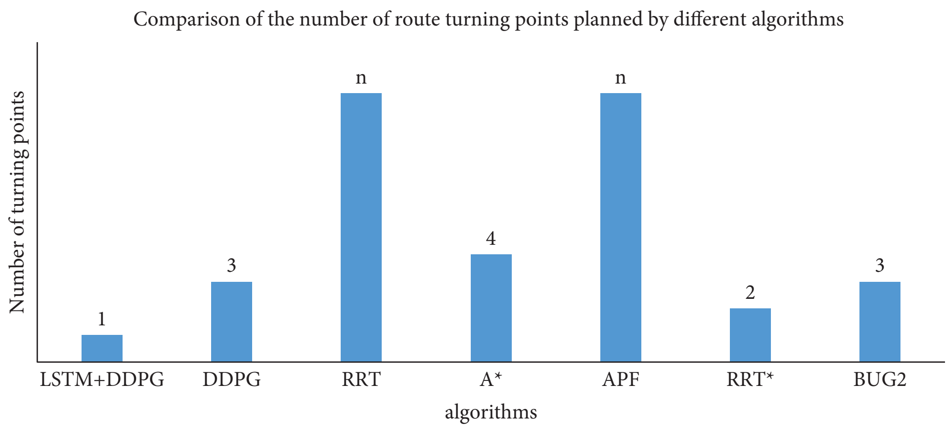


Figure 14: The number of path turning points planned by different algorithms.

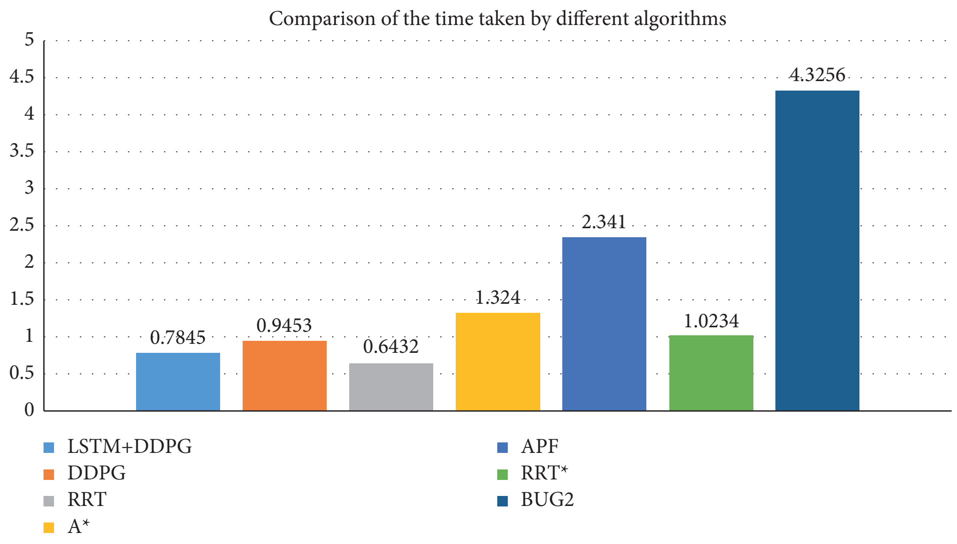


Figure 15: Comparison of the time taken by different algorithms.

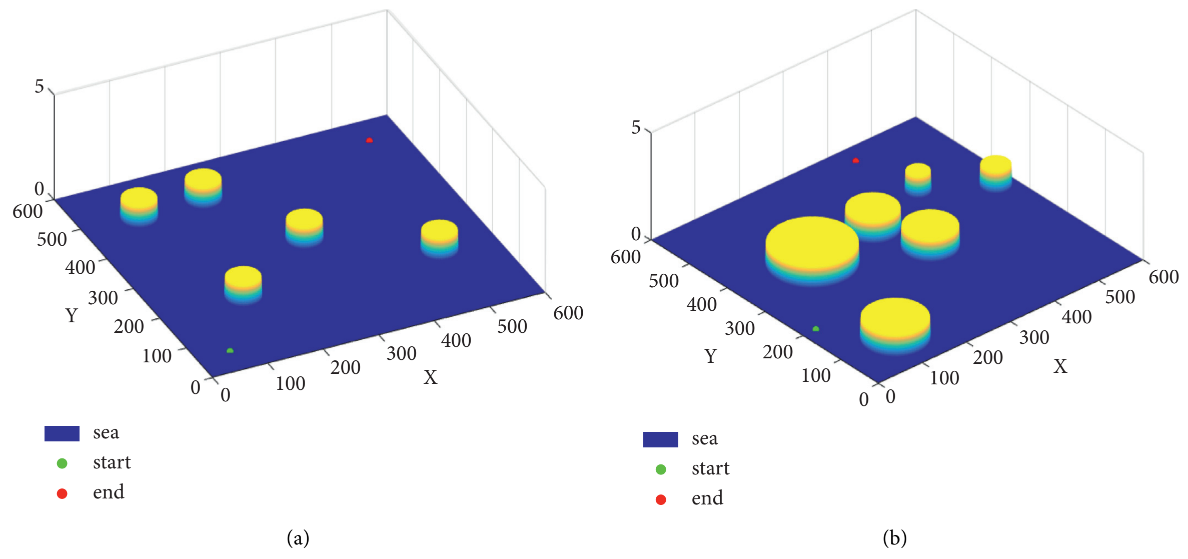
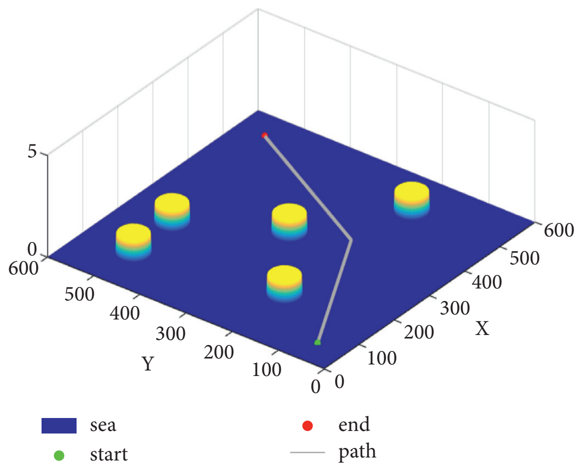
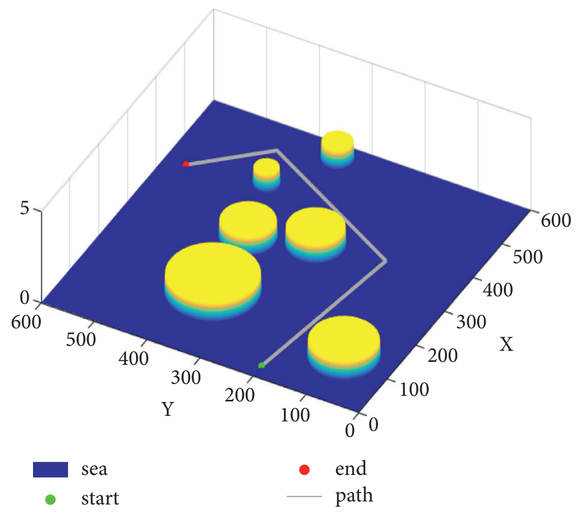


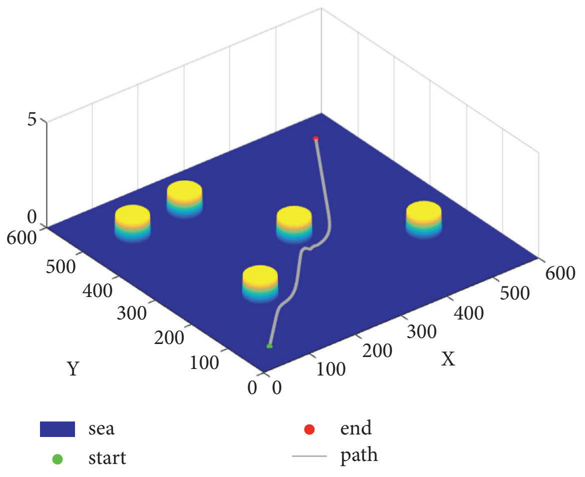
Figure 16: Environments with different levels of complexity. (a) Environment 1. (b) Environment 2.



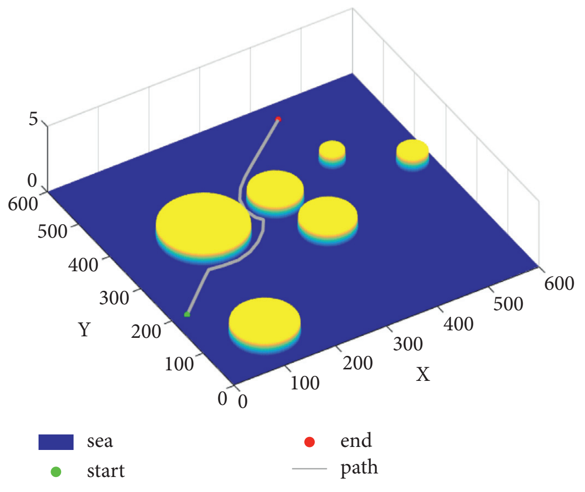
(a)



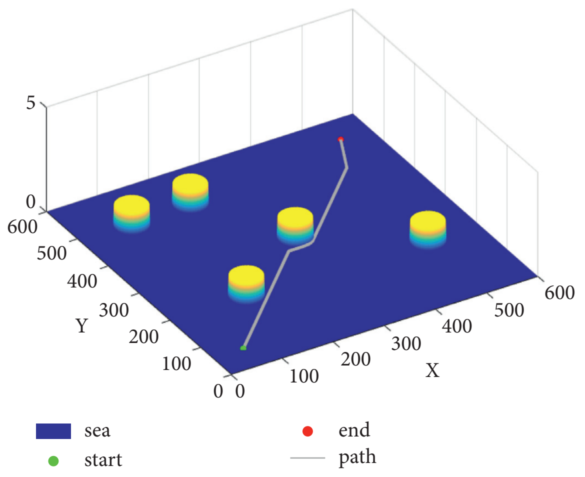
(b)



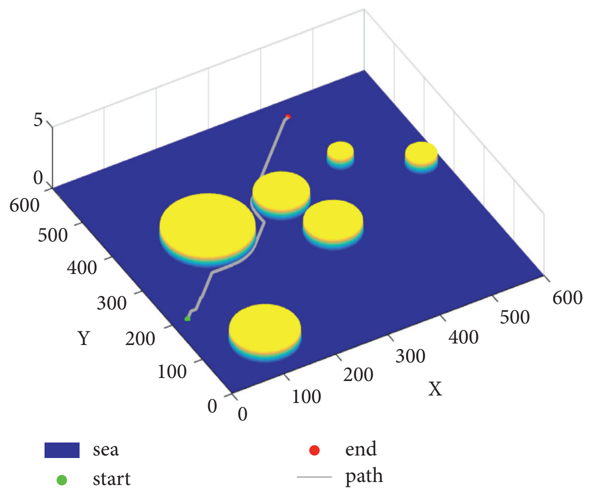
(c)



(d)

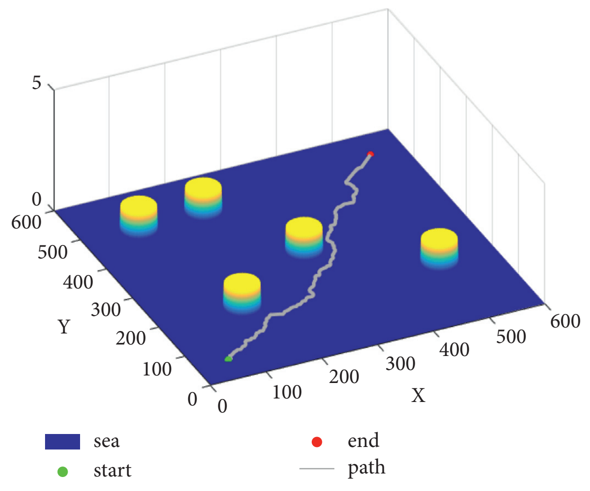


(e)

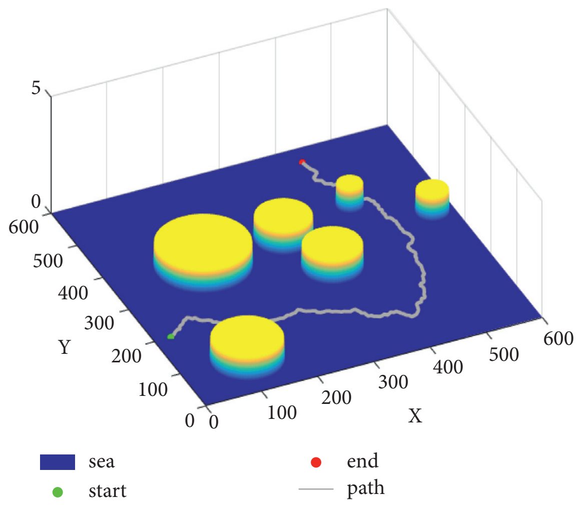


(f)

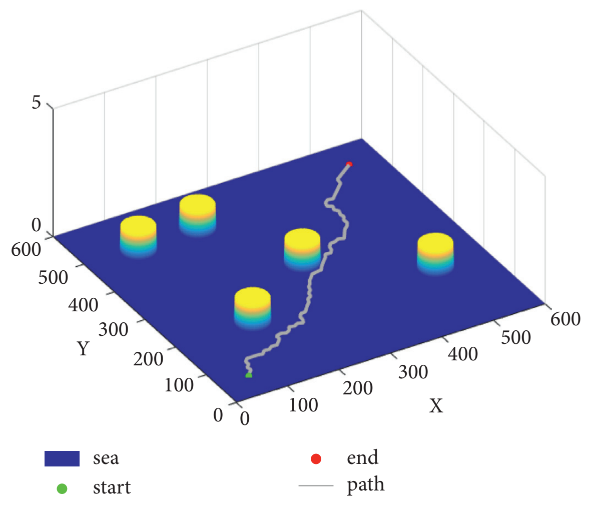
Figure 17: Continued.



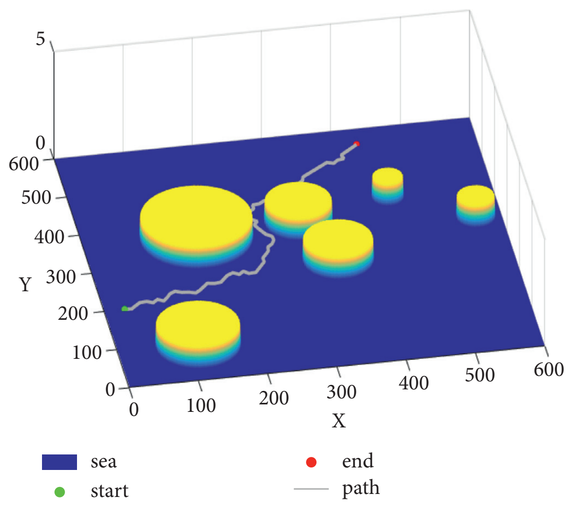
(g)



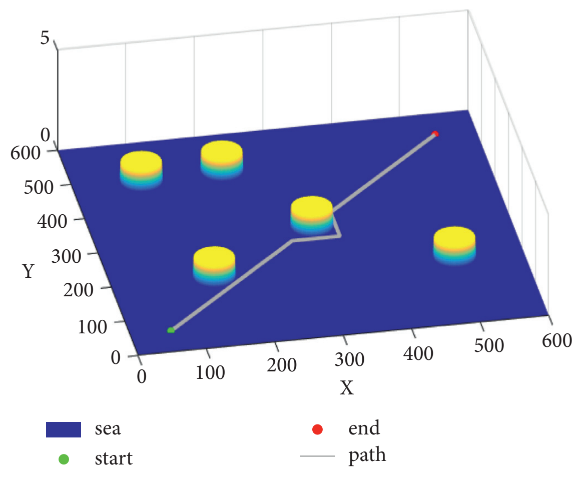
(h)



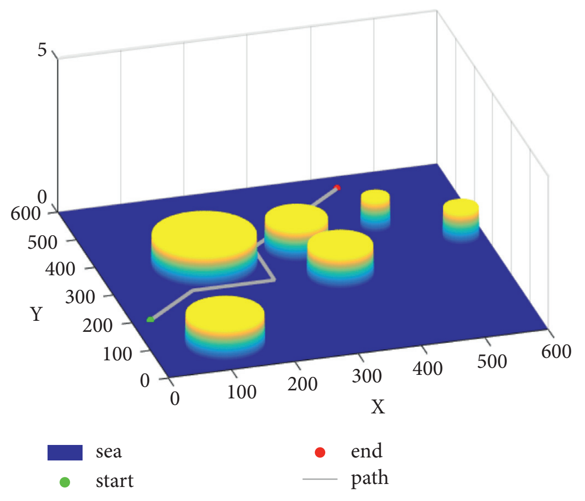
(i)



(j)



(k)



(l)

Figure 17: Paths planned for different environments. (a) The path planned by this paper algorithm in environment 1. (b) The path planned by this paper algorithm in environment 2. (c) The path planned by APF algorithm in environment 1. (d) The path planned by APF algorithm in environment 2. (e) The path planned by A* algorithm in environment 1. (f) The path planned by A* algorithm in environment 2. (g) The path planned by RRT algorithm in environment 1. (h) The path planned by RRT algorithm in environment 2. (i) The path planned by RRT* algorithm in environment 1. (j) The path planned by RRT* algorithm in environment 2. (k) The path planned by BUG2 algorithm in environment 1. (l) The path planned by BUG2 algorithm in environment 2.

through the obstacles, which is in line with the actual navigation specifications of ships.

6. Conclusions

In order to improve the safety and economy of coastal ship path planning, this paper proposes a coastal ship path planning method based on improved DDPG and DP. This method realizes the path planning of ships through improved DDPG and optimized reward function. Compared with traditional DDPG, this method improves the convergence speed of the algorithm and the utilization of data. In addition, the improved DP algorithm is used to further optimize the planned path, which solves the problem that there may be more inflection points in the planned path, making the ship's navigation safer and more economical, and the planned path is more in line with the actual sailing requirements of the ship. Experimental comparison with other path planning algorithms and verification results in different environments show that the path planned by the method proposed in this research has obvious advantages in terms of path length and number of inflection points.

However, the method in this paper still cannot solve the following problems, and these problems are also the key parts that need to be studied in the next part of this paper:

(1) This paper cannot deal with the situation of dynamic obstacles in the sea at present. This part is the next research work.

(2) This paper cannot solve the problem when the ship encounters another ship during the voyage, that is, the collision avoidance operation needs to be combined with the collision avoidance rules, which is another work in the next research.

Data Availability

All the data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant no. 2018YFB1601502), the Liao Ning Revitalization Talents Program (Grant no. XLYC1902071), and the Fundamental Research Funds for the Central Universities (Grant no. 3132019313).

References

[1] K. Wróbel, J. Montewka, and P. Kujala, "Towards the assessment of potential impact of unmanned vessels on maritime transportation safety," *Reliability Engineering & System Safety*, vol. 165, pp. 155–169, 2017.

[2] T. Wang, Q. Wu, J. Zhang, B. Wu, and Y. Wang, "Autonomous decision-making scheme for multi-ship collision avoidance with iterative observation and inference," *Ocean Engineering*, vol. 197, Article ID 106873, 2020.

[3] S. Wang, Y. Zhang, and L. Li, "A collision avoidance decision-making system for autonomous ship based on modified velocity obstacle method," *Ocean Engineering*, vol. 215, Article ID 107910, 2020.

[4] Y. Guo, Q. Pan, Q. Sun, C. Zhao, D. Wang, and M. Feng, "Cooperative game-based multi-agent path planning with obstacle avoidance," in *Proceedings of the 2019 IEEE 28th International Symposium on Industrial Electronics (ISIE)*, pp. 1385–1390, Vancouver, Canada, June 2019.

[5] Z. Liu, Y. Zhang, X. Yu, and C. Yuan, "Unmanned surface vehicles: an overview of developments and challenges," *Annual Reviews in Control*, vol. 41, pp. 71–93, 2016.

[6] H. Shen, C. Guo, T. Li, and Y. Yu, "An intelligent collision avoidance and navigation approach of unmanned surface vessel considering navigation experience and rules," *Journal of Harbin Engineering University*, vol. 39, pp. 998–1005, 2018.

[7] B. C. Shah and S. K. Gupta, "Long-distance path planning for unmanned surface vehicles in complex marine environment," *IEEE Journal of Oceanic Engineering*, vol. 45, no. 3, pp. 813–830, 2020.

[8] S. Wang, Y. Zhang, and Y. Zheng, "Multi-ship encounter situation adaptive understanding by individual navigation intention inference," *Ocean Engineering*, vol. 237, Article ID 109612, 2021.

[9] Y. Zheng, X. Zhang, Z. Shang, S. Guo, and Y. Du, "A decision-making method for ship collision avoidance based on improved cultural particle swarm," *Journal of Advanced Transportation*, vol. 2021, Article ID 8898507, 31 pages, 2021.

[10] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] Q. Liu, J. Zhai, Z. Zhang et al., "A survey on deep reinforcement learning," *Chinese Journal of Computers*, vol. 41, pp. 1–27, 2018.

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

[13] C. Chen, X.-Q. Chen, F. Ma, X.-J. Zeng, and J. Wang, "A knowledge-free path planning approach for smart ships based on reinforcement learning," *Ocean Engineering*, vol. 189, Article ID 106299, 2019.

[14] P. Bhopale, F. Kazi, and N. Singh, "Reinforcement learning based obstacle avoidance for autonomous underwater vehicle," *Journal of Marine Science and Application*, vol. 18, no. 2, pp. 228–238, 2019.

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

[16] S. Guo, X. Zhang, Y. Zheng, and Y. Du, "An autonomous path planning model for unmanned ships based on deep reinforcement learning," *Sensors*, vol. 20, no. 2, p. 426, 2020.

[17] X. Cao, C. Sun, and M. Yan, "Target search control of AUV in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96549–96559, 2019.

[18] U. Orozco-Rosas, K. Picos, and O. Montiel, "Hybrid path planning algorithm based on membrane pseudo-bacterial potential field for autonomous mobile robots," *IEEE Access*, vol. 7, pp. 156787–156803, 2019.

[19] U. Orozco-Rosas, O. Montiel, and R. Sepúlveda, "Mobile robot path planning using membrane evolutionary artificial

potential field," *Applied Soft Computing*, vol. 77, pp. 236–251, 2019.

[20] M. Zhu, X. Wang, and Y. Wang, "Human-like autonomous car-following model with deep reinforcement learning," *Transportation Research Part C: Emerging Technologies*, vol. 97, pp. 348–368, 2018.

[21] G. Xiong, L. Niu, and Y. Tian, "Path planning system for smart cars used in education[C]//2020 15th IEEE conference on industrial electronics and applications (ICIEA). IEEE," pp. 229–234, 2020.

[22] F. Gao, H. Zhou, and Z. Yang, "Global path planning for surface unmanned ships based on improved A* algorithm," *Application Research of Computers*, vol. 37, 2020.

[23] P. Chen, Y. Huang, E. Papadimitriou, J. Mou, and P. van Gelder, "Global path planning for autonomous ship: a hybrid approach of fast marching square and velocity obstacles methods," *Ocean Engineering*, vol. 214, Article ID 107793, 2020.

[24] X. Fan, Y. Guo, H. Liu, B. Wei, and W. Lyu, "Improved artificial potential field method applied for AUV path planning," *Mathematical Problems in Engineering*, vol. 2020, Article ID 6523158, 21 pages, 2020.

[25] T. Wang, X. P. Yan, Y. Wang, and Q. Wu, "Ship domain model for multi-ship collision avoidance decision-making with COLREGs based on artificial potential field," *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, vol. 11, no. 1, pp. 85–92, 2017.

[26] J. Xiang, H. Wang, Z. Ouyang, and H. Yi, "Research on local path planning algorithm of unmanned boat based on improved Bidirectional RRT," *Shipbuilding of China*, vol. 61, pp. 157–166, 2020.

[27] S. Lavalle, "Rapidly-exploring random trees : a new tool for path planning," *Research Report*, vol. 98, pp. 293–308, 1998.

[28] W. Tao, S. Yan, F. Pan, and G. Li, "AUV path planning based on improved genetic algorithm," in *Proceedings of the 2020 5th International Conference on Automation, Control and Robotics Engineering (CACRE)*, pp. 195–199, Dalian, China, September 2020.

[29] J. Jiang, X. Yao, E. Yang, and J. Mehnen, "An improved adaptive genetic algorithm for mobile robot path planning analogous to TSP with constraints on city priorities," in *Proceedings of the IEEE World Congress on Computational Intelligence 2020*, pp. 19–24, Glasgow, UK, July 2020.

[30] F. Ding, Z. Zhang, M. Fu, Y. Wang, and C. Wang, "Energy-efficient path planning and control approach of USV based on particle swarm optimization," in *Proceedings of the Conference on OCEANS MTS/IEEE Charleston*, pp. 1–6, Charleston, SC, USA, October 2018.

[31] A. Lazarowska, "Ship's trajectory planning for collision avoidance at sea based on ant colony optimisation," *Journal of Navigation*, vol. 68, no. 2, pp. 291–307, 2015.

[32] S. Xie, X. Chu, M. Zheng, and C. Liu, "Ship predictive collision avoidance method based on an improved beetle antennae search algorithm," *Ocean Engineering*, vol. 192, Article ID 106542, 2019.

[33] D. Zhao, K. Shao, Y. Zhu, D. Li, and Y. Chen, "Review of deep reinforcement learning and discussion on the development of computer Go," *Control Theory & Applications*, vol. 33, pp. 701–717, 2016.

[34] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[35] R. Zhang, P. Tang, Y. Su, X. Li, G. Yang, and C. Shi, "An adaptive obstacle avoidance algorithm for unmanned surface vehicle in complicated marine environments," *IEEE/CAA Journal of Automatica Sinica*, vol. 1, no. 4, pp. 385–396, 2014.

[36] Z. Zhou, X. He, L. Xu, and C. Qu, "USV path planning and simulation based on deep reinforcement learning," in *Proceedings of the 2020 China Simulation Conference*, pp. 27–29, Beijing, China, November 2020.

[37] L. Huang, H. Qu, M. Fu, and W. Deng, "Reinforcement learning for mobile robot obstacle avoidance under dynamic environments," in *Proceedings of the 15th Pacific Rim International Conference on Artificial Intelligence (PRICAI)/15th Pacific Rim Knowledge Acquisition Workshop (PKAW)*, pp. 441–453, Nanjing, China, August 2018.

[38] H. Nicolas, J. Jonathan, P. Timothy, and S. David, "Memory-based control with recurrent neural networks," 2015, https://arxiv.org/abs/1512.04455.

[39] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," https://arxiv.org/abs/1507.06527.

[40] S. Guo, X. Zhang, Y. Du, Y. Zheng, and Z. Cao, "Path planning of coastal ships based on optimized DQN reward function," *Journal of Marine Science and Engineering*, vol. 9, no. 2, p. 210, 2021.

[41] D. Zhang and X. Zhang, "Compression algorithm of GPS trajectory data based on the temporal characteristics," *Journal of Transport Information and Safety*, vol. 3, pp. 6–9, 2013.

[42] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, and S. S. Ravi, "Compression of trajectory data: a comprehensive evaluation and new approach," *GeoInformatica*, vol. 18, no. 3, pp. 435–460, 2014.

[43] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.