Hua *et al. Intell Robot* 2024;4(4):363-84
DOI: 10.20517/ir.2024.22

Intelligence & Robotics

Research Article

Open Access

Improved DDPG algorithm-based path planning for unmanned surface vehicles

Menglong Hua, Weixiang Zhou, Hongying Cheng, Zihao Chen

College of Information Engineering, Shanghai Maritime University, Shanghai 200000, China.

Correspondence to: Dr. Weixiang Zhou, College of Information Engineering, Shanghai Maritime University, No. 1550, Harbor Avenue, Pudong New Area, Shanghai 200000, China. E-mail: zhouwx@shmtu.edu.cn

How to cite this article: Hua M, Zhou W, Cheng H, Chen Z. Improved DDPG algorithm-based path planning for unmanned surface vehicles. *Intell Robot* 2024;4(4):363-84. http://dx.doi.org/10.20517/ir.2024.22

Received: 7 Jul 2024 **First Decision:** 26 Aug 2024 **Revised:** 31 Oct 2024 **Accepted:** 2 Nov 2024 **Published:** 13 Nov 2024

Academic Editor: Simon Yang **Copy Editor:** Pei-Yun Wang **Production Editor:** Pei-Yun Wang

Abstract

As a promising mode of water transportation, unmanned surface vehicles (USVs) are used in various fields owing to their small size, high flexibility, favorable price, and other advantages. Traditional navigation algorithms are affected by various path planning issues. To address the limitations of the traditional deep deterministic policy gradient (DDPG) algorithm, namely slow convergence speed and sparse reward and punishment functions, we proposed an improved DDPG algorithm for USV path planning. First, the principle and workflow of the DDPG deep reinforcement learning (DRL) algorithm are described. Second, the improved method (based on the USVs kinematic model) is proposed, and a continuous state and action space is designed. The reward and punishment function are improved, and the principle of collision avoidance at sea is introduced. Dynamic region restriction is added, distant obstacles in the state space are ignored, and the nearby obstacles are observed to reduce the number of algorithm iterations and save computational resources. The introduction of a multi-intelligence approach combined with a prioritized experience replay mechanism accelerates algorithm convergence, thereby increasing the efficiency and robustness of training. Finally, through a combination of theory and simulation, the DDPG DRL is explored for USV obstacle avoidance and optimal path planning.

Keywords: DDPG deep reinforcement learning, unmanned surface vehicles, obstacle avoidance, path planning

1. INTRODUCTION

An unmanned surface vehicle (USV) is a vessel that can autonomously navigate on the surface of water and perform various tasks. Research on USVs is of great significance for marine development and monitoring because it can effectively improve efficiency, reduce risks and costs, and support the rational utilization and protection of marine resources. As a type of water transportation with broad application prospects, unmanned vessels have attracted considerable attention from researchers and industries, and they are used in various professional domains[1]. These vehicles are capable of performing various tasks in the marine environment, such as patrol security, anti-submarine warfare, marine survey, underwater detection, and rescue operations[2]. The study of USVs can aid in understanding the marine environment and resource allocation, as these vehicles can perform tasks at sea for a long time without being limited by factors such as personnel fatigue and safety. This method can also be used to perform tasks instead of personnel in rescue, reducing the risk of casualties. It is more advantageous in terms of operation and maintenance costs and does not require personnel to drive, which reduces training costs. It is favored for its efficiency, safety, and low cost. Research on path planning can help USVs to avoid obstacles, optimize navigation routes, improve efficiency, and ensure vessel safety[3]. In the dynamic and complex water environment, USVs have to travel to destinations autonomously, and therefore, they have more rigorous requirements in terms of route planning and route adjustment depending on the actual situation[4]. Traditional route planning methods have limitations and cannot adapt to real-time changes and uncertainties in complex environments. Therefore, determining how to use reinforcement learning algorithms to realize autonomous obstacle avoidance path planning has become a topic of focus in current research. For example, Su *et al.* proposed multi-intelligence reinforcement learning and used it in unmanned ground vehicles to capture escaping targets[5]. Moreover, they utilized the soft actor-critic (SAC) algorithm and introduced an attention mechanism to prevent collision of the capturing vehicle. Additionally, they proposed a reward function strategy by combining individual and collaborative rewards. In this context, the capturing vehicle was deemed the obstacle, the escaping vehicle was deemed the destination, and path planning was performed to reach the goal point. For example, Feng *et al.* proposed reinforcement learning to solve the UAV navigation problem[6]. The UAV navigation task is modeled as a Markov decision process (MDP) with parameterized actions. In addition, the sparse reward problem is also taken into account. To address these issues, develop the HER-MPDQN by combining multi-pass deep Q-network (MP-DQN) and hindsight experience replay (HER).

Many traditional path planning algorithms have been proposed, such as the A* algorithm for global path planning, which can find solutions in a limited map space. Yu *et al.* combined a speed barrier with A* calculation, unlike the traditional A* algorithm, to propose a new collision-proof path planning method that limits the risk of being trapped in a local minimum[7]. Zhang *et al.* proposed to reduce the number of turning points by increasing the cost function of the turning points to smooth a path[8]. These approaches represent significant improvements over the traditional A* algorithm, but their memory consumption is excessive when dealing with large-scale problems, and the search needs to be re-trained if the obstacles change. Yu *et al.* proposed the improved D*lite algorithm with an enhanced path cost function to reduce the range of nodes and shorten the path length by using the inverse distance weighted interpolation approach[9]. Jin *et al.* combined conflict-based search algorithms for path planning in the presence of map transformations[10]. The approaches proposed in the literature[9,10] are a significant improvement over traditional D*lite, D*lite is based on discrete grids, and therefore, continuous space processing needs to be optimized. If a large number of obstacles are encountered, it will fall into local optimization. For example, the dynamic window approach (DWA) is associated with local path planning algorithms. Zhang *et al.* proposed vector field adaptive DWA to clarify the obstacle information in all directions[11]. This method differs from the traditional DWA algorithm, which improves the evaluation function, solves the DWA route bypass problem, and provides the maximum acceleration to limit dynamic performance. Cao *et al.* designed velocity transformation and safe distance evaluation coefficients by considering the velocity and direction of movement between a robot and an obstacle[12]. They modified the DWA algorithm, which is computationally less intensive and offers real-time capabilities. However, in more complex environments, DWA falls into local minima points, and the local path planning method is more re-

stricted in other environments. In summary, the traditional global and local path planning algorithms lack generalization ability for new environments, tend to fall into local optima, and consume excessive memory. In addition, they have insufficient ability to handle continuous space.

Reinforcement learning has been considered widely by scholars at home and abroad since it was proposed, and it is the current research hotspot in relation to path planning. The algorithms proposed thus far include Q-learning, deep Q-network (DQN), and deep deterministic policy gradient (DDPG). Zhou *et al.* introduced a Q-table initialization method by incorporating a new action-selection policy and a new reward function, in addition to adapting the root mean square propagation (RMSprop) method for learning rate adjustment[13]. Wang *et al.* combined a radial basis function (RBF) neural network with the Q-learning algorithm for function approximation to improve the convergence speed of the algorithm from literature[14]. In addition, they redesigned the reward function to take into account the USV's heading angle and turning performance and proposed a safety threshold. The improved Q-learning algorithm converged faster and planned better paths, but the computational cost of the approach was higher for high-dimensional state spaces. Moreover, when using function approximation, it may face problems such as overfitting and gradient explosion. Yu *et al.* defined the state and action spaces in the double DQN architecture, in addition to designing a reward function to evaluate and guide model training[15]. The DQN algorithm is an extension of Q-learning that uses deep neural networks to approximate the Q-function and handle complex, high-dimensional state spaces. Although these approaches largely improved the traditional DQN algorithm, they were adapted to discrete spaces and required increased computational effort for continuous actions. Moreover, the improved approaches were more sensitive to hyperparameters. Therefore, we adopt the DDPG algorithm, which can deal with higher-dimensional state spaces and continuous action spaces, instead of the previously described reinforcement learning algorithm. The DDPG algorithm is stable, and the inclusion of two additional target networks helps reduce jitter in the training process and improves stability. Liu *et al.* and Zhou *et al.* proposed multi-intelligence reinforcement learning path planning, and their results indicated that multi-intelligence can improve system stability in complex environments to a greater extent than single intelligence[16,17]. This shortens the training time and increases computational efficiency. Liu *et al.* proposed multi-intelligence and combinatorial-prioritized experience replay mechanisms to solve the problem of inefficient experience reuse[16]. However, when using a large number of intelligences, the coordination and cooperation between them must be considered, which will increase the computational burden. Moreover, a large amount of space to store the experiences of these intelligences in large-scale environments. Zhou *et al.* proposed the novel task decomposed multi-agent twin delayed deep deterministic policy gradient (TD-MATD3) and redesigned the reward function achieving good results in complex dynamic environments, it still did not completely solve the problem of heavy computational burden of multi-intelligence agent, and it did not involve the International Regulations for Preventing Collisions at Sea (COLREGs), did not introduce constraints, and did not conform to the actual navigation situation[17]. Wu *et al.* proposed combines the original sparse reward de-coupling with an artificial potential field method to design new reward and punishment functions to solve the problem of reward sparsity and improve algorithm efficiency[18]. However, the problem of sensitive parameter settings remained unsolved, and in complex environments, the artificial potential field may not be adjusted efficiently and would be extremely time-consuming with a high probability of falling into a local optimum. Chen *et al.* attempted to reduce Q-value overestimation and enhance the ability of an agent to explore the global optimum[19]. A warm-up stage was introduced to improve stability at the beginning of training and accelerate the convergence speed of training. However, excessive reliance on the warm-up phase, which requires substantial debugging to find the right value in more complex environments, can increase the training time. Moreover, reducing Q-value overestimation can lead to local optimality in complex environments. Feng *et al.* introduced intrinsic motivation and added HER to the DDPG algorithm to improve its exploration ability[20]. They used the beluga whale optimization algorithm for hyperparameter optimization to improve the algorithm training process. However, the combination of intrinsic motivation and HER increases the computation burden, which can lead to over-exploration, and HER needs to select the appropriate replay target, failing which the learning rate can decrease. Yang *et al.*

proposed the use of the deep reinforcement learning (DRL) and velocity obstacle (VO) methods, introduced the principle of COLREGs, and added the VO algorithm to the reward function to avoid collisions based on the sailing speed[21]. In addition, they solved the problem of sparse rewards by using the fuzzy theory of computation and introduced the navigational influence factor (NIF) to construct the state space. Although the introduction of the speed barrier method is more consistent with the actual navigation situation, application of the fuzzy theory may lead to uncertainty in the decision-making process, and the influence factor increases the complexity of the state space and decreases the processing speed. Although the abovementioned studies largely improved the DDPG algorithm, a few shortcomings persist. For example, the reward design is overly complicated; the warm-up phase has been introduced, but a balance between exploration and utilization is yet to be achieved; and coordinating the emergence of multiple intelligences needs to be realized. In addition, the improved algorithm converges faster, but it is more complex, which leads to other problems.

According to the literature, this paper is integrated to provide different opinions. The DDPG reinforcement learning algorithm improves the combinatorial reward function, sets the dynamic region restriction, uses dual intelligences combined with a prioritized experience replay mechanism, sets the action space by considering the actual situation of USV navigation, and incorporates the COLREGs principle.

The contributions of this paper can be summarized as follows:
1. A DDPG reinforcement learning framework based on actor critics is proposed. This framework uses the properties of the DDPG algorithm and is applied to a continuous action space. Considering the problem of reward sparsity, a new continuous reward function is designed to improve the motivation of intelligent bodies, such that there exists a corresponding reward or punishment for each step. The convergence speed of the algorithm is improved.
2. To save computational resources, we employ the dynamic region restriction method, which is designed to let an intelligent agent ignore faraway obstacles and retain only a small number of obstacles around itself to reduce the computational burden.
3. Dual intelligences are combined with the prioritized experience replay mechanism to allow the intelligences to share their experiences with each other, which greatly reduces the training time, increases the overall planning efficiency, and improves the learning speed. Dual intelligences help avoid the scenario in which a large number of intelligences increases the computational burden, and yet, they are more efficient than a single intelligence.
4. To ensure greater consistency with the actual USV navigation situation, we set up the action space to prevent USV movement from being in line with actual large turns, such as 180 degrees, per the COLREGs principle. Doing so enhances the safety and optimization of navigation routes.

The remainder of this paper is summarized as follows: Section 1 describes the working principle of the DDPG reinforcement learning algorithm, and Section 2 describes how we improve the DDPG algorithm, with each subsection explaining individual improvements. In Section 3, we describe experimental simulations conducted using different-sized maps and obstacles to demonstrate algorithm feasibility and compare the proposed approach to traditional DDPG algorithms, namely deep Q-network (DON) and combined prioritized experience replay with multi-intelligent depth deterministic policy gradient (CPER-MADDPG). In the final section, the future outlook is summarized.

2. DDPG DRL ALGORITHMS

The DDPG algorithm entails reinforcement learning, which can be represented by Markov decision making. This algorithm mainly involves an intelligent agent, an environment, states, rewards, and punishments. The intelligent agent obtains a state in the current environment, and based on this state, it performs the next action and obtains a reward or punishment value. The agent interacts continuously with the environment to obtain

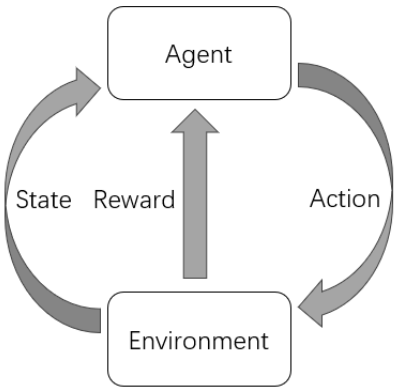

Figure 1. Basic framework of reinforcement learning.

as many rewards as possible. The basic framework is illustrated in Figure 1.

The DDPG algorithm has prominent advantages over other reinforcement learning algorithms. It can handle complex inputs such as high-dimensional state spaces, and it can use images as a function approximator through neural networks. It is more capable at handling continuous action space problems, and it uses a deterministic policy that outputs continuous actions, which helps improve the convergence of continuous action space. The DDPG algorithm solves the continuous action problem, drawing mainly on the methods of DQNs, namely experience replay and goal networks. Moreover, the DDPG algorithm adds a policy network on top of DQNs to output actions directly, which is different from other reinforcement learning algorithms. The policy network and the Q-network are essentially actor-critic (AC) structures. The policy network performs as an actor, the Q-network scores each action of the actor, and the policy network adjusts its policy continuously according to the scores of the Q-network; that is, it updates the parameters of the neural network.

The optimal strategy for a DQN is to learn a very good Q-network that will hopefully pick an action to maximize the Q-value. The actors cater to the critics, and therefore, the gradient of the optimal strategy network aims to maximize the Q-value, whereas the loss function aims to cause Q to take a negative sign, as given in

$$loss = -Q_{\omega}(s, a), \tag{1}$$

where variable ω represents the parameters of the Q-network, s denotes the current state, and a denotes the current action. We optimize the Q-network in the same way as we optimize the DQN, that is, by using $r + \gamma Q_{\bar{\omega}}(s', a')$ to fit the future reward Q_target and letting the output of the Q-network $Q_{\omega}(s, a)$ approximate the Q_target, such that the loss function of the optimized Q-network can be used to determine the mean-square deviation of the two values, as determined by

$$loss = [Q_{\omega}(s, a), r + \gamma Q_{\bar{\omega}}(s', a')], \tag{2}$$

where r stands denotes real reward, γ denotes the discount factor, $\bar{\omega}$ denotes the $target\_Q$ network and $target\_P$ network, s' denotes the next state, and a' denotes the next action. However, DDPG draws on the deep Q network and will inevitably have some defects of the deep Q network. For instance, if the Q_target is unstable, the next $Q_{\bar{\omega}}(s', a')$ is unstable as well. $Q_{\bar{\omega}}(s', a')$ is also a prediction. To increase its stability, the DDPG algorithm builds target networks for the Q network and the policy network, respectively (the $target\_Q$ network and $target\_P$ policy network), as illustrated in Figure 2. The $target\_Q$ network is used to calculate the next $Q_{\bar{\omega}}(s', a')$ value, and the $target\_P$ network is used to output the next action $a' = \mu_{\bar{\theta}}(s')$. The next $Q_{\bar{\omega}}(s', a')$ value can be found only by using the next state and action. DDPG comprises four networks, and the

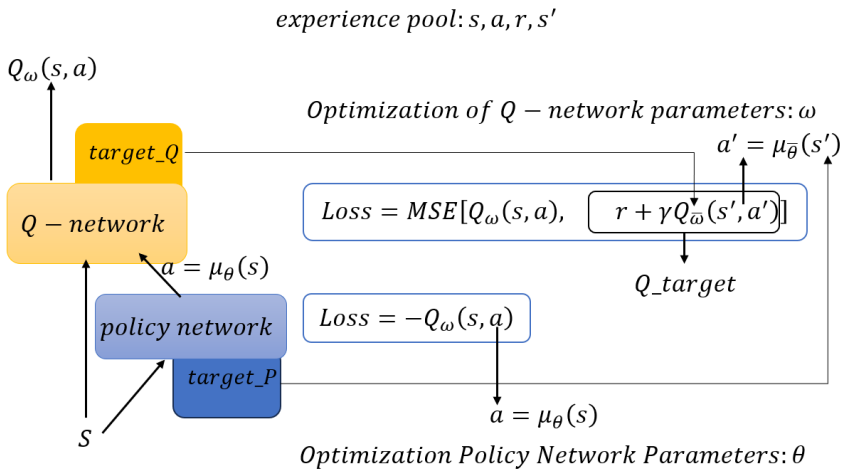

Figure 2. Basic structure of DDPG algorithm. DDPG: Deep deterministic policy gradient.

two additional networks are used to ensure that the calculation of Q_target is more stable. These two target networks fix the parameters for a certain period before synchronizing the latest parameters with the evaluation network, that is, ω for the Q network and θ for the strategy network, such that the target value fluctuates to a lesser extent and prevents the estimates from increasing. This fixation period is called the update interval.

The training data required by the DDPG algorithm include the current state, current action, reward and punishment value, and next state, which is stored using a playback buffer and is sampled and trained to achieve optimization. The basic structure of the DDPG algorithm is depicted in Figure 2.

3. IMPROVING THE DDPG DRL ALGORITHM

In this paper, we improve and optimize the original DDPG algorithm, and in this section, we focus on the state and action space, combined excitation function design, dynamic region restriction, prioritized empirical replay mechanism, multi-intelligence agent, and introduction of Gaussian noise.

3.1 USVs kinematics model

We present a schematic of the coordinate system of USVs in Figure 3. Then, the second-order kinematic equation of the USVs is established, as follows:

$$\left\{\begin{array}{c} \dot{v}_i = a_v \\ \dot{\omega}_i = a_\omega \\ \dot{x}_i = v_i \cos\psi_i \\ \dot{y}_i = v_i \sin\psi_i \\ \dot{\psi}_i = \omega_i \end{array}\right\}, \tag{3}$$

where v_i denotes speed, ψ_i denotes heading, ω_i is angular velocity, a_v is acceleration, and a_ω represents angular acceleration.

In this USV kinematics model, we use second-order kinematics equations that consider only the speed, heading, angular velocity, acceleration, and angular acceleration; that is, only the forward speed of the USV, range of the horizontal left and right directions, forward acceleration, and acceleration of the left and right steering are considered. The heading speed v_i and angular velocity ω_i are limited by the constraints $0 \leqslant v_i \leqslant v_{\max}$, $-\omega_{\max} \leqslant \omega_i \leqslant \omega_{\max}$. The specific settings are presented in 3.2.2 Action space. The setting range allows the USV to conform to the actual situation, which ensures that forward movement is smooth, turning action

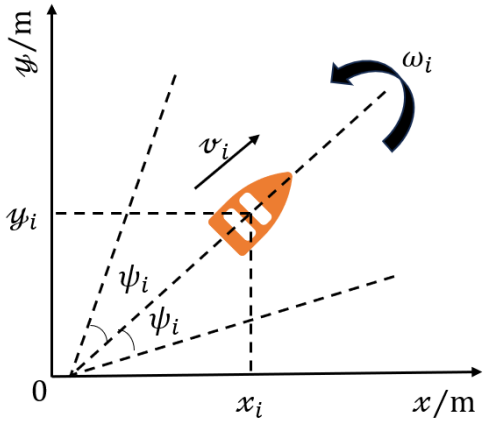

Figure 3. Schematic diagram of coordinate system of USV. USV: Unmanned surface vehicle.

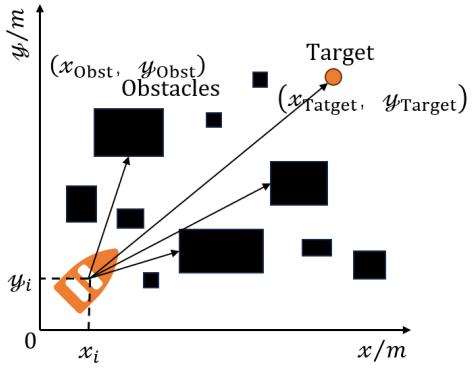

Figure 4. USV state space. USV: Unmanned surface vehicle.

is not too fast, and steering action amplitude is not too large, avoiding the situation of instant 180 turn. The proposed model is a relatively simple second-order equation that does not consider a few situations such as wind and waves. We hope to add more situations in a subsequent study.

3.2 Designing state and action spaces

3.2.1 State spaces

In the DRL process, intelligence is based on the state information received from the environment to determine what action to take next. Therefore, a reasonable state space must be designed. The state space should contain the current position of the unmanned ship, position of the obstacle, and target position. We use the plane right-angle coordinate system, that is, X-axis and Y-axis, to represent the positions of the USV, obstacle, and target by coordinates, which indicate the distance between the current USV position and the obstacle and target points. The state space of environmental observation information is presented in Figure 4.

3.2.2 Action spaces

The DDPG is used to solve the continuous action space problem, where the action space needs to be designed considering the possible actions that the USV may take to avoid obstacles and reach the target point. The USV can change its heading by controlling the rudder angle to avoid obstacles and reach the target point. The common continuous range for heading design is $[-90°, 90°]$. During training, if the steering angle is excessively large, the USV path may not be smooth, the training time may be long, and other problems may be encountered. In this paper, the designed heading range is $[-30°, 30°]$, continuous angular velocity range is $[-20°/s, 20°/s]$, and angular acceleration is set to $20°/s^2$. Meanwhile, the propulsion magnitude is adjusted

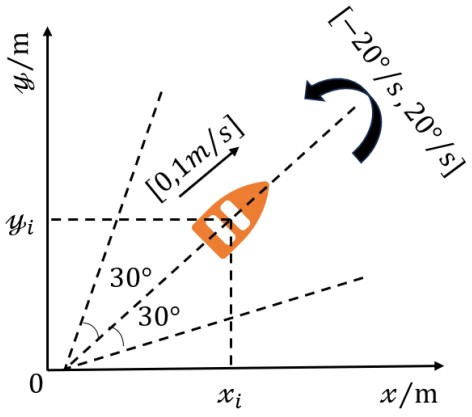

Figure 5. USV action space. USV: Unmanned surface vehicle.

to control the USV speed, which is adjusted in the obstacle avoidance process. The simulation map is a 20×20 grid constructed using Python, and therefore, the continuous propulsion speed range could be designed as $[0, 1m/s]$, acceleration is set to an output of $0.8m/s^2$ continuous action in DDPG, and a tanh function layer is added to the output layer. This function limits the output to $[-1, 1]$ and then adds, subtracts, and scales the output according to the actual action. The rudder angle and speed are combined to form a two-dimensional (2D) action space, and they can be adjusted simultaneously in the USV obstacle avoidance process to increase flexibility, as illustrated in Figure 5.

3.3 Improved combined reward and penalty functions

After designing the USV state and action spaces, we design the continuous reward and punishment functions based on the motion, position, obstacles, start point, and endpoint of the USV. The reward and punishment functions are used to evaluate the success of DRL decisions, which affect the convergence of the entire algorithm and play a decisive role in the whole process. The common DDPG reward and punishment functions are given as

$$R = \left\{ \begin{array}{l} -2\ , hit \\ 5, arrival \\ 0, other \end{array} \right\}, \tag{4}$$

where R denotes the overall reward. To solve the reward sparsity problem, we design a new continuous reward and punishment function to introduce the COLREGs principle, and we add the USV heading angle and maintain a safe distance. Therefore, the USV can obtain the corresponding reward and punishment each time it takes an action.

When the USV touches an obstacle, it is penalized r_1. When it reaches the target point, it is rewarded r_2. The closer the target point, the higher is the reward, and the distance between the USV and target point is computed as follows:

$$d_1 = \sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}, \tag{5}$$

where d_1 denotes the straight-line distance between the USV and the target point. The current position coordinates of the USV are (X_1, Y_1) and those of the target point are (X_2, Y_2). Then, the reward is designed using the following exponential function:

$$r_3 = te^{-d_1}, \tag{6}$$

where t is a positive constant set to 10. The reward is lower when d_1 is greater and higher when the distance is smaller until the value of the exponential function reaches t, which indicates arrival at the target point. Herein,

Table 1. Reward and punishment functions

Reward	Current status of USVs
$r_1 = 10$	Get to the target point
$r_2 = -5$	Hit an obstacle
$r_3 = te^{-d_1}$	Distance to target point
$r_4 = -1$	$0 \leqslant d_i \leqslant 1/6$
$r_5 = k\cos\theta$	Angle between forward direction and target point
0	Other

USVs: Unmanned surface vehicles.

the map designed using the distance formula is a 20×20 environment, and therefore, it is set to assign a penalty of r_4 when $0 \leqslant d_i \leqslant 1/6$, as follows:

$$d_i = \sqrt{(X_1 - X_i)^2 + (Y_1 - Y_i)^2}, \tag{7}$$

where i denotes each obstacle, (X_1, Y_1) denote the current position coordinates of the USV, and (X_i, Y_i) denote the position coordinates of the obstacle. The rewards are designed based on the cosine function, intercept 0 to $180°$. The cosine function 0 to $180°$ is a monotonically decreasing function. As the angle θ decreases, the USV moves toward the target point, and the reward increases. When the angle is greater than $180°$, a penalty is assigned (the greater the angle, the higher is the penalty). When the angle is $180°$, the USV is moving in the direction opposite to that of the target point. The reward and penalty functions are expressed as follows:

$$r_5 = k\cos\theta, \tag{8}$$

where k is a positive constant set to 5, θ denotes the angle between the direction of advance of the USV and the target point. The details are summarized in Table 1, and the final reward function is expressed as follows:

$$R = r_1 + r_2 + r_3 + r_4 + r_5 \tag{9}$$

3.4 Dynamic area restrictions

On top of the original foundation of state-space obstacles, we performed dynamic region restriction. The dynamic region limiting mechanism allows us to ignore the influence of those obstacles that are farther from the USV. Only the obstacles near the USV are observed, and the information of these obstacles is removed from the state space, so that the USV can explore little by little and adapt better to the complexity of the environment. State space tailoring is performed to adjust the state space according to the training process and changes in the environment around the USV at the current location. In this work, the USV is designed to observe five nearby obstacles, ignore other obstacles, reduce the number of algorithm iterations, and reduce computer resource usage, as illustrated in Figure 6.

3.5 Replay buffer

When using the DDPG algorithm for path planning, after completion of the design of the above sections, the design of the experience pool is considered, which largely determines the convergence speed of the algorithm and whether the planned path is shorter and smoother. The traditional replay buffer uses uniform random sampling batches, and consequently, certain experiences are not sampled at one time or are sampled multiple times. Intelligent agents collect samples with different degrees of importance, and to speed up the training of intelligent agents and increase training efficiency, all the experiences in the experience pool are sampled at least once by incorporating a prioritized experience replay mechanism, as follows:

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}, \tag{10}$$

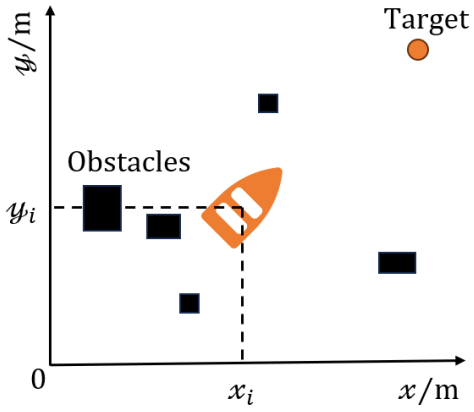

Figure 6. USVs observation space. USVs: Unmanned surface vehicles.

where $P(i)$ is the sampling probability of each experience, and p_i is the priority of the i experience. When Equation (2) is smaller, the better the strategy, the higher is the priority. The weight of the priority to control the degree of priority is α. When α is 0, the sampling is uniform. The total number of samples in the experience pool is represented by k. In priority sampling, uneven samples, which are biased, are utilized. To solve this problem, importance is assigned to sampling weights, as follows:

$$\omega_i = \left(\frac{1}{N}\frac{1}{p(i)}\right)^{\beta}. \tag{11}$$

where ω_i is the importance of sampling weights for each experience, N denotes the experience size of each batch, and β is the degree to which the control introduces offsetting biases.

3.6 Bidirectional search method

We use prioritized experience replay, on the basis of which we introduce the multi-intelligence agent idea. Two common methods are used for multi-intelligence agent experience replay. In the first method, an independent experience replay buffer is set up for each intelligent agent, where each intelligent agent uses its own prioritized experience replay mechanism. In the second method, the experiences of multiple intelligences are shared and merged into a shared replay buffer, and then, a prioritized experience replay is implemented for this buffer. In the prioritized experience replay mechanism introduced herein, dual intelligences and distributed prioritized replay, which applies the prioritized experience replay mechanism in a distributed environment by using the experience of multiple intelligences to update the priority level together, are employed. Multi-intelligences can store experiences in a buffer for sample training and learn from shared experiences. The shared experience buffer increases the diversity of experiences, provides more sample data, and accelerates the optimization of strategies in cooperative tasks, which helps increase the training efficiency. Moreover, a shared buffer can reduce the amount of maintenance required compared to individual buffers, which reduces resource consumption, just as a single intelligence does. The two intelligent agents begin at the start and endpoints and move toward the end and start points, respectively, before they collide in the middle and finally find the optimal path, as illustrated in Figure 7.

3.7 Gaussian noise

Upon the introduction of noise in the DDPG algorithm, the policy network outputs a deterministic action value instead of randomly sampled actions. However, to add explorability to the DDPG algorithm, Gaussian noise is added to the deterministic action. A randomized Gaussian noise process is used to maintain a degree of exploration while maintaining continuity and smoothness.

The action value output from the strategy network is added to the random noise obtained by sampling the

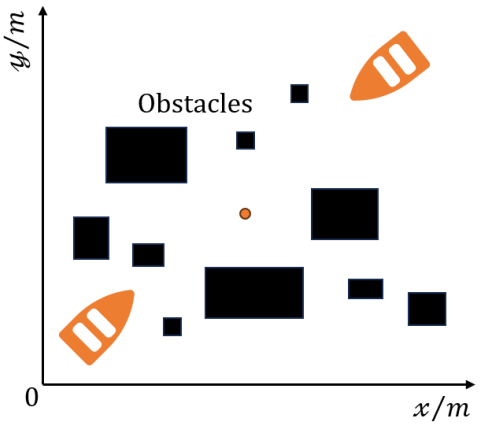

Figure 7. USV bidirectional search. USV: Unmanned surface vehicle.

Gaussian distribution, and the result is then used as the final action output. Noise variance is reduced gradually during the training process, and in the later training stages, the output of the strategy network itself is relied upon. As the number of training steps increases, the noise variance decreases, as follows:

$$\varphi = \frac{1}{step}, \tag{12}$$

where φ denotes noise variance, and $step$ denotes the number of steps in the current training process.

By introducing random Gaussian noise and gradually decreasing its variance, we can balance the exploratory and exploitative aspects and improve the performance and stability of the algorithm to some extent.

3.8 Improved DDPG algorithm pseudo-code

To enhance the understanding of the running process of the improved DDPG reinforcement learning algorithm 1, its pseudocode is provided, as follows:

3.9 Comparison of advantages and disadvantages of existing algorithms

In Table 2, the strengths and weaknesses of the existing DDPG algorithms described in the Introduction are presented concisely and clearly.

The advantages of the algorithm proposed herein compared to the existing algorithms are as follows: A new continuity reward function is designed considering the problem of reward sparsity. A dynamic region restriction method is introduced to reduce the computational burden by allowing an intelligent agent to ignore faraway obstacles and retain only a few obstacles around itself. A dual-intelligent agent combined with a prioritized experience replay mechanism method is added, where the dual-intelligent agent avoids the problem of too many intelligences leading to a larger burden and improves the efficiency compared to that of a single intelligent agent. Intelligent agents are allowed to share experiences with each other, which reduces the training time substantially and increases the overall planning efficiency and learning speed. To ensure greater consistency with the actual USV sailing situation, we set up the action space such that large USV movements such as 180-degree turns are avoided; the COLREGs principle is introduced; and the planned path is away from obstacles to improve safety and optimize sailing routes.

Algorithm 1: Improving the DDPG reinforcement learning algorithm

Initialization: Parameters of the Q-network and the policy network for each intelligence.
Two target network parameters for each intelligence.
Store, the experience playback pool, sets the prioritization adjustment parameter α and the importance sampling parameter β.

```
for episode in range(M): do
    Initialize the noise function step for action exploration.
    Initialize environment initialization S.
    Initialize dynamic region limit = 5.
    for t in range(T) do
        for i in range(Two_agents) do
            Consider dynamic region restriction to select action a, a = dynamic region restriction (a).
        end
        Execute action a and observe reward r, next s_next.
        for i in range(Two_agents) do
            compute improved reward function, R = compute improved reward function (s, a, r,
              s_next).
        end
        Store to the prioritized experience playback buffer Store=(s,a,R,s_next) and take a small batch
          from it.
        for i in range(Two_agents) do
            Optimize the gradient loss function of the policy network with loss so that Q takes a
              negative sign loss = −Q_ω (s, a).
            Optimize the loss function loss for the Q network is to find the mean square deviation of
              these two values loss = [Q_ω (s, a) , r + γQ_ω̄ (s′, a′)].
            Update the priority in the priority experience playback buffer.
        end
        Update the target network.
        Update current observation s=s_next.
    end
end
```

4. RESULTS

4.1 Modeling of the mission environment

We used Python to construct a 2D raster map representing the environment of the USV moving on the surface of water. The size of the raster map was set as 20×20. In Figure 8, the black part indicates the obstacles on the water surface and those under the water; red dot indicates the starting point; distant green point indicates the endpoint; and empty white part indicates the free passage area.

4.2 Parameter setting

A few specific parameter settings of the improved DDPG algorithm proposed herein are presented in Table 3.

4.3 Algorithm comparison

The start point is $(0, 0)$, endpoint is $(19, 19)$, and path planning is performed using the improved DDPG algorithm, original DDPG algorithm, and DQN algorithm proposed herein.

As illustrated in Figure 9, from the path planning perspective, the original DDPG algorithm and the improved DDPG algorithm proposed herein were able to reach the destination from the starting point. The proposed improved DDPG algorithm planned a smoother path than that planned by the original DDPG algorithm; the

Table 2. Comparison of the existing algorithms

Existing algorithms	Advantages	Disadvantages
Liu *et al.* [16]	A combination of multi-intelligentsia and combinatorial prioritization of experience replay mechanisms	Multi-intelligence coordination and cooperation. Requires a lot of space to store experience
Zhou *et al.* [17]	Multi-intelligent agent TD-MATD3 binding, a new reward function	Multi-intelligence coordination and cooperation. Requires a lot of space to store experience
Wu *et al.* [18]	Artificial potential field method for designing reward and punishment functions to improve prioritized experience replay structure	The parameter settings are sensitive, and the manual potential field is not efficiently adjusted, time-consuming, and trapped in a local optimum
Chen *et al.* [19]	Reducing the overestimation of Q, improving the reward function, and introducing a warm-up phase	Relies on a warm-up phase that requires extensive debugging to increase training time. Reduces overestimation of Q and falls into local optimality in complex environments
Feng *et al.* [20]	Introducing intrinsic dynamics HER, beluga whale optimization algorithm for hyperparameter optimization	The combination of intrinsic motivation and HER increases computation, leading to over-exploration, and HER requires suitable targets for replay
Yang *et al.* [21]	DRL and speed barriers combined, COLREGs introduced, VO algorithm incorporates reward functions, fuzzy theory calculations, NIF constructs state spaces	The application of fuzzy theory leads to uncertainty in the decision making process and the influence factors increase the state space complexity

TD-MATD3: Task decomposed multi-agent twin delayed deep deterministic policy gradient; HER: hindsight experience replay; DRL: deep reinforcement learning; COLREGs: International Regulations for Preventing Collisions at Sea; VO: velocity obstacle; NIF: navigational influence factor.

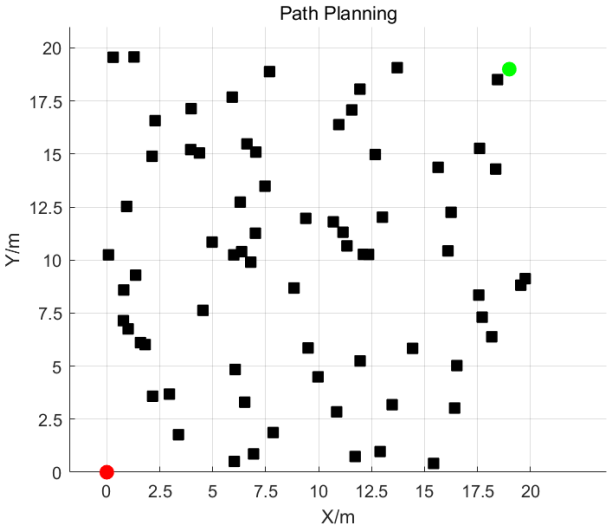

Figure 8. 2D grid environment. 2D: Two-dimensional.

Table 3. Parameter setting

Parameters	Retrieve a value
$Actor-lr$	0.001
$Critic-lr$	0.001
$gamma$	0.99
tau	0.005
$seed$	10
$noise$	0.1
$N1$	512
$N2$	512
$step$	300
$obstacle\_num$	50
$agent\_size$	0.3
$obstacle\_size$	0.5
$max\_visible\_obstacles$	5

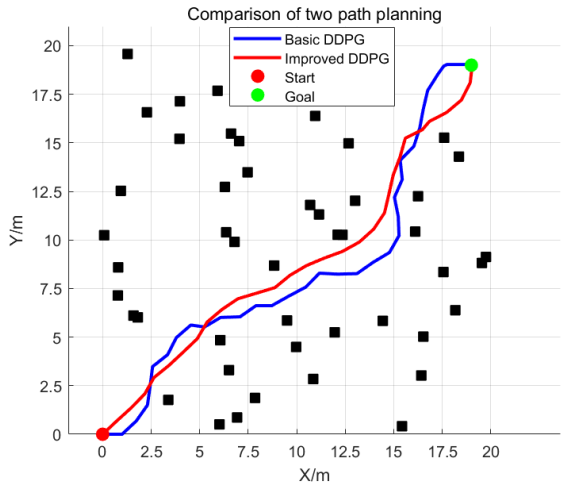

Figure 9. Paths planned using original and improved DDPG algorithms. DDPG: Deep deterministic policy gradient.

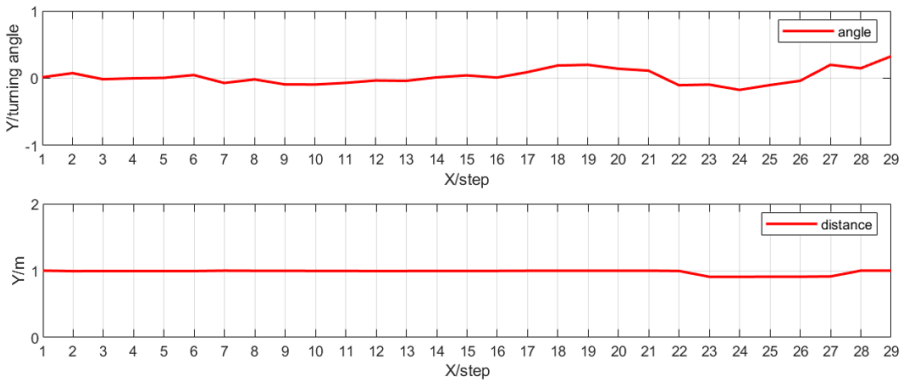

Figure 10. Steering angle and forward distance.

planned paths were shorter and did not exhibit excessive jitter.

Meanwhile, depending on the actual situation, the reward functions of the action space and continuity were changed such that the planned path was far away from obstacles and smoother without excessive steering. The steering angle and forward distance of the final path planned using the improved DDPG algorithm are illustrated in Figure 10, where the steering angle is not excessively jittery and is stable, and the forward distance is maintained at 0.99 m.

Figure 11 depicts the number of coordinate points of the X- and Y-axes of the path planned using the improved DDPG algorithm.

As illustrated in Figure 12, herein, we use dual intelligences combined with a prioritized experience replay mechanism. The proposed improved DDPG algorithm can reach relatively stable values within a shorter time in the early stages of training compared to the original DDPG algorithm, and the combined rewards for each episode in the early stage of training are higher than those of the original algorithm. The improved DDPG algorithm tends to reach higher values at around 500 episodes and stabilizes at 670 episodes, while the original DDPG algorithm stabilizes at 900 episodes.

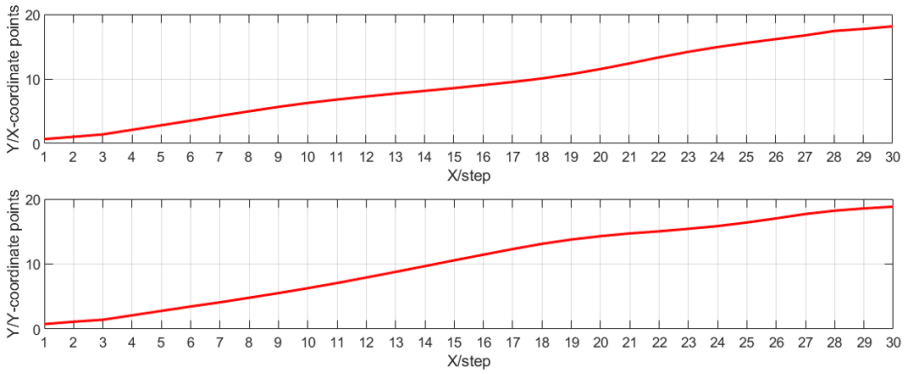

Figure 11. X- and Y-axes' coordinate points.

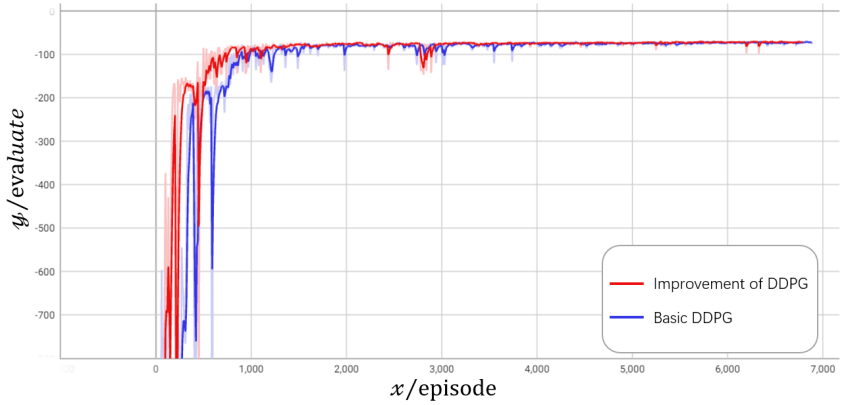

Figure 12. Algorithm convergence comparison.

In the same environment, we apply the DQN algorithm, and the results are presented in Figure 13, along with a comparison of the paths planned by the three algorithms. The DQN algorithm is trained, and in the current environment from the starting point, the jitter of the planned path is excessive, and the path does not reach the target point. The convergences of the DQN, DDPG, and the proposed improved DDPG algorithms are illustrated Figure 14.

The improved DDPG algorithm was compared to the original DDPG and DQN algorithms. The results indicated that the DQN algorithm planned an incomplete path, and it did not converge. The proposed improved DDPG algorithm changed the state and action spaces and redesigned the continuity reward and punishment functions such that the ranges of steering and advancement were narrowed. This, to a certain extent, restricted excessive fluctuation of the USV trajectory and smoothed and shortened the planned path, as illustrated in Figure 13. Simultaneously, the multi-intelligence agent priority experience replay mechanism was used, one from the middle to the starting point and one from the starting point to the endpoint. This reduced the planning distance to a certain extent. In addition, the dynamic area restriction was used to observe five obstacles in the vicinity while ignoring the other obstacles to reduce the number of algorithmic iterations, save computer resources, and accelerate the convergence speed of the algorithm, as depicted in Figure 14. The path planned by the improved DDPG was 16.81% shorter than that planned by the traditional DDPG algorithm. Moreover, the convergence speed of the proposed algorithm was 34.32% faster than that of the traditional algorithm. The details are presented in Table 4.

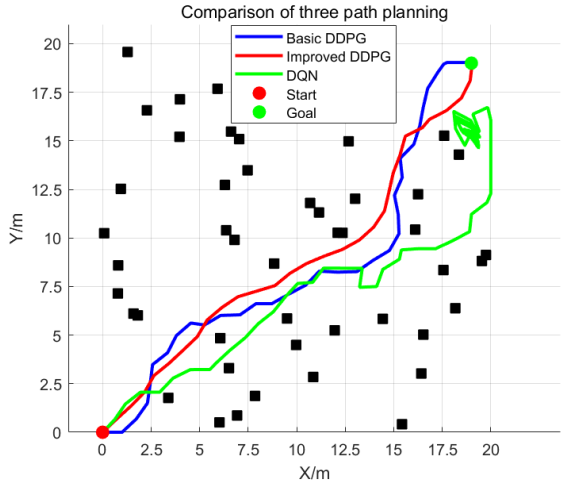

Figure 13. Comparison of three paths.

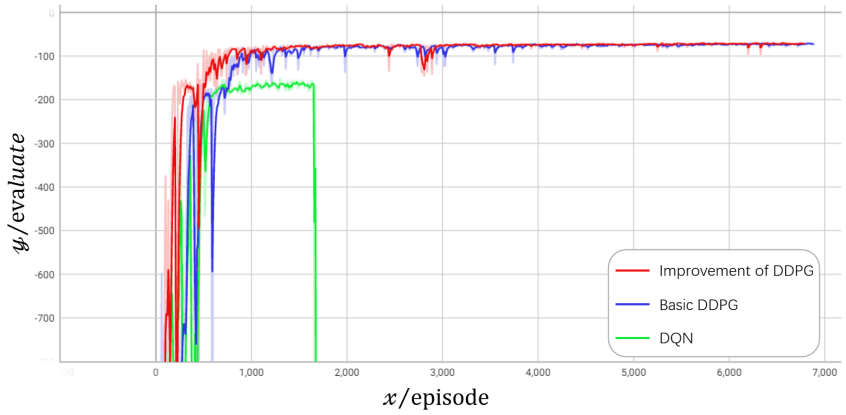

Figure 14. Comparison of convergences of three algorithms.

Table 4. Comparison of traditional and proposed DDPG algorithms

	Path length	Comprehensive incentives
DQN	Unfinished	Not converged
DDPG	34.05	900
Improvement of DDPG	29.15	670

DDPG: Deep deterministic policy gradient; DQN: deep Q-network.

The parameters *noise* and *tau* were revised considering the original parameters, as summarized in Table 3. The *noise* value was changed to 0.2 to increase the magnitude of noise and add exploration. The value of *tau* was changed to 0.009 to increase the speed of parameter update of the target network. The path planning results obtained using the above parameter settings are presented in Figure 15.

The length of the planned path was 31.35, which was longer and slightly more jittery than the path planned before changing the parameters. Additionally, the proposed improved DDPG had a consistently higher convergence speed than that of the algorithm with the changed parameters, as illustrated in Figure 16. Changing the values of other parameters yielded unsatisfactory results, and the results indicated that the parameter settings

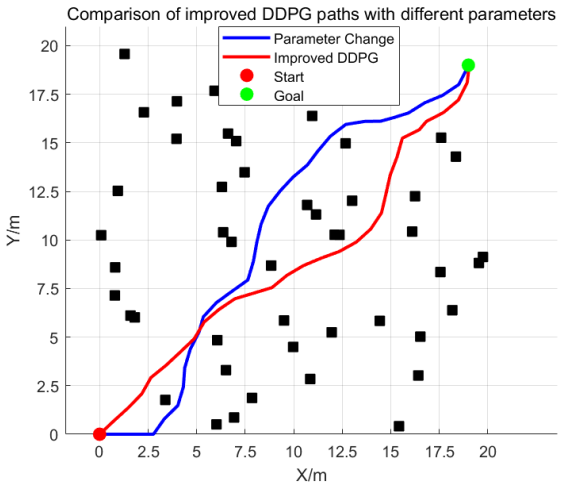

Figure 15. Comparison of paths planned using the improved DDPG with parameter changes. DDPG: Deep deterministic policy gradient.

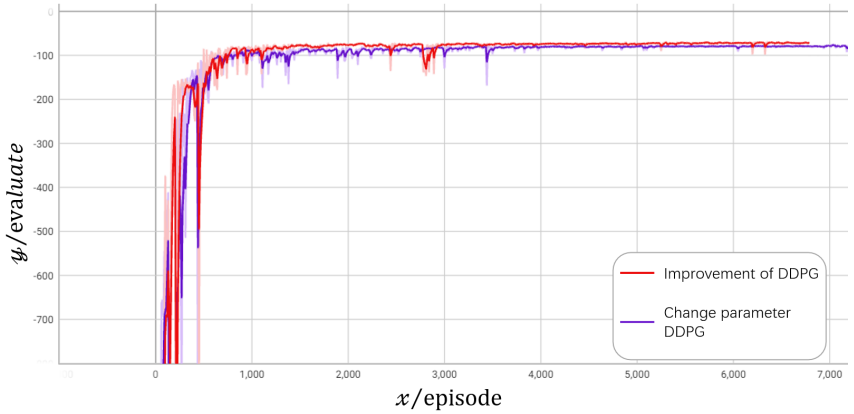

Figure 16. Convergences of improved DDPG algorithm and the algorithm with changed parameters. DDPG: Deep deterministic policy gradient.

presented in Table 3 were more appropriate.

Based on the original environment, the starting and endpoints were changed to $(5, 15)$, endpoint was $(19, 2.5)$, and triangular obstacles were added randomly to verify the feasibility of the algorithm. The results are presented in Figure 17, indicating that the improved algorithm was able to avoid the obstacles and find the endpoint.

To further verify the effectiveness of the proposed improved algorithm, based on the initial environment, we added 20 more obstacles to increase the complexity of the environment, as shown in Figure 18. The improved algorithm was still able to find the target point, the planned path did not have much jitter, and the path length was 33.98.

In the above experiments, we increased only the number and type of obstacles and changed the starting and endpoints. To explore further, we increased the map size to 25×25, where the endpoint was $(24, 24)$, and set the number of obstacles to 80 to verify the feasibility of the improved DDPG algorithm. As shown in Figure 19, the improved algorithm was still able to plan the path, and the jitter was not excessive.

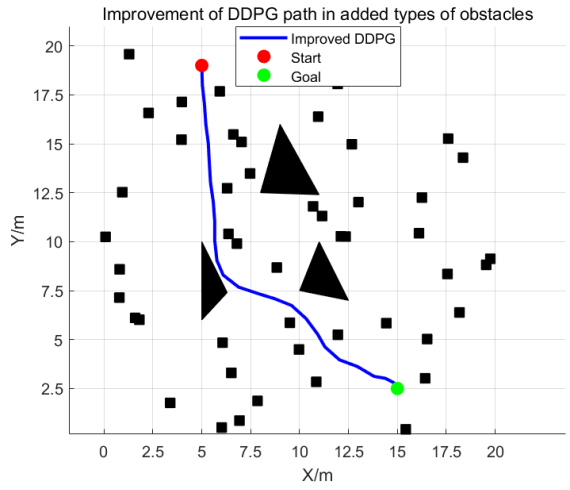

Figure 17. Adding obstacles and changing the starting point.

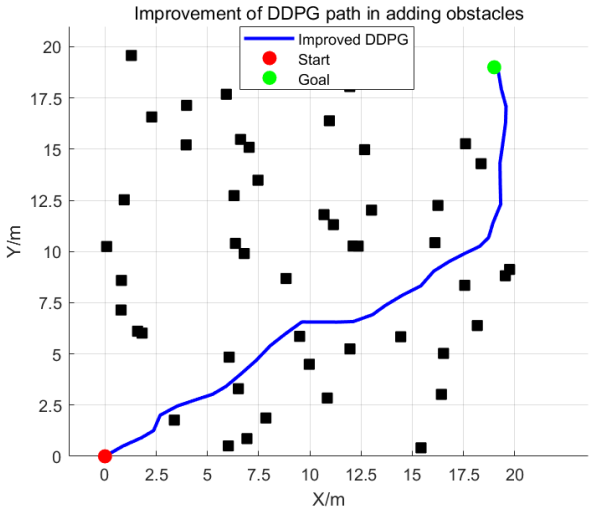

Figure 18. Improving DDPG algorithm path in complex environments. DDPG: Deep deterministic policy gradient.

The results of the above experiments indicate that the proposed improved DDPG algorithm performed path planning effectively. To further validate this result, we used the method proposed by Liu *et al.*, which mainly adopts MADDPG combined with a prioritized empirical replay mechanism, and makes a small change to the prioritized replay mechanism to redefine the prioritized ordering[16]. We extracted only the method (CPER-MADDPG) and applied it to USV path planning in this work. The method proposed by Liu *et al.* has some similarities to the improved method proposed herein, and therefore, it was selected for comparison[16].

Because only the method was extracted, to verify whether the method could plan the path in the environment considered herein, the experimental environment was changed to a simpler 15×15 map, but the number of obstacles remained unchanged at 50. As depicted in Figure 20, the performance of the proposed improved algorithm was still considerable. We used normal priority experience replay and added dynamic region restriction to save computer resources and accelerate algorithm convergence. The reward function was modified to ensure that the planned paths were less jittery and distant from obstacles.

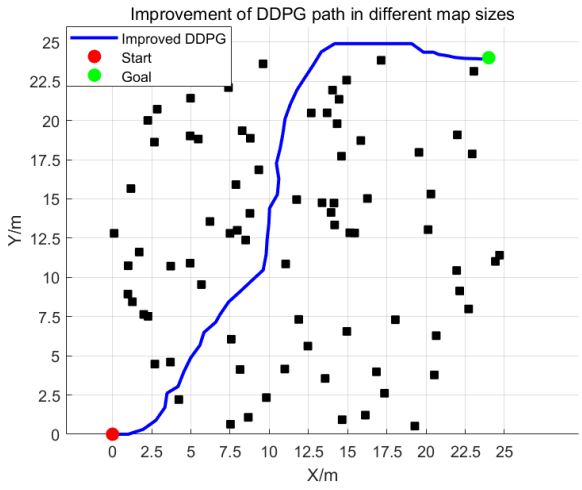

Figure 19. Paths obtained by applying improved DDPG algorithm to different map sizes. DDPG: Deep deterministic policy gradient.

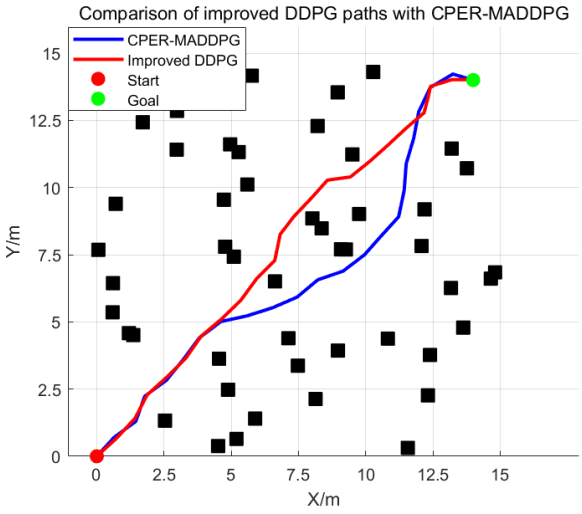

Figure 20. Comparison of the paths planned using the improved DDPG algorithm and the CPER-MADDPG algorithm. DDPG: Deep deterministic policy gradient; CPER-MADDPG: combined prioritized experience replay with multi-intelligent depth deterministic policy gradient.

The results of the above experiments verified the feasibility of the proposed method, and therefore, the environment was changed to the initial environment for comparison; the results are presented in Figure 21. The improved DDPG algorithm planned a smoother path with less jitter and fewer sharp corners compared to that planned by the CPER-DDPG algorithm. The lengths of the paths planned by the improved DDPG algorithm and CPER-DDPG algorithm were 29.15 and 30.92, respectively, indicating that the proposed algorithm planned a shorter path. The proposed algorithm changed the continuity reward function and action space, such that the planned paths were shorter and less jittery. As depicted in Figure 22, the convergence rate of the CPER-DDPG algorithm was consistently slower than that of the proposed algorithm in the early stages, and it reached the same level gradually at around $1,500$ iterations. The proposed algorithm stabilized after 670 iterations, but the CPER-DDPG algorithm stabilized after 860 iterations. The CPER-DDPG algorithm used a multi-intelligence agent combined with a priority experience replay mechanism, while the proposed algorithm additionally used dynamic region restriction to reduce the number of iterations and accelerate convergence.

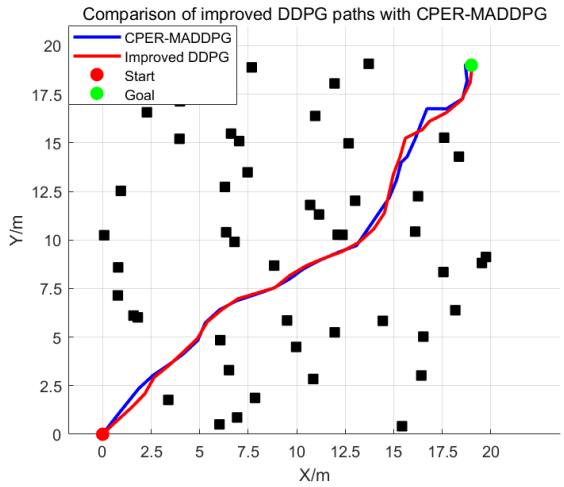

Figure 21. Comparison of paths planned using the proposed improved DDPG algorithm and CPER-MADDPG for different maps. DDPG: Deep deterministic policy gradient; CPER-MADDPG: combined prioritized experience replay with multi-intelligent depth deterministic policy gradient.

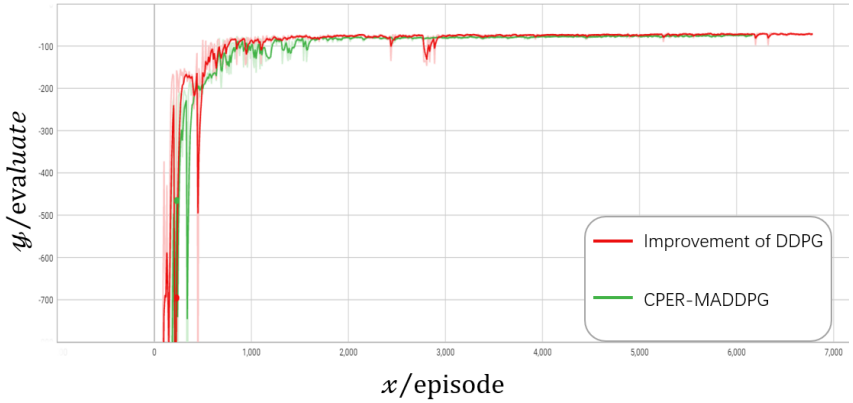

Figure 22. Convergences of improved DDPG and CPER-MADDPG algorithms. DDPG: Deep deterministic policy gradient; CPER-MADDPG: combined prioritized experience replay with multi-intelligent depth deterministic policy gradient.

In sum, we compared the proposed algorithm with the traditional DDPG algorithm and DQN algorithm. By changing the parameter settings of the proposed improved DDPG algorithm for comparison, increasing the number of irregular triangular obstacles in the original environment, using different map sizes, and extracting the method proposed in [16] and applying it to the experimental environment of this paper for comparison, we verified the feasibility of the proposed improved DDPG algorithm.

5. CONCLUSIONS

In this paper, the improved DDPG algorithm was proposed for USV path planning. Based on the DDPG algorithm, the continuity reward function was redesigned, and the COLREGs principle was introduced to solve the reward sparsity problem. Dynamic region restriction was added to reduce the number of algorithm iterations and save computational resources. The multi-intelligence body bidirectional search method and distributed priority replay mechanism were introduced to accelerate algorithm convergence. Through experiments, it was demonstrated that the improved DDPG algorithm converged faster, and the paths planned using it were smoother and shorter, thereby verifying the feasibility of the improved algorithm. The improved DDPG al-

gorithm increased the efficiency and accuracy of path planning, as well as the safety of USV navigation by introducing COLREGs, which helps USVs navigate complex environments. In terms of the dynamics of the setup, the state space still does not consider the real sailing situation. Meanwhile, in the future, the performance of the proposed algorithm should be explored in different environments by considering more real-time factors such as the wind, waves, and weather conditions. Additionally, the algorithm could also be extended to other types, such as financial trading to optimize the timing of automated buying and selling, and healthcare to dynamically adjust drug dosages. Furthermore, deep learning techniques could be incorporated to optimize the continuous reward function and dynamic region restriction.

DECLARATIONS

Authors' contributions

Carried out the numerical simulation, wrote the improved DDPG code, and wrote the draft: Hua M
Guided the work, and proofread the draft: Zhou W
Helped collect and process the data: Cheng H, Chen Z

Availability of data and materials

Not applicable.

Financial support and sponsorship

This work was supported by the National Natural Science Foundation of China (No. 62203293). It was also sponsored by the Shanghai Rising-Star Program (No. 22YF1416100)

Conflicts of interest

Zhou W is the Guest Editor Assistant of the Special Issue of "Intelligent Control and Decision-making of Unmanned Ships" of the journal *Intelligence & Robotics*, while the other authors have declared that they have no conflicts of interest.

Ethical approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

REFERENCES

1. Tabish N, Chaur-Luh T. Maritime autonomous surface ships: a review of cybersecurity challenges, countermeasures, and future perspectives. *IEEE Access* 2024;12:17114–36. DOI
2. Sun Q, Liu ZW, Chi M, Ge MF, He D. Robust coverage control of multiple USVs with time-varying disturbances. *Intell Robot* 2023;3:242-56. DOI
3. Liang Y. Route planning of aids to navigation inspection based on intelligent unmanned ship. In: 2021 4th International Symposium on Traffic Transportation and Civil Architecture (ISTTCA); 2021 Nov 12-14; Suzhou, China. IEEE; 2021. pp. 95-8. DOI
4. Li H, Yang Z. Incorporation of AIS data-based machine learning into unsupervised route planning for maritime autonomous surface ships. *Transport Res E Log* 2023;176:103171. DOI
5. Su M, Pu R, Wang Y, Yu M. A collaborative siege method of multiple unmanned vehicles based on reinforcement learning. *Intell Robot* 2024;4:39-60. DOI
6. Feng S, Li X, Ren L, Xu S. Reinforcement learning with parameterized action space and sparse reward for UAV navigation. *Intell Robot* 2023;3:161-75. DOI
7. Yu D, Roh MI. Method for anti-collision path planning using velocity obstacle and A* algorithms for maritime autonomous surface ship. *Int J Nav Arch Ocean* 2024;16:100586. DOI

8. Zhang B, Cao J, Li X, et al. Minimum dose walking path planning in a nuclear radiation environment based on a modified A* algorithm. *Ann Nucl Energy* 2024;206:110629. DOI
9. Yu J, Yang M, Zhao Z, et al. Path planning of unmanned surface vessel in an unknown environment based on improved D*Lite algorithm. *Ocean Eng* 2022;266:112873. DOI
10. Jin J, Zhang Y, Zhou Z, Jin M, Yang X, Hu F. Conflict-based search with D* lite algorithm for robot path planning in unknown dynamic environments. *Comput Electr Eng* 2023;105:108473. DOI
11. Zhang J, Zhu Z, Xue Y, Deng Z, Qin H. Local path planning of under-actuated AUV based on VADWA considering dynamic model. *Ocean Eng* 2024;310:118705. DOI
12. Cao Y, Nor NM. An improved dynamic window approach algorithm for dynamic obstacle avoidance in mobile robot formation. *Decis Anal* 2024;11:100471. DOI
13. Zhou Q, Lian Y, Wu J, Zhu M, Wang H, Cao J. An optimized Q-Learning algorithm for mobile robot local path planning. *Knowl Based Syst* 2024;286:111400. DOI
14. Wang Y, Lu C, Wu P, Zhang X. Path planning for unmanned surface vehicle based on improved Q-Learning algorithm. *Ocean Eng* 2024;292:116510. DOI
15. Yu Y, Liu Y, Wang J, Noguchi N, He Y. Obstacle avoidance method based on double DQN for agricultural robots. *Comput Electron Agr* 2023;204:107546. DOI
16. Liu P, Ma X, Ding J, Gu C. Multi-agent collaborative path planning algorithm with reinforcement learning and combined prioritized experience replay in Internet of Things. *Comput Electr Eng* 2024;116:109193. DOI
17. Zhou Y, Kong X, Lin KP, Liu L. Novel task decomposed multi-agent twin delayed deep deterministic policy gradient algorithm for multi-UAV autonomous path planning. *Knowl Based Syst* 2024;287:111462. DOI
18. Wu X, Zhao Z, Ge Q, Luo Z. Unmanned aerial vehicle path planning based on DP-DDPG algorithm. In: 2023 IEEE International Conference on Unmanned Systems (ICUS); 2023 Oct 13-15; Hefei, China. IEEE; 2023. pp. 251–6. DOI
19. Chen Q, Yang J, Mao J, Liang Z, Lu C, Sun P. A path following controller for deep-sea mining vehicles considering slip control and random resistance based on improved deep deterministic policy gradient. *Ocean Eng* 2023;278:114069. DOI
20. Feng Z, Wang C, An J, et al. Emergency fire escape path planning model based on improved DDPG algorithm. *J Build Eng* 2024;95:110090. DOI
21. Yang X, Lou M, Hu J, et al. A human-like collision avoidance method for USVs based on deep reinforcement learning and velocity obstacle. *Expert Syst Appl* 2024;254:124388. DOI