Journal of *Marine Science and Engineering*

MDPI

Article

Research on Ship Trajectory Control Based on Deep Reinforcement Learning

Lixin Xu [1,2], Jiarong Chen [3], Zhichao Hong [1,2,*], Shengqing Xu [3], Sheng Zhang [4] and Lin Shi [5]

[1] Ocean College, Jiangsu University of Science and Technology, Zhenjiang 212003, China; lxu@just.edu.cn
[2] Jiangsu Marine Technology Innovation Center, Nantong 226000, China
[3] School of Naval Architecture and Ocean Engineering, Jiangsu University of Science and Technology, Zhenjiang 212000, China; cjrwn513@163.com (J.C.); shengqingxu7@gmail.com (S.X.)
[4] Zhenjiang Yuanli Innovation Technology Co., Ltd., Zhenjiang 212008, China; zhangsheng_1990@163.com
[5] China Construction Civil Infrastructure Co., Ltd. (CSCIC), Beijing 100029, China; shi_lin@cscec.com
* Correspondence: hzciavip@163.com; Tel.: +86-186-4092-1905

Abstract: Ship trajectory tracking controllers based on deep reinforcement learning (DRL) are widely applied in various fields such as autonomous driving and robotics due to their strong adaptive learning capabilities and optimization decision-making ability. However, ship trajectory control faces challenges such as long training cycles and poor convergence performance. These issues are primarily caused by the unreasonable design of algorithm models and reward functions, which limit the performance optimization and energy efficiency improvements in real-world navigation. In this paper, we propose a ship trajectory tracking control algorithm based on deep reinforcement learning. The proposed algorithm introduces maximum entropy theory and experience replay techniques. Additionally, it enhances the reward function module by adding reward terms and fitting weight designs. A three-dimensional simulation environment is constructed to validate the proposed method. The results demonstrate that the controller designed in this study outperforms traditional DRL controllers in terms of fast convergence, convergence stability, and final reward values. The controller meets the requirements for tracking conventional trajectories and shows stable and efficient performance in both wide-area water search experiments and river channel traversal experiments. These experimental results provide valuable insights for future research directions.

Keywords: trajectory control; deep reinforcement learning; trajectory tracking controller; reward function

check for updates

Received: 9 March 2025
Revised: 13 April 2025
Accepted: 14 April 2025
Published: 16 April 2025

Citation: Xu, L.; Chen, J.; Hong, Z.; Xu, S.; Zhang, S.; Shi, L. Research on Ship Trajectory Control Based on Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2025**, *13*, 792. https://doi.org/10.3390/jmse13040792

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

1.1. Background

With the development of the global shipping industry and the continuous advancement of related technologies, intelligent ship technology is gradually becoming one of the main research directions in the industry [1,2]. Trajectory tracking control, as one of the core technologies for intelligent ships, plays a significant role in automatic navigation and collision avoidance [3,4].

Traditional ship trajectory tracking control methods primarily include PID control, fuzzy control, sliding mode control, and others [5]. These methods can meet the basic control requirements of ships to some extent, but their control accuracy and robustness are often limited in the complex and ever-changing marine environment. Deep reinforcement learning (DRL) technology is gradually demonstrating its unique advantages in the

maritime field [6,7]. DRL does not require high-precision models and performs well in handling complex, nonlinear ship trajectory tasks [8,9]. However, it also faces challenges such as long training cycles and low interpretability.

1.2. Related Work

Traditional ship control is usually based on PID, etc. Fossen et al. utilized a line-of-sight- (LOS) based underactuated ship path tracking control method, which is one of the earliest and most widely used strategies for ship motion control [10]. Their team introduced a nonlinear Lyapunov function to design the controller, effectively addressing issues posed by the model [11]. In this framework, DO [12] introduced a nonlinear Lyapunov function [13] to design the controller, effectively addressing issues posed by the model. Shen et al. [14] modeled fully actuated ships and, to tackle external environmental disturbances, proposed an adaptive dynamic surface sliding mode control method with a disturbance observer. They performed simulation experiments to verify its feasibility. Jiao et al. [15] introduced a performance function with constraints for controller design, utilizing radial basis function (RBF) neural networks to correct jittering parameters. Their simulation results showed that the trajectory tracking error converged within the specified range, verifying the effectiveness and superiority of the proposed control strategy. Qi [16] used a high-gain observer to estimate the system's velocity vector and combined this with the approximation capability of RBF neural networks and the backstepping method to design a controller that successfully completed trajectory tracking for the ship. Shen et al. [17], under input constraints, designed a sliding mode recursive control law by constraining the ship's trajectory, considering both the position and velocity errors, which enhanced the system's robustness. Finally, they integrated neural networks to complete the trajectory tracking control of the ship. Liu et al. [18] employed a disturbance observer to estimate low-frequency disturbances in the dynamics, designed adaptive laws to estimate unknown time-varying current velocities, and used an auxiliary power system to compensate for input saturation constraints in the brake system. Chen et al. [19] proposed a fixed-time fractional order synovial control method to address uncertainty issues in the model and environment. This method can effectively track and reduce the shaking phenomenon of synovium. He et al. [20] proposed an autonomous ship collision avoidance path planning method suitable for multi-ship encounter scenarios, achieving real-time ship course adjustment through fuzzy adaptive PID control.

In recent years, research on trajectory control has primarily focused on deep reinforcement learning methods. Zhao L et al. [21] designed a trajectory controller based on the PPO algorithm, achieving good tracking performance in unknown environments. Song et al. [22] developed a trajectory controller using carrot tracking guidance combined with the PPO algorithm, which avoids complex parameter calculations and demonstrates high tracking accuracy in disturbed environments. Zhang et al. [23] and Zhu et al. [24] designed a trajectory controller using line-of-sight (LOS) guidance combined with the DDPG algorithm, with simulation results showing good tracking performance; however, interference issues were not considered in the tests. Wang [25] designed a trajectory controller based on DDPG-H, achieving precise tracking under both interference-free and disturbed conditions, with relatively smooth rudder angle outputs. Zhao Y et al. [26] proposed a trajectory control method based on DQN smooth convergence. Chen et al. [27] proposed a ship route planning method that integrates the A* algorithm with a Double Deep Q-Network (A-DDQN), demonstrating improvements in reducing fuel consumption and carbon emissions. Cui et al. [28] incorporated LSTM and MHA mechanisms into the TD3 algorithm network, enhancing its attention to historical state information and achieving superior ship performance compared to conventional methods in complex encounter scenarios. Wang

et al. [29] proposed a SAC-based multi-path tracking controller that significantly improves path-following accuracy and success rate. Experimental results showed that the proposed algorithm converges faster and has better control effects compared to traditional DQN algorithms, which has practical significance.

In practical ship control applications, vessel trajectory tracking primarily relies on PID control and Model Predictive Control (MPC). Due to the inherent limitations of these two control algorithms, some real-world vessels adopt a hybrid "PID+MPC" architecture. For instance, Kongsberg's Dynamic Positioning (DP) system, deployed on thousands of offshore vessels, traditionally uses PID for thrust allocation. Meanwhile, Hyundai Heavy Industries has applied MPC technology in LNG ship berthing systems, where its core functionality lies in multi-step prediction and constrained optimization to ensure safe docking in complex environments. However, these methods exhibit limited adaptability in harsh sea conditions and require high parameter-tuning costs, prompting the industry to explore smarter solutions.

Deep reinforcement learning, owing to its advantages in adaptive decision-making, has gradually transitioned from academic research to practical applications in recent years. Researchers at the Norwegian University of Science and Technology developed a DRL-based ship navigation controller and conducted experiments using a 1:75.5 scale physical ship model. Results showed that under wind-free conditions, the controller achieved precise tracking of a 40 m square trajectory, with measured position and heading angles aligning closely with simulation data. Even under strong wind disturbances, the system demonstrated robustness by completing complex maneuvers such as "figure-8" paths, despite initial deviations, thereby validating the feasibility of DRL.

A review of current research in ship motion control reveals that while traditional trajectory control algorithms can accomplish tracking tasks, they suffer from limited adaptability and high tuning costs. DRL algorithms effectively address these shortcomings while remaining compatible with other parallel research efforts. However, as studies progress, challenges such as prolonged training cycles and decision under- or overfitting have become apparent. Given these research characteristics, this study develops a Soft Actor–Critic algorithm to enhance training efficiency in ship control. This approach not only addresses a key challenge in ship trajectory tracking control but also constitutes the core contribution of this paper.

2. Mathematical Model of the Ship

2.1. Kinematic Equations

In this study, since the research task focuses on the trajectory tracking control of an unmanned ship and primarily addresses its horizontal motion, only three degrees of freedom are considered: roll, surge, and heave. Additionally, two coordinate systems are established: the body-fixed coordinate system and the North–East coordinate system, see Figure 1 for details.

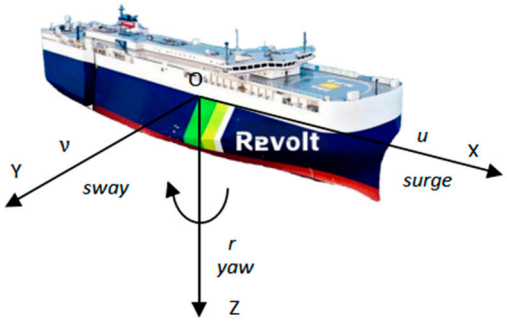


Figure 1. Body-fixed coordinate system.

Thus, the conversion relationship between the ship's velocity in the North–East-Down (NED) coordinate system and the velocity in the body-fixed coordinate system (Body) is shown in Figure 2. The conversion formula is as follows:

$$\begin{cases} \dot{x} = u\cos\psi + v\sin\psi \\ \dot{y} = u\sin\dot{\psi} - v\cos\psi \\ \dot{\psi} = r \end{cases} \tag{1}$$

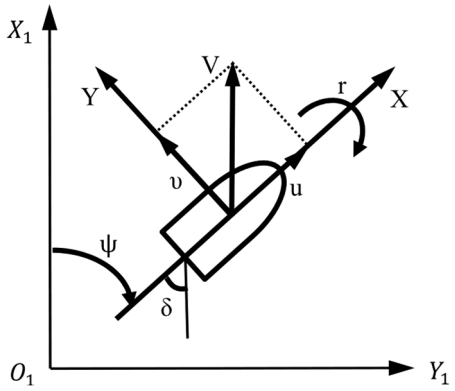


Figure 2. Transformation relationship of the coordinate system.

The vectors $v = [u, v, r]^T$ and $\eta = [x, y, \psi]^T = [N, E, \psi]^T$ represent the ship's velocity, position, and heading angle information, which are used to describe the ship's state. The ship's kinematic equations can be expressed as follows:

$$\dot{\eta} = R(\psi)v \tag{2}$$

In the equation, $\dot{\eta}$ represents the ship's velocity in the North–East-Down (NED) coordinate system, and $R(\psi) \in R^{3\times3}$ is the coordinate transformation matrix, which satisfies the following:

$$R^T(\psi) = R^{-1}(\psi) \text{ and } \|R(\psi)\| = 1 \tag{3}$$

$$R(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

2.2. Dynamics Model

When the ship is tracking at high speed, if the speed exceeds 4 knots, it exhibits underactuated characteristics, lacking sufficient sway control force. Tracking is achieved solely through the coordination of surge and yaw. Generally, in the study of track and heading control, it is assumed that the ship is sailing at a constant or slowly changing speed, and the speed is close to the surge speed:

$$U = \sqrt{u^2 + v^2} \approx u \tag{5}$$

To simplify the model, it is typically assumed that the ship is symmetric about its longitudinal axis, allowing the separation of the longitudinal motion equation from the lateral motion and yaw motion equations. The ship's speed is not influenced by the other two equations.

$$(m - X_{\dot{u}})\dot{u} - X_u u - X_{|u|u}|u|u = \tau_u \tag{6}$$

Assuming that the speed of the propeller is constant and only the sway and yaw equations are considered, the dynamic equation can be simplified as follows:

$$M\dot{\upsilon} + N(u_o)\upsilon = b\delta \tag{7}$$

In the equation, $\upsilon = [v, r]^T$ is the state vector, M is the mass matrix, $N(u_o)$ contains velocity and linear term matrices, b is the control input matrix, and δ is the steering angle.

In ship maneuvering motion, Equation (7) is usually simplified, and the simplified response equation is:

$$T_1T_2\ddot{r} + (T_1 + T_2)\dot{r} + r = K\delta + KT_3\dot{\delta} \tag{8}$$

Applying the Laplace transform to Equation (8) yields the transfer function:

$$\frac{r}{\delta}(s) = \frac{K(1 + T_3s)}{(1 + T_1s)(1 + T_2s)} \tag{9}$$

In the formula, K, T_1, T_2, T_3 are the manipulative indices.

In this paper, considering the large inertia and low frequency characteristics of the ship, as well as the steering delay response phenomenon, combined with the steering engine model and the environmental disturbance term, a first-order linear response model was obtained:

$$\begin{cases} T\ddot{r} + r = K\delta + \omega \\ T_E\dot{\delta} = \delta_E - \delta \end{cases} \tag{10}$$

where T_E, δ_E, δ are the rudder time constant, the commanded rudder angle, and the actual rudder angle, respectively. ω is the environmental disturbance term.

3. Method

3.1. MDP Model of USV (Unmanned Surface Vehicle) Ship Trajectory Control

In order to realize the track control of unmanned ships, it is necessary to build a Markov decision process (MDP) model [30] that integrates the construction environment and data collection. The model is composed of four tuples (P, S, A, R), where A represents the action space, S represents the state space, P represents the state transition probability, and R represents the reward function. The planning system process of the MDP model is as follows: update the initial state information S_0 of the unmanned ship, output action $a_t \in A$. According to strategy π within each time step t, the unmanned ship enters a new state S_{t+1} and evaluates the action effect through the reward function R_t.

3.2. PER-Soft Actor–Critic

3.2.1. Soft Actor–Critic

The SAC algorithm was proposed by Tuomas haarnoja et al. Compared with the traditional Actor–Critic (AC) algorithm, SAC introduced the maximum entropy framework and added the entropy regularization term to the objective function, which improved the exploration efficiency and stability of the algorithm, and solved the problem that the traditional reinforcement learning could not solve in the continuous action space.

The introduction of maximum entropy helps to accelerate policy learning and reduce the likelihood of the agent becoming stuck in local optima. Therefore, entropy regularization is added to the objective function of DRL, and its optimization objective function is:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, E_\pi\left[\sum_t r(s_t, a_t) + \alpha H(\pi(\cdot|s_t))\right] \tag{11}$$

where π^* is the optimal strategy, α is the regularization coefficient, and $H(\pi(\cdot|s_t))$ is the degree of randomness in the state s_t.

The regularization of entropy: By adjusting the coefficient, the larger the value, the greater the exploration rate of the algorithm, so as to improve the convergence speed of the strategy and reduce the possibility of falling into the local optimum.

The selection of the entropy regularization coefficient (temperature coefficient) is very important in this algorithm. The choice of the regularization coefficient should be different in different states and environments. When the optimal state solution is uncertain, the entropy weight should be appropriately increased. When confirming some state solutions, it should be smaller. The goal of reinforcement learning is that the mean value of constraint entropy is greater than H_O. Then we can obtain the following loss function:

$$L(\alpha) = E_{s_t \sim D, a_t \sim \pi(.|s)}[-\alpha \log \pi(a_t|s_t) - \alpha H_O] Q(s_t, a_t) \pi(s_t, a_t) \tag{12}$$

When the entropy of the strategy is lower than H_O, the objective function will increase the regularity coefficient α and improve the exploratory ability of the agent. On the contrary, it will reduce the coefficient to ensure the improvement in the reward.

The SAC algorithm is solved by strategy iteration. Therefore, the soft Bellman equation is used:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma E_{s_{t+1}}[V(s_{t+1})] \tag{13}$$

The stable value function is represented as follows:

$$V(s_t) = E_{a_t \sim \pi}[Q(s_t, a_t) - \alpha \log \pi(a_t|s_t)] = E_{a_t \sim \pi}[Q(s_t, a_t) + \alpha H(\pi(\cdot|s_t))] \tag{14}$$

According to the soft Bellman equation, the final strategy can converge to the finite state and action space, so the improved formula of the Soft strategy is:

$$\pi_{new} = \underset{\pi'}{\operatorname{argmin}} D_{KL}\left(\pi'(\cdot|s), \frac{\exp\left(\frac{1}{\alpha} Q^{\pi_{old}}(s, \cdot)\right)}{Z^{\pi_{old}}(s, \cdot)}\right) \tag{15}$$

In the SAC algorithm, there are two parts: Actor and Critic networks, corresponding to the action value function $Q(s_t, a_t)$ and the strategy function $\pi(s_t, a_t)$. When selecting the Q value, a smaller value should be selected to avoid overestimating the Q value. The TD method is used to calculate network loss. The loss function of the Critic-Q network is as follows:

$$L_Q(\phi) = E_{(s_t, a_t, r_t, s_{t+1}) \sim D}\left[\frac{1}{2}\left(Q_\phi(s_t, a_t) - \left(r_t Y\left(\min_{j=1,2} Q_{\overline{\phi}}(s_{t+1}, a_{t+1}) \alpha \log \pi(a_{t+1}|s_{t+1})\right)\right)\right)^2\right] \tag{16}$$

where D is the sample, $\phi, \overline{\phi}$ is the network parameter and the target network parameter, respectively.

The strategy function obtains the loss function through KL divergence:

$$L_\pi(\theta) = E_{s_t \sim D, a_t \sim \pi_\theta}\left[\alpha \log(\pi_\theta(a_t|s_t)) - Q_\phi(s_t, a_t)\right] \tag{17}$$

In a continuous action environment, the sampling process itself is non-differentiable, and sampling actions directly from the policy network result in a non-differentiable. To address this issue, the Reparameterization Trick was employed to transform the sampling process into a differentiable form. The specific method is detailed in Table 1.

Table 1. Reparameterization Trick.

Reparameterization Trick
1. Generate Gaussian distribution parameters: The strategy network outputs the mean $\mu_\theta(s_t)$ and standard deviation $\sigma_\theta(s_t)$ of Gaussian distribution;
2. Add random noise: Generating Gaussian random noise $\xi \sim N(0,1)$; By ξ generating specific actions: $\widetilde{a_t} = \mu_\theta(s_t) + \sigma_\theta(s_t)\cdot\xi$;
3. Scaling using a bitangent sine function: To ensure that the action is within a reasonable range, zoom: $\widetilde{a_t} = \tanh(\mu_\theta(s_t) + \sigma_\theta(s_t)\cdot\xi)$.

The reparameterized action $\widetilde{a_t}$ is substituted into the Q network to calculate the Q value, and a new loss function is obtained:

$$L_\pi(\theta) = E_{s_t\sim D,\xi\sim N}\left[\alpha\log\left(\left(\pi_\theta\left(\widetilde{a_\theta}(s_t,\xi)\right)\middle|s_t\right)\min_{j=1,2} Q_{\phi_j}\left(s_t,\widetilde{a_\theta}(s_t,\xi)\right)\right)\right] \tag{18}$$

3.2.2. Priority Experience Replay

The prioritized experience replay (PER), proposed by Schaul et al. in 2016 [31], is a technique to improve the efficiency of deep reinforcement learning algorithms. The main idea of this technique is to enable the model to learn more from experiences that are more informative; that is, those resulting in larger temporal difference (TD) errors. The larger the error, the more the model needs to be corrected, making that sample more valuable. The TD error is defined as follows:

$$\delta_t = r_t + \gamma Q_1(s_{t+1}, a_{t+1}; \theta) - Q_2(s_t, a_t; \theta) \tag{19}$$

where r_t is the instant reward, γ is the discount factor, Q_1 and Q_2 are the target Q network and the current Q network, respectively. When sampling, the greater the δ_t value, the greater the probability. When sampling for new samples, the samples are stored according to the size of the error value δ_t. The new samples are added to the priority queue, and the samples with the lowest δ_t value in the queue are deleted. Priority p_i is expressed as follows:

$$p_i = |\delta_i| + \epsilon \tag{20}$$

where $|\delta_i|$ is the absolute value of the TD error and ϵ is a small positive value, which is used to avoid the case where the priority p_i is 0.

The sampling probability of the sample is expressed as follows:

$$p_i = \frac{p_i^\alpha}{\sum_i p_i^\alpha} \tag{21}$$

where p_i^α is the α power of priority and $\sum_i p_i^\alpha$ is the sum of α power of all priorities. α is a super parameter used to control the importance of priority. When $\alpha = 0$, all samples are uniformly sampled. When $\alpha = 1$, sampling is entirely based on priority. However, the model will produce deviations when sampling. In order to correct the deviation caused by priority sampling, the importance sampling weight ω_i was introduced:

$$\omega_i = \left(\frac{1}{N}\cdot\frac{1}{p(i)}\right)^\beta \tag{22}$$

$$L = \frac{1}{B}\sum_{i} \omega_i \delta_i^2 \tag{23}$$

where n is the size of the experience pool, and β is a super parameter, which is used to control the degree of importance sampling. Generally speaking, β gradually increases from a small value to 1 to gradually reduce the impact of the deviation. L is the final loss function.

3.2.3. Combination of PER and Soft Actor–Critic

The final algorithm is PER-SAC, and the specific algorithm process is shown in Algorithm 1.

Algorithm 1 PER-SAC Algorithm

1. Set policy parameters, initialize the corresponding target network, and initialize experience pool D;
2. For episode e = 1-E do:
   For each step in the time step: execute the action $a_t \in \pi_\theta$, check the experience pool capacity, obtain the reward r_t and the next state s_{t+1}, store the track (s_t, a_t, r_t, s_{t+1}), and update the experience pool;
   Sample x pieces of experience with sampling probability from experience pool D;
   Calculate importance sampling weight;
   Update the two Q networks and the policy network π, and update the temperature coefficient at the same time;
3. End.

Figure 3 is the network structure diagram describing the PER-SAC algorithm:

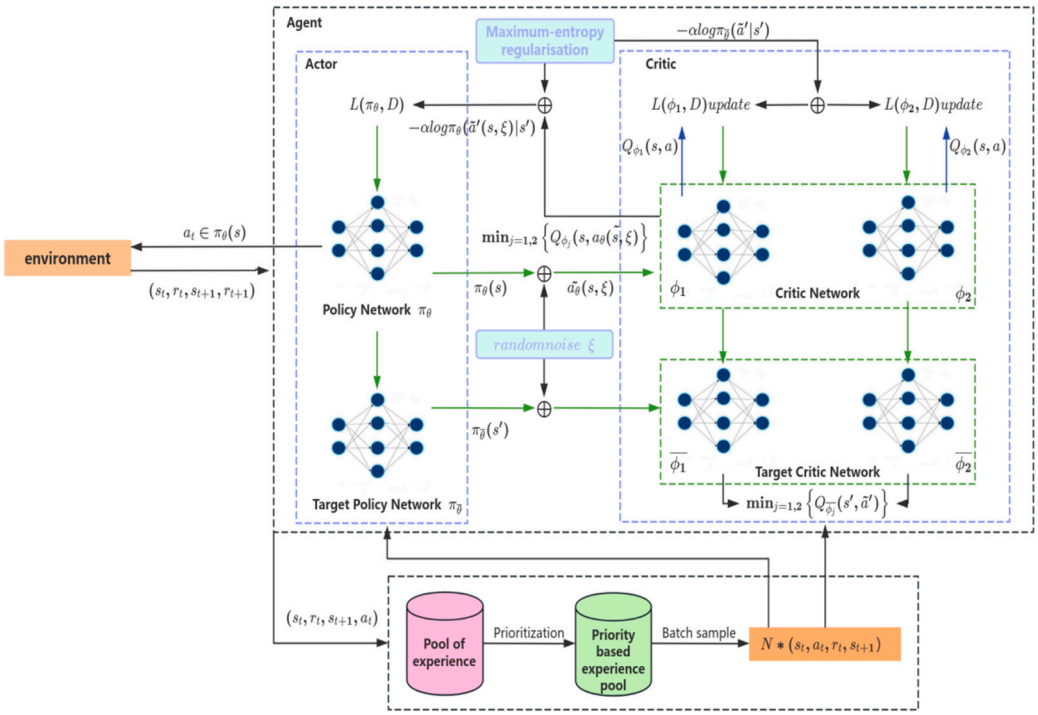


Figure 3. PER-SAC network structure diagram.

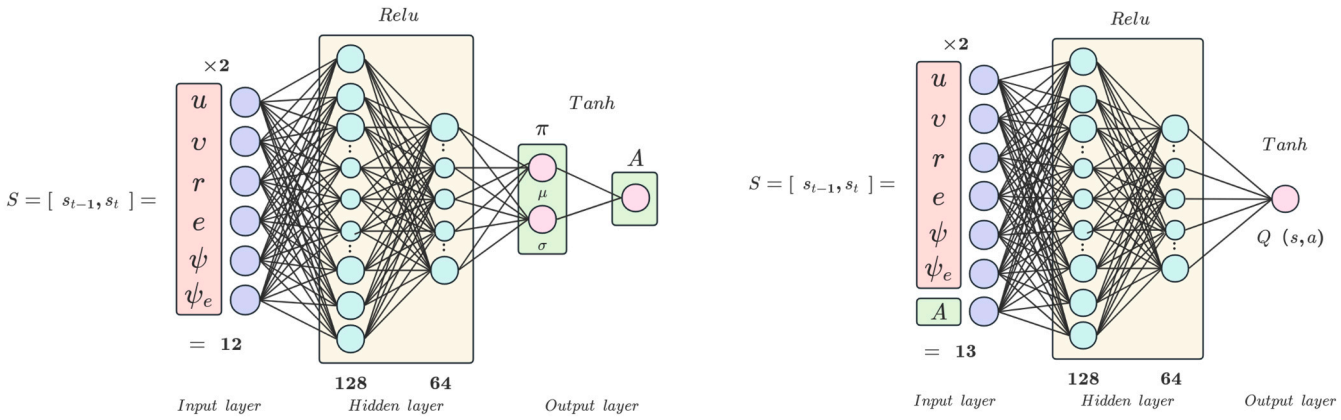
3.3. Network Architecture Design

The algorithm used in this study is designed based on the Actor–Critic (AC) framework, requiring specific designs for both the Actor and Critic networks according to the complexity of the control task, including the type of neural network, neurons, and activation functions.

In the AC framework, neural networks are primarily used to approximate the policy and value functions, with common types including the Multilayer Perceptron (MLP), Recurrent Neural Network (RNN), the Long Short-Term Memory Network (LSTM), and the Convolutional Neural Network (CNN). This study employed MLP, also known as the Artificial Neural Network (ANN), which consists of an input layer, hidden layers, and an output layer, with fully connected layers between them.

Neural networks have two important metrics: width and depth. Increasing depth enables the network to better approximate complex features, but may also lead to overall performance degradation. Increasing the width enhances the learning capability of each layer but raises computational costs. Considering the complexity of the trajectory tracking task while balancing these two metrics, the network adopted a uniform structure with two hidden layers containing 128 and 64 neurons, respectively. Activation functions introduce nonlinearity to neural networks, with common choices, including Tanh, Sigmoid, ReLU, and LeakyReLU, whose output ranges are confined to $[-1, 1]$, $[0, 1]$, $[0, +\infty)$, and $(-\infty, +\infty)$, respectively. For trajectory control tasks, where action outputs may be positive or negative but within a limited range, the output layer uses the Tanh function, while hidden layers employ ReLU to achieve rapid convergence and mitigate gradient vanishing.

The detailed network structure is shown in Figure 4.



(**a**) Actor Network (**b**) Critic Network

Figure 4. Network architecture diagram.

The Actor network took 12 consecutive states as the input at each timestep, processed them through two hidden layers activated by ReLU functions, and finally output actions sampled from the policy distribution through a Tanh activation function. The Critic network, at each timestep, took the same 12 states plus 1 action as input, processed them through two ReLU-activated hidden layers, and output the state–action value Q through a Tanh activation function.

3.4. Design of Reward Function Module

The reward function is the core component of the reinforcement learning algorithm, which determines the behavior of agents and affects learning efficiency and performance.

This paper combined PER with the reward function to optimize ship task training and improve the efficiency of policy fitting.

When designing the reward function R, we typically categorize it into two types based on its distribution characteristics: dense rewards and sparse rewards. Sparse rewards refer to only providing positive or negative feedback when the agent performs critical actions, while giving zero reward at other times. In contrast, dense rewards evaluate and provide feedback for every action taken by the agent. In practical applications, a balance between these two reward mechanisms is necessary. Sparse rewards may lead to insufficient feedback, making it difficult to discover optimal actions, whereas dense rewards provide timely guidance. Therefore, the design should incorporate both dense and sparse rewards while considering the impact of positive and negative rewards on the agent.

In this study, to ensure that the vessel simultaneously considers heading error and tracking error during trajectory following, we designed both positive and penalty dense rewards. Specifically, tracking error was associated with positive rewards to encourage exploration, while heading error was assigned negative values to prompt timely adjustments. The reward function was formulated as follows:

$$R_1 = -|\psi_e| + \alpha \times 0.5^{|e|} \tag{24}$$

In the equation, α, β represent the weight parameters.

The trajectory tracking control route of the ship is discrete points, so there are straight lines between the discrete points. A reward function was added as guidance on the straight line. The reward item is:

$$R_2 = \zeta + \text{growth_rate} * x, x = \text{len_percent} \tag{25}$$

$$\text{growth_rate} = 1 + \left(\frac{x}{0.1} + \kappa\right) \tag{26}$$

where len_percent is the path completion degree, and its value range is $(0, 1)$. ζ, κ is the function base value and the function base growth rate coefficient, respectively. See Figure 5 for details.

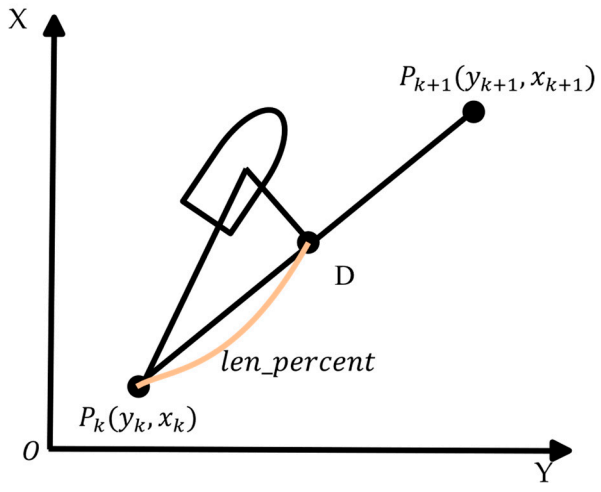


Figure 5. len_percent schematic diagram.

In the track control, the ultimate goal is to complete the tracking of the target trajectory. In order to avoid the unmanned ship from deviating too much from the design trajectory and reduce the training cycle time, a penalty value was given when the error was too large, and the sparse motion boundary reward was set:

$$R_3 = \begin{cases} -100, & \text{if } |e| > 10 \\ 0, & \text{else} \end{cases} \tag{27}$$

Therefore, the overall reward for track control is $R_{toal} = R_1 + R_2 + R_3$.

The deep reinforcement learning ship trajectory tracking control system has the problem of long training cycles. The orthogonal experiment method can be used to analyze multiple weights without losing the experimental information and reducing the number of experiments as much as possible.

In order to consider the influence of various factors, the $L_9(3^4)$ orthogonal table was introduced, and four factors were considered. Each factor had three levels. See Table 2 for each factor and level. The minimum number of rounds required for the reward and incentive value to gradually converge to a relatively stable level in the ship training task was calculated, and the results were analyzed directly. See Table 3 and Figure 6.

Table 2. Factors and levels of the orthogonal table.

	Factor A	Factor B	Factor C	Factor D
Horizontal Serial Number	**Bow Error**	**Distance Error**	**Track Completion Degree**	**Angle**
1	−1.5	0.1	0.2	45
2	−5	0.5	1	90
3	−10	2	2	135

Table 3. Intuitive analysis results.

	Factor A	Factor B	Factor C	Factor D
K_{1J}	233	265	191	262
K_{2J}	117	138	167	157
K_{3J}	178	125	170	108.9
R_J	116	140	24	153.1

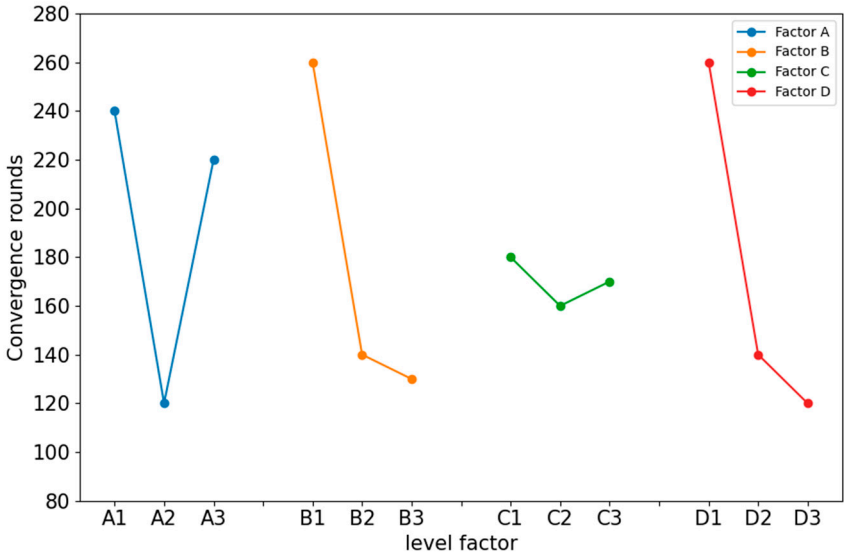


Figure 6. Visual analysis diagram.

In the Table 3, R_J is the range, K_{iJ} is the mean of factor j at level I, abbreviated as horizontal mean.

It can be observed from the above figure and table that although the ranges of each factor differed, they were all within the same order of magnitude. The influence of factor

A and factor B on the training cycle of the algorithm was basically the same. Factor D had a certain influence, but its influence degree was slightly lower than that of factor A and factor B. Factor C played a guiding role in reward and had a certain impact, but it cannot be compared with heading and distance errors. To sum up, the optimal parameter combination of the reward function was $A_2B_3C_2$.

Factor A represents the heading error. From an intuitive analysis, it is clear that A did not converge, so further analysis of A was conducted while keeping factors B and C fixed, as shown in Table 4. The result showed that A reached its optimal value around level 2. Factor B represents the distance error, and its weight coefficient showed convergence. Although factor C had a smaller influence than A and B, proper track completion can promote the stability and efficiency of the training process. As for factor D, it can be observed that the controller showed decreasing difficulty in the training angle.

To sum up, the final coefficient selection combination of the controller reward weight was $A_2B_3C_2$.

When a USV executes track control training, the process of obtaining the reward function is shown in Figure 7.

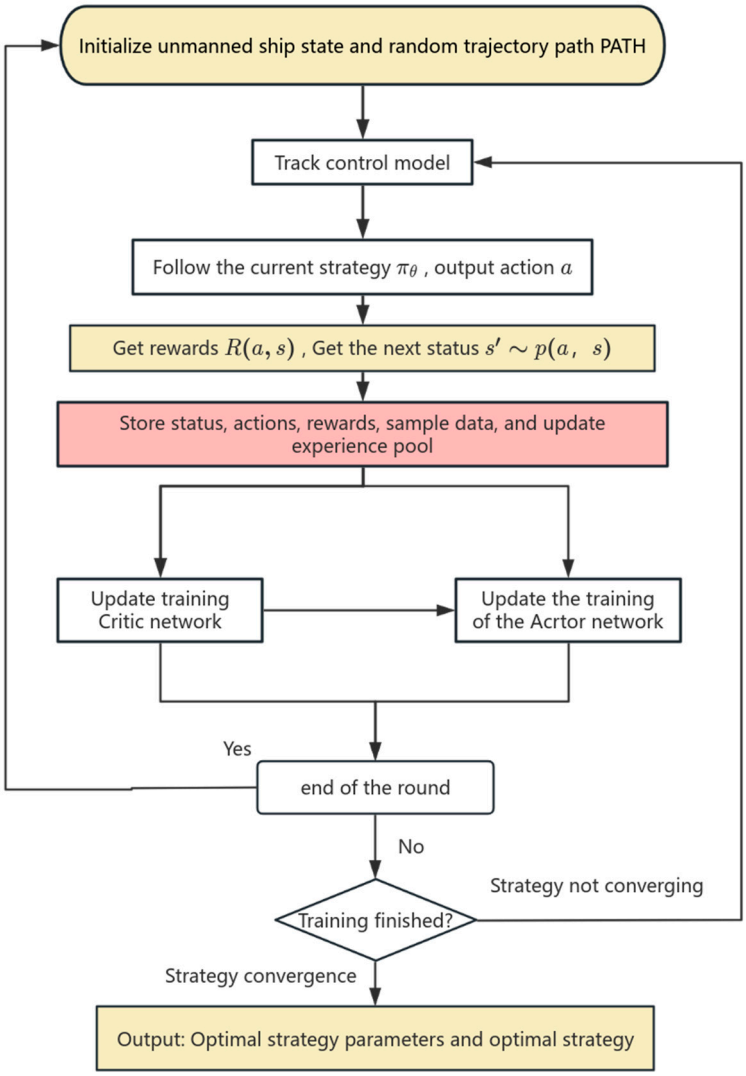


Figure 7. Training process and reward acquisition.

Table 4. Factor a convergence analysis.

PATH = [(−80,−80), (−50,20), (20,40), (45,10), (60,55)]	
Factor A	**Convergence Rounds**
−5	160
−10	395
−30	2893
−50	5000 + (Not converging)

4. Analysis of Simulation Results

This paper used the PyBullet simulation platform (Version: 3.2.5) and the Gym (Version: 0.19.0) learning framework to build a three-dimensional visual simulation environment for unmanned ship motion control. Simulation experiments were designed to verify the feasibility of the ship trajectory tracking controller based on deep reinforcement learning in various water conditions, and the convergence efficiency of the new controller was verified by comparing it with other RL algorithms.

The parameter settings of the training phase of the deep reinforcement learning algorithm are shown in Table 5 below.

Table 5. SAC training parameter settings.

Parameter	Value
Discount factor γ	0.99
Actor network learning rate	0.0003
Critic network learning rate	0.0003
Replay memory size	100,000
Neural network hidden layer	2
Number of neurons in each hidden layer	128
Action noise standard deviation	0.1
Maxepisodes M	40×10^7

4.1. Network Training and Comparative Test

Before testing, the controller needed to be pre-trained to ensure that the SAC algorithm had learned the basic control strategy. After completing the pre-training, the training process was analyzed and evaluated. Various algorithms were then compared with the training path shown in Figure 8 for the evaluation results. The initial state of the unmanned vessel was set to $\eta = \left[80, 80, 0^{\circ}\right]^{T}$, and the path was PATH = [(−80,−80), (−40,0), (0,0), (40,40), (80,40)], with the units in meters.

For the convenience of analysis, the reward values of each algorithm were processed using a moving average. The four algorithms analyzed were: DDPG, PPO, TD3, and PER-SAC. In terms of convergence speed, PER-SAC and DDPG converged quickly, followed by TD3, while PPO was the slowest, starting to converge at around 580 episodes. Among the four algorithms within the limit of 1000 rounds, only PER-SAC converged stably. PER-SAC finally began to converge and stabilize at 330 rounds; DDPG began to converge at 110 rounds and reached its highest value at 580 rounds, but ultimately did not converge well and experienced significant fluctuations. TD3 converged after 360 rounds; PPO started to converge last, but ultimately did not have stable convergence. From the reward values, it can be observed that PER-SAC and PPO exhibited better stability, with the reward values of PER-SAC slightly higher than those of PPO. DDPG, in terms of the reward function trend, was able to converge at a rate comparable to PER-SAC, but its volatility was the most significant, lacking stability. In summary, based on both convergence speed and reward values, it can be concluded that the PER-SAC algorithm, with its prioritized experience

replay mechanism, is able to make better use of experience and eventually converge to a better policy, demonstrating the ability to promote stability and efficiency in the learning process. Based on the completed episodes, we analyzed the task completion rate and the average and maximum values of the task completion rewards, as shown in Table 6.

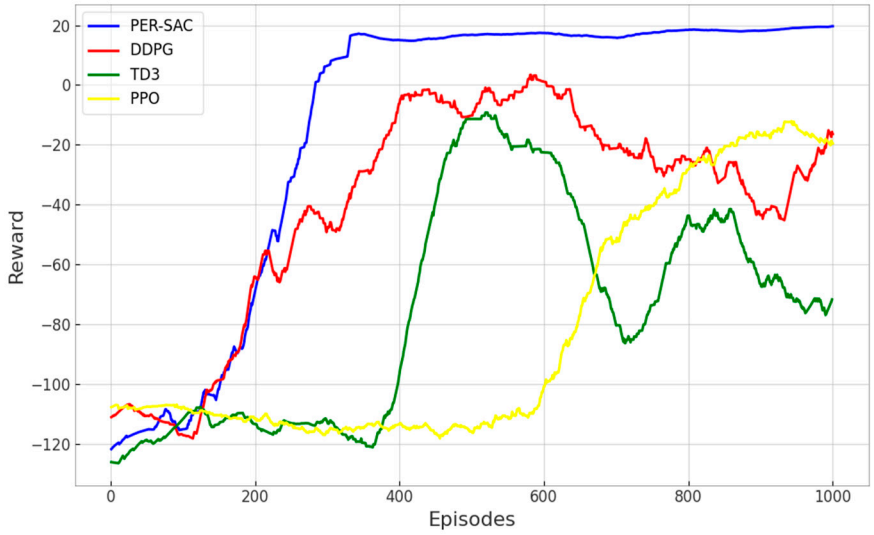


Figure 8. Comparison of rewards for trajectory control training.

Table 6. Training results.

Algorithm	Task Completion Rate	Average Value	Maximum Value
PER-SAC	79.4%	16.55	25.88
DDPG	57.6%	12.43	20.56
TD3	33.5%	−1.07	10.21
PPO	26%	2.62	16.59

During the evaluation of the algorithm, the stability of the loss function curve based on the neural network is also an important evaluation metric, as it directly reflects the quality of the network architecture. The loss values generated during the training process were stored, and the results are depicted in Figures 9 and 10.

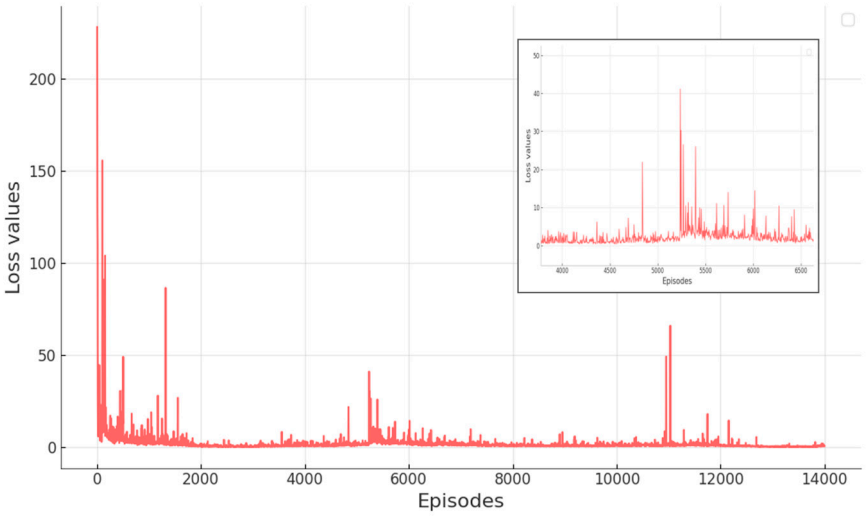


Figure 9. The loss curve of the Critic network in the PER-SAC algorithm.

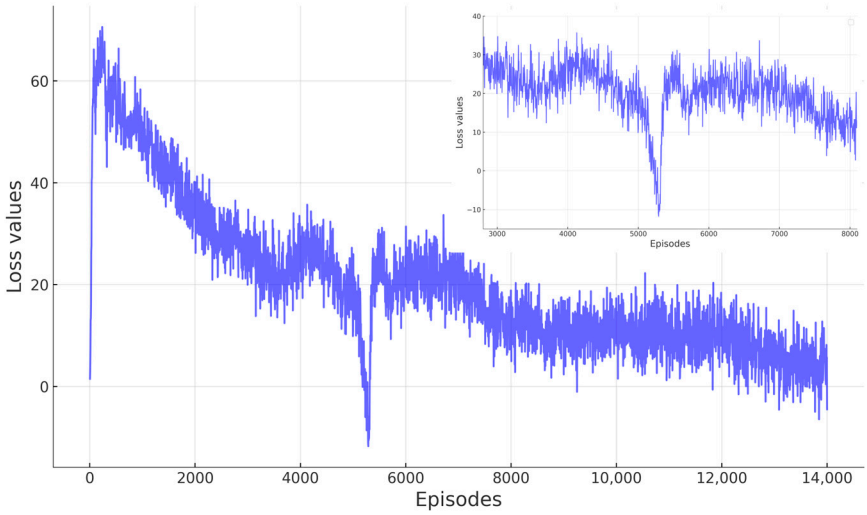


Figure 10. Loss curve of the Actor network in the PER-SAC algorithm.

At the beginning of training, the Critic network exhibited a relatively high loss value, with some values exceeding 200. This indicates that, in the early stages of training, the Critic network has a very limited understanding of the environment, and the predicted Q-values deviate significantly from the actual values. At the same time, the loss value fluctuated greatly in the early stages, reflecting the network's attempt to learn and adjust its parameters. The significant decrease in loss values during the initial phase suggests that the learning rate is appropriate and that gradient descent is being effectively applied. After 2000 iterations of learning, the loss value gradually decreased, with some fluctuations, but the overall trend was downward. Particularly after 13,000 steps, the loss value approached zero, indicating that the Critic network had largely converged and learned an effective policy. Occasional fluctuations during the convergence process were due to the algorithm exploring new strategies. However, overall, the loss value remained at a low level, suggesting that the network is now able to predict the environment's rewards with relatively high accuracy.

The evaluation of the Actor network's loss curve showed that, in the first 3500 episodes, the loss curve exhibited a significant downward trend. Around episode 5700, a sharp drop occurred, but it stabilized afterward, suggesting that the network was continuously learning and improving. As training progressed, this volatility persisted, with some noticeable spikes, indicating that the model was constantly adjusting its policy to adapt to changes in the environment. After episode 8000, the loss value gradually stabilized, and the network began to converge, with the convergence speed becoming more stable.

4.2. Verification Simulation

After completing the training, the trained model was saved for subsequent evaluation. The test results for the PER-SAC algorithm were designed for single-objective, multi-objective, and special route scenarios. The initial position of the ship in the simulation was set to (−80,−80), with an initial heading angle of 45°.

Simulation Experiment 1: The straight-line and zigzag trajectories were designed to observe the controller's tracking performance and response to sharp angle changes. The trajectory coordinates were [(−80,−80), (80,80)] for the straight line and [(−80,−80), (−40,0), (0,0), (40,−80), (80,80)] for the zigzag pattern, with all units in meters, The result is shown in Figure 11.

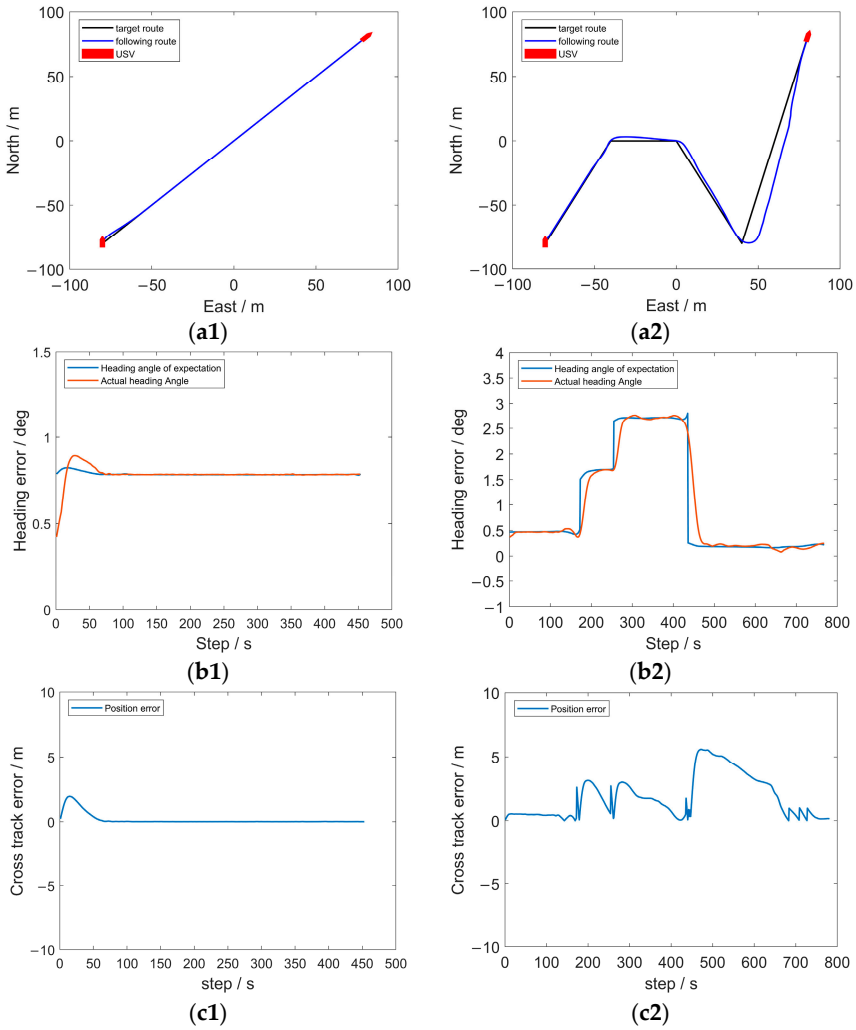


Figure 11. Experimental results of straight and sharp path tracking: (**a1**,**a2**) Path tracking visualization; (**b1**,**b2**) Comparison chart of heading angle error; (**c1**,**c2**) Distance error diagram.

From the tracking results of the two trajectories, it can be seen that the ship was able to complete the trajectory tracking task within a certain error margin, proving the feasibility of the controller. Although there was a slight deviation in the path at the beginning, this was due to a 45-degree deviation between the ship's heading and the preset path direction, which aligned with the ship's motion dynamics. The ship exhibited error fluctuations at the corners, which were consistent with the dynamic constraints of the USV. From the heading error tracking chart, it is evident that the ship can essentially achieve the desired heading angle with minimal error. The error fluctuations corresponded to specific steps, indicating that the fluctuations occurred at the USV's starting point and path turning points, which were consistent with the dynamic constraints, and the error quickly converged to zero.

Simulation Experiment 2: To simulate the trajectory of a traditional vessel, the designed trajectory points were (−80,−80), (−50,−50), (−30,50), (30,50), (50,−50) and (−80,−80), (−30,−20), (−30,60), (40,10), (80,80). The tracking performance and result are shown in Figure 12.

From the test results and the error table, it can be concluded that, similar to the verification results in Experiment 1, the USV, based on the execution strategy trained in this paper, was able to follow the curved path effectively. Although the USV experienced slight yaw, and there were changes in motion parameters at the start and end points of the

path, it quickly returned to the target instructions and the target path. The control error was within an acceptable range.

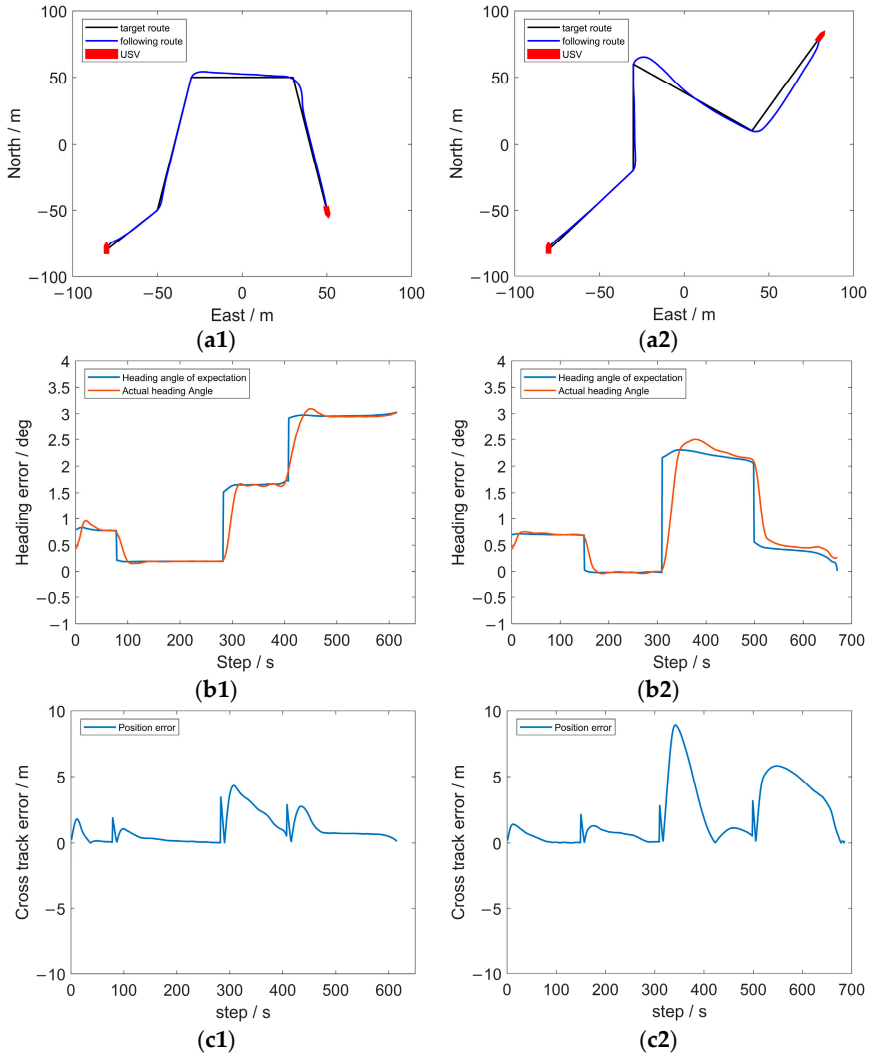


Figure 12. Experimental results of conventional path tracking: (**a1**,**a2**) Path tracking visualization; (**b1**,**b2**) Comparison chart of heading angle error; (**c1**,**c2**) Distance error diagram.

4.3. Characteristic Water Area Simulation

To validate the performance of the controller in different actual water environments, it was tested in various scenarios. The wide-area water search experiment simulated the ship's trajectory tracking ability in vast and complex water regions, while also evaluating the controller's path-following capability under simulated obstacle avoidance conditions. The river channel search traversal experiment, on the other hand, focused on narrow, constrained path tracking tasks, aiming to examine the performance of the control system in following a specific channel under high precision requirements. The experimental results are depicted in Figure 13, the yellow dashed line in Figure 13a represents the simulated water boundary.

Through the simulation analysis of the two classic paths mentioned above, it is clear that the vessel was able to complete the trajectory tracking task within a certain error margin, demonstrating the feasibility of this controller. In Figure 13a1,a2, by observing the ship's path tracking simulation image, it can be seen that the vessel completed the trajectory control with a small error. Although there was some deviation at the trajectory switching points, it followed the laws of ship motion. Comparing the heading angle

tracking graph, it is evident that the controller was still able to track the optimal heading in narrow water areas.

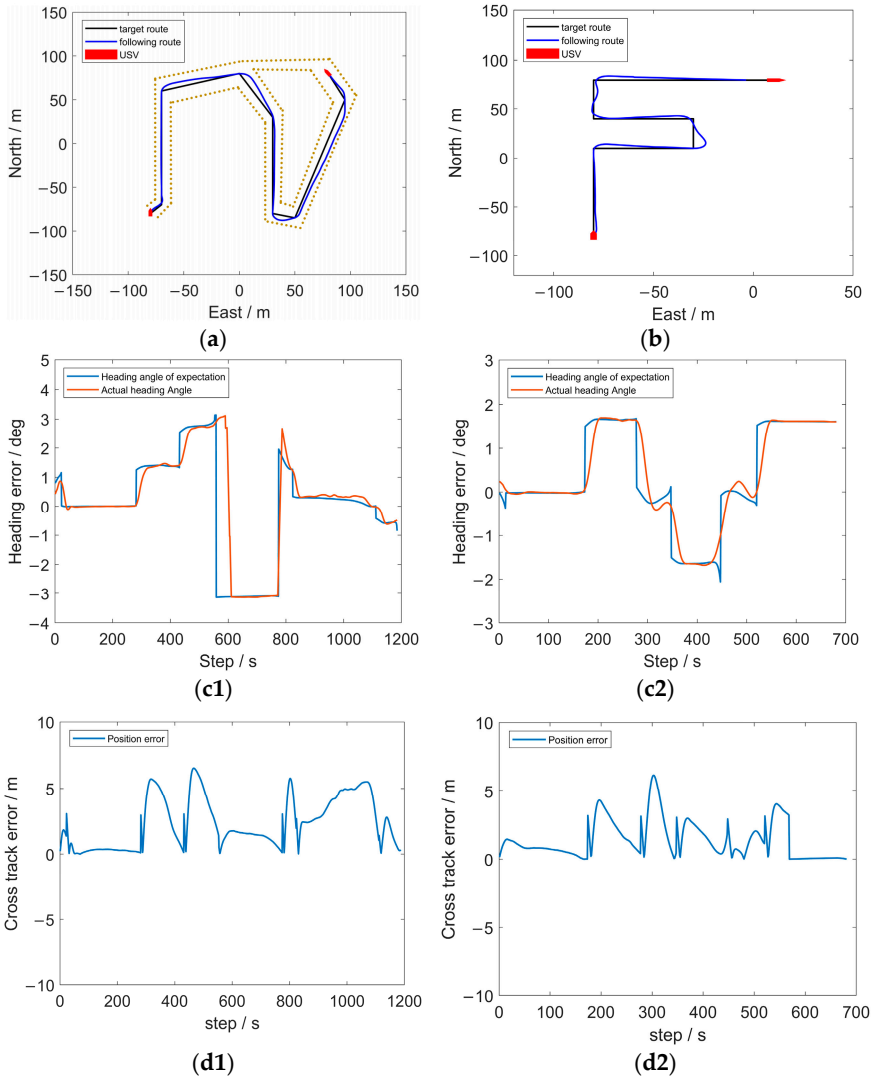


Figure 13. Characteristic water area path tracking experiment: (**a**) River channel search and traversal experiment; (**b**) Wide area water search experiment; (**c1**,**c2**) Comparison chart of heading angle error; (**d1**,**d2**) Distance error diagram.

The river channel search traversal experiment was selected to simulate the ship's rapid turning control when encountering obstacles. The vessel was able to quickly complete the turn and avoid obstacles. Even with the ship's continuous sharp turns, it still completed the trajectory task within a certain level of precision.

In summary, the controller demonstrates excellent performance in both complex water regions and narrow channels. It effectively avoids obstacles and maintains heading, showing high precision and stability. These results provide strong validation for the controller's performance in real-world applications.

5. Conclusions

This paper addressed the issues of long training periods and sparse rewards in the USV trajectory tracking task by designing a controller that integrates the Soft Actor–Critic algorithm and the prioritized experience replay strategy. Additionally, the reward function module was designed, and simulation comparison experiments were conducted. The main contributions are as follows:

- To tackle the problem of sparse rewards in the USV trajectory control task, the prioritized PER algorithm is introduced. This enhanced sample utilization and training stability, while also reducing the interaction cost between the agent and the environment.
- The reward module was designed with specific attention given to the formulation of reward components and tuning appropriate weight parameters. This helps promote a balance between exploration and exploitation, ensuring the stability and efficiency of the learning process. It also influences the convergence speed of the algorithm and the quality of the policy.
- The performance of the PER-SAC controller was compared with other reinforcement learning controllers in various aspects to validate the feasibility of the proposed controller.

The research showed that the PER-SAC controller, compared to other RL controllers, can improve training efficiency and achieve faster convergence of the policy. However, further research is still required, and work will be continued in the following aspects. In the training environment, the simulation used in this study was based on a static water body, without considering the impact of environmental factors such as wind, waves, and currents on the controller. Additionally, due to certain constraints, no experimental validation in real-world water bodies was conducted in this study. Conducting field experiments in real water environments is crucial for improving the algorithm and validating the model. In future work, we will account for external uncertain environmental disturbances, design a more robust controller with stronger anti-disturbance capabilities, and conduct experiments in real water bodies.

Author Contributions: Conceptualization, L.X. and J.C.; methodology, L.X. and J.C.; software, J.C. and Z.H.; validation, Z.H., S.Z. and S.X.; formal analysis, Z.H., L.S. and S.Z.; investigation, L.X., J.C. and Z.H.; resources, L.X. and Z.H.; data curation, J.C.; writing—original draft preparation, J.C.; writing—review and editing, Z.H., L.S. and S.Z.; visualization, L.X., J.C. and S.X.; supervision, L.X. and Z.H.; project administration, L.X.; funding acquisition, L.X. and Z.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key R&D Program of China (2022YFC2806600, 2022YFC2806604) and the Jiangsu Marine Technology Innovation Center Industry Incubation Project (N0. MTIC-2023-RD-0002).

Data Availability Statement: The datasets presented in this article are not readily available because part of the data still need to be studied later. Requests to access the datasets should be directed to cjrwn513@163.com.

Conflicts of Interest: Author Sheng Zhang was employed by the company Zhenjiang Yuanli Innovation Technology Co., Ltd. Author Lin Shi was employed by the company China Construction Civil Infrastructure Co., Ltd. (CSCIC). The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Zhang, Y.T.; Xu, L.X.; Cao, L.; Hong, Z.; Wang, M.; Li, N. Current status and prospects of technical development for intelligent ships. *Ship Eng.* **2023**, *45* (Suppl. S1), 185–187+192.
2. Gong, J.Y.; Xu, J.Y.; Hong, Z.C. Enhancing motion forecasting of ship sailing in irregular waves based on optimized LSTM Model and Principal component of wave-height. *Front. Mar. Sci.* **2025**, *12*, 1497956. [CrossRef]
3. Pei, Z.Y.; Dai, Y.S.; Li, L.G.; Jin, J.; Shao, F. A review of motion control methods for unmanned surface vehicles. *Mar. Sci.* **2020**, *44*, 153–162.

4. Chen, X.; Zheng, J.; Li, C.; Wu, B.; Wu, H.; Montewka, J. Maritime traffic situation awareness analysis via high-fidelity ship imaging trajec tory. *Multimed. Tools Appl.* **2024**, *83*, 48907–48923. [CrossRef]
5. Mehrzadi, M.; Terriche, Y.; Su, C.L.; Othman, M.B.; Vasquez, J.C.; Guerrero, J.M. Review of dynamic positioning control in maritime microgrid systems. *Energies* **2020**, *13*, 3188. [CrossRef]
6. Wang, Y.H.; Zhang, X.Y.; Wang, C.L. A review of dynamic positioning control technology for ship mooring. *J. Harbin Eng. Univ.* **2023**, *44*, 172–180.
7. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *IEEE Signal Process. Mag.* **2017**, *34*, 26–38. [CrossRef]
8. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Hassabis, D. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
9. Chen, Z.; Zhang, Y.; Zhang, X. Deep reinforcement learning for ship maneuvering control in complex marine environments. *IEEE Trans. Neural Netw. Learn. Syst.* **2021**, *32*, 2026–2037.
10. Fossen, T.I.; Breivik, M.; Skjetne, R. Line-of-sight path following of underactuated marine craft. *IFAC Proc. Vol.* **2003**, *36*, 211–216. [CrossRef]
11. Fossen, T.I.; Pettersen, K.Y.; Galeazzi, R. Line-of-sight path following for Dubins paths with adaptive sideslip compensation of drift forces. *IEEE Trans. Control Syst. Technol.* **2014**, *23*, 820–827. [CrossRef]
12. Do, K.D.; Jiang, Z.P.; Pan, J. Robust adaptive path following of underactuated ships. *Automatica* **2004**, *40*, 929–944. [CrossRef]
13. Beard, R.W.; McLain, T.W. Multiple vehicle coordination using Lyapunov methods. *Proc. IEEE* **2003**, *91*, 1059–1075.
14. Zhang, X.; Shen, Z.; Bi, Y. Adaptive dynamic surface sliding mode control for ship trajectory tracking with disturbance ob-server. *Ship Eng.* **2018**, *40*, 81–87.
15. Jiao, J.; Bao, D.; Hu, Z. Ship trajectory tracking based on preset performance adaptive neural networks. *J. Huazhong Univ. Sci. Technol. (Nat. Sci. Ed.)* **2022**, *50*, 77–82.
16. Qi, L.; Li, H.; Zhang, Q. Ship trajectory tracking backstepping control based on high-gain observer and RBF neural network. *Control Decis.* **2017**, *32*, 145–152.
17. Shen, Z.; Bi, Y.; Wang, Y.; Guo, C. Adaptive recursive sliding mode control for trajectory tracking of input-output constrained ships. *Control Theory Appl.* **2020**, *37*, 1419–1427.
18. Liu, L.; Wang, D.; Peng, Z.; Wang, H. Predictor-based LOS guidance law for path following of underactuated marine surface vehicles with sideslip compensation. *Ocean Eng.* **2016**, *124*, 340–348. [CrossRef]
19. Chen, D.; Zhang, J.; Li, Z. A novel fixed-time trajectory tracking strategy of unmanned surface vessel based on the fractional sliding mode control method. *Electronics* **2022**, *11*, 726. [CrossRef]
20. He, Y.X.; Li, Z.R.; Mou, J.M.; Hu, W.X.; Li, L.L.; Wang, B. Collision-avoidance path planning for multi-ship encounters considering ship manoeuvrability and COLREGs. *Transp. Saf. Environ.* **2021**, *3*, 103–113. [CrossRef]
21. Zhao, L.; Roh, M.I.; Lee, S.J. Control method for path following and collision avoidance of autonomous ship based on deep reinforcement learning. *J. Mar. Sci. Technol.* **2019**, *27*, 1.
22. Song, D.; Gan, W.; Yao, P.; Zang, W.; Qu, X. Surface path tracking method of autonomous surface underwater vehicle based on deep reinforcement learning. *Neural Comput. Appl.* **2023**, *35*, 6225–6245. [CrossRef]
23. Zhang, F.; Li, B.; Ruan, Z. Navigation control of unmanned boats based on deep reinforcement learning. *Meas. Technol.* **2018**, *38* (Suppl. S1), 207–211.
24. Zhu, K.; Huang, Z.; Wang, X.M. Intelligent ship track tracking control based on deep reinforcement learning. *Chin. J. Ship Res.* **2021**, *16*, 105–113.
25. Wang, Y. Modeling and Path Tracking Control of Unmanned Surface Vehicles. Master's Thesis, Zhejiang University, Hangzhou, China, 2019.
26. Zhao, Y.; Qi, X.; Ma, Y.; Li, Z.; Malekian, R.; Sotelo, M.A. Path following optimization for an underactuated USV using smoothly-convergent deep rein-forcement learning. *IEEE Trans. Intell. Transp. Syst.* **2020**, *22*, 6208–6220. [CrossRef]
27. Chen, X.Q.; Hu, R.Y.; Luo, K.; Wu, H.F.; Biancardo, S.A.; Zheng, Y.W.; Xian, J.F. Intelligent ship route planning via an A∗ search model enhanced double-deep Q-network. *Ocean Eng.* **2025**, *327*, 120956. [CrossRef]
28. Cui, Z.W.; Guan, W.; Zhang, X.K.; Zhang, G.Q. Autonomous collision avoidance decision-making method for USV based on ATL-TD3 algorithm. *Ocean Eng.* **2024**, *312 Pt 3*, 119297. [CrossRef]
29. Wang, Z.; Wu, Y.; Song, W. Multi-path Following for Underactuated USV Based on Deep Reinforcement Learning. In Proceedings of the 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022), Xi'an, China, 23–25 September 2022; Fu, W., Gu, M., Niu, Y., Eds.; Lecture Notes in Electrical Engineering. Springer: Singapore, 2023; Volume 1010.

30. Shi, H.B.; Sun, Y.R.; Li, G.Y. Model-based DDPG for Motor Control. In Proceedings of the 2017 IEEE International Conference on Progress in Informatics and Computing (PIC 2017), Nanjing, China, 15–17 December 2017; pp. 284–288.
31. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized Experience Replay. *arXiv* **2016**, arXiv:1511.05952.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.