

Article

A Novel Dynamically Adjusted Entropy Algorithm for Collision Avoidance in Autonomous Ships Based on Deep Reinforcement Learning

Guoquan Chen , Zike Huang, Weijun Wang * and Shenhua Yang * 

Navigation College, Jimei University, Xiamen 361021, China; cgq0802@foxmail.com (G.C.); zk_huang07@163.com (Z.H.)

* Correspondence: wjwang@jmu.edu.cn (W.W.); yangshh@163.com (S.Y.)

Abstract: Decision-making for collision avoidance in complex maritime environments is a critical technology in the field of autonomous ship navigation. However, existing collision avoidance decision algorithms still suffer from unstable strategy exploration and poor compliance with regulations. To address these issues, this paper proposes a novel autonomous ship collision avoidance algorithm, the dynamically adjusted entropy proximal policy optimization (DAE-PPO). Firstly, a reward system suitable for complex maritime encounter scenarios is established, integrating the International Regulations for Preventing Collisions at Sea (COLREGs) with collision risk assessment. Secondly, the exploration mechanism is optimized using a quadratically decreasing entropy method to effectively avoid local optima and enhance strategic performance. Finally, a simulation testing environment based on Unreal Engine 5 (UE5) was developed to conduct experiments and validate the proposed algorithm. Experimental results demonstrate that the DAE-PPO algorithm exhibits significant improvements in efficiency, success rate, and stability in collision avoidance tests. Specifically, it shows a 45% improvement in success rate per hundred collision avoidance attempts compared to the classic PPO algorithm and a reduction of 0.35 in the maximum collision risk (CR) value during individual collision avoidance tasks.



Citation: Chen, G.; Huang, Z.; Wang, W.; Yang, S. A Novel Dynamically Adjusted Entropy Algorithm for Collision Avoidance in Autonomous Ships Based on Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2024**, *12*, 1562. <https://doi.org/10.3390/jmse12091562>

Academic Editor: Sergei Chernyi

Received: 27 July 2024

Revised: 18 August 2024

Accepted: 1 September 2024

Published: 5 September 2024



Copyright: © 2024 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/>).

Keywords: unmanned ship; deep reinforcement learning; dynamically adjusted entropy; collision avoidance; COLREGs

1. Introduction

As the global economy rapidly expands, the number of maritime vessels has significantly increased, marking them as barometers of the global economic climate [1]. This growth has led to increasingly crowded maritime routes, escalating concerns about navigational safety. Against this backdrop, the development of effective collision avoidance mechanisms for ships has become a critical issue in ensuring navigational safety in complex maritime environments. The International Regulations for Preventing Collisions at Sea (COLREGs) play a pivotal role in this [2]. However, statistics indicate that over 80% of maritime collisions are caused by human error, including the crew's failure to fully and effectively comply with the COLREGs at critical moments [3]. The primary research goal for autonomous maritime collision avoidance algorithms is to enable ships to make effective collision avoidance decisions in complex environments, thus minimizing or preventing accidents.

Various algorithms for ship collision avoidance automation have been proposed, such as the A* algorithm [4], genetic algorithm (GA) [5], and artificial potential field (APF) [6]. These classic path-planning techniques have supported the preliminary exploration of intelligent maritime navigation. However, these algorithms, while useful, encounter several limitations, including high computational complexity, strong dependence on environmental models and heuristic functions, and a lack of learning capabilities. These issues are

particularly problematic in scenarios involving unknown environments, moving obstacles, and other unquantifiable factors that are crucial in unmanned maritime path planning.

In contrast, recent advancements in automation and artificial intelligence have introduced new avenues for addressing these challenges in maritime collision avoidance. Deep reinforcement learning (DRL) algorithms, known for their adaptability and versatility, are expected to address these shortcomings and are becoming a focal point in collision avoidance research. Guo Siyu et al. [7] employed a combination of reinforcement learning algorithms, such as deep deterministic policy gradient (DDPG) and APF, to enhance learning efficiency and convergence speed. However, their research did not account for ship models and actual environmental conditions, necessitating further validation in complex settings. Yin Cheng et al. [8] designed a comprehensive reward function for deep reinforcement learning tailored to underactuated unmanned marine vessels, taking into account environmental disturbances and vessel motion characteristics. They successfully implemented path planning in complex maritime areas with numerous static obstacles; however, the applicability of this method in scenarios with dynamic obstacles remains to be evaluated. Zhijian Huang et al. [9] adjusted the standard Deep Q-Network (DQN) architecture to better handle the complexity and partial observability of marine environments, although further validation is required to assess the algorithm's performance in dynamic and multi-obstacle settings. Xu Xinli et al. [10] developed a COLREGs-compliant intelligent collision avoidance algorithm (CICA) that tracks and updates network weights. The design of the action space involves three discrete values, resulting in a lack of continuity in actions.

Compared to other DRL algorithms, the proximal policy optimization (PPO) algorithm offers greater stability and efficiency in managing continuous action spaces, making it particularly well-suited for handling the continuous control challenges in ship maneuvering [11]. Qianhao Xiao et al. [12] introduced a novel distributed sampling strategy that enhances the balance and diversity of sample collection through regional segmentation. They incorporated a Beta strategy to address action boundary issues in continuous action spaces, thereby increasing the success rate of path planning and reward accumulation. Eivind Meyer et al. [13] proposed a new observation vector and reward function design, including a feasibility pooling algorithm for real-time sensor data dimension reduction. They also introduced a reward trade-off parameter λ , enabling the agent to dynamically adjust its navigation strategy based on the current policy. Wei Guan et al. [14] proposed an improved algorithm that integrates generalized advantage estimation into the loss function of the PPO algorithm, verifying that unmanned ships can autonomously navigate and avoid collisions without human intervention. Chuanbo Wu et al. [15] developed a collision avoidance path planning method combining the dynamic window approach with the PPO algorithm, which has been validated through simulation experiments to be effective in near-shore navigation.

Although the PPO algorithm has made significant progress in maritime collision avoidance, there remains room for improvement in exploration efficiency and policy performance. Particularly, it exhibits limitations in balancing exploration and exploitation. In complex maritime environments, the algorithm may become trapped in local optima, limiting further performance enhancements. To overcome this challenge, introducing appropriate exploration mechanisms is crucial, especially when dealing with complex or high-dimensional decision spaces. Such mechanisms can help the algorithm escape local optima and explore a broader solution space, thereby improving overall decision quality and strategy robustness.

Entropy regularization is an effective method to avoid local optima by encouraging agents to engage in broader exploration during training, fostering the discovery of new possibilities, and maintaining the randomness of strategies [16,17]. However, applying this technique to complex environments at sea and balancing entropy parameters to achieve optimal performance between exploration and exploitation remains a nuanced challenge. To address these issues, this paper introduces an improved PPO algorithm with dynamically adjusted entropy (DAE-PPO). This research introduces a quadratically decreasing entropy

approach designed to maintain a high entropy coefficient early in training, allowing for extensive exploration, thus preventing convergence to local optima and adapting to more complex training environments. As training progresses, the entropy coefficient gradually decreases, mitigating abrupt changes in exploration intensity, ensuring a smooth transition during the training process, reducing instability, and accelerating model convergence. The main contributions of this paper are as follows:

- (1) An enhanced proximal policy optimization (PPO) algorithm based on dynamic entropy adjustment is proposed. This approach optimizes the entropy regularization framework to improve the exploration efficiency and policy performance of the PPO algorithm without introducing additional hyperparameters. A PPO network framework specifically designed for maritime collision avoidance has been developed, with various improvements analyzed and compared based on a comprehensive training environment.
- (2) A collision risk (CR) metric is introduced into the reward function, based on the distance to the closest point of approach (DCPA) and time to the closest point of approach (TCPA). Regulations from COLREGs are integrated, and factors influencing the collision avoidance process are considered, constructing a refined reward signal tailored for training unmanned ships.
- (3) The proposed algorithm is implemented on the Unreal Engine 5 (UE5) physics engine platform, creating a simulation environment that mirrors maritime navigation characteristics. Experimental validation is conducted to demonstrate the effectiveness and practicality of the proposed method.

This study contributes a robust and compliant approach to collision avoidance in autonomous ship navigation, enhancing safety and operational efficiency in complex maritime environments.

The organization of this paper is as follows: Section 2 presents the ship dynamics model, COLREGs, collision risks, and mathematical models of ships. Section 3 covers deep reinforcement learning methods, network frameworks, PPO optimization techniques, the design of action and state spaces, and reward function design. Section 4 describes the establishment of the training environment and the presentation of training results, and it includes tests conducted on the improved ship collision avoidance algorithm within the UE5 environment. The paper concludes with a summary and outlook.

2. Problem Description

2.1. Ship Motion Dynamics

This study utilizes the mathematical models of ship dynamics proposed by Yasukawa and Yoshimura (2015) [18], Sandeepkumar et al. (2022) [19], and Sivaraj et al. (2022) [20] to construct the simulation environment, as shown in Equation (1):

$$\begin{cases} (m + m_x)\dot{u} - (m + m_y)vr - x_Gmr^2 = X_H + X_R + X_P + X_W \\ (m + m_y)\dot{v} - (m + m_x)ur + x_Gm\dot{r} = Y_H + Y_R + Y_W \\ (I_zG + x_G^2m + J_z)\dot{r} + x_Gm(\dot{v} + ur) = N_H + N_R + N_W \end{cases} \quad (1)$$

where m represents the mass of the ship, m_x , m_y denote the added mass in the x and y directions, u , v , r , $\dot{u}$, $\dot{v}$, $\dot{r}$ represent the velocities and accelerations in the x, y, and yaw directions, respectively. I_z and J_z represent the moment of inertia and the additional moment of inertia, respectively, along the z-axis, x_G is the center of gravity along the x-axis. The hydrodynamic forces are divided into components caused by the hull (H), propeller (P), rudder (R), and waves (W).

During navigation, the ship acquires relevant parameters from other vessels and calculates collision avoidance parameters, as shown in Figure 1. The positions of one's own ship (OS), target ship (TS), and goal are (x_0, y_0) , (x_{i_n}, y_{i_n}) , and (x_g, y_g) , respectively. The OS's speed is v_0 , the blue arrow indicates the direction of speed, its heading is φ_0 , and its rudder angle is δ_0 . The target ship's speed is v_{i_n} , its heading is φ_{i_n} , its bearing relative to

one's own ship is α_{T_n} , the relative bearing from OS to the goal is α_g , and the distance to the nearest point on the reference path is D_r .

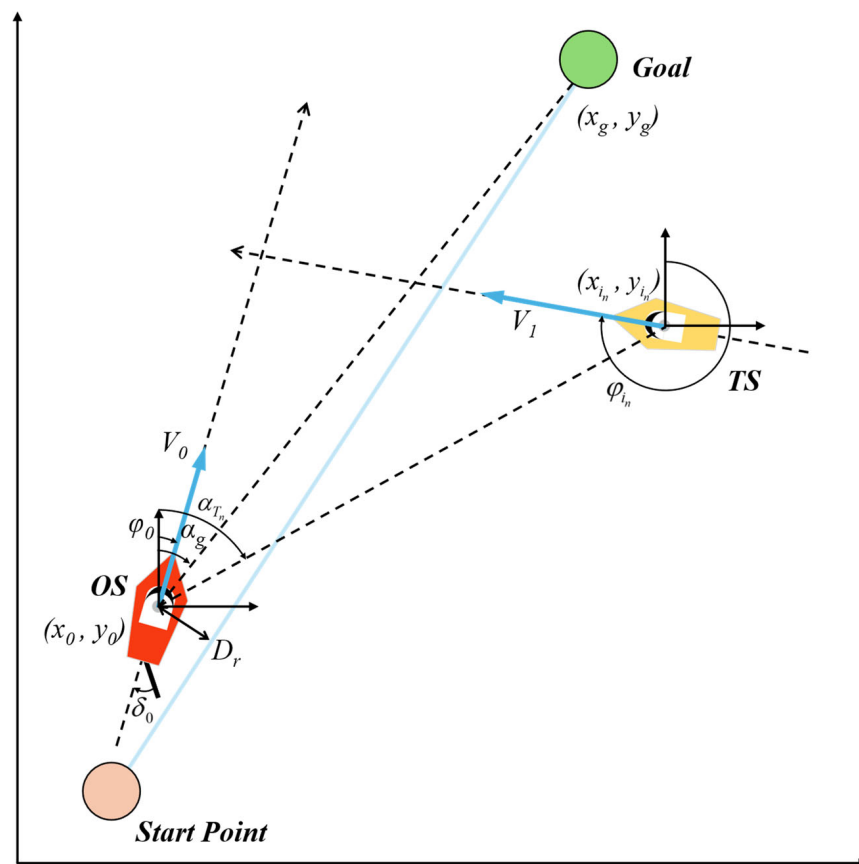


Figure 1. Collision avoidance schematic diagram.

2.2. COLREGs

It is crucial for the development of autonomous ship collision avoidance systems to accurately understand behavioral norms under different collision scenarios and to consider maritime collision avoidance rules. Based on the COLREGs and an in-depth analysis of steering and navigation rules, we have systematically organized the collision avoidance strategies that should be followed when ships encounter other moving vessels, as shown in Figure 2, ensuring logical consistency and clear expression.

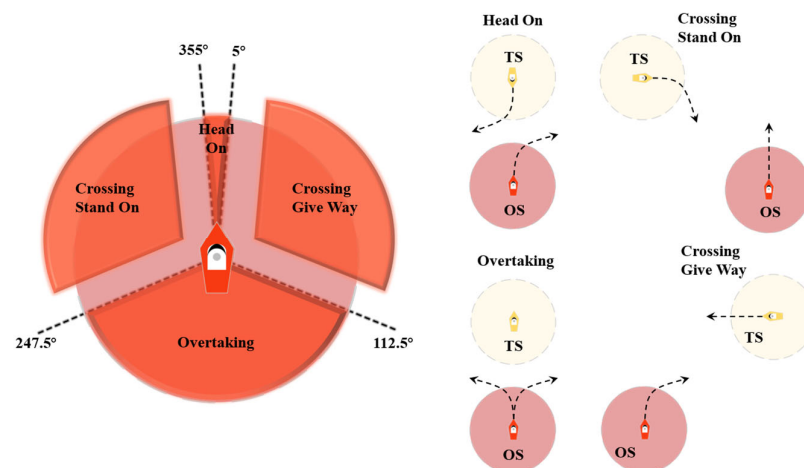


Figure 2. Collision avoidance according to the COLREGs for each situation.

- (1) Overtaking: When a ship is located within the rear quadrant of another moving ship (112.5° to 247.5°), it is in an overtaking encounter situation. If the overtaking ship is faster than the ship being overtaken, it must take appropriate measures to avoid a collision, safely maneuvering by altering course to port or starboard to pass at a safe distance between the two ships.
- (2) Head-on: When two ships are on opposing or nearly opposing courses, and the bearing of the approaching ship is within the 0° to 5° or 355° to 360° range relative to one's own ship, it is defined as a head-on situation. In this case, both ships should alter course to starboard, passing port to port to prevent a collision.
- (3) Crossing-stand-on: If a moving ship is located on the port side of one's own ship, within a bearing range of 247.5° to 355° , it is defined as a crossing-stand-on. In this scenario, one's own ship, being the stand-on ship, should maintain its course and speed.
- (4) Crossing-give-way: When a moving ship is situated on the starboard side of one's own ship, with a bearing between 0° and 112.5° , it is defined as a crossing-give-way. In this case, one's own ship should take action to alter course to starboard to ensure a safe distance between the ships and to avoid the crossing ship.

2.3. Collision Risk

The introduction of collision risk (CR) not only simplifies the decision-making process for collision avoidance but also enhances the predictability and controllability of collision avoidance actions. The calculation of CR is based on the quantitative analysis of key parameters such as TCPA and DCPA. As shown in Figure 3, D represents the distance between the two ships, φ_r is the relative heading between one's own ship and the target ship, and α_{T_n} indicates the bearing of the target ship relative to one's own ship. v_0 , v_{i_n} , and v_r denote the speed of one's own ship, the target ship, and their relative speed, respectively. The blue arrow indicates the direction of speed. These parameters form a comprehensive risk assessment system for ship collision avoidance, providing an intuitive and reliable method to assess collision risk.

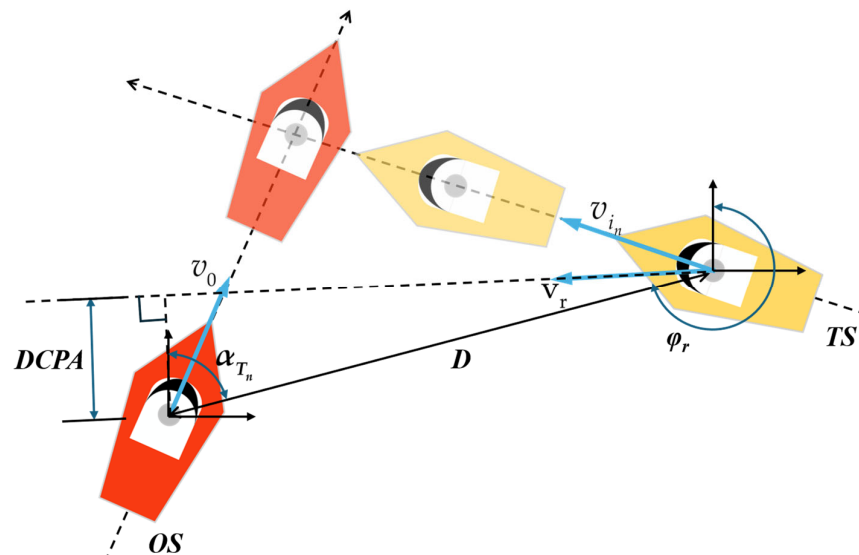


Figure 3. Concepts of DCPA, and TCPA.

The formula for calculating the distance to the closest point of approach (DCPA) between two ships is as follows:

$$DCPA = D \sin(\varphi_r - \alpha_r - \pi) \quad (2)$$

The formula for calculating the time to the closest point of approach (TCPA) between two ships is as follows:

$$TCPA = \frac{D \cos(\varphi_r - \alpha_r - \pi)}{v_r} \quad (3)$$

To accurately assess the CR and effectively implement avoidance strategies, it is essential to consider the ship domain (SD) [21] as a critical factor. The size and shape of the ship domain directly determine the outcome of the CR assessment, defining scenarios where ships are deemed to encounter potential collision risks. Numerous methods exist for determining ship domains [22,23], ranging from complex theoretical algorithms to detailed adjustments based on expert knowledge. In this paper, we employ a circular ship domain as a simple and intuitive method, which is both practical and efficient, as shown in Figure 2, illustrating the ship domains of OS and TS with a radius, r . The circular ship domain is easy to calculate and apply, significantly simplifying operational procedures and enhancing the efficiency and accuracy of ship collision avoidance decisions. Despite its simplicity, this method offers significant advantages in navigational safety and decision-making efficiency, providing a clear approach to addressing challenges in complex maritime conditions. Therefore, the CR assessment formula designed in this paper is as follows [24,25]. The objective function to be maximized is:

$$CR = \exp\left(-\frac{DCPA}{c_1}\right) - \exp\left(-\frac{TCPA}{c_2}\right) \quad (4)$$

To determine the coefficients for DCPA and TCPA, the size and speed of the OS and the TS, as well as the operational distance at which actual collision avoidance maneuvers commence, were considered. These coefficients are set to ensure that when the TS is at the edge of the OS's recognition distance (dr), a constant collision risk threshold, defined as allowable CR (CR_{al}), is maintained. The dr is defined as the distance at which one's own ship starts to identify and monitor the target ship, while the CR_{al} is used to determine when one's own ship should take action to avoid a collision with the target ship. To validate the effectiveness of the designed method, a KVLCC 2 tanker [26] was chosen as the example vessel. Table 1 summarizes the main parameters of this example ship.

Table 1. Principal dimensions of KVLCC2 and key parameters for collision avoidance.

Items	Value	Items	Value
Length (L)	320 m	Ship domain	900 m
Speed (V)	13 knots	Recognition distance (dr)	2000 m
Breadth	58 m	Allowable CR (CR_{al})	0.35
Turning radius	900 m	Coefficient 1 (c_1)	1588
Max rudder angle	35°	Coefficient 2 (c_2)	284

3. Collision Avoidance for Unmanned Ships Based on Deep Reinforcement Learning

In the field of machine learning, reinforcement learning (RL) is a core algorithmic paradigm focused on how agents learn through interaction with the environment to maximize cumulative rewards. Policy gradient methods are representative algorithms in this domain, which evaluate the performance of potential strategies to optimize decision-making processes. A significant advantage of these methods is their ability to directly parameterize strategies, particularly suitable for exploring high-dimensional action spaces [27,28]. However, significant changes in policy parameters can lead to performance instability in practice.

To address the challenges of policy gradient methods, researchers have developed trust region policy optimization (TRPO). The TRPO algorithm, inspired by natural policy gradients, centers on the idea of limiting the extent of policy updates to ensure gradual improvement in stability and performance. TRPO employs Kullback-Leibler (KL) divergence

to control the magnitude of policy updates, thereby preventing excessively large update steps [29,30].

$$L^{\text{TRPO}}(\theta) = \hat{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \quad (5)$$

where $\pi_\theta(a_t|s_t)$ is the probability of action a_t under state s_t according to the new policy. $\hat{A}_t$ is the advantage function, used to assess the superiority of action a_t compared to the average policy.

TRPO maintains stability during policy iteration by constructing a trust region, which prevents performance degradation due to overly large update steps. However, the application of TRPO often involves complex second-order optimization computations, limiting its wider application and scalability.

To address the limitations of TRPO, Schulman et al. [31] introduced proximal policy optimization (PPO). The PPO algorithm regulates the policy update process as follows:

$$L^{\text{PPO}}(\theta) = \hat{E}_t [\min(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t)] \quad (6)$$

where $r_t(\theta)$ represents the ratio of the new policy to the old policy (importance sampling ratio). Clipping operations constrain changes in this ratio, preventing excessively large updates and, thus, enhancing the stability of the algorithm. This approach not only simplifies the implementation process but also shows significant advantages in terms of sample complexity, scalability, and performance.

PPO effectively mitigates severe fluctuations in performance that could result from policy updates by limiting the likelihood ratio between new and old policies. The implementation of PPO consists of two fundamental steps: first, executing the policy and collecting data, and second, using a gradient ascent algorithm to optimize the agent's objective function.

In the PPO algorithm, the loss function consists of three parts: the policy objective function $L_t^{\text{PPO}}(\theta)$, the value function term L_t^{VF} , and the entropy term $H(\pi(a_t|s_t))$ [32]. These components are combined into the total loss function of PPO, where coefficients C_1 and C_2 are used to adjust the weights of the value function and entropy term. The policy objective function aims to optimize decision-making effectiveness, the value function term improves the accuracy of state estimation, and the entropy term enhances policy exploration to prevent premature convergence to local optima.

Specifically, the loss function in PPO can be expressed as follows:

$$L_t(\theta) = \hat{E}_t [L_t^{\text{PPO}}(\theta) - C_1 L_t^{\text{VF}} + C_2 H(\pi(a_t|s_t))] \quad (7)$$

where $L_t^{\text{PPO}}(\theta)$ is the policy objective function, L_t^{VF} is the value function loss, and $H(\pi(a_t|s_t))$ represents the policy entropy, which is the entropy of the probability distribution of choosing an action, a_t , under a given state s_t according to policy π . Adjusting C_1 and C_2 controls the influence of the value function and entropy on the total loss, balancing optimization, and exploration in the policy.

While simplifying implementation, PPO demonstrates significant advantages in terms of sample complexity, scalability, and performance. By limiting the extent of policy updates, PPO effectively avoids potential instabilities during training. Compared to TRPO, PPO simplifies the complex second-order optimization problem, thus enhancing training efficiency. However, PPO also has some limitations, such as relatively low sample utilization and sensitivity to hyperparameters, which, although less critical than in TRPO, still require careful tuning of multiple hyperparameters to ensure optimal performance.

Chaudhari and others [33] introduced entropy as a quantitative measure of randomness into the PPO objective function, increasing the uncertainty of the policy and thereby encouraging exploratory behavior. An increase in entropy implies a more exploratory

policy, while a decrease leads to a more deterministic approach. In PPO, the objective function can be expressed as follows:

$$L(\theta) = L^{\text{PPO}}(\theta) + \alpha \cdot H(\pi(a_t|s_t)) \quad (8)$$

where the entropy regularization term $H(\pi(a_t|s_t))$ is added to the objective function, where α is a tuning hyperparameter that balances the relationship between the PPO objective function $L^{\text{PPO}}(\theta)$ and the entropy regularization. Its range is from 0 to 1, with higher values encouraging more exploratory behaviors and increasing policy randomness, thereby avoiding early policy convergence. The introduction of entropy $H(\pi(a_t|s_t))$ effectively promotes exploratory behavior during policy training, especially in situations with large action spaces or complex state transitions. The introduction of policy entropy as a regularization term in the loss function helps prevent the policy from prematurely focusing on a few actions and avoids early convergence to local optima, thus motivating the algorithm to explore a wider range of state–action pairs during training. This mechanism is particularly useful in scenarios with large action spaces or complex state transitions, aiding in the discovery of better long-term strategies.

3.1. The Proposed DAE-PPO Algorithm

This algorithm optimizes the entropy regularization framework within the PPO algorithm without introducing additional hyperparameters, thus promoting more efficient exploration and improved policy performance. To further enhance the exploration capability of the policy, we propose a dynamic entropy adjustment method using a quadratically decreasing entropy approach to unify reward maximization with exploration uncertainty. This refinement allows for more precise entropy adjustments. The improved objective function incorporating the quadratically decreasing entropy term is represented as follows:

$$L(\theta) = L^{\text{PPO}}(\theta) + Q(t) \cdot H(\pi(a_t|s_t)) \quad (9)$$

Here, $Q(t)$ represents the entropy coefficient as a function of time t , defined as follows:

$$Q(t) = \lambda_{\max} + (\lambda_{\min} - \lambda_{\max}) \cdot \left(\frac{t}{t_{\max}} \right)^2 \quad (10)$$

where $\lambda_{\max}$ represents the maximum exploration coefficient set during training, $\lambda_{\min}$ the minimum exploration coefficient, t the current timestep, and $t_{\max}$ the total number of timesteps. The entropy coefficient decreases quadratically from $\lambda_{\max}$ to $\lambda_{\min}$ as t progresses toward $t_{\max}$. This coefficient decreases quadratically over time, providing more exploration space in the early stages of training and preventing premature convergence. As training progresses, the entropy coefficient gradually diminishes, facilitating a smooth transition from exploration to exploitation and ensuring the policy quickly converges to the global optimum during the later stages of training.

The core advantage of the quadratically decreasing entropy method lies in its high entropy coefficient settings at the beginning of training, which provides ample exploration space for the model and helps prevent the strategy from converging prematurely to local optima. Compared to simple linearly decreasing methods, this approach maintains higher entropy values longer during the early training phase, thereby enhancing the breadth of policy exploration and increasing the likelihood of discovering global optima, especially in complex and variable training environments. As training progresses, the gradual decay of the entropy coefficient aids in a smooth transition to the exploitation phase, reducing training instabilities caused by sudden changes in exploration intensity. In the later stages of training, the reduced entropy coefficient encourages the strategy to utilize the knowledge acquired, thereby facilitating rapid convergence of the model and enhancing policy performance. Furthermore, the dynamic entropy adjustment method employed in this study not only improves the exploration efficiency of the PPO algorithm but also enhances

the adaptability and robustness of the strategy in varied environments. With a carefully designed entropy adjustment strategy, we can more effectively guide agents in making decisions within complex reinforcement learning tasks, providing a novel optimization approach for solving practical problems.

3.2. Network Architecture and Initial Settings

In this study, we employ an actor-critic architecture to implement the PPO algorithm, where the actor network is responsible for generating a probability distribution of actions and sampling to determine the actual actions, while the critic network estimates the state values. The dimensions of the input layer of this architecture are determined by the observation vector of the environment, while the dimensions of the output layer are dictated by the action vector. Specifically, the actor network outputs the means and standard deviations of the actions, making the output layer's dimension twice the number of actions; conversely, the critic network outputs a single state value, with an output layer dimension of 1. In terms of hidden layer design, both the actor and critic networks employ a multi-layer perceptron (MLP) structure, with each hidden layer containing 128 units, split across two layers.

The initial steps of the training process include creating a Replay Buffer to store interaction experiences, followed by initializing the actor and critic networks, and deciding whether to use initial network weights based on the configuration file. Subsequently, both networks are configured with the Adam optimizer, and an exponential learning rate scheduler is used to adjust the learning rate.

During the training loop, experience data are received from the trainer and stored in the replay buffer. The network parameters are then updated using the PPO strategy, with the updated strategy fed back to the trainer, ensuring synchronous updates of the critic network.

The design of this network architecture aims to leverage the dual advantages of the actor-critic method: parallel optimization of policy and value functions to accelerate the learning process. The actor network's output of action probability distributions facilitates natural policy exploration, while the introduction of entropy loss and regularization loss further incentivizes exploratory behavior, helping to prevent the strategy from prematurely converging to local optima. The critic network provides a value estimate that offers a stable reference objective for policy optimization. Furthermore, updates to network parameters are performed through a dedicated optimizer, incorporating core features of the PPO algorithm, such as clipping of probability ratios and truncation of value functions. These mechanisms work together to safeguard the update process, preventing instability due to excessively large steps.

Figure 4 illustrates the update process of the PPO algorithm. The PPO algorithm effectively captures complex feature relationships through a multi-layer perceptron structure and uses a replay buffer to store experiences, ensuring sample diversity and stability. Through clipping mechanisms and synchronous update strategies, the PPO algorithm achieves stability in policy and consistency in value estimates during training. This design not only enhances the model's flexibility and adaptability but also ensures the robustness and reliability of the learning process.

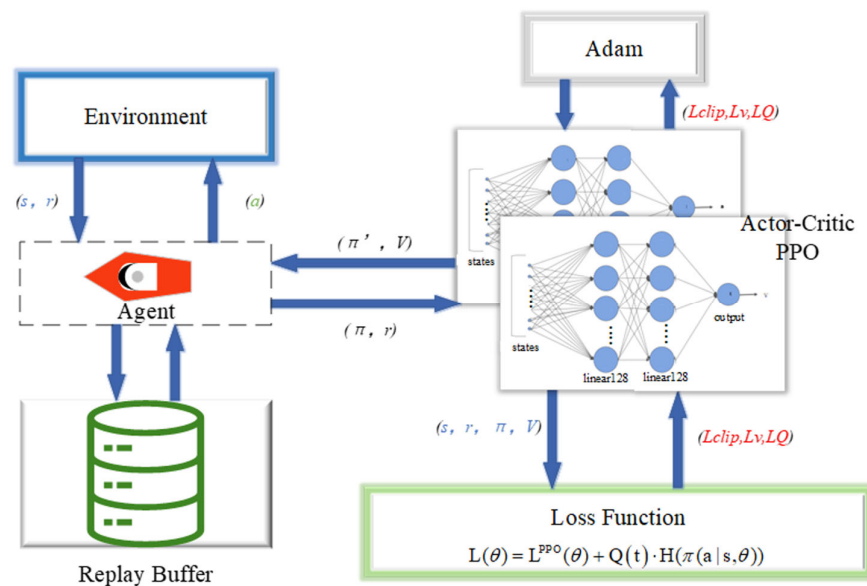


Figure 4. Algorithm architecture.

3.3. State and Action Space Design

In the reinforcement learning framework, the environment provides the agent with observations of its current state, based on which the agent selects actions. Subsequently, the environment responds to the agent's actions, providing feedback that includes new state information and corresponding rewards. In the context of autonomous collision avoidance tasks, unmanned ships act as agents, while obstacles, the marine environment, and other vessels constitute their environment. To ensure effective deployment and application in real-world settings, the design of the state space must closely reflect data available from actual sensors. This study has designed a multidimensional state space, as follows:

$$S = \left\{ \varphi_0, \dot{\varphi}_0, \delta_0, \dot{\delta}_0, v_0, x_0, y_0, \alpha_g, x_g, y_g, \alpha_{T_1}, v_{i_1}, \varphi_{i_1}, x_{i_1}, y_{i_1}, CR_{i_1}, \dots, \alpha_{T_n}, v_{i_n}, \varphi_{i_n}, x_{i_n}, y_{i_n}, CR_{i_n}, D_r \right\} \quad (11)$$

which includes the following four parts: 1. The agent's own navigational status: This part involves the unmanned ship's heading, rate of heading change, rudder angle, rudder angular velocity, speed, and position coordinates, all of which collectively describe the ship's immediate navigational state. 2. Goal-related state: This part includes the relative bearing and coordinates of the goal in relation to the unmanned ship, providing target-oriented navigation information. 3. Target ship state information: for each TS in the environment, the state space includes its true bearing, speed, heading, position coordinates, and CR relative to the OS. This information is crucial for assessing and avoiding collision risks. 4. Reference path state: this concerns the distance between the unmanned ship and the nearest point on the reference path, providing a reference for path planning and navigation.

To ensure safe navigation at sea, mariners continuously monitor collision risks and make decisions based on extensive navigation experience to make timely adjustments to the vessel's course. Through comprehensive training in a simulation environment, the unmanned ship can learn and master these collision avoidance decision-making skills. In this study, the rudder angle serves as the action chosen by the agent through the strategy, capable of changing the unmanned ship's direction and path.

$$\delta_0 \leftarrow \delta_0 + \dot{\delta}_0 \quad (12)$$

The rudder angle is designed in a continuous space, δ_0 from -35° to 35° , and $\dot{\delta}_0$ from -5° to 5° . This design closely mimics the physical characteristics of actual ship maneuvering, enhancing the flexibility and adaptability of the unmanned ship's navigation.

3.4. Reward Function Design

The reward function plays a crucial role in reinforcement learning, serving as a metric to evaluate the agent's behavior and guiding the agent to act in a way that maximizes its cumulative rewards. As the training process iterates, the agent learns how to act within the explored environment to maximize its expected future benefits, eventually converging toward a stable behavior strategy. To ensure that the trained strategies effectively accomplish the autonomous collision avoidance tasks, this study divides the reward function into four components: destination reward, navigation reward, collision avoidance reward, and rule compliance reward. The reward values are calculated and accumulated at each frame. Here is the specific design of each part of the reward function:

- (1) Destination reward: This reward is designed to guide the agent toward the destination.

$$R_{goal} = \frac{D_{goal}^{t-1} - D_{goal}^t}{V^t \cdot dt}, D_{goal}^t = \|P_{goal} - P_O^t\| \quad (13)$$

where D_{goal}^{t-1} and D_{goal}^t are the distances between the destination and the ship's previous and current positions, respectively. P_{goal} is the goal position, and P_O^t is the OS's current position. V^t is the OS's forward speed, and dt is the timestep. The numerator represents the difference in distance to the destination across each timestep, while the denominator represents the actual displacement of the ship per timestep. Thus, the equation effectively measures the ratio of the ship's effective displacement toward the destination to its actual displacement. The higher the ratio, the more aligned the ship's movement is with the direct line to the destination. Figure 5 illustrates the design principles of the reward function.

- (2) Navigation reward: This reward encourages the agent to navigate efficiently along the reference path.

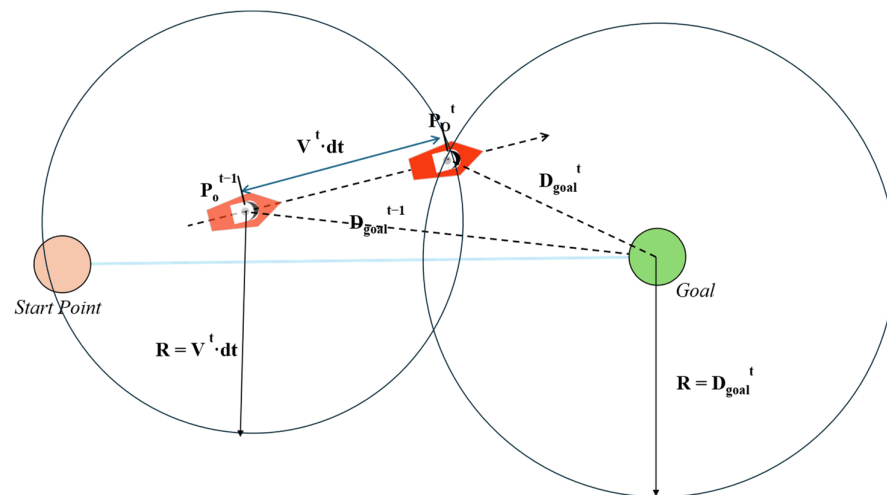


Figure 5. Navigation reward.

$$R_{navigation} = \begin{cases} \frac{4L - D_r^t}{4L} & \text{if } D_r^t < 8L \\ \frac{90 - |\alpha_g^t|}{90} & \text{if } |\alpha_g^t| \leq 90 \\ -1 & \text{if } |\alpha_g^t| > 90 \end{cases} \quad (14)$$

where D_r^t is the shortest distance between the ship and the reference path at time t , L is the ship's length, and the boundary distance is set to eight times L . The reward increases as the ship sails closer to the reference path, with the formula using four times the ship's length as the denominator. α_g^t is the angle between the ship's course at time t and the line connecting the ship to the destination. The reward increases as the ship's velocity direction more closely aligns with the destination. If α_g^t exceeds 90° , indicating the ship is moving away from the destination, a reward of -1 is given to discourage this behavior. Figure 6 illustrates the design principles of the reward function.

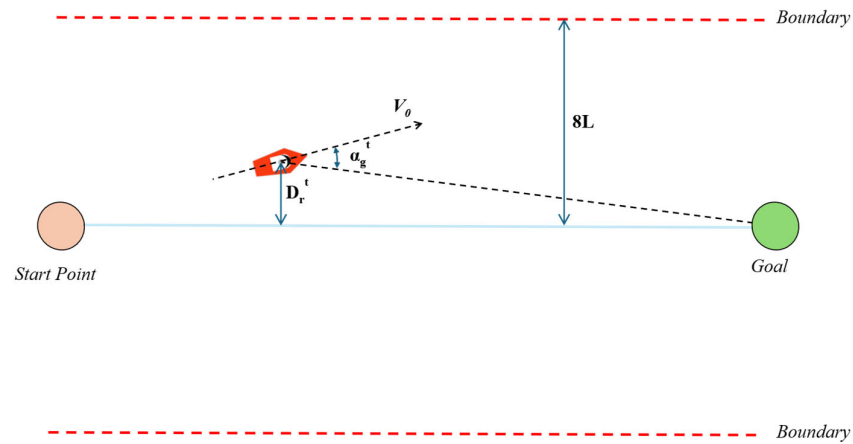


Figure 6. Simulation experiment scenes in Unreal Engine 5.

- (3) Collision avoidance reward: This reward is based on the current collision risk value and a critical threshold, aimed at preventing collisions with obstacle vessels.

$$R_{\text{Collision}} = \begin{cases} 1 & \text{if } CR < CR_{al} \\ \frac{1+CR_{al}-\frac{2CR}{1-CR_{al}}}{1-CR_{al}} & \text{if } CR_{al} \leq CR \leq 1.0 \\ -1 & \text{if } 1.0 \leq CR \end{cases} \quad (15)$$

where CR represents the current collision risk value, and CR_{al} is the critical collision risk threshold. If CR is less than CR_{al} , the ship is considered relatively safe from the target ship. If CR exceeds CR_{al} , it is considered that there is a significant risk of collision, and collision avoidance decisions should be executed promptly.

- (4) Rule compliance reward: According to the COLREGs, this reward guides autonomous ships in making collision avoidance decisions that conform to rules.

$$R_{\text{COLREGs}} = \begin{cases} 1 & \text{if satisfies standards} \\ 0 & \text{otherwise} \end{cases} \quad (16)$$

COLREGs constrain the behavior of autonomous ships. First, refer to the standards in Figure 2 to assess encounter scenarios, and then determine if the OS's decisions comply with COLREG rules 13–17 to appropriately avoid conflicts and provide corresponding rewards. If the OS decides to alter course in a head-on or starboard-crossing situation when $\delta_0 > 0$, it is considered compliant with COLREGs, receiving a reward value of 1; in all other cases, the reward is 0.

Through meticulous design of state spaces, action spaces, and reward functions, this study has established a comprehensive reinforcement learning training system. At each timestep, the agent's state information is fed into the neural network, which outputs value estimates for each possible action based on current parameters and states. The agent selects the best action based on these estimates, leading to updates in the environmental state. This design ensures that the agent can learn effective collision avoidance strategies in complex maritime environments.

4. Experimental Results and Analysis

4.1. Training Environment

The simulation experiments in this study were conducted using the Unreal Engine 5.3.2 virtual simulation platform, renowned for its advanced graphical processing capabilities and physics engine. The hardware configuration of the experimental environment comprised a high-performance NVIDIA RTX 3080 GPU, customized and manufactured by ASUS, a renowned electronics company based in Taipei, Taiwan, and an Intel Core i7 4.50 GHz CPU processor, sourced from Intel Corporation, headquartered in Santa Clara, California, USA. This combination provided robust computational support for the experiments. On the software side, we utilized Python 3.7 programming language, paired with the TensorFlow deep learning framework, to construct and train our experimental models. As shown in Figure 7, we built the calm sea simulation experiment scenes in Unreal Engine 5 (UE5), ensuring the visualization and interactivity of the experiments.

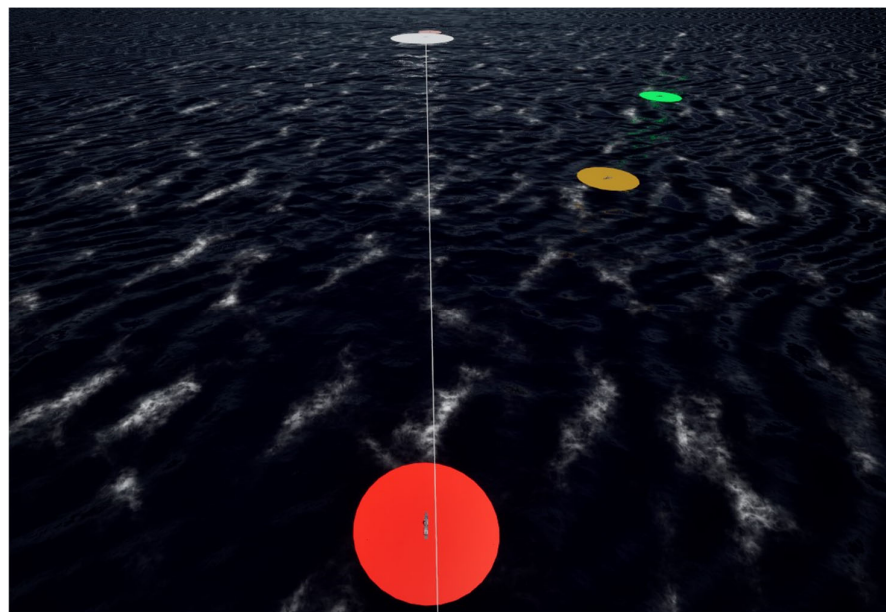


Figure 7. Simulation experiment scenes in Unreal Engine 5.

As depicted in Figure 8a, for convenience, both the OS and TS are assumed to be the same ship (KVLCC2). The main specifications and parameters of the validated sample ship are summarized in Table 1. Each training session randomly generates two obstacle ships in specified states. The obstacle ships near the destination have 10 initial positions, uniformly distributed along a circular arc formed by their starting positions, with an interval β of 1° between them. This setup aims to simulate the continuity of situations that might be encountered during ship navigation. Obstacle ships near the target ship have 360 initial positions, evenly spaced by interval β along a circle formed by their starting positions. This is used to simulate various encounter scenarios. The light-colored circular area around the ship represents the ship's domain, with a radius R closely related to the calculation of the CR value. The initial position of the target ship and its line to the destination is defined as the navigation line, which serves as an input reference for the reward function. Figure 8b shows the termination signal for the state, i.e., the end conditions for experimental training. The termination signal triggers in three scenarios: if the target ship collides with an obstacle ship, deviates beyond the maximum allowed distance from the planned route, or reaches the destination, at which point the system marks the training session as "Done".

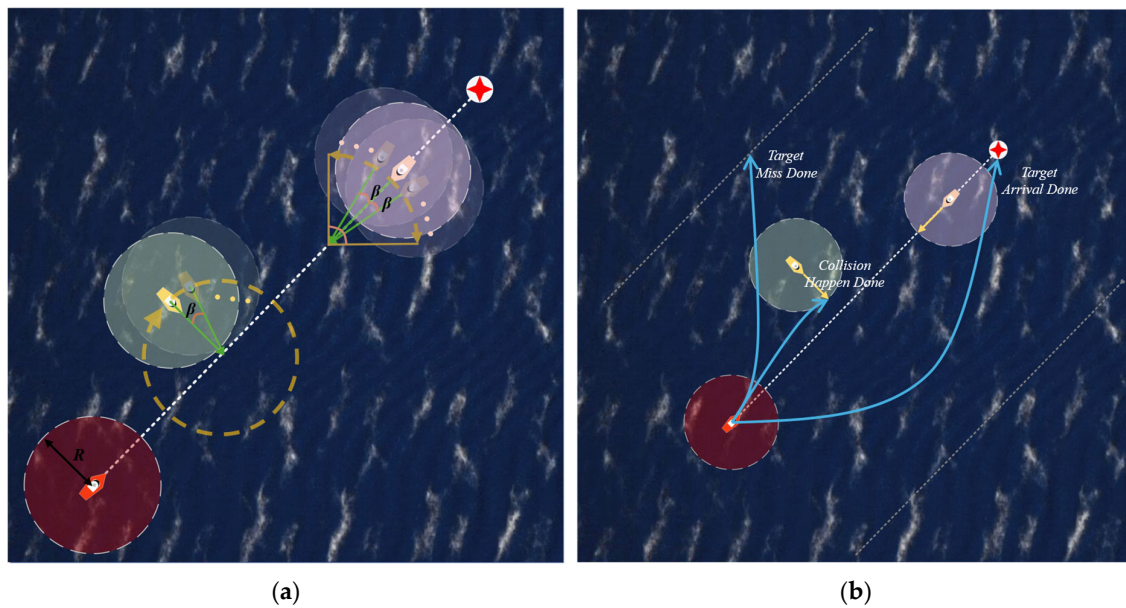


Figure 8. Training environment. (a) Training scene generation; (b) training termination conditions.

4.2. Algorithm Hyperparameters

Based on comprehensive considerations of the algorithm architecture and simulation environment, the selected training hyperparameters are presented in Table 2. The learning rate critic is the learning rate used to update the value function (managed by the critic network). An appropriate learning rate is crucial for ensuring the stability of the training process. The learning rate policy refers to the learning rate of the policy network, which adjusts the agent's behavior in response to environmental feedback. The magnitude of this learning rate directly influences the dynamics of training. The learning rates for the actor and critic networks are 0.001 and 0.01, respectively. The learning rate decay is set to 0.99, which causes the learning rate to exponentially decay as training progresses, aiding in model convergence and reducing the risk of significant weight updates in the later stages of training. In the entropy-inclusive PPO algorithm, the entropy coefficient is set to 0.05, which quantifies the randomness of the strategy and encourages exploratory behavior. Its optimal value was determined through a series of debugging experiments. In the DAE-PPO algorithm, the entropy coefficient start and entropy coefficient end are set to 0.09 and 0.01, respectively, to facilitate dynamic entropy adjustment during training. Epsilon Clip (ϵ -Clip) is set at 0.2 to limit the magnitude of policy updates, preventing training instability due to overly large updates. Lambda (λ) is set at 0.95 to adjust the variance and bias in the generalized advantage estimation (GAE), ensuring the stability of value estimates.

Table 2. Hyperparameters for training.

Hyperparameter	Value
Learning rate critic	0.01
Learning rate policy	0.001
Learning rate decay	0.99
Entropy coefficient (entropy-PPO)	0.05
Entropy coefficient start (DAE-PPO)	0.09
Entropy coefficient end (DAE-PPO)	0.01
Epsilon clip	0.2
Lambda (GAE)	0.95
Discount factor (γ)	0.99

4.3. Training Process

Figure 9 shows the trends in average reward values and the corresponding confidence intervals for three algorithms over 20,000 training episodes. The red curve represents the PPO algorithm with dynamically adjusted entropy (DAE-PPO), the green curve represents the entropy-inclusive PPO algorithm, and the blue curve represents the classic PPO algorithm. The shaded areas in the graph indicate the confidence intervals for each algorithm. In the early stages of training, DAE-PPO has a wider confidence interval, indicating that the algorithm adopts a broader exploration strategy initially. This strategy helps the algorithm quickly identify behavior patterns that yield higher rewards. As training progresses, after about 5000 episodes, the average reward for DAE-PPO begins to significantly increase, and its confidence interval starts to narrow, demonstrating the algorithm's performance stability. By 8000 training episodes, the average reward for DAE-PPO stabilizes and ultimately reaches a high level. The DAE-PPO algorithm, by employing a quadratic decay interpolation of entropy, demonstrates significant advantages in autonomous ship navigation and collision avoidance tasks. This entropy decay strategy optimizes the balance between exploration and exploitation, not only enhancing the algorithm's average reward but also its stability and reducing convergence time. In contrast, both the traditional PPO algorithm and the entropy-inclusive PPO algorithm have not surpassed DAE-PPO in terms of reward values and stability, thus confirming the superiority of DAE-PPO in such tasks.

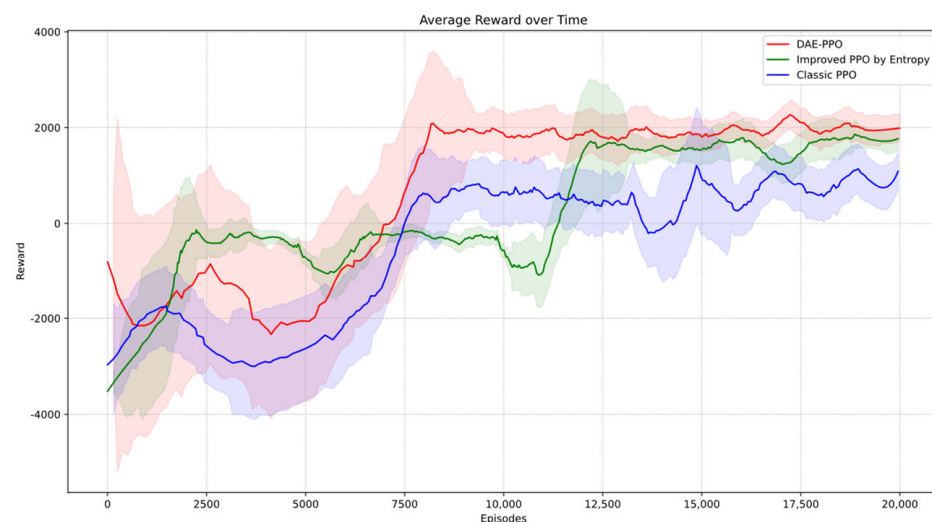


Figure 9. Average reward.

Figure 10 provides a visual display of the agent's performance across various training stages, offering a detailed tracking and evaluation of the entire learning process.

Figure 10a shows the agent's training in the first episode. Due to the random initialization of neural network parameters and the agent's unfamiliarity with the environment, its movement is based on random exploration, leading to circling behavior at the starting position.

Figure 10b displays the agent's training in episode 1432. The agent learns from feedback that circling at the initial position is not a reward-driven behavior, thus it begins to explore the scenario. It then collides with an obstacle ship and the training ends. Although this training session failed, the interaction experience is crucial for future exploration.

Figure 10c illustrates the training effect in episode 3646, where the agent successfully avoids ST2 and begins to try more behaviors to explore the scenario.

Figure 10d shows the training outcome in episode 5600, where the agent's behavior aligns more closely with the intended guided rewards. The failed experience of colliding with obstacle ship ST1 is essential for achieving better training results.

As shown in Figure 11a,b, the initial positions of OS, TS1, and TS2 are (6117.98, 1863.74), (4832.98, 4510.49), and (687.41, 7271.13), respectively, with the obstacle ships maintaining constant speed and course. Figure 11a,b, respectively, show the trajectory and the relative distance curves of the OS and TS for the classic PPO training model. The classic PPO model exhibits significant fluctuations in ship heading, showing a lack of stability, and the closest distances to TS1 and TS2 are both less than 600 m, indicating a high risk of collision. Moreover, the decision-making during encounters does not comply with the COLREGs.

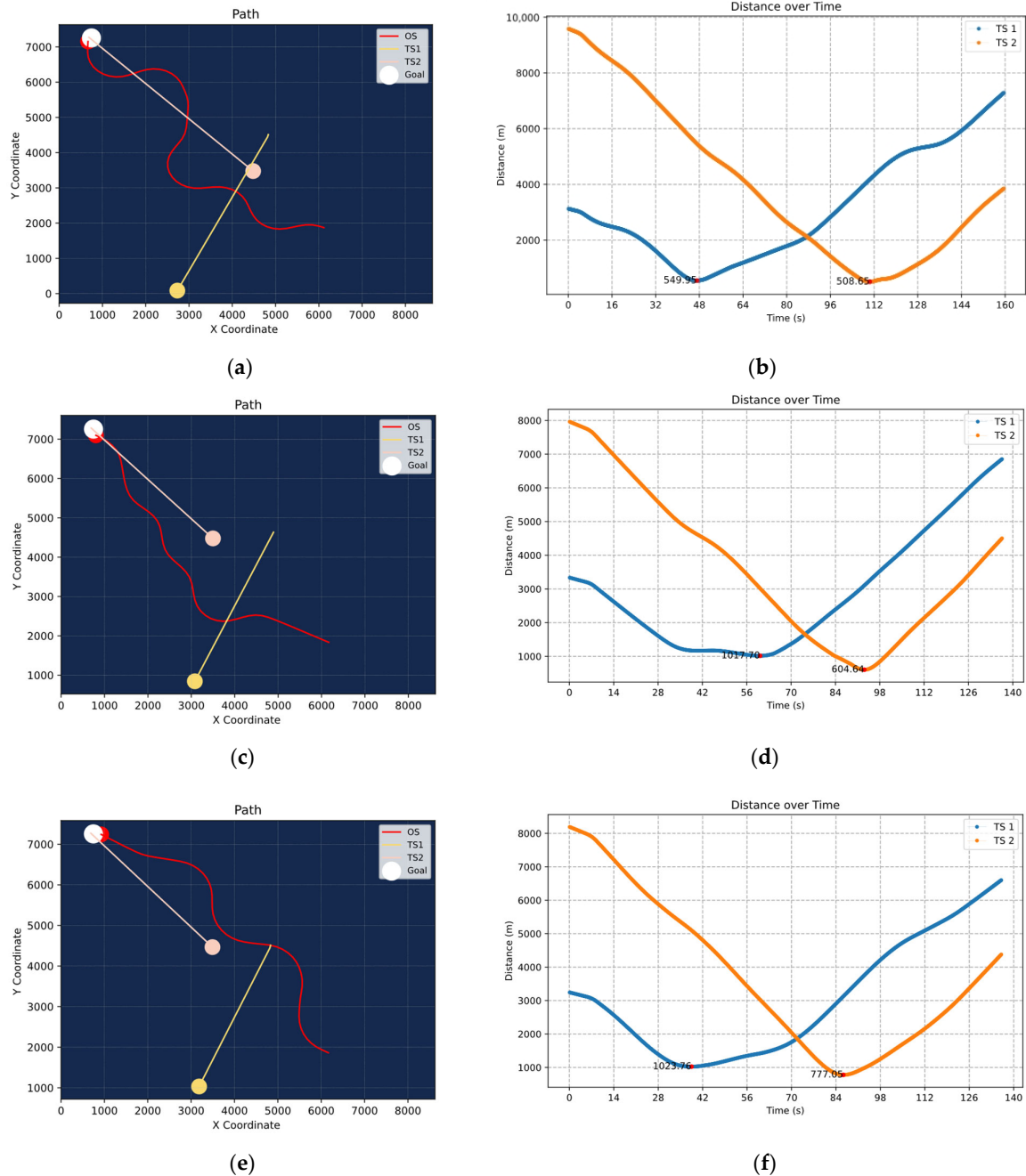


Figure 11. Encounter situation. (a) Path planned by classic PPO; (b) distance curve between OS and TSs for classic PPO; (c) path planned by entropy-PPO; (d) distance curve between OS and TSs for entropy-PPO; (e) path planned by DAE-PPO; (f) distance curve between OS and TSs for DAE-PPO.

Figure 11c,d present the trajectory and the relative distance curves of the OS and TS for the entropy-PPO training model. Compared to the classic PPO model, the entropy-PPO model's course is more stable, maintaining a higher safety distance with TS, yet the closest

point remains at a fairly close 604.64 m. Actions during encounters with TS1 and TS2, such as turning left too little or lacking clear movement, do not comply with COLREGs.

Figure 11e,f showcase the trajectory and relative distance curves of OS and TS for the DAE-PPO training model. The trajectory indicates that DAE-PPO exhibits greater stability in collision avoidance decisions compared to the previous two models. The encounter decisions comply with COLREGs and are clearly and succinctly executed. As seen in Table 3, the DAE-PPO algorithm performs excellently in collision avoidance, with the widest separation and the lowest collision risk among the algorithms, and it achieves this in the shortest time of 135 s, indicating its strong capability in complex encounter scenarios.

Table 3. Results of collision avoidance of Example.

Algorithms	Minimum Distance between Ships [m]	Maximum Collision Risk	Time
DAE-PPO	777.05	0.72	135
Entropy-PPO	604.64	0.81	137
Classic PPO	508.65	1.07	159

To verify the generalization ability of the algorithm, 100 sets of random experiments were conducted under the same random seed. In the same environment, 100 collision avoidance tests were conducted, with the number of successful incidents recorded. As shown in Table 4, the DAE-PPO algorithm significantly outperformed other algorithms in terms of success rate in multiple experiments, further validating its stability and reliability.

Table 4. Results of 100 times experiments.

Algorithm	DAE-PPO	Entropy-PPO	Classic PPO
Successful arrival	96	81	51

Figure 12 displays a challenging encounter scenario involving three obstacle ships, designed to assess the performance of collision avoidance models in handling complex situations.

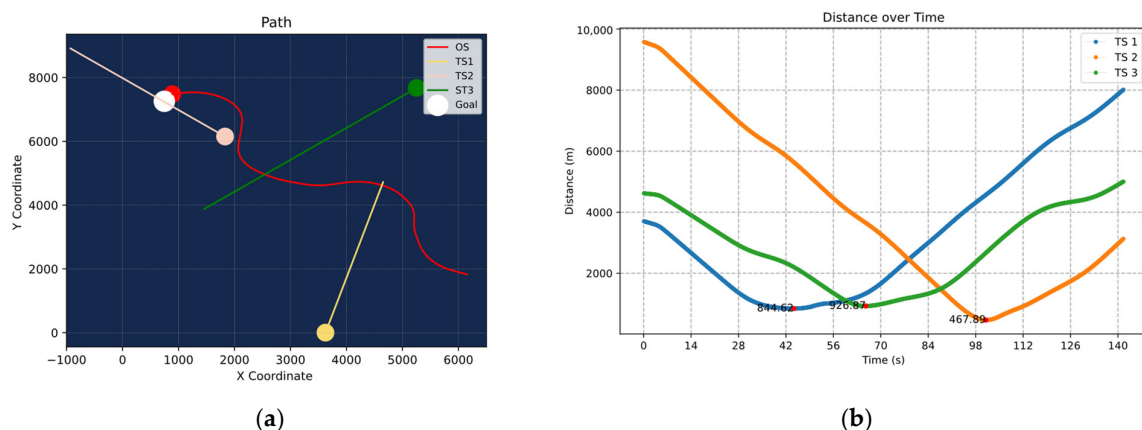


Figure 12. Collision avoidance environment. (a) Path planned; (b) distance curve between OS and TSs.

In this scenario, the initial positions of OS are (6121.76, 1857.45), with TS1, TS2, and TS3 starting at (4654.17, 4721.57), (−928.20, 8911.22), and (1458.61, 3880.10), respectively. During the simulation, the course and speed of the obstacle ships remain constant. Figure 12a shows the completed path of OS. When OS and TS1 formed a crossing-give-way situation with a potential collision risk, the agent made a significant right turn. The minimum distance between the two ships was recorded as 766.25 m, successfully achieving safe

avoidance, and the decision strictly followed the COLREGs. After the avoidance maneuver, OS corrected its rudder angle and, after a stable course, performed a crossing-stand-on encounter with TS3. At this point, the agent decided to turn left. During this process, the minimum distance between the two ships was 1098.60 m. After avoiding TS3, OS continued to navigate smoothly toward the destination. Subsequently, OS and TS2 encountered a head-on situation. OS quickly made a significant right turn, and during this process, the minimum distance between the two ships was again 1098.60 m, with the avoidance behavior also complying with COLREGs. After completing the avoidance, OS began to resume its course and smoothly reached its destination.

Through this simulation, we validated the effectiveness and compliance of the collision avoidance model in handling complex maritime traffic scenarios, ensuring safe and compliant navigation in varied maritime environments.

5. Conclusions

This study addresses the issue of ship collision avoidance in maritime transport environments by proposing an improved PPO algorithm (DAE-PPO) based on dynamically adjusted entropy. The algorithm optimizes the exploration mechanism through a quadratically decreasing entropy method, effectively avoiding local optima and enhancing strategic performance, all while considering the COLREGs. Simulation results indicate that the DAE-PPO algorithm significantly outperforms in efficiency, success rate, and stability in collision avoidance tests.

Moreover, the reward function designed in this study is subdivided into destination, navigation, collision avoidance, and rule compliance rewards, effectively guiding the agent in making effective collision avoidance decisions in complex maritime environments.

Despite the significant achievements of the DAE-PPO algorithm in simulated environments, future research needs to delve deeper into several areas: While simulation environments can emulate various scenarios, the complexity and unpredictability of actual maritime environments are higher. Future work should include testing and validating the algorithm in real maritime conditions. Maritime traffic often involves interactions among multiple ships; research could explore collaborative collision avoidance strategies within multi-agent systems to enhance the efficiency and safety of overall maritime traffic. The exploration of elliptical ship domain models, which can dynamically adjust based on the speed of unmanned ships, will more accurately simulate real-world collision avoidance scenarios. Through these subsequent studies, we will aim to further enhance the performance of autonomous collision avoidance algorithms for unmanned ships, advance unmanned ship technology, and ultimately achieve safe and efficient autonomous maritime navigation.

Author Contributions: Conceptualization, G.C. and W.W.; methodology, Z.H.; software, Z.H. and W.W.; validation, G.C., Z.H. and W.W.; writing—review and editing, Z.H., G.C. and W.W.; data curation, Z.H.; visualization, G.C. and Z.H.; supervision, S.Y., G.C. and W.W.; project administration, S.Y.; funding acquisition, G.C., W.W. and S.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China (no. 52371369), the Key Projects of the National Key R&D Program (no. 2021YFB390150), the Natural Science Project of Fujian Province (no. 2022J01323, 2023J01325, 2023I0019), the Science and Technology Plan Project of Fujian Province (No. 3502ZCQXT2021007), the Natural Science Foundation of Xiamen, China (grant no. 502Z202373038), and the Funds of Fujian Province for Promoting High-Quality Development of the Marine and Fisheries Industry (No. FJHYF-ZH2023-10).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Millefiori, L.M.; Braca, P.; Zissis, D.; Spiliopoulos, G.; Marano, S.; Willett, P.K.; Carniel, S. COVID-19 Impact on Global Maritime Mobility. *Sci. Rep.* **2021**, *11*, 18039. [\[CrossRef\]](#) [\[PubMed\]](#)
2. International Maritime Organization. *Convention on the International Regulations for Preventing Collisions at Sea, 1972 (COLREGs)*; International Maritime Organization: London, UK, 1972.
3. Tang, P.; Zhang, R.; Liu, D.; Huang, L.; Liu, G.; Deng, T. Local Reactive Obstacle Avoidance Approach for High-Speed Unmanned Surface Vehicle. *Ocean. Eng.* **2015**, *106*, 128–140. [\[CrossRef\]](#)
4. Hart, P.; Nilsson, N.; Raphael, B. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cyber.* **1968**, *4*, 100–107. [\[CrossRef\]](#)
5. Holland, J.H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*; The MIT Press: Cambridge, MA, USA, 1992; ISBN 978-0-262-27555-2.
6. Khatib, O. Real-Time Obstacle Avoidance for Manipulators and Mobile Robots. *Int. J. Robot. Res.* **1986**, *5*, 90–98. [\[CrossRef\]](#)
7. Guo, S.; Zhang, X.; Zheng, Y.; Du, Y. An Autonomous Path Planning Model for Unmanned Ships Based on Deep Reinforcement Learning. *Sensors* **2020**, *20*, 426. [\[CrossRef\]](#)
8. Cheng, Y.; Zhang, W. Concise Deep Reinforcement Learning Obstacle Avoidance for Underactuated Unmanned Marine Vessels. *Neurocomputing* **2018**, *272*, 63–73. [\[CrossRef\]](#)
9. Huang, Z.; Lin, H.; Zhang, G. The USV Path Planning Based on an Improved DQN Algorithm. In Proceedings of the 2021 International Conference on Networking, Communications and Information Technology (NetCIT), Manchester, UK, 26–27 December 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 162–166.
10. Xu, X.; Lu, Y.; Liu, X.; Zhang, W. Intelligent Collision Avoidance Algorithms for USVs via Deep Reinforcement Learning under COLREGs. *Ocean. Eng.* **2020**, *217*, 107704. [\[CrossRef\]](#)
11. Peng, X.; Han, F.; Xia, G.; Zhao, W.; Zhao, Y. Autonomous Obstacle Avoidance in Crowded Ocean Environment Based on COLREGs and POND. *J. Mar. Sci. Eng.* **2023**, *11*, 1320. [\[CrossRef\]](#)
12. Xiao, Q.; Jiang, L.; Wang, M.; Zhang, X. An Improved Distributed Sampling PPO Algorithm Based on Beta Policy for Continuous Global Path Planning Scheme. *Sensors* **2023**, *23*, 6101. [\[CrossRef\]](#) [\[PubMed\]](#)
13. Meyer, E.; Robinson, H.; Rasheed, A.; San, O. Taming an Autonomous Surface Vehicle for Path Following and Collision Avoidance Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 41466–41481. [\[CrossRef\]](#)
14. Guan, W.; Cui, Z.; Zhang, X. Intelligent Smart Marine Autonomous Surface Ship Decision System Based on Improved PPO Algorithm. *Sensors* **2022**, *22*, 5732. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Wu, C.; Yu, W.; Li, G.; Liao, W. Deep Reinforcement Learning with Dynamic Window Approach Based Collision Avoidance Path Planning for Maritime Autonomous Surface Ships. *Ocean. Eng.* **2023**, *284*, 115208. [\[CrossRef\]](#)
16. Mnih, P.V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
17. Williams, R.J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Mach. Learn.* **1992**, *8*, 229–256. [\[CrossRef\]](#)
18. Yasukawa, H.; Yoshimura, Y. Introduction of MMG Standard Method for Ship Maneuvering Predictions. *J. Mar. Sci. Technol.* **2015**, *20*, 37–52. [\[CrossRef\]](#)
19. Sandeepkumar, R.; Rajendran, S.; Mohan, R.; Pascoal, A. A Unified Ship Manoeuvring Model with a Nonlinear Model Predictive Controller for Path Following in Regular Waves. *Ocean. Eng.* **2022**, *243*, 110165. [\[CrossRef\]](#)
20. Sivaraj, S.; Rajendran, S.; Prasad, L.P. Data Driven Control Based on Deep Q-Network Algorithm for Heading Control and Path Following of a Ship in Calm Water and Waves. *Ocean. Eng.* **2022**, *259*, 111802. [\[CrossRef\]](#)
21. Fujii, Y.; Tanaka, K. Traffic Capacity. *J. Navig.* **1971**, *24*, 543–552. [\[CrossRef\]](#)
22. Coldwell, T.G. Marine Traffic Behaviour in Restricted Waters. *J. Navig.* **1983**, *36*, 430–444. [\[CrossRef\]](#)
23. Goodwin, E.M. A Statistical Study of Ship Domains. *J. Navig.* **1975**, *28*, 328–344. [\[CrossRef\]](#)
24. Mou, J.M.; Tak, C.V.D.; Ligteringen, H. Study on Collision Avoidance in Busy Waterways by Using AIS Data. *Ocean. Eng.* **2010**, *37*, 483–490. [\[CrossRef\]](#)
25. Ha, J.; Roh, M.-I.; Lee, H.-W. Quantitative Calculation Method of the Collision Risk for Collision Avoidance in Ship Navigation Using the CPA and Ship Domain. *J. Comput. Des. Eng.* **2021**, *8*, 894–909. [\[CrossRef\]](#)
26. Sakamoto, N.; Ohashi, K.; Araki, M.; Kume, K.; Kobayashi, H. Identification of KVLCC2 Manoeuvring Parameters for a Modular-Type Mathematical Model by RaNS Method with an Overset Approach. *Ocean. Eng.* **2019**, *188*, 106257. [\[CrossRef\]](#)
27. Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. Benchmarking Deep Reinforcement Learning for Continuous Control. In Proceedings of the International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016.
28. Ilyas, A.; Engstrom, L.; Santurkar, S.; Tsipras, D.; Janoos, F.; Rudolph, L.; Madry, A. A Closer Look at Deep Policy Gradients. *arXiv* **2020**, arXiv:1811.02553.
29. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust Region Policy Optimization. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 1889–1897.
30. Kakade, S.M. A Natural Policy Gradient. In *Advances in Neural Information Processing Systems*; MIT Press: Cambridge, MA, USA, 2001; Volume 14.

31. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
32. Williams, R.J.; Peng, J. Function Optimization Using Connectionist Reinforcement Learning Algorithms. *Connect. Sci.* **1991**, *3*, 241–268. [[CrossRef](#)]
33. Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J.; Sagun, L.; Zecchina, R. Entropy-SGD: Biasing Gradient Descent into Wide Valleys. *J. Stat. Mech.* **2019**, *2019*, 124018. [[CrossRef](#)]

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.