Journal of *Marine Science and Engineering*

MDPI

Article

An Intelligent Algorithm for USVs Collision Avoidance Based on Deep Reinforcement Learning Approach with Navigation Characteristics

Zhe Sun [1,2], Yunsheng Fan [1,2,*] and Guofeng Wang [1,2]

[1] College of Marine Electrical Engineering, Dalian Maritime University, Dalian 116026, China; rickey@dlmu.edu.cn (Z.S.); dmuwgf@dlmu.edu.cn (G.W.)
[2] Key Laboratory of Technology and System for Intelligent Ships of Liaoning Province, Dalian 116026, China
* Correspondence: yunsheng@dlmu.edu.cn; Tel.: +86-155-2478-9899

Abstract: Many achievements toward unmanned surface vehicles have been made using artificial intelligence theory to assist the decisions of the navigator. In particular, there has been rapid development in autonomous collision avoidance techniques that employ the intelligent algorithm of deep reinforcement learning. A novel USV collision avoidance algorithm based on deep reinforcement learning theory for real-time maneuvering is proposed. Many improvements toward the autonomous learning framework are carried out to improve the performance of USV collision avoidance, including prioritized experience replay, noisy network, double learning, and dueling architecture, which can significantly enhance the training effect. Additionally, considering the characteristics of the USV collision avoidance problem, two effective methods to enhance training efficiency are proposed. For better training, considering the international regulations for preventing collisions at sea and USV maneuverability, a complete and reliable USV collision avoidance training system is established, demonstrating an efficient learning process in complex encounter situations. A reward signal system in line with the USV characteristics is designed. Based on the Unity maritime virtual simulation platform, an abundant simulation environment for training and testing is designed. Through detailed analysis, verification, and comparison, the improved algorithm outperforms the pre-improved algorithm in terms of stability, average reward, rules learning, and collision avoidance effect, reducing 26.60% more accumulated course deviation and saving 1.13% more time.

Keywords: unmanned surface vehicles; deep reinforcement learning; autonomous collision avoidance; COLREGs

Citation: Sun, Z.; Fan, Y.; Wang, G. An Intelligent Algorithm for USVs Collision Avoidance Based on Deep Reinforcement Learning Approach with Navigation Characteristics. *J. Mar. Sci. Eng.* **2023**, *11*, 812. https://doi.org/10.3390/jmse11040812

Academic Editor: Gerasimos Theotokatos

Received: 25 February 2023
Revised: 14 March 2023
Accepted: 20 March 2023
Published: 11 April 2023

1. Introduction

With the higher demand for unmanned surface vehicle (USV) intelligent technology, countries have taken measures to improve the effect of autonomous collision avoidance while safeguarding public life and property. Intelligence navigation refers to a USV that can use a sensing system to obtain current maritime navigation information and autonomously generate USV navigation decisions to achieve safe and reliable water navigation. Research on intelligent USV navigation is of great practical importance for the shipping industry and even human society.

USVs have been used for missions at sea due to their small size, high speed, low cost, and no risk of human casualties. In ocean mapping, USVs are used to carry GNSS and other equipment to assist in achieving accurate positioning and mapping [1]. The USV can efficiently achieve the detection of submarine geomorphology and underwater objects by carrying side scan sonar [2]. In hydrological monitoring, the detection area is covered by the navigation and control of the USVs, and the hydrological parameters are monitored by the sensor equipment [3,4]. In addition, many studies for path-tracking control [5], trajectory tracking [6], dynamic positioning [7], and collision avoidance [8] problems are a permanent

basis for any application of USVs, and they can forcefully advance the development of the problem of autonomous USV navigation.

There are many ways of designing autonomous collision avoidance algorithms for USV, such as A*, artificial potential field, velocity obstacle, dynamic window, fast marching method, etc. Ren et al. [9] use the velocity obstacle method for collision avoidance algorithm design, optimizing the way of setting up the velocity obstacle region for multi-ship collision avoidance. Fan et al. [10] use an improved cuckoo search algorithm designed with an adaptive update step, which optimizes the global search capability and can plan a better collision avoidance strategy. Guan et al. [11] use the A* method and dynamic window method to design a collision avoidance algorithm for static and dynamic obstacles, optimizing the weight coefficient of the dynamic window algorithm by deep Q network. These methods have unique advantages in specific environments and are very effective in USV collision avoidance algorithms, but their disadvantages are also distinct. Firstly, the operational anthropomorphism degree of these algorithms is not sufficient. Secondly, the generalization ability of some algorithms is poor, and the tuning of parameters is complicated. Moreover, some algorithms simplify the actual situation when applied in training and are difficult to be applied in practice. However, the model-free deep reinforcement learning approach based on learning styles can effectively compensate for these shortcomings.

With the development of the deep reinforcement learning approach, it has achieved great results in many fields due to its outstanding perception and decision-making capabilities, such as Go [12], video games [13], autonomous navigation [14], and medicine [15]. It is based only on itself, without any human knowledge of collision avoidance beyond navigation rules. However, it can make excellent decisions in many challenging domains. Especially in USV collision avoidance, reliable samples for learning are hard to obtain and expensive. Therefore, independently, starting tabula rasa, deep reinforcement learning can fully compensate for these problems and complete collision avoidance tasks in complex USV encounter situations.

On the issues of USV collision avoidance, many pieces of research on autonomous collision avoidance based on the deep reinforcement learning approach have been developed. Most researchers focus on the model-free method because this direction is easy to implement, and for another reason, the model-based method is difficult. The model-free method used in the USVs collision avoidance algorithm is divided into value-function-based and policy-gradient-based [16]. The former is good at dealing with discrete action space. Chen et al. [17] provide a representative paradigm for the discrete ship movements and devise a unique way of training. Li et al. [18] employ the traditional artificial potential field (APF) method to optimize the reward signal of the DQN method, resulting in a more scientific reward signal. Shen et al. [19] improve the neural network framework for more efficient learning, obtaining a better collision avoidance effect than the algorithm before improvement. Zhou et al. [20] improve the collision algorithm to solve the sparse reward problem, using two networks to generate actions and evaluate behavior. Compared to the DQN algorithm, the training effect under the improved algorithm is better.

Research on the latter, the policy gradient method, is also widely studied. Du et al. [21] propose an improved algorithm based on the deep deterministic policy gradient (DDPG) algorithm [22] that combines with Douglas–Peucker (DP) algorithm to plan the path. The Long Short-Term Memory (LSTM) architecture and rich reward function are designed to improve the speed and stability of convergence. Xu et al. [23] also choose the DDPG algorithm and establish a risk assessment model, improving the network structure. Their algorithm has a good collision avoidance effect and real-time performance. Additionally, Chen et al. [24] propose a cooperative multi-agent deep reinforcement learning algorithm for ship collision avoidance, resulting in a good collision avoidance effect under simple typical ship encounter situations. Considering the results of this collision avoidance research, based on the reinforcement learning algorithm, there are some problems worth further consideration:

(1) In many pieces of research, the training environment in each episode is fixed, lacking practical significance, whether complex or not.
(2) Most researchers are not selecting more random seeds to verify the superiority and reliability of their algorithm.
(3) Some researchers are not considering the maneuverability characteristics of USVs adequately.

Deep reinforcement learning theory provides an extraordinary way for USVs' autonomous collision avoidance safety and efficiency. Compared with the traditional methods, it performed better in complex environments with simple designs. Furthermore, it does not require any prior knowledge from the expert navigator about avoiding the obstacle well; surprisingly, it is not even necessary to provide fully observed training environments, to accomplish the complex collision task.

Aiming at the above problems and characteristics, in this paper, a USV collision avoidance algorithm based on the deep reinforcement learning approach, DRLCA, is designed. The main contributions of this paper are as follows:

(1) This paper considers the restriction of maneuverability and international regulations for preventing collisions at sea (COLREGs) in the training process. A suitable training environment with stochasticity and complexity is designed based on the deep reinforcement learning approach. Additionally, considering the collision avoidance process for factors, a meticulous reward signal for USVs training is constructed, which makes training more practical.
(2) Double Q learning method is used to reduce overestimation, dueling neural network architecture to improve training effect, and prioritized experience replay to optimize sampling. The results of various improvements are analytically compared under an abundant training environment based on multiple random number seeds.
(3) Aiming at the hard-exploration problem caused by the training environment with strong randomness, the noisy network method is introduced, which can enhance the detection capability. Experimentally, the best way of noise adding in USV collision avoidance training is confirmed. Considering the characteristics of the USVs collision avoidance problem, the restriction of the dynamic area is introduced in training for calculation reduction and the clip of neural network state input for training effect improvement.

This paper is organized as follows. Kinematic parameters, COLREGs, ship domain, and USV mathematical models are in Section 2. Section 3 is about the deep reinforcement learning approach and its optimization methods. Section 4 describes the establishment of the training environment. In Section 5, the improved algorithm for USVs collision avoidance is tested in the Unity environment. The last section is the summary and prospect.

2. USV Collision Avoidance Parameters

2.1. USV Collision Avoidance Characteristics

As shown in Figure 1, $Y(N)$ and $X(E)$ point to the north and east. (x_U, y_U), (x_O, y_O), and (x_T, y_T) are the positions of our own USV, obstacle USV, and terminal. φ_U, φ_T, φ_O, ϕ, α_O, and θ are our USV course, absolute azimuth of the terminal and our own USV, obstacle USV course, relative azimuth of the terminal and our own USV, absolute azimuth of the obstacle USV and our own USV, and the relative azimuth of the obstacle USV and our own USV. The distance between obstacle USV and our own USV is d. The speed of our own USV and obstacle USV are V_U and V_O. To reflect the USV collision avoidance training characteristics, the following Norrbin ship mathematical model is selected [25],

$$\begin{cases} T\dot{\eta} + \eta + \alpha\eta^3 = K\delta \\ \eta = \dot{\varphi} \end{cases} \tag{1}$$

where, T, K, and α are related to USV maneuverability, and the course change caused by rudder angle change can be well calculated. Because the research object of this paper is the "Lan Xin" USV, which has a vector propulsion system, it is necessary to consider

the influence of the characteristics of the steering gear. The vector propulsion system is a type of thruster that can change direction to achieve maximum speed propulsion in any direction and can obtain additional control torque to achieve control of ship attitude change. Therefore, the following equation, the second-order propulsion angle response model, is selected in this paper,

$$\ddot{\delta} + 2\zeta\omega_n\dot{\delta} + {\omega_n}^2\delta = k{\omega_n}^2\delta_r \tag{2}$$

where, ω_n, ζ, and k are intrinsic frequency, damping ratio and proportionality coefficient as $\omega_n = 0.958$, $\zeta = 0.811$ and $k = 0.923$. δ_r is the target rudder angle.

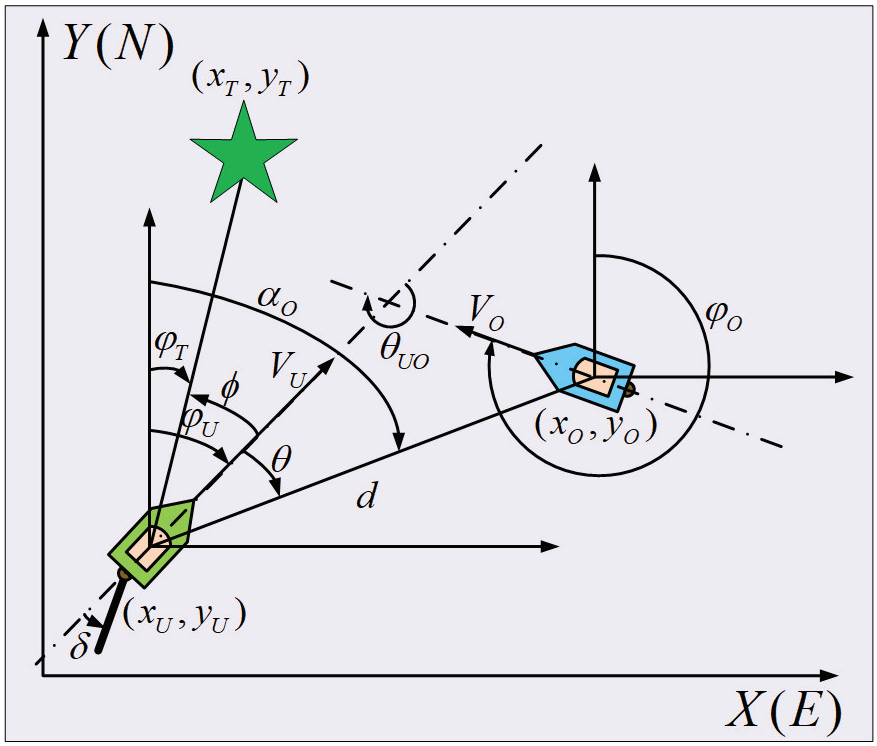


Figure 1. USV collision avoidance kinematic.

The accurate division of USV encounter situations is crucial for collision avoidance agent training, and it is divided into the following six conditions in this paper [26]:

(1) As shown in Figure 2a, when the obstacle USV_O and own USV_U are at the relative azimuth of $[355°, 360°) \cup [0°, 5°)$, it is the head-on encounter situation. According to the COLREGs, when there is a hazard of USV collision, both USVs have to avoid each other and should turn to the port side as they pass.

(2) As shown in Figure 2b, when USV_O is at the $[247.5°, 355°)$ relative azimuth of USV_U, and there is a risk of collision, it is the crossing-stand-on encounter situation. USV_U should stand on the course, and USV_O should turn to starboard.

(3) As shown in Figure 2c, when USV_O is at the $[5°, 112.5°)$ relative azimuth of USV_U, and there is a risk of collision, it is the crossing-give-way encounter situation. USV_U should turn to starboard, and USV_O should stand on the course.

(4) As shown in Figure 2d, when USV_U is at the $[112.5°, 247.5°)$ relative azimuth of USV_O, and there is a risk of collision, it is the overtaking encounter situation. USV_U should avoid the collision, and turns to starboard or port are allowed.

(5) As shown in Figure 2e, when USV_O is at the $[112.5°, 247.5°)$ relative azimuth of USV_U, and there is a risk of collision, it is the overtaking encounter situation. USV_U should stand on the course.

(6) Additionally, when the obstacle USV is in breach of rules, our own USV should avoid it proactively.

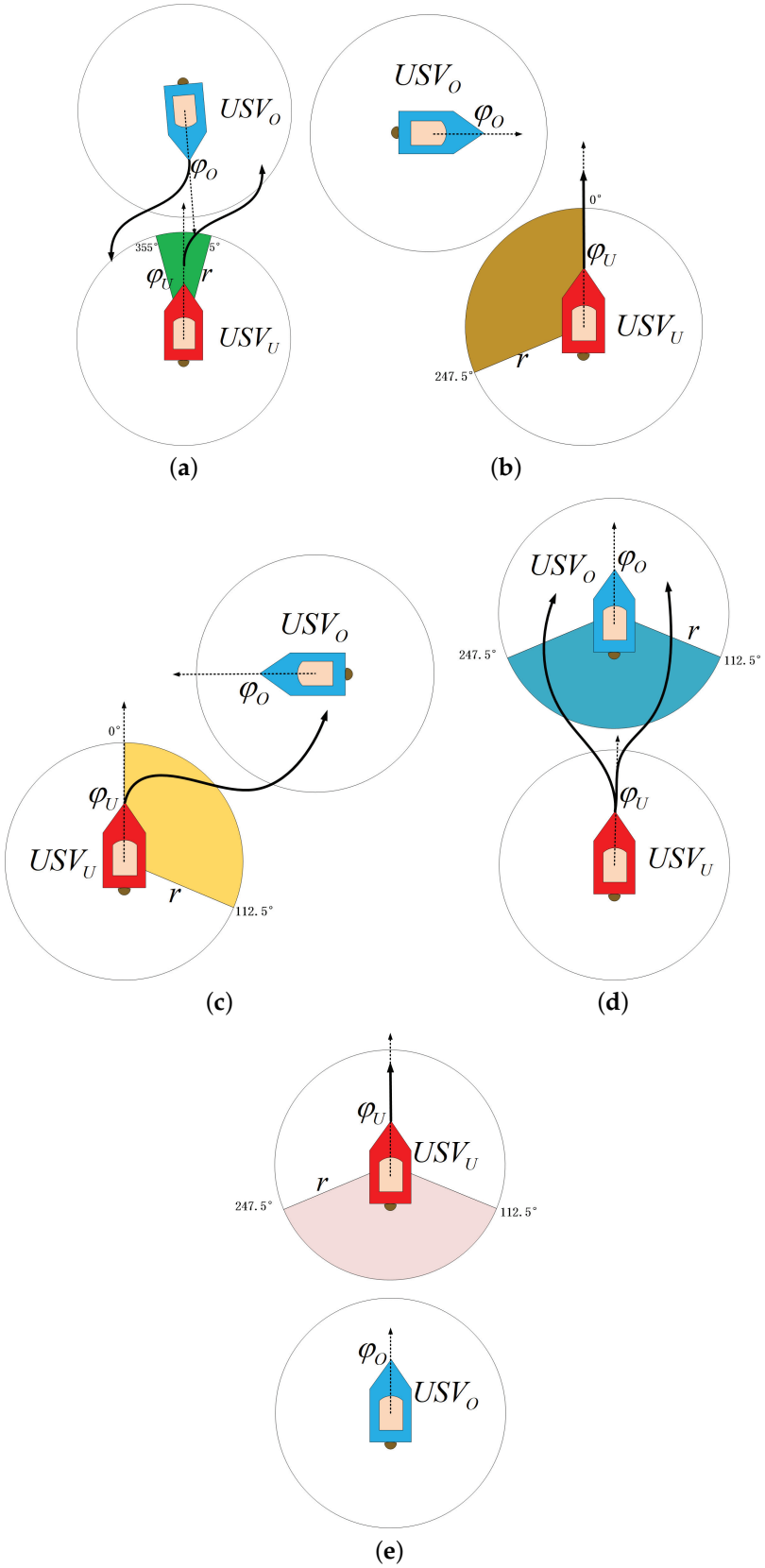


Figure 2. Encounter situation. (**a**) Head-on; (**b**) Crossing-stand-on; (**c**) Crossing-give-way; (**d**) Overtaking-give-way; (**e**) Overtaking-stand-on.

2.2. USV Collision Avoidance Characteristics

In Figure 2, our USV and the obstacles also have a region with the radius of r, called ship domain (SD) [27]. This domain is for each USV and is used to determine the collision when another USV invades. There are three main ways of defining the ship domain, theoretical analyses, experts' knowledge, and determined empirically. The first two are mainly based on a variable number of parameters to regulate the shape and size of the ship domain, which is complex and detailed. The third one lacks some details but is more concise. In this paper, the third way of a circular ship domain is chosen, which is a clean and practical ship domain shape. It is essential for deep reinforcement learning USV collision avoidance algorithm training. Another essential concept is the dynamic area (DA), recorded as R, planning a circular range around USV. When the obstacle USV is not in this range, there is no collision risk between the USVs. These two parameters are used to calculate the collision risk index (CRI), which can not only visualize the current risk for navigation intuitively but is also the key to guiding collision avoidance behavior. The distance at the closest point of approaching (DCPA) and time to the closest point of approaching (TCPA) are defined as shown in Figure 3, and they can be calculated as follows,

$$\begin{cases} DCPA = d_{OU}\sin(\lambda) \\ TCPA = d_{OU}\cos(\lambda)/V_{OU} \end{cases}. \tag{3}$$

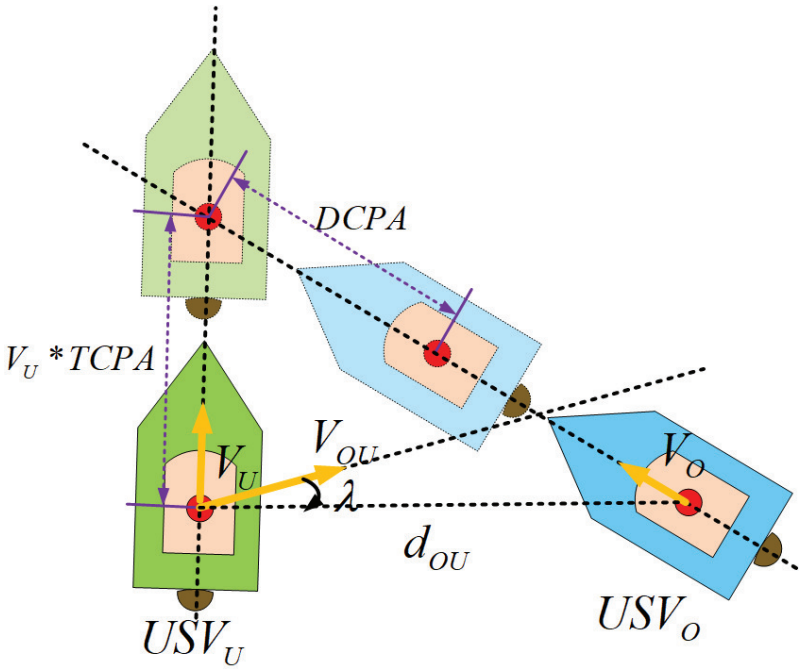


Figure 3. DCPA and TCPA.

Their membership functions u_D and u_T can be calculated as follows,

$$u_D = \begin{cases} 1, & |DCPA| \le r \\ 0.5 - 0.5\sin[\frac{\pi}{R-r} \times \frac{DCPA(R+r)}{2}], & r < |DCPA| \le R \\ 0, & |DCPA| > R \end{cases} \tag{4}$$

If TCPA > 0,

$$u_T = \begin{cases} 1, & TCPA \le T_1 \\ [\frac{T_2 - TCPA}{T_2 - T_1}]^2, & T_1 < TCPA \le T_2 \\ 0, & TCPA > T_2 \end{cases} \tag{5}$$

If TCPA ≤ 0,

$$u_T = \begin{cases} 1, & |TCPA| \leq T_1 \\ [\frac{T_2+TCPA}{T_2-T_1}]^2, & T_1 < |TCPA| \leq T_2 \\ 0, & |TCPA| > T_2 \end{cases} \quad (6)$$

T_1 and T_2 are expressed as follows,

$$T_1 = \begin{cases} \frac{\sqrt{D_1^2+DCPA^2}}{V_{OU}}, & D_1 \geq |DCPA| \\ 0, & D_1 < |DCPA| \end{cases} \quad (7)$$

$$T_2 = \begin{cases} \frac{\sqrt{D_2^2-DCPA^2}}{V_{OU}}, & D_2 \geq |DCPA| \\ 0, & D_2 < |DCPA| \end{cases} \quad (8)$$

Therefore, the USV collision risk u_{CRI} can be calculated as follows,

$$u_{CRI} = \begin{cases} 0, & u_D = 0 \\ 0, & u_D \neq 0, u_T = 0 \\ \max(u_D, u_T), & u_D \neq 0, u_T \neq 0 \end{cases} \quad (9)$$

3. Deep Reinforcement Learning

Deep reinforcement learning theory focuses on the interaction in learning, which addresses how an agent can maximize their reward through learning conspicuous behavior in different states under a specific environment. It is worthwhile to note that it requires only a small amount of prior knowledge provided by humans, but it can complete many challenging problems. Such as path planning in a grid [28], imitating humans playing video games [29], and controlling the movement of vehicles [30]. The deep reinforcement learning approach does not need to investigate internal connection and hidden architecture to the object. Through trial-and-error and delay reward, it can perform control and environment identification tasks simultaneously [31]. The reinforcement learning theory with strong decision-making ability is very suitable for use in the research of USVs. Based on the USV agent and training maritime environment, this algorithm can accomplish the task of USV collision avoidance well in a complex environment.

3.1. Deep Q Learning

Reinforcement learning theory is very suitable for use in unstructured and intricate environments because it is almost impossible for other algorithms to accurately describe the implicit relationship between the environment and the agent. In this way, the component that can make decisions is called the agent, and all the others are the environment. All the frames in reinforcement learning theory are around the constant interaction between these two components [32]. The core of agent-environment interaction consists of the following four elements [33]. The first element is the policy π. It describes the mapping relationship from state to action, similar to a functional relationship, and determines the selected action in the current state. The second element is reward signal G_t. It is a scalar form of feedback from the environment to the agent's behavior, and it defines the long-standing goal of the agent in reinforcement learning processing. The third element is the value function v. It is similar to the reward signal, a quantitative description of advantages based on the latest state. The value function analyzes the better choice from a longer perspective. Finally is the model, which reflects whether the agent has the ability to react to the external environment. The method of driving the interaction process is through the Markov decision process (MDP) [34], which consists of the following three parts. The first part is the state S, which describes the agent and environment at the current moment t. The second part is the

action A, which represents all possible ways an agent can influence the environment. The third part is the reward function R, which describes the value of the reward for taking a particular action in a given state. The last part is transition probability $p(s', r|s, a)$, which selects a specific probability distribution for each state and action. In every step t, the agent influences the environment by its action A_t, basing the state S_{t-1} and reward R_{t-1} of the last step, and then getting a new state S_t and reward R_t. Through such a learning process, the reinforcement learning agent can continue understanding the environment deeper and achieve clever solutions to complex control problems.

Q learning is an algorithm [35] that not only contains the bootstrapped learning idea from Dynamic Programming algorithm [36], learning without waiting for the end of an episode but also allows continuous interaction with the environment without modeling, as in the Monte Carlo method [37]. The Q learning algorithm is updated using the following functions,

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{10}$$

Though Q learning has performed well in many control problems, there are still many limitations because of the updating form of the value function based on the Q table. In many complex control problems, the reinforcement learning approach is used to deal with problems that are common, complex, and high-dimensional, such as tasks with huge state space or a complex combination of forms, whereas traditional Q learning is inadequate. Therefore, the Q table is fitted using a specific neural network. The gradient descent technique is used instead of the Q table to update the neural network, which perfectly makes up for the defects of traditional Q learning [38].

Figure 4 shows an update schematic diagram of the DQN algorithm. At every training step t, the agent interacts with the environment, constantly enriches its knowledge, and improves their behavior. The state S is fed into the neural network θ and influences the environment by maximizing the action value obtained at this training step t. Then, the state S changes to S' as the environment changes, and the agent can obtain the corresponding rewards signal. Whereafter, the information of interaction (s, a, r, s') is saved in the experience replace buffer for sample and learning. The evaluating Q network output $Q(s, a; \theta)$, and target Q network output $Q(s', a'; \theta^-)$. In this way, all the parameters of the loss function needed for neural network training can be obtained, and the gradient descent method is used to update the evaluating Q network. Finally, the parameters of evaluating the Q network are copied to the target network every N step to achieve the real policy update.

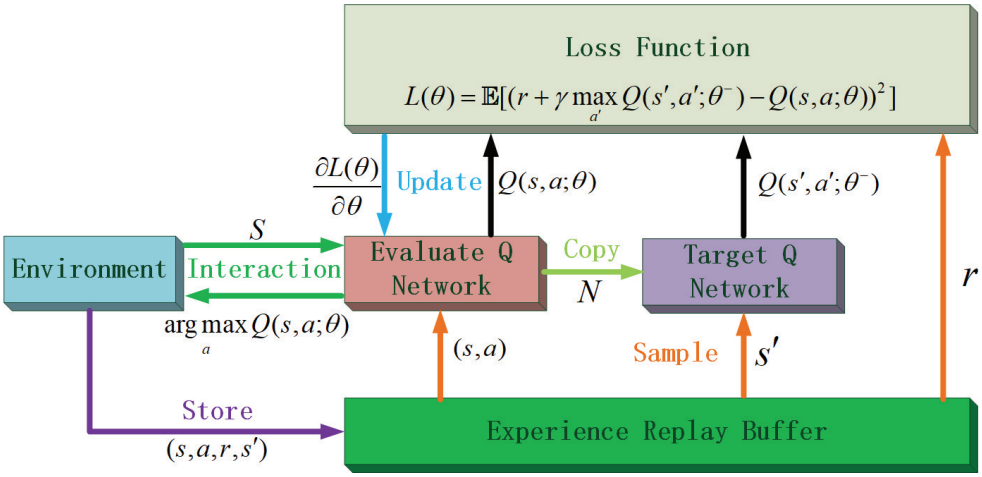


Figure 4. DQN algorithm architecture.

3.2. Collision Avoidance Algorithm for USV

When building a USV collision avoidance training framework with the deep reinforcement learning approach, a complete set of state, action, and reward signals is essential and can facilitate efficient training.

The first part is the design of state space. The USVs autonomous collision avoidance system must be able to face complex and time-varying maritime conditions. With reliable sensors, USVs can perceive the real-time information of obstacles within a range and can be used for USV collision avoidance behavior training. So, the following state space is designed,

$$S = \{\varphi_U, \dot{\varphi}_U, \delta_U, \dot{\delta}_U, \vartheta_t, d_t, d_{O1}, \vartheta_{O1}, \varphi_{O1}, \ldots, d_{Om}, \vartheta_{Om}, \varphi_{Om}, d_{S1}, \vartheta_{S1}, \ldots, d_{Sn}, \vartheta_{Sn}\} \quad (11)$$

The state space can be divided into four parts. The first part is the state of our own USV, which reflects the navigation information of the USV. It contains its own USV course φ_U, the change rate of our USV course $\dot{\varphi}_U$, its own USV rudder angle δ_U, and the change rate of the USV's rudder angle $\dot{\delta}_U$. The second part is the terminal state, which describes the navigation destination of each episode. It contains the absolute azimuth of the terminal and USV_U, and the distance between USV_U and the terminal. The third part is the state of obstacle USVs, which reflects the navigation information of obstacle USVs near our USV, and for m obstacle USVs, there are m groups. It contains the absolute azimuth of USV_O and USV_O, the distance between our own USV and the obstacle USVs, and the obstacle USVs' course. The fourth part is the state of static obstacles with n sets of information for n obstacles USV. It contains the absolute azimuth of static obstacles and USV_O, and the distance between our USV and static obstacles.

The second part is the building of the action space. It is appropriate to design some actions in action space as the change of rudder angle because the USV changes its navigation state by rudder changes, and it can be designed as the following action space,

$$A = \{\Delta\delta_1, \Delta\delta_2, \Delta\delta_3, \Delta\delta_4, \ldots, \Delta\delta_k\} \quad (12)$$

After selecting the action, the change of target rudder angel $\Delta\delta_k$ is obtained, and the new target rudder angle is as follows,

$$\delta_r \leftarrow \delta_r + \Delta\delta_k \quad (13)$$

In this paper, 11 different sizes of actions are designed in the action space to enable the USV to adopt various behaviors, such as not changing the rudder angle, slightly changing, and hard changing. Therefore, the specific designs of action space are as follows,

$$A = \{-5^\circ, -4^\circ, -3^\circ, -2^\circ, -1^\circ, +0^\circ, +1^\circ, +2^\circ, +3^\circ, +4^\circ, +5^\circ\} \quad (14)$$

The third part is the design of the reward signal, which evaluates the various USV behaviors at each training step. The training is to continuously learn about the new environmental conditions and maximize their estimation of future benefits, but this estimation is derived from the design of reliable reward signals. The reward design of this paper is divided into the following two parts,

(1) The reward for goal
   (a) Terminal reward
   The terminal is where the end of USV navigation is on each training episode. The design of the terminal reward can encourage this good behavior and affect the whole training environment through bootstrap. When $\sqrt{(x_U - x_t)^2 + (y_U - y_t)^2} < r_{\min} + r_t$, it is considered that reaching the terminal, and getting the reward, R_t.
   (b) Collision reward
   Avoiding obstacles is another important goal in training. Punishment for collision can teach the trained USV to keep a safe distance from obstacles. When

$\sqrt{(x_U - x_O)^2 + (y_U - y_O)^2} < r$ or $\sqrt{(x_U - x_O)^2 + (y_U - y_O)^2} < r_O$, it is considered that colliding the obstacle USV. The collision reward obtained is R_O.

When $\sqrt{(x_U - x_O)^2 + (y_U - y_O)^2} < r + r_S$, it is considered that colliding the static obstacle. The collision reward is R_S.

(c) COLREGs reward

COLREGs provide a constraint for USV behaviors. Integrating COLREGs into the training process in a reward signal can endow the trained USV agent with regularized avoidance behavior. When $E \in \{E_3, E_4\}$ and $a \notin \{0°, +1°, +2°, +3°, +4°, +5°\}$, the reward signal $R_C = k_C u_{CRI}$ can be obtained. The more dangerous the moment of breaking the COLREGs, the higher the penalty for USV. When $E \in \{E_3, E_4\}$ and $a \in \{0°, +1°, +2°, +3°, +4°, +5°\}$, or $E \in \{E_1, E_2, E_5, E_6\}$, there are the conditions that the our USV should go straight or turn left or right. The designed reward signal is 0.

(d) Seamanship reward

When there are no obstacles or no duty to give way, our USV should keep straight as far as possible. Therefore, the following seamanship reward is designed to restrain the navigation behavior of the USV: When $a \notin \{0°\}$ and $u_{CRI} = 0$, the reward is R_δ.

(2) Guiding reward

The guiding reward can enrich the reward signal in a training environment and avoid the training difficulty caused by the sparse rewards problem.

(a) Course reward

The course that points more toward the terminal is considered to be a better state, so the course reward signal is designed as follows,

$$R_\varphi = \mathrm{k}_\varphi(\varphi_\mathrm{k} - |\varphi_U - \varphi_T|) \tag{15}$$

where φ_k is the critical value of the positive or negative reward.

(b) Course better reward

The agent's behavior is positive if it makes the course more pointed toward the terminal after an action, so the course better reward signal is designed as follows,

$$R_{\Delta\varphi} = \begin{cases} r_s & \text{, if } \phi \text{ smaller} \\ r_b & \text{, if } \phi \text{ bigger} \\ r_{e1} & \text{, if } \phi \text{ doesn't change, and } \phi = 0 \\ r_{e2} & \text{, if } \phi \text{ doesn't change, and } \phi \neq 0 \end{cases} \tag{16}$$

Thus, the complete reward signal function can be expressed as,

$$R = R_t + R_O + R_S + R_C + R_\delta + R_\varphi + R_{\Delta\varphi} \tag{17}$$

After designing the state space, action space, and reward signal, the training system is completed. Figure 5 shows the complete training architecture. At each step, the state information is input into the neural network, then the value of all actions based on the current network parameters and state is obtained through the neural network. Then the selected action is obtained, resulting in the environment update.

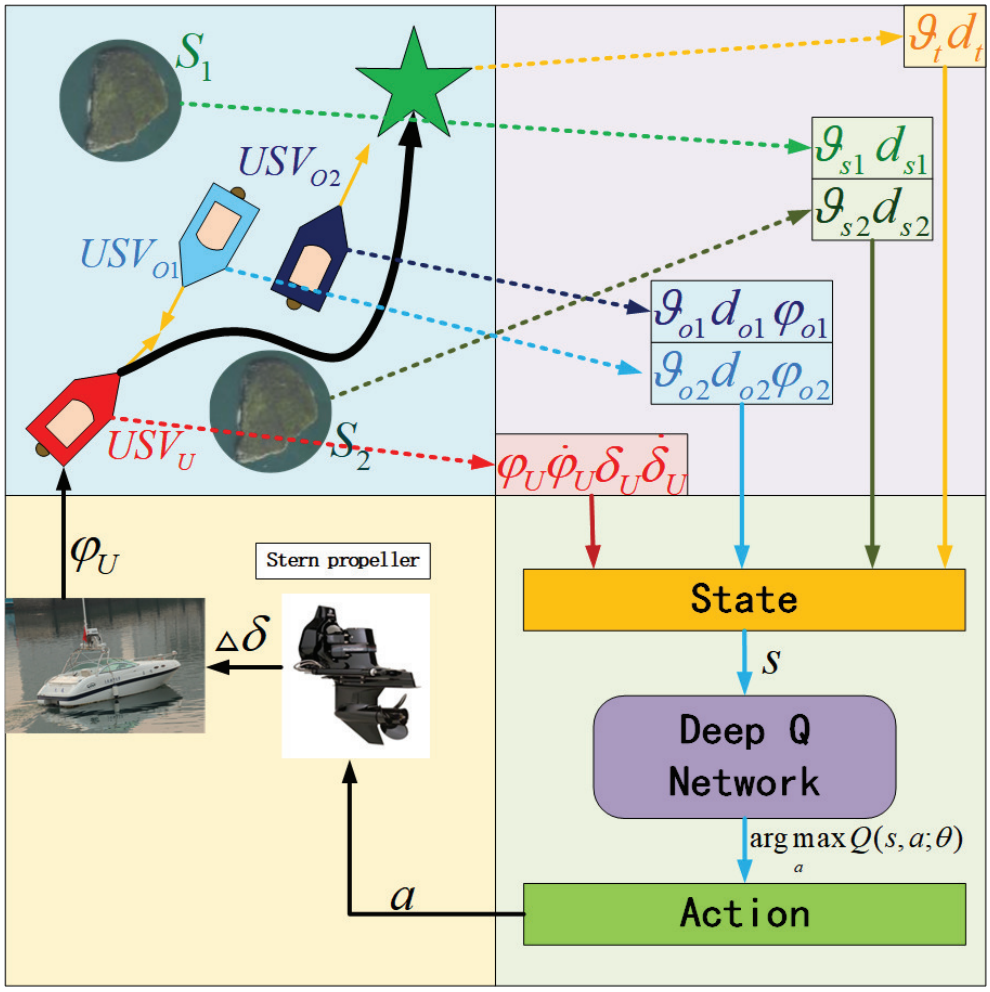


Figure 5. Training Architecture.

4. Improvement for USV Collision Avoidance Algorithm

4.1. Double DQN

In the training process of the DQN algorithm, uncontrollable overestimation of the action value generally exists, which is caused by the unknown of the real action value in the learning process, resulting in the performance of the training being affected and even falling into local optimal [39]. The DQN algorithm uses a greedy policy based on the action with the maximum action value in the action space, which will introduce significant maximization bias. This kind of overestimation is common. However, the influence of overestimation on the optimal policy can be reduced as much as possible by Double Q learning. In the traditional DQN algorithm, as shown in Equation (18), the action is chosen through the target network, while the value estimation output is also. This operation is the root cause of the overvaluation. Therefore, it can be decoupled by two neural network outputs. The action is selected by the evaluated network rather than based on the target network to reduce the impact of the overestimation problem [40]. A new calculation method can be obtained as shown in Equation (19).

$$Y_t^{DQN} = r + \gamma \max_{a'} Q(s', a'; \theta^-) \tag{18}$$

$$Y_t^{Double\_DQN} = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta); \theta^-) \tag{19}$$

4.2. Dueling DQN

Dueling neural network architecture is an outstanding method [41]. As shown in Equation (20), the action value function is divided into a state value function and an advantage function. The former describes the worth of a state, while the latter describes the relative importance of each action. It can distinguish the value of different states and actions, leading to more robust training.

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha) \tag{20}$$

4.3. Prioritized Experience Replay

As a pivotal part of the DQN algorithm, the experience replay buffer can reduce sample correlation and improve sample utilization. The traditional DQN algorithm uniformly samples the experience from the replay buffer after storing samples through action policy. However, this form may not be optimal. In different training conditions and steps, the value of each interaction for network training is distinct. If interactions have different sampling probability weights, the interactions that are more valuable to the current USV training will be assigned higher weights, which will be more conducive to agent learning [42]. Here a significant index, TD-error, forms an essential part of the loss function and a basis for gradient descent. This index provides easily accessible and valid evidence for the definition of priority, making the interactions in the experience replay buffer with a certain tilt. This random-priority method makes learning from the experience more robust. TD-error is as follows,

$$TD = r + \gamma \max_{a'} Q(s',a';\theta^-) - Q(s,a;\theta) \tag{21}$$

Interactions with high sampling weights are also not guaranteed to be sampled at any step, while those with low sampling weights are not necessarily not sampled. The priority only makes the samples that need to be chosen much easier. The probability of sampling is as follows,

$$P(i) = \frac{p_i^\alpha}{\sum_m p_m^\alpha} \tag{22}$$

where $P(i)$ is the probability of sampling for each interaction, and there are m interactions in the experience replay buffer. α is used to adjust the effect of priority. When $\alpha \to 0$, it is uniform sampling, and the higher α is, the more prominent the effect of priority on sampling. This paper adopts proportional prioritization as follows,

$$p_i = |TD_i| + \varepsilon \tag{23}$$

where, TD_i is the TD-error after normalization of $TD_i = \frac{TD_i}{TD_{\max}}$, and ε is a parameter to avoid the problem of zero denominators.

The off-policy method requires importance sampling weights (ISW) to correct the bias of estimation based on different samples. However, the transfer probability of the Q-learning algorithm is not dependent on policy but on the environment. Therefore, it does not need ISW. Nevertheless, the prioritized experience replay method breaks this advantage because the unbiasedness of the expected value estimation depends on the sample with the same distribution as the expected value. The prioritized experience replay changes the distribution of the samples in an uncontrolled way. Therefore, it is necessary to compensate for this bias by adding ISW, which is calculated as follows,

$$\omega_i = \left(\frac{1}{N}\frac{1}{P(i)}\right)^\beta \tag{24}$$

This normalized ISW can make the updating process more stable. β can adjust the impact of prioritized experience replay. When $\beta \to 1$, it cancels out the inconsistent probability effect of prioritized experience replay. So, its loss function is shown as follows,

$$L(\theta) = \mathbb{E}\Big[\omega_i TD_i{}^2\Big]. \tag{25}$$

4.4. Noisy Network

Because the training environment considered in this paper is multivariate, it places high demands on the exploratory capabilities of the algorithm. Although promoting the exploration with ε greedy policy is easy to implement and their effect is acceptable in most cases, it is not a reasonable enough choice because the agent acting up to the greedy-action with a specific probability value is too aimless. So, it can drive exploration by adding a learnable noise instead of ε greedy policy [43], which has not been used in USV collision avoidance. The weight and bias of the neural network become uncertain due to the noise parameters, which will increase the uncertainty of neural network output and promote agent exploration. After adding noise, the neural network can be expressed as follows,

$$y \doteq (\mu^w + \sigma^w \odot \varepsilon^w)x + \mu^b + \sigma^b \odot \varepsilon^b \tag{26}$$

where, weight w is divided into two parts, the weight μ^w without noise and the weight σ^w with noise. ε^w is the noise parameter, and $\odot$ represents element-wise multiplication for adding the noise to each weight σ^w in the neural network. Similarly, μ^b is the bias without noise, and σ^b is the bias with the noise parameter of ε^w. The noisy network forms a way of exploration by reasonably controlling the noise added to each network parameter, which can meet the different exploration needs in different training conditions. Factorized Gaussian noise is chosen to construct the noise in the neural network, which has a lower computational cost, that is suitable for the DQN algorithm. The noise can be constructed as follows,

$$\begin{cases} \varepsilon^w_{i,j} = \zeta(\varepsilon_i)\zeta(\varepsilon_j) \\ \varepsilon^b_j = \zeta(\varepsilon_j) \end{cases} \tag{27}$$

where, $\zeta(a) = \mathrm{sgn}(a)\sqrt{|a|}$ is a function for construction of noise. All the value of parameters ε_i and ε_j are obeyed Gaussian distribution as $\varepsilon \sim N(0, k)$. The variance k limits the noise size, and the higher variance means a greater ability to explore. The mean of zero means introducing noise parameters will not bias the original policy. Introducing a noisy network instead of the traditional neural network is more beneficial for environment exploration. The noise drives the exploration making the algorithm more flexible, reasonable, and efficient. So, its loss function is as follows,

$$\overset{'}{L}(\zeta)_{Noisy} = \mathbb{E}_{\varepsilon,\varepsilon^-}[\mathbb{E}_{(s,a,r,s')}[r + \gamma \max_{a'} Q(s', a', \varepsilon^-; \zeta^-) - Q(s, a, \varepsilon; \zeta)]^2]. \tag{28}$$

4.5. Improvements with USV Characteristics

Since the input to the neural network in the form of USV states contains scalar values with many differences in the order of magnitude, such as rudder angle values, direction values, and distance values, it is not reasonable to put them directly into training. USV state clipping is a benefit for improving the network training efficiency. By normalizing the input, the potential problem of the large descent gradient caused by the differential input can be avoided as much as possible, which makes neural network training more robust.

If the obstacles are too far from our own USV, the changes in the distance between the USVs do not affect the navigation of the USV. Therefore, it is not worth wasting more computational resources on learning these conditions. In this paper, a simplified way is designed, using the collision avoidance parameter of DA, the distances between USVs and

obstacles or the terminal more than the DA value R are limited to R, which can effectively improve the training effect.

The code for this Algorithm 1 is as follows,

Algorithm 1 DRLCA algorithm code

Initialize USV training environment
Initialize experience replay buffer H to capacity of C
Initialize evaluation neural network in θ
Initialize target network in $\theta^- = \theta$
Initialize variance of noise in k
For episode = 1, n do
  Initialize initial states of each USV and static obstacles
  Initialize speed of our USV and obtain ship domain size r in current episode
  While true
    Update the USV collision avoidance training environment
    Generate ε_i and ε_j
    Get noise parameters
    Select the action with $a_m = \arg\max_a Q(s', \arg\max_a Q(s', a; \theta), \varepsilon^-; \zeta^-)$
    Changing rudder angel by execute action $a^* = a_m$ in environment, and obtain s_{m+1}
    Obtain collision avoidance reward signal $R = R_t + R_O + R_S + R_C + R_\delta + R_\varphi + R_{\Delta\varphi}$
    Store current transition (s_m, a_m, r_m, s_{m+1}) in experience replay buffer H
    Assign current transition to highest priority $p_{\max}$
    By priority for each transition $P(i)$, sample the random minibatch of transitions (s_k, a_k, r_k, s_{k+1}) from H for learning
    Caculate ISW ω_i for each transition in minibatch
    Caculate TD-error TD_i.
    Obtain $y_i = \begin{cases} r_j, & if \ \mathrm{j}+1 \ is \ the \ terminal \\ r_j + \gamma \max_{a'} Q(s_j, a', \varepsilon^-; \zeta^-), & otherwise \end{cases}$
    Using gradient descent with ISW, Update evaluate network parameters θ
    Update $p_i = |TD_i| + \varepsilon$ for all samples
    If it is target network updating step xN
      update the weight $\theta^- = \theta$
    End if
    The number of steps counted plus 1
  End while
End for
Return the weight $\theta^* = \theta^-$ of target network

5. Experiments

5.1. Training Environment

Figure 6 shows a collision avoidance training environment designed with a single obstacle USV. The left part of this figure shows how the obstacle USV is generated, with 360 initial positions, spaced the degree of $\eta = 1^\circ$ evenly distributed around the circumference. The right part of this picture shows how the static obstacles are generated, with eight potential locations for each initial location of obstacle USV. Six of these locations are evenly distributed around the circumference of the circle, and the other two are at the intersection of line P1P3, line P4P6, and line P2P5. In each episode, randomly select two locations from these eight locations to generate static obstacles. It constitutes a simulation environment with $360 \times 8 \times 7 = 20{,}160$ random combinations of obstacles. In addition, the simulation in this paper is based on the Unity virtual marine simulation training and testing platform.

This experiment is based on 2× RTX 2080Ti GPUs, Xeon Gold 5218 CPU, Python 3.6, and Tensorflow 1.15 for environment building, algorithm training, and collision avoidance testing.

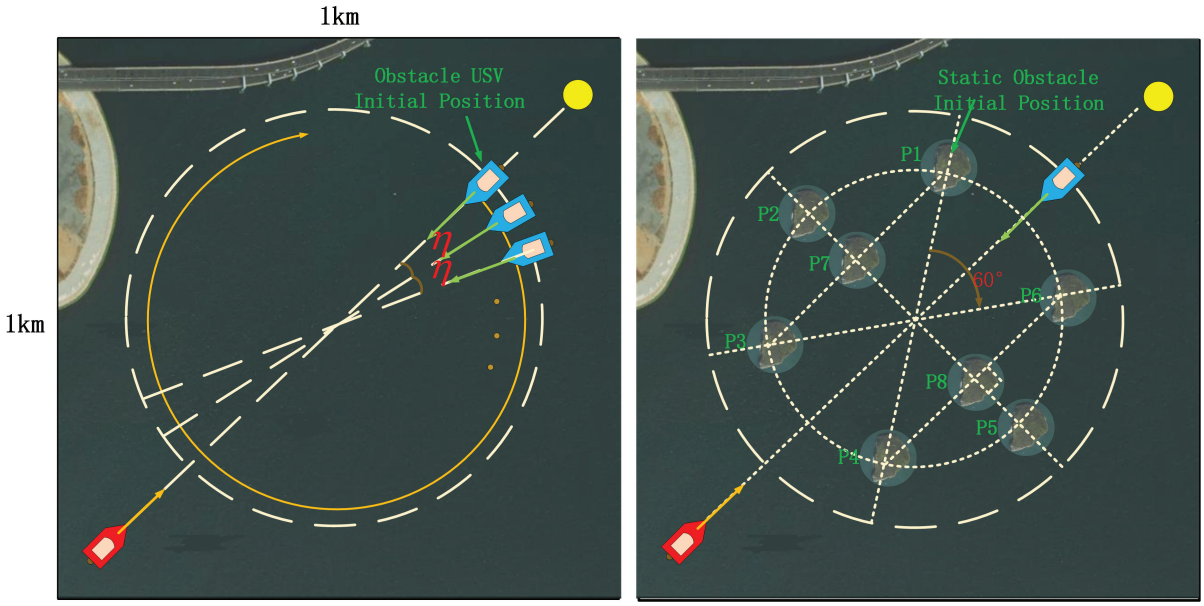


Figure 6. Training environment.

5.2. Framework for Training

Based on the various improvements of the algorithm structure and simulation environment, the training hyperparameters selected in this paper are shown in Table 1. For more stable training, the learning rate is designed to be very low and the discount factor very high, 0.0001 and 0.99, respectively. The update frequency of the target network is 4096, and the sampling number is 600. The noise variance is 0.1, and the mean value is 0, which ensures a suitable exploration capability while avoiding the bias caused by noise. The ISW is initially 0.5, and the priority experience replay factor is 0.4. For better training, start training on the 2000th step. The experience replay memory size of 1,000,000 ensures that no experience is dropped.

Table 1. Hyperparameters for training.

Hyperparameter	Value
Learning Rate	0.0001
Discount Factor	0.99
Target Network Update Frequency	4096
Replay Memory Size	1,000,000
Batch Size	600
Noise Variance	0.1
Noise Mean	0.0
Greedy Value	1.0
Importance Sampling	0.5
Linearly Anneal of Importance Sampling	1.25×10^{-6}
Priority Experience Replay	0.4
Replay Start Size	2000

Figure 7 shows the neural network architecture in this paper. With 900 neurons in each layer, the green part is the traditional layer of the Q network, the yellow is the noise layer, the blue is the dueling network architect, the orange is the addition operation, the red is state input, and the black is the action output. The optimizer is Adam. The activation function is Lecky_ReLU.

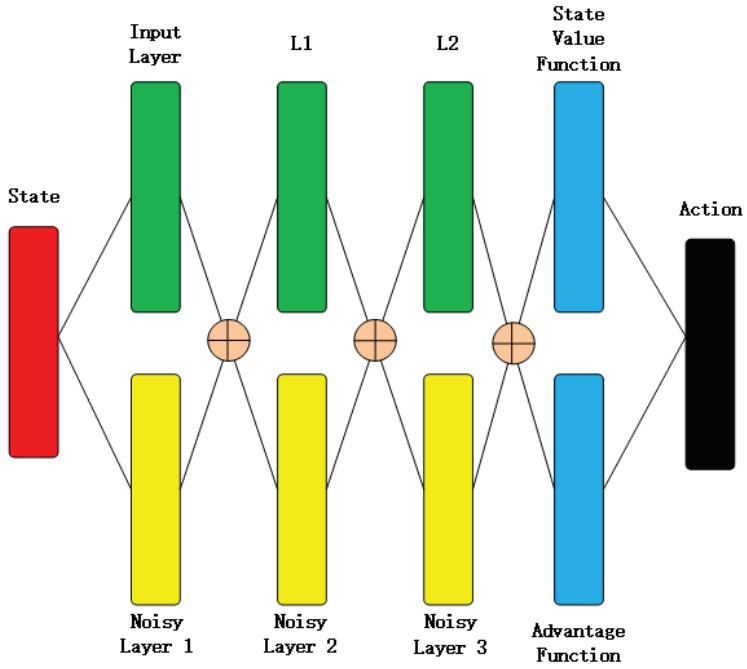


Figure 7. Neural network architecture.

5.3. Training

The average reward in training is shown in Figure 8. The same ten sets of random number seeds are selected for all algorithms for ten simulations, and the average reward graph containing confidence regions is plotted. In ten simulations, the average reward per 40 episodes is averaged to one value, forming ten values, and the point on the curve is the average of these ten values. The shaded part in this figure shows the confidence region, whose upper and lower bounds are the maximum and minimum values among these ten values. The means of the average reward value of each algorithm over different training stages is shown in Table 2. An individual improvement is limited, but the algorithm proposed in this paper, DRLCA, combining all improvement methods, has a very significant collision avoidance performance. The average reward of DRLCA rises very fast in the early stage, especially the first point on the curve, which is already much higher than others and smooths out at a higher average reward value position in the later stage of training.

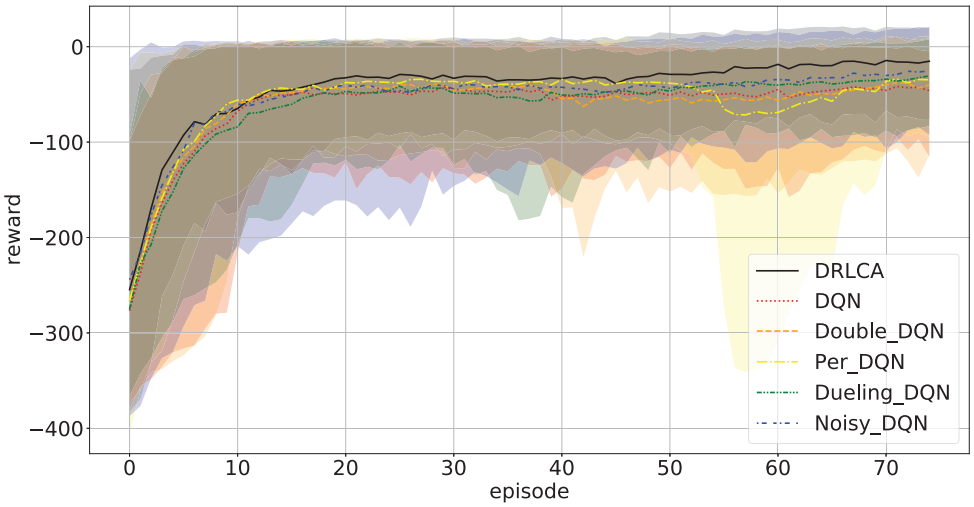


Figure 8. Average reward.

Table 2. Mean of rewards at different stages of training.

Alogrithm	First Third of Training	Middle Third of Training	Last Third of Training
DRLCA	−75.86	−32.61	−20.83
DQN	−91.05	−48.37	−47.26
Double DQN	−85.33	−47.62	−49.55
Per DQN	−81.92	−37.46	−51.17
Dueling DRLCA	−96.42	−48.82	−38.18
Noisy DQN	−81.97	−43.11	−34.09

As Figure 9a shows the condition of USV training in the first episode, the USV will keep rotation because the agent has no knowledge about this environment and is basing its movement on random exploration. Figure 9b shows the training effect in the 9th episode, where the USV tries to collect more experience. Figure 9c shows the training effect in the 20th episode. The USV has tried more behaviors to explore this environment. Figure 9d shows the training effect in the 46th episode, where the USV reaches the terminal for the first time, which is very important for the training and proves that the guidance reward designed in this paper is very effective. Figure 9e shows the effect in the 124th episode, where the failed collision experience with the obstacle USV is crucial for better learning. As shown in Figure 9f, the effect in the 358th episode, the USV constantly optimizes its behavior, and within the next thousands of training episodes, the optimal collision avoidance policy is approached continuously.

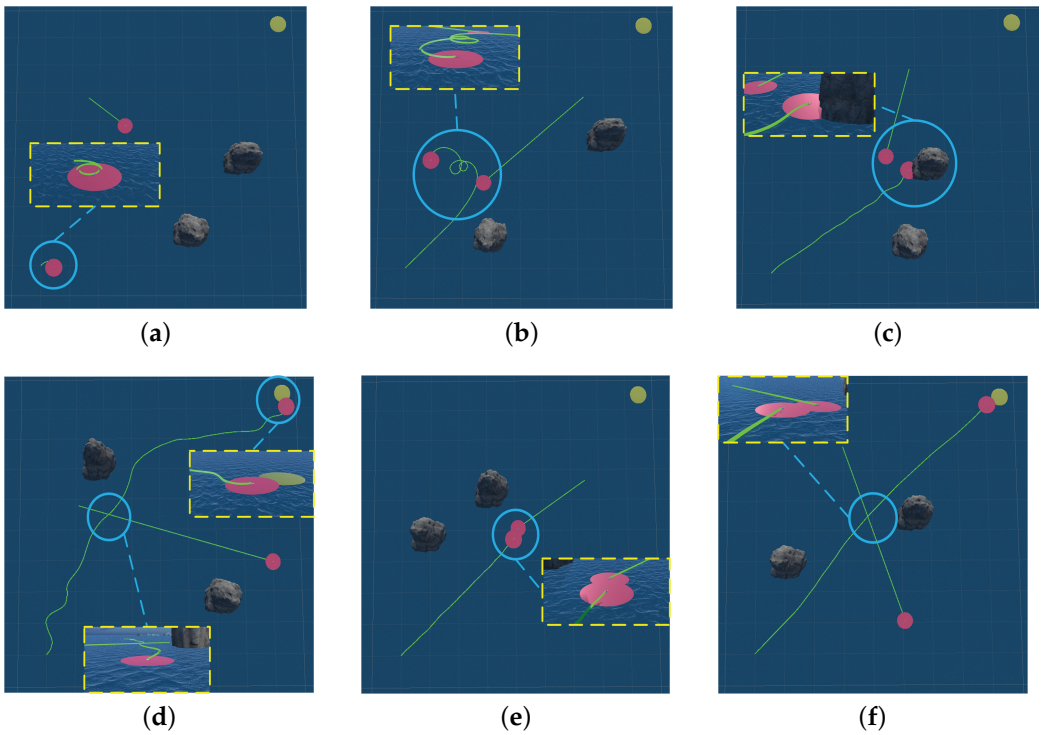


Figure 9. Training. (**a**) First episode; (**b**) 9th episode; (**c**) 20th episode; (**d**) 46th episode; (**e**) 124th episode; (**f**) 358th episode.

To verify whether all the improvements have positive effects and whether removing one leads to better training, an ablation study is performed. The average reward of the ablation study is shown in Figure 10. The means of the average reward value of each algorithm over different ablation study stages is shown in Table 3. The average reward height and increase rate of the algorithms with any one improvement removed are lower than the DRLCA, verifying all the improvement methods are complementary.

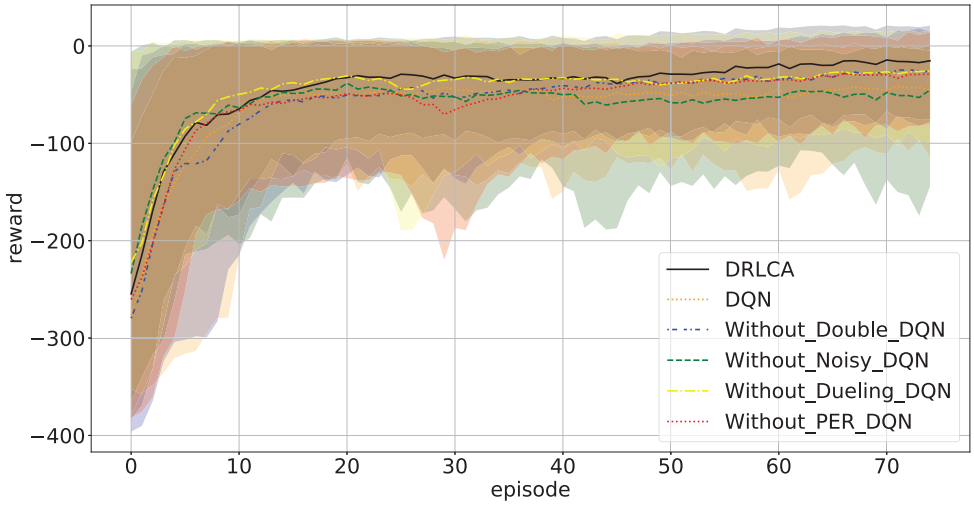


Figure 10. Ablation study.

Table 3. Mean of rewards at different stages of ablation study.

Alogrithm	First Third of Training	Middle Third of Training	Last Third of Training
DRLCA	−75.86	−32.61	−20.83
DQN	−91.05	−48.37	−47.26
Without Double DQN	−97.36	−43.94	−31.61
Without Per DQN	−88.91	−49.68	−33.53
Without Dueling DRLCA	−69.30	−35.92	−31.44
Without Noisy DQN	−73.84	−52.57	−52.04

5.4. Test

The first test environment is shown in Figure 11, where our USV is in the No. 6 encounter situation with the USV_O, and USV_U should avoid the USV_O and steers both to port and starboard are allowed. Concurrently, the two static obstacles do not obviously block the navigation of our own USV that can be used to test whether the USV collision avoidance agent has learned to ignore the non-hazardous obstacles. Figure 11a shows the effect of collision avoidance of the DRLCA algorithm and Figure 11b DQN algorithm. The initial position of the USV_O, static obstacles 1 and 2 are $(402.33, 783.66)$, $(216.34, 402.33)$, and $(442.76, 205.51)$. The course of obstacle USV is 161°. Figure 11c shows the change of rudder angle and course of collision avoidance of the DRLCA, Figure 11d DQN algorithm. Figure 11e shows the changes in the distance of collision avoidance of the DRLCA algorithm, Figure 11f DQN algorithm. By adding up $180 - \phi$ (if $\phi > 180°$) or ϕ (if $\phi \leq 180°$) in each second, the accumulated course deviation can be obtained, and the DRLCA is 1730.28°and DQN is 3781.80°. The closest distances to the three obstacles are 43.89 m, 136.66 m, and 162.22 m for DRLCA, and 129.81 m, 169.04 m, and 104.55 m for DQN.

The second test environment is shown in Figure 12, where it is in the No. 1 encounter situation, and USV_U should avoid the USV_O and steers both to port and starboard are allowed. In addition, there are two static obstacles, one of which does not affect our USV's navigation, and the other does. It tests the collision avoidance ability when encountering static obstacles and USV at the same time. Figure 12a,b show the effect of collision avoidance of DRLCA and DQN. The initial position of the USV_O, static obstacles 1 and 2 are $(654.51, 757.15)$, $(505.24, 799.95)$, and $(628.56, 422.74)$. The course of obstacle USV is 211°. Figure 12c,d shows the change of rudder angle and course of collision avoidance of DRLCA and DQN. Figure 12e,f shows the distance in collision avoidance of DRLCA and DQN. In the index of accumulated course deviation, the DRLCA is 707.51°, and DQN is 2357.59°.

The closest distances to the three obstacles are 44.32 m, 245.9 m, and 99.51 m for DRLCA and 57.6 m, 274.29 m, and 66.81 m for DQN.

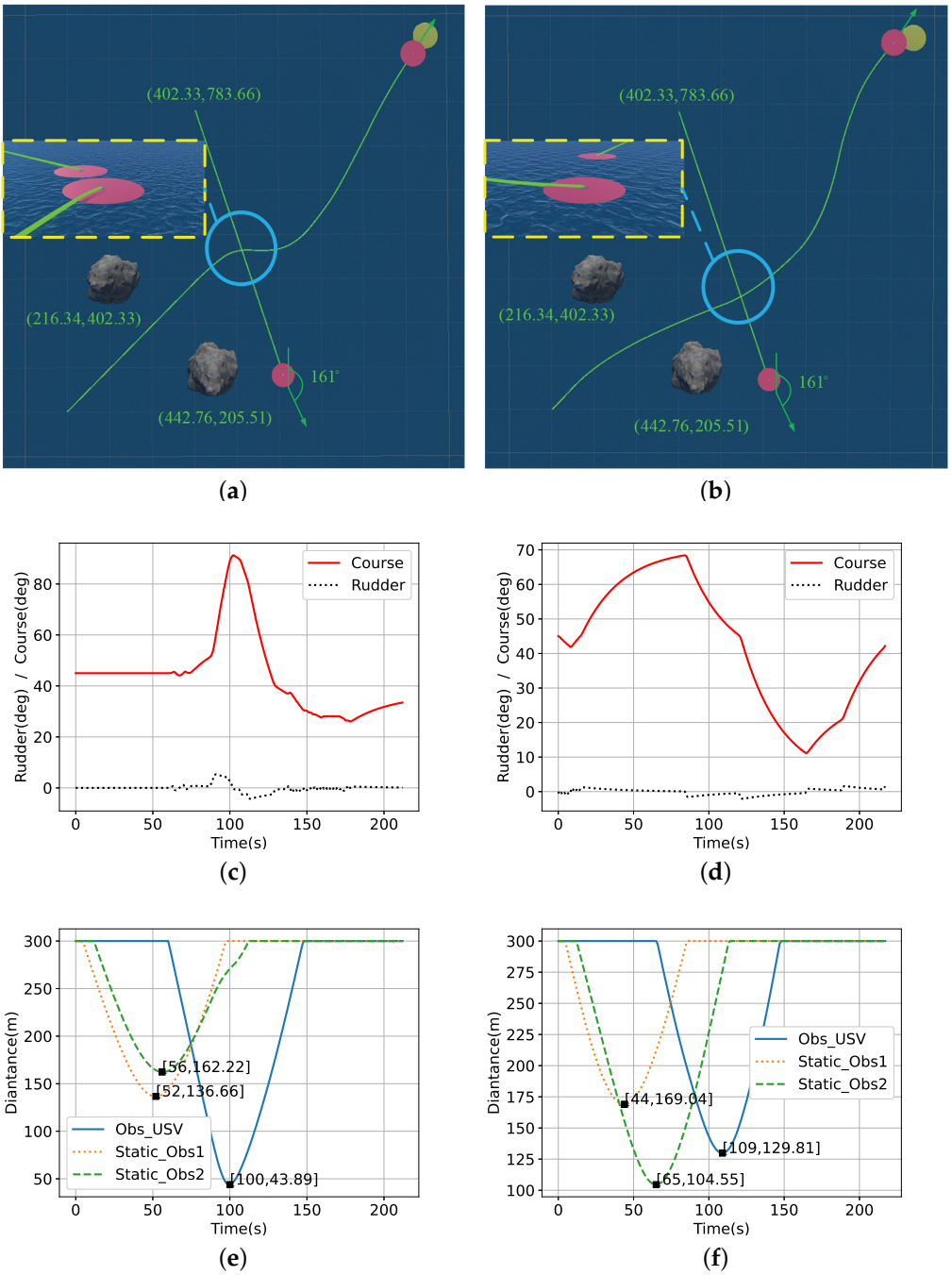


Figure 11. Encounter situation 1. (**a**) Path planned by DRLCA; (**b**) Path planned by DQN; (**c**) Course and rudder for DRLCA; (**d**) Course and rudder for DQN; (**e**) Distance for DRLCA; (**f**) Distance for DQN.

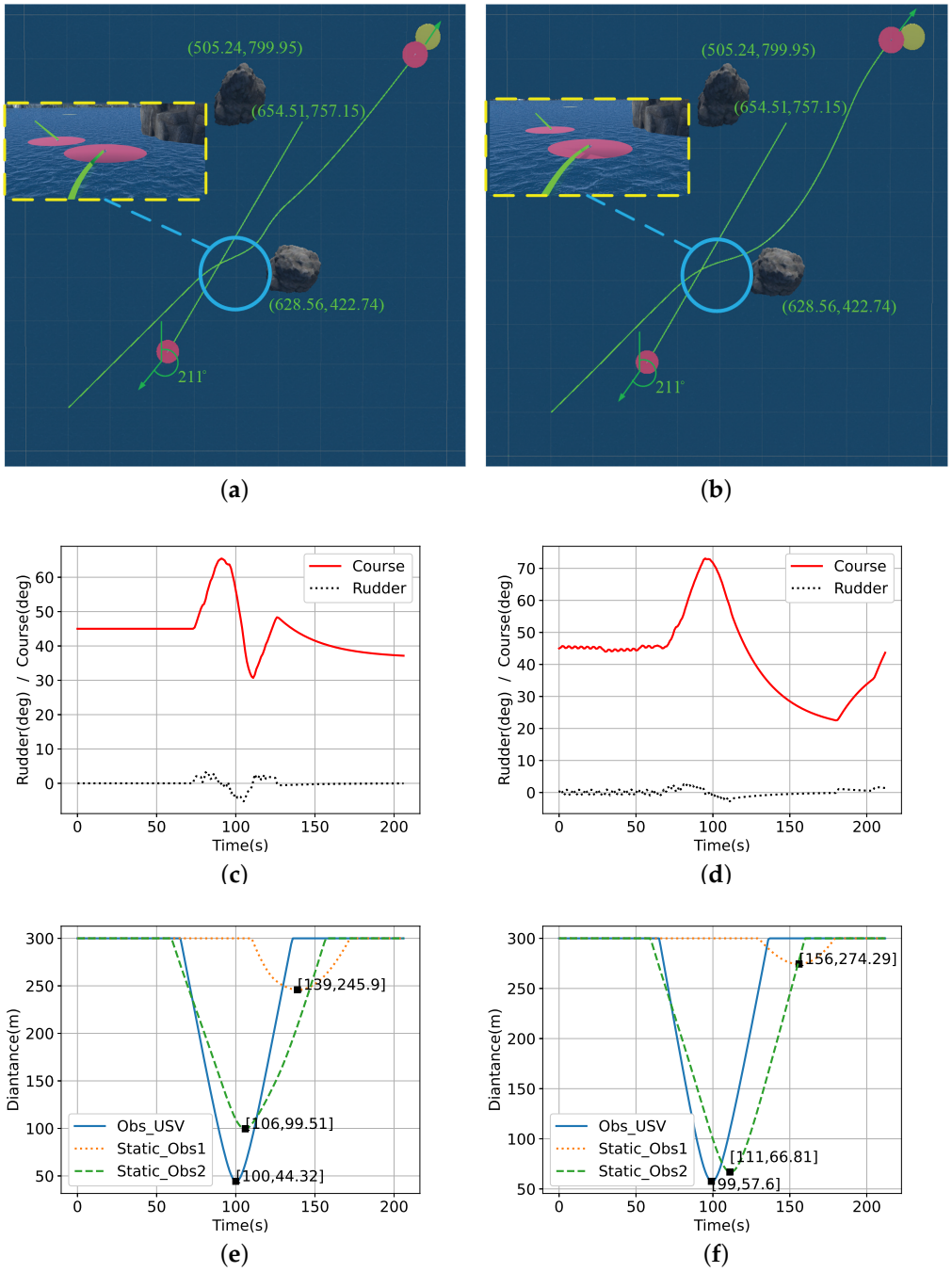


Figure 12. Encounter situation 2. (**a**) Path planned by DRLCA; (**b**) Path planned by DQN; (**c**) Course and rudder for DRLCA; (**d**) Course and rudder for DQN; (**e**) Distance for DRLCA; (**f**) Distance for DQN.

The third test environment is shown in Figure 13. The interference from the obstacle USV is not significant, which can verify the ability to avoid static obstacles. Figure 13a,b show the effect of collision avoidance of DRLCA and DQN. The initial position of the USV_O, static obstacles 1 and 2 are $(789.77, 422.35)$, $(538.82, 644.89)$, and $(461.18, 355.11)$. The course of obstacle USV is 285. Figure 13c,d shows the change of rudder angle and course of collision avoidance of DRLCA and DQN. Figure 13e,f shows the distance in collision avoidance of DRLCA and DQN. The accumulated course deviation is 3371.51° for DRLCA, and DQN could not compare because of the incomplete navigation. The closest distances to the three obstacles are 136.02 m, 100.57 m, and 138.48 m for DRLCA and 36.45 m, 65.79 m, and 103.96 m for DQN.

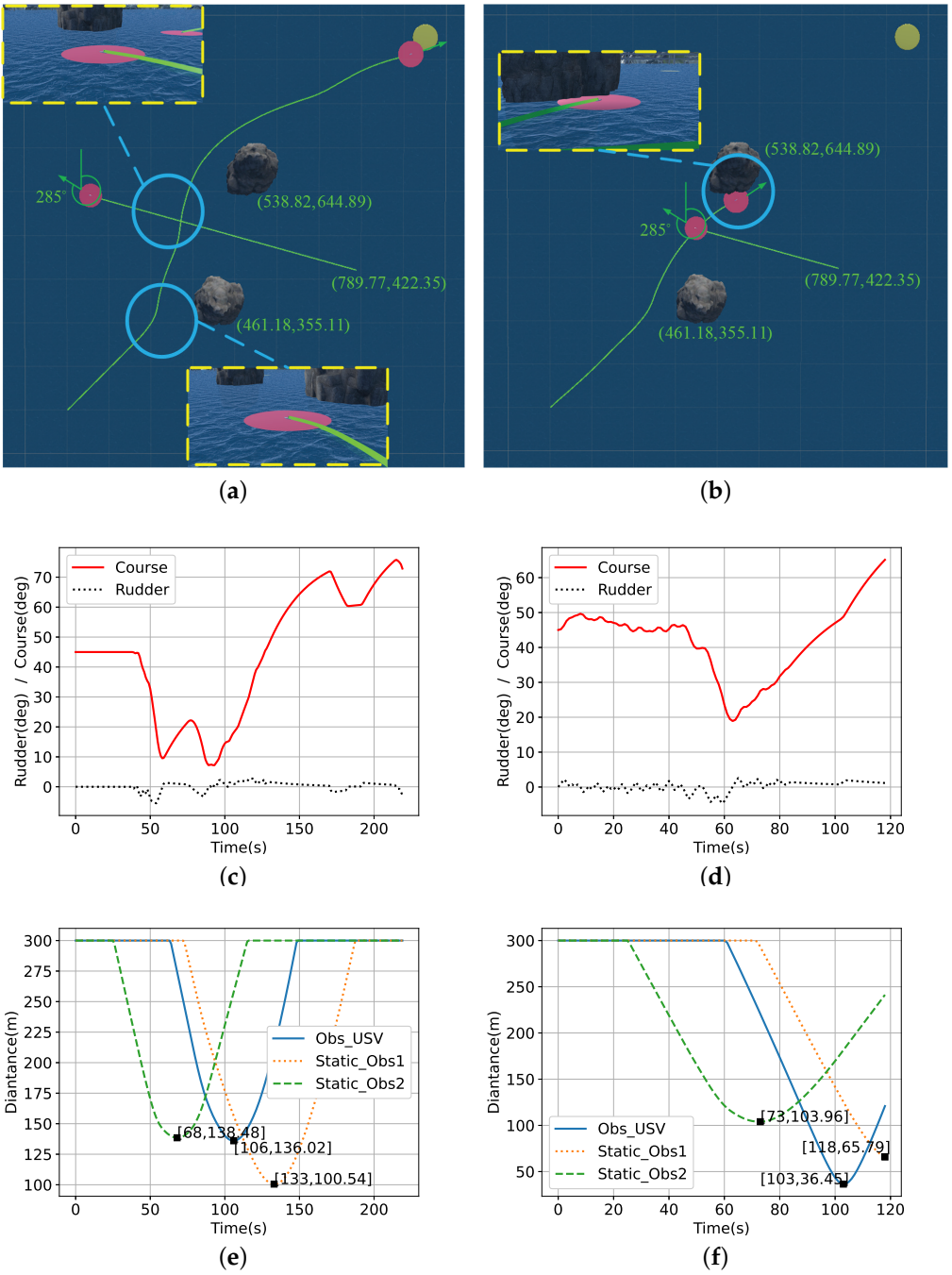


Figure 13. Encounter situation 3. (**a**) Path planned by DRLCA; (**b**) Path planned by DQN; (**c**) Course and rudder for DRLCA; (**d**) Course and rudder for DQN; (**e**) Distance for DRLCA; (**f**) Distance for DQN.

The fourth test environment is shown in Figure 14. Our USV is in the No. 4 encounter situation with the obstacle USV, and there is no interference from the static obstacles. Figure 14a,b show the effect of collision avoidance of DRLCA and DQN. The initial position of the USV_O, static obstacles 1 and 2 are $(363.80, 232.70)$, $(232.70, 636.20)$, and $(633.65, 431.90)$. The course of obstacle USV is 27°. Figure 14c,d shows the change of rudder angle and course of collision avoidance of DRLCA and DQN. Figure 14e,f shows the distance in collision avoidance of DRLCA and DQN. The accumulated course deviation is 909.19° and 2074.92° for DRLCA and DQN. The closest distances to the three obstacles are 55.11 m, more than 300 m, and 88.98 m for DRLCA and 33.16 m, 265.91 m, and 183.41 m for DQN.

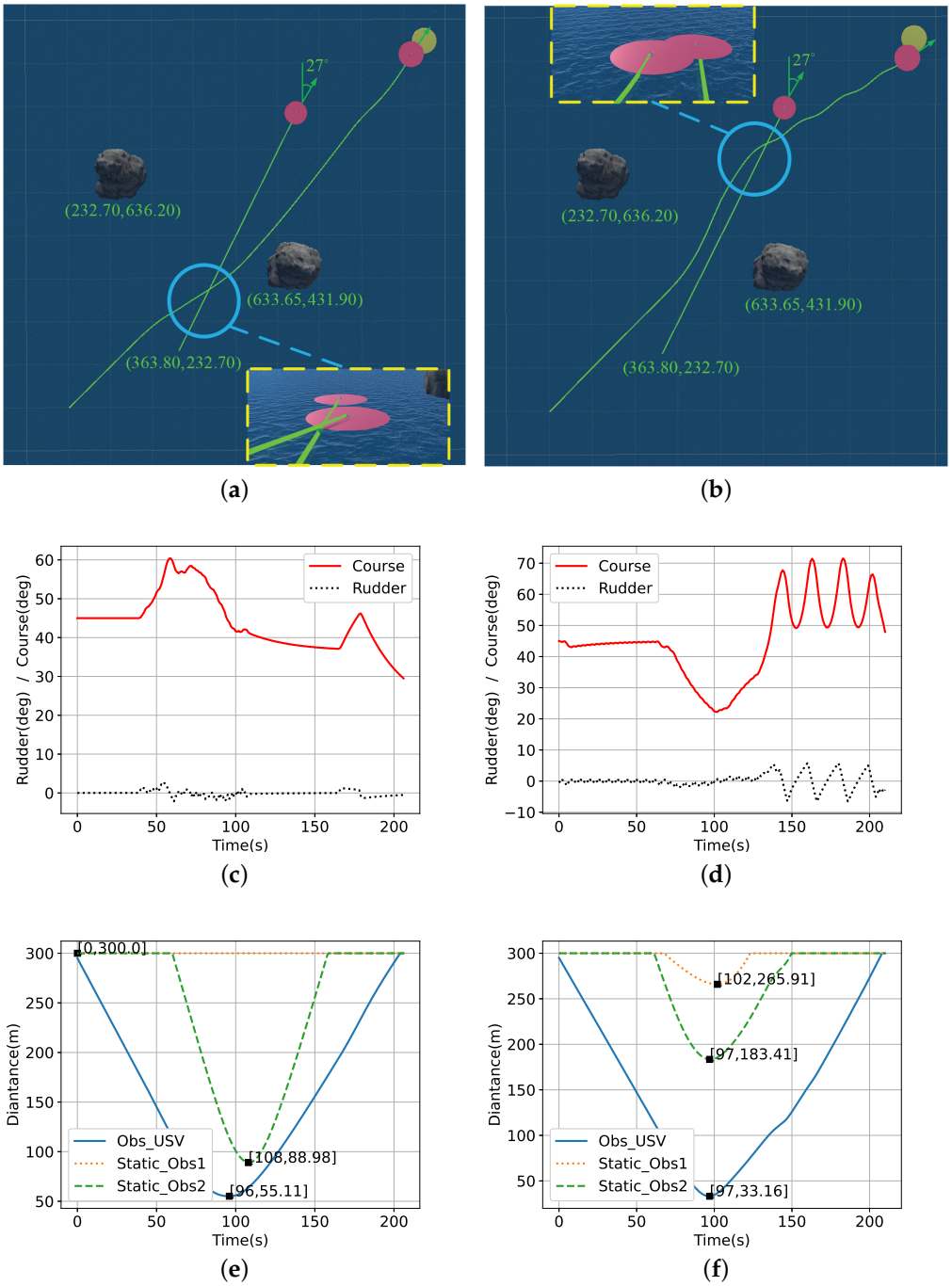


Figure 14. Encounter situation 4. (**a**) Path planned by DRLCA; (**b**) Path planned by DQN; (**c**) Course and rudder for DRLCA; (**d**) Course and rudder for DQN; (**e**) Distance for DRLCA; (**f**) Distance for DQN.

Table 4 shows the comparison results for four experiments. In groups 1, 2, and 4 experiments, the DRLCA reduced the total course deviation by 54.25%, 70.00%, and 56.18%, respectively, compared with the DQN, with an average improvement of 60.14%.

In total, 100 experiments of collision avoidance under the same random number seed are carried out. As shown in Table 5, the results of these experiments are recorded. The number of successful arrivals, out-of-bounds, and collisions are also recorded, and the success rate of the DRLCA is much higher. Because the failed collision avoidance will affect the result of the accumulated course deviation, only the experiments that reach the terminal are used for the calculation. The accumulated course deviation for DRLCA is 2150.02°, and DQN is 2929.36°, improving 26.60%. Finally, the average time per experiment is compared

for DRLCA and DQN, which are 211.875 and 214.304 s, improving by 1.13%. The improved algorithm performs better on collision avoidance problems in this environment.

Table 4. Results of four experiments.

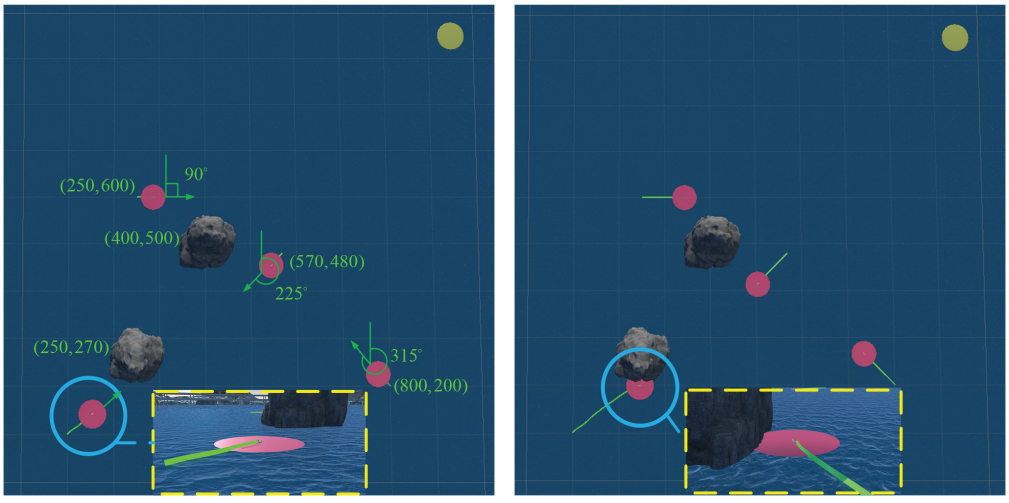
Experiment	Result	Course Deviation
Test environment 1, DRLCA	arrival	1730.28°
Test environment 1, DQN	arrival	3781.80°
Test environment 2, DRLCA	arrival	707.51°
Test environment 2, DQN	arrival	2357.59°
Test environment 3, DRLCA	arrival	3371.51°
Test environment 3, DQN	collision	/
Test environment 4, DRLCA	arrival	909.19°
Test environment 4, DQN	arrival	2074.92°

Table 5. Results of 100 times experiments.

Algorithm	Successful Arrival	Out of Bound	Collision	Average Accumulated Deviation of Course	Average Time
DRLCA	97	3	0	2150.02°	211.875 s
DQN	56	30	14	2929.36°	214.304 s

5.5. Multi-Obstacle USV Collision Avoidance

A test effect diagram of multi-obstacle USV collision avoidance is shown in Figure 15. As shown in Figure 15a, the first stage shows the initial condition of a test environment. The obstacle USVs are at the position of $(250, 600)$, $(570, 480)$, and $(800, 200)$. The static obstacles are at $(250, 270)$ and $(400, 500)$. The course of obstacle USVs are 90°, 225°, and 315°. As shown in Figure 15b, stage 2 is a condition when our own USV avoids a static obstacle. As shown in Figure 15c, our USV encounters USV_{O2} in the No. 1 encounter situation, turns to the starboard, and successfully avoids the obstacle USV. As shown in Figure 15d, our USV encounters the USV_{O3} in the No. 3 encounter situation. Our USV successfully turned to the starboard to avoid the obstacle USV according to the COLREGs. As shown in Figure 15e, stage 5, the USV_U is close to the USV_{O1}, but there is no hazard of collision. Therefore, our USV continued to navigate to the terminal. Finally, as shown in Figure 15f, our USV arrives at the terminal.



(**a**) (**b**)

Figure 15. *Cont.*

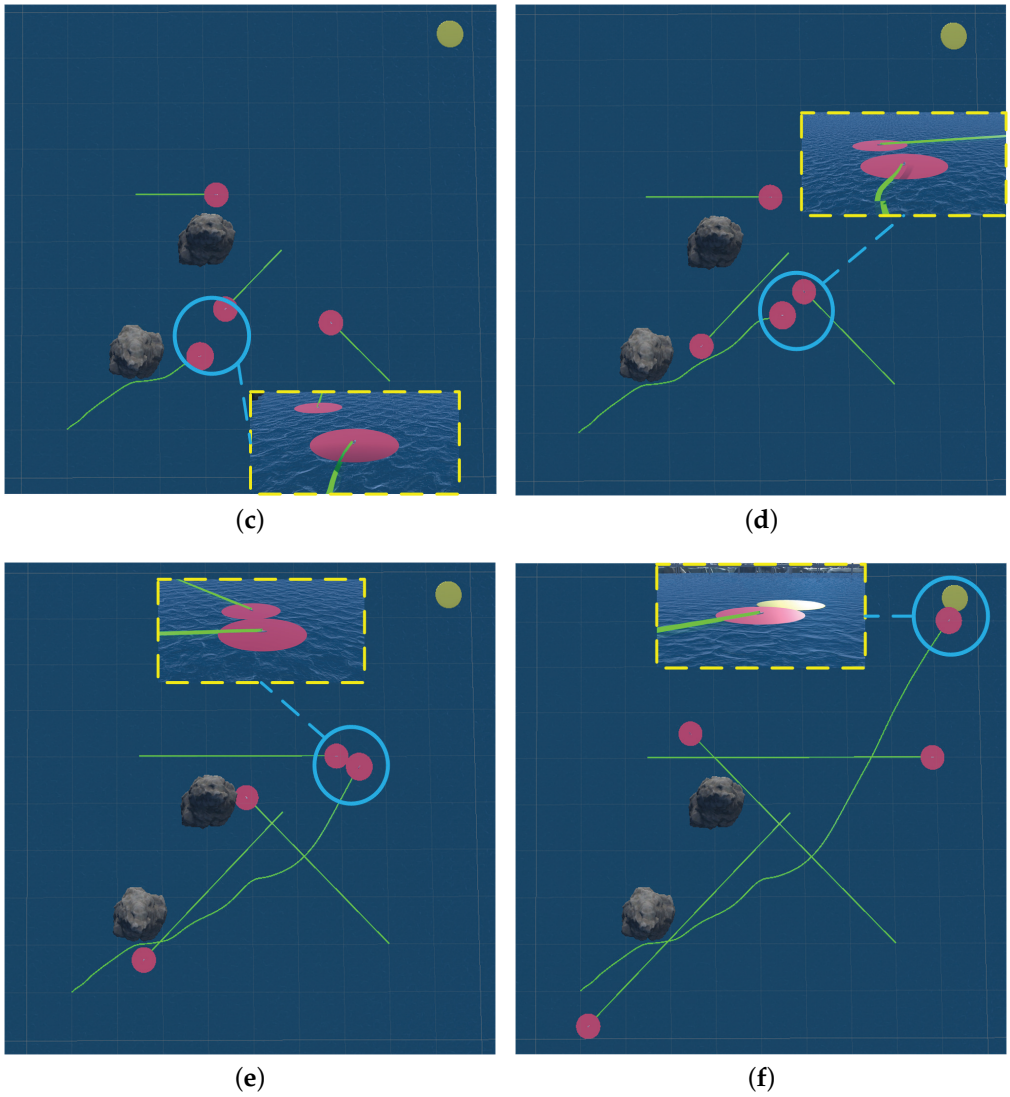


Figure 15. Multi-USVs collision avoidance environment. (**a**) Test stage 1; (**b**) Test stage 2; (**c**) Test stage 3; (**d**) Test stage 4; (**e**) Test stage 5; (**f**) Test stage 6.

6. Conclusions

This paper proposes an autonomous USV collision avoidance framework, DRLCA, which can be applied to USV navigation. The collision avoidance characteristics and maneuverability of USV are considered, and an efficient method for collision avoidance agent training is designed accordingly. A dueling architecture and a double learning method are used to improve training efficiency. Prioritize experience replay method is used instead of the uniform sampling method to improve sample utilization. The noisy network method, which has not been applied to the USV collision avoidance problem, is used to increase the exploration capability in USV training, verifying the feasibility of this method. Combining the characteristics of USV collision avoidance, two effective improvement methods are proposed in this paper, namely USV state clipping and DA distance restriction. Combined with the Unity virtual marine platform, which has realistic physical characteristics, the effect of the DRLCA is reliably verified and compared. The result shows that the improved collision avoidance algorithm proposed in this paper has a superior USV collision avoidance effect.

In the future, the ship domain will be replaced by an ellipse that can vary with speed to achieve a more accurate simulation of realistic collision avoidance situations. At the same time, the reward signal will be designed to be more detailed and associated with different

information, such as distance, speed, and angle. How the stability and generalization ability of the algorithm can be further improved will be investigated to cope with the situation that about 3% of the DRLCA algorithm proposed in this study still does not arrive at the terminal.

Author Contributions: Funding acquisition, G.W.; Writing—original draft, Z.S.; Review and editing, Y.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by "National Natural Science Foundation of China" (Grant number 61976033), "Key Development Guidance Program of Liaoning Province of China" (Grant number 2019JH8/10100100), "Pilot Base Construction and Pilot Verification Plan Program of Liaoning Province of China" (Grant number 2022JH24/10200029), "China Postdoctoral Science Foundation" (Grant number 2022M710569).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author, [Yunsheng Fan], upon reasonable request.

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

Abbreviations

The following abbreviations are used in this manuscript:

USV	unmanned surface vehicle
DQN	deep Q network
DRL	deep reinforcement learning
COLREGs	international regulations for preventing collisions at sea
DRLCA	deep reinforcement learning collision avoidance
DDPG	deep deterministicpolicy gradient
LSTM	long short-term memory
SD	ship domain
DA	dynamic area
CRI	collision risk index
DCPA	the distance at the closest point of approaching
TCPA	time to the closest point of approaching

References

1. Dabrowski, P.S.; Specht, C.; Specht, M. Integration of multi-source geospatial data from GNSS receivers, terrestrial laser scanners, and unmanned aerial vehicles. *Can. J. Remote Sens.* **2021**, *47*, 621–634. [CrossRef]
2. Kurowski, M.; Thal, J.; Damerius, R. Automated survey in very shallow water using an unmanned surface vehicle. *IFAC-PapersOnLine* **2019**, *52*, 146–151. [CrossRef]
3. Li, C.; Jiang, J.; Duan, F. Modeling and experimental testing of an unmanned surface vehicle with rudderless double thrusters. *Sensors* **2019**, *19*, 2051. [CrossRef]
4. Luis, S.Y.; Reina, D.G.; Marín, S.L.T. A multiagent deep reinforcement learning approach for path planning in autonomous surface vehicles: The Ypacaraí lake patrolling case. *IEEE Access* **2021**, *9*, 17084–17099. [CrossRef]
5. Mu, D.; Wang, G.; Fan, Y. Adaptive trajectory tracking control for underactuated unmanned surface vehicle subject to unknown dynamics and time-varing disturbances. *Appl. Sci.* **2018**, *8*, 547. [CrossRef]
6. Stateczny, A.; Specht, C.; Specht, M. Study on the positioning accuracy of GNSS/INS systems supported by DGPS and RTK receivers for hydrographic surveys. *Energies* **2021**, *14*, 7413. [CrossRef]
7. Gao, S.; Liu, C.; Tuo, Y. Augmented model-based dynamic positioning predictive control for underactuated unmanned surface vessels with dual-propellers. *Ocean Eng.* **2022**, *266*, 112885. [CrossRef]
8. Li, Y.; Zhang, H. Collision Avoidance Decision Method for Unmanned Surface Vehicle Based on an Improved Velocity Obstacle Algorithm. *J. Mar. Sci. Eng.* **2022**, *10*, 1047. [CrossRef]
9. Ren, J.; Zhang, J.; Cui, Y. Autonomous obstacle avoidance algorithm for unmanned surface vehicles based on an improved velocity obstacle method. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 618. [CrossRef]

10. Fan, Y.; Sun, X.; Wang, G. Collision avoidance controller for unmanned surface vehicle based on improved cuckoo search algorithm. *Appl. Sci.* **2021**, *11*, 9741. [CrossRef]
11. Guan, W.; Wang, K. Autonomous Collision Avoidance of Unmanned Surface Vehicles Based on Improved A-Star and Dynamic Window Approach Algorithms. *IEEE Intell. Transp. Syst. Mag.* **2023**, 2–17. [CrossRef]
12. Silver, D.; Schrittwieser, J.; Simonyan, K. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef] [PubMed]
13. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
14. Wang, N.; Gao, Y.; Zhao, H. Reinforcement learning-based optimal tracking control of an unknown unmanned surface vehicle. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *32*, 3034–3045. [CrossRef]
15. Bastani, H.; Drakopoulos, K.; Gupta, V. Efficient and targeted COVID-19 border testing via reinforcement learning. *Nature* **2021**, *599*, 108–113. [CrossRef]
16. Kiran, B.R.; Sobh, I.; Talpaert, V. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 4909–4926. [CrossRef]
17. Chen, C.; Chen, X.Q.; Ma, F. A knowledge-free path planning approach for smart ships based on reinforcement learning. *Ocean. Eng.* **2019**, *189*, 106299. [CrossRef]
18. Li, L.; Wu, D.; Huang, Y. A path planning strategy unified with a COLREGS collision avoidance function based on deep reinforcement learning and artificial potential field. *Appl. Ocean. Res.* **2021**, *113*, 102759. [CrossRef]
19. Shen, H.; Hashimoto, H.; Matsuda, A. Automatic collision avoidance of multiple ships based on deep Q-learning. *Appl. Ocean. Res.* **2019**, *86*, 268–288. [CrossRef]
20. Zhou, C.; Wang, Y.; Wang, L. Obstacle avoidance strategy for an autonomous surface vessel based on modified deep deterministic policy gradient. *Ocean Eng.* **2022**, *243*, 110166. [CrossRef]
21. Du, Y.; Zhang, X.; Cao, Z. An Optimized Path Planning Method for Coastal Ships Based on Improved DDPG and DP. *J. Adv. Transp.* **2021**, *2021*, 7765130. [CrossRef]
22. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
23. Xu, X.; Cai, P.; Ahmed, Z. Path planning and dynamic collision avoidance algorithm under COLREGs via deep reinforcement learning. *Neurocomputing* **2022**, *468*, 181–197. [CrossRef]
24. Chen, C.; Ma, F.; Xu, X. A Novel Ship Collision Avoidance Awareness Approach for Cooperating Ships Using Multi-Agent Deep Reinforcement Learning. *J. Mar. Sci. Eng.* **2021**, *9*, 1056. [CrossRef]
25. Norrbin, N.H. Theory and observations on the use of a mathematical model for ship manoeuvring in deep and confined waters. In *Publication 68 of the Swedish State Shipbuilding Experimental Tank, Proceedings of the 8th Symposium on Naval Hydrodynamics, Pasadena, CA, USA, 24–28 August 1970*; Elanders Boktryckeri Aktiebolag: Göteborg, Sweden, 1971; pp. 807–905.
26. Fan, Y.; Sun, Z.; Wang, G. A Novel Reinforcement Learning Collision Avoidance Algorithm for USVs Based on Maneuvering Characteristics and COLREGs. *Sensors* **2022**, *22*, 2099. [CrossRef]
27. Fujii, Y.; Tanaka, K. Traffic capacity. *J. Navig.* **1971**, *24*, 543–552. [CrossRef]
28. Piray, P.; Daw, N.D. Linear reinforcement learning in planning, grid fields, and cognitive control. *Nat. Commun.* **2021**, *12*, 4942. [CrossRef]
29. Aytar, Y.; Pfaff, T.; Budden, D. Playing hard exploration games by watching YouTube. In Proceedings of the 32nd International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 3–8 December 2018; pp. 2935–2945.
30. Bellemare, M.G.; Candido, S.; Castro, P.S. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature* **2020**, *588*, 77–82. [CrossRef]
31. Sutton, R.S.; Batro, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
32. Dabney, W.; Kurth-Nelson, Z.; Uchida, N. A distributional code for value in dopamine-based reinforcement learning. *Nature* **2020**, *577*, 671–675. [CrossRef]
33. Bain, A. *The Emotions and the Will*; John W. Parker and Son: London, UK, 1859.
34. Alagoz, O.; Hsu, H.; Schaefer, A.J. Markov decision processes: A tool for sequential decision making under uncertainty. *Med. Decis. Mak.* **2010**, *30*, 474–483. [CrossRef]
35. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef]
36. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37. [CrossRef]
37. Metropolis, N.; Ulam, S. The monte carlo method. *J. Am. Stat. Assoc.* **1949**, *44*, 335–341. [CrossRef] [PubMed]
38. Mnih, V.; Kavukcuoglu, K.; Silver, D. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602.
39. Hasselt, H. Double Q-learning. In Proceedings of the 23rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–9 December 2010; pp. 2613–2621.
40. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the 13th AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
41. Wang, Z.; Schaul, T.; Hessel, M. Dueling network architectures for deep reinforcement learning. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1995–2003.

42. Schaul, T.; Quan, J.; Antonoglou, I. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
43. Fortunato, M.; Azar, M.G.; Piot, B. Noisy networks for exploration. *arXiv* **2017**, arXiv:1706.10295.