




Article

# Research on Obstacle Avoidance Planning for UUV Based on A3C Algorithm

Hongjian Wang <sup>1</sup>, Wei Gao <sup>1,\*</sup>, Zhao Wang <sup>1</sup>, Kai Zhang <sup>1</sup>, Jingfei Ren <sup>1</sup>, Lihui Deng <sup>1,2</sup> and Shanshan He <sup>1</sup>

<sup>1</sup> College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, China; cctime99@163.com (H.W.); promotion5@foxmail.com (K.Z.); gongchengrendlh@163.com (L.D.)

<sup>2</sup> Tianjin Navigation and Instrument Institute, Tianjin 300130, China

\* Correspondence: gg19961996@foxmail.com

**Abstract:** Deep reinforcement learning is an artificial intelligence technology that combines deep learning and reinforcement learning and has been widely applied in multiple fields. As a type of deep reinforcement learning algorithm, the A3C (Asynchronous Advantage Actor-Critic) algorithm can effectively utilize computer resources and improve training efficiency by synchronously training Actor-Critic in multiple threads. Inspired by the excellent performance of the A3C algorithm, this paper uses the A3C algorithm to solve the UUV (Unmanned Underwater Vehicle) collision avoidance planning problem in unknown environments. This collision avoidance planning algorithm can have the ability to plan in real-time while ensuring a shorter path length, and the output action space can meet the kinematic constraints of UUVs. In response to the problem of UUV collision avoidance planning, this paper designs the state space, action space, and reward function. The simulation results show that the A3C collision avoidance planning algorithm can guide a UUV to avoid obstacles and reach the preset target point. The path planned by this algorithm meets the heading constraints of the UUV, and the planning time is short, which can meet the requirements of real-time planning.

**Keywords:** A3C; UUV; collision avoidance; path planning



**Citation:** Wang, H.; Gao, W.; Wang, Z.; Zhang, K.; Ren, J.; Deng, L.; He, S. Research on Obstacle Avoidance Planning for UUV Based on A3C Algorithm. *J. Mar. Sci. Eng.* **2024**, *12*, 63. <https://doi.org/10.3390/jmse12010063>

Academic Editors: Sébastien Lafond and Sepinoud Azimi

Received: 27 November 2023

Revised: 17 December 2023

Accepted: 19 December 2023

Published: 26 December 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The UUV has strong civilian and military value due to its autonomous navigation ability, which can obtain information on ocean and seabed environments and detect and identify moving targets [1]. The capability of obstacle avoidance planning is an important influencing factor in determining whether UUVs can independently complete preset tasks [2]. Autonomous obstacle avoidance in UUVs refers to the UUV detecting surrounding obstacles and collecting their status information, and planning its paths according to a certain algorithm to avoid obstacles and ultimately reach the preset target point [3]. Deep reinforcement learning algorithms have good decision-making ability and can learn independently, which is used to solve continuous decision-making problems. So Deep reinforcement learning algorithms are very suitable for solving obstacle avoidance planning problems [4].

Deep reinforcement learning is one of the most focused directions in the field of artificial intelligence in recent years, which combines the perceptual ability of deep learning with the decision-making ability of reinforcement learning. It can directly control the behavior of agents through high-dimensional perceptual input learning, providing ideas for solving perceptual decision-making problems in complex systems [5]. In recent years, research on deep reinforcement learning algorithms has mainly focused on the DQN (Deep Q-network) algorithm and its related improvements. Mnih et al., from DeepMind, propose the DQN algorithm [6], through which the agent can learn to play video games by simply obtaining raw pixels from images [7]. Mnih introduces a separate Q-function network and introduces iterative updates into DQN to reduce the correlation between the target value and the current value, proposing an improved version of DQN, namely Nature

DQN [8]. Wang et al. propose the Dueling DQN model, which separates state values and action dominance values, enabling better integration of network architecture and RL algorithms [9]. Hasselt proposes using adaptive normalization learning to address the issue of not changing the scale of the approximation function during the learning process in DQN, which affects the quality of the algorithm in different game applications by cutting the feedback to a predetermined range [10]. Lillicrap proposes the DDPG (Deep Deterministic Policy Gradient) algorithm, which is a reinforcement learning algorithm of the Actor-Critic framework that combines the policy gradient algorithm and DQN algorithm, and can update the policy network in a single step through an off-policy method to achieve the maximum total reward [11]. Mnih et al. propose the A3C algorithm, which introduces the technique of distributed parallel training based on the classic Actor-Critic framework [12]. The parallel threads of the A3C algorithm can explore different actions and effectively improve learning efficiency.

There are many types of traditional collision avoidance planning algorithms, such as the Dynamic Window method [13], Rapidly-Exploring Random Tree [14], and the artificial potential field method [15], but these lack the ability to learn and have poor adaptability to the environment. Heuristic search algorithms commonly used to solve collision avoidance planning problems include the genetic algorithm [16], Particle Swarm Optimization Algorithm [17], and Ant Colony Algorithm [18]. It is difficult to achieve real-time planning for collision avoidance planning based on heuristic search algorithms. The robot needs to replan the next path point for each time step they take, while heuristic search algorithms require a longer time for each planning, making it difficult to conduct real-time planning in the environment. And some heuristic search algorithms are prone to falling into local optima, resulting in the robot being unable to reach the target point [19]. Some scholars use neural network algorithms to solve path planning problems. Changjian Lin et al. propose an improved recurrent neural network for UUV online obstacle avoidance [20]. This algorithm obtains shorter paths, uses less energy through their actuators, and is insensitive to noise. However, the collision avoidance strategy learned by this algorithm is based on expert data and lacks adaptability to the environment. The quality of expert data has a significant impact on collision avoidance effectiveness. Once the expert knowledge is unreasonable, the collision avoidance strategies learned by the neural network are less likely to achieve good results.

The collision avoidance planning policy based on deep reinforcement learning is autonomously learned by the agent, and the training dataset comes from the interaction between the agent and the environment. Reasonable setting of the reward function can achieve good collision avoidance effects. The collision avoidance planning algorithm based on deep reinforcement learning has good real-time performance and can meet the heading constraints of robots. Prashant Bhopale proposed a modified Q-learning obstacle avoidance algorithm for UUVs [21]. This method reduces the chances of a collision with obstacles, but the action space of this method is discrete, which may result in the resulting path not being optimal. Jiawei Wang proposed the APF-DQN collision avoidance planning algorithm [22]. This method introduces direction reward to solve the problem of sparse reward in the DQN algorithm, but the action space of this method is discrete, and the obstacle environment is relatively simple. Janani Bodaragama proposed a collision avoidance planning algorithm based on the Random Network Distillation algorithm [23]. However, the action space in the text is discrete, with an angular velocity of 1.5 rad set, which is very detrimental to the control of actual robots.

As a type of deep reinforcement learning algorithm, the A3C (Asynchronous Advantage Actor-Critic) algorithm can effectively utilize computer resources and improve training efficiency by synchronously training Actor-Critic in multiple threads. The A3C algorithm can effectively solve problems in the field of mobile robots. Yoko Sasaki proposed the A3C Motion Learning algorithm for an autonomous mobile robot [24]. The experimental results indicate that the robot can acquire short-term, simple collision avoidance motion. However, the research object in the article is omnidirectional mobile robots, which reduces the

difficulty of designing collision avoidance algorithms. Z. Zhou proposed a path planning algorithm based on deep reinforcement learning, aiming at the demand for flexibility and real-time performance in an unknown aquatorium [25]. This method discretizes the angular velocity, which is not conducive to USV control and may result in suboptimal paths.

In order to improve the real-time performance of collision avoidance planning algorithms and ensure that the planned path meets the kinematic constraints of UUVs, this article will use the A3C algorithm to solve the collision avoidance planning problem of UUVs in unknown environments. This algorithm adopts a continuous action space and restricts the heading angle of UUVs.

The structure of this article is as follows: Section 2 describes the basic principles of deep reinforcement learning and the kinematic model of the UUV. Section 3 designs a UUV collision avoidance planning algorithm based on A3C. Section 4 conducts simulation experiments in different obstacle environments. The conclusion is given in Section 5.

## 2. Materials

This section mainly introduces the kinematic modeling of the UUV and the basic principles of the A3C algorithm.

### 2.1. UUV Model

This article takes the UUV as the research object, with a total length of 4.5 m, a total width of 1.1 m, and a height of 0.6 m. In order to study the motion law of the UUV and determine its underwater motion position information, it is necessary to first establish a suitable coordinate system and analyze the force and maneuverability of the UUV under this coordinate system. Usually, the two basic coordinate systems for UUV motion modeling are fixed coordinate systems and motion coordinate systems, as shown in Figure 1.

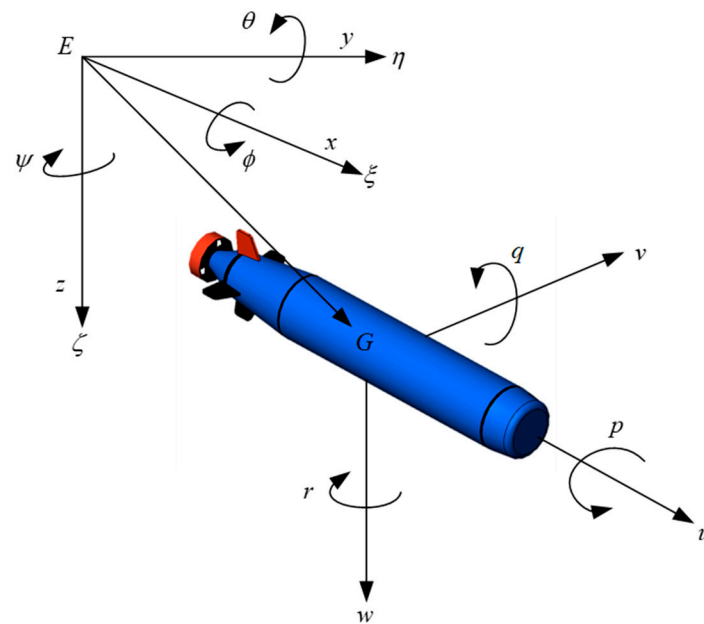


Figure 1. The UUV coordinate system.

For the convenience of research, this article simplifies the six-degree-of-freedom motion model of UUVs, only considering the motion of the underactuated UUV in the longitudinal, transverse, and bow directions. Therefore, the assumptions are made:

- (1) Neglecting the influence of third-order and higher-order hydrodynamic coefficients on the UUV;
- (2) Neglecting the influence of roll, pitch, and heave movements of the UUV on horizontal motion.

The movement of the UUV is mainly carried out in three degrees of freedom: longitudinal, transverse, and bow. This article defines the vector  $\eta = [x \ y \ \psi]^T$  to represent the generalized position information of the UUV in a fixed coordinate system, The vector  $v = [u \ v \ r]^T$  represents the generalized velocity of the UUV, where  $u$  is the longitudinal velocity of the UUV,  $v$  is the lateral velocity of the UUV, and  $r$  is the raw rate. The schematic diagram of the horizontal movement of the UUV is shown in Figure 2, where  $v_t = \sqrt{u^2 + v^2}$  is the UUV's synthesis speed and  $\beta = \arctan(v/u)$  is the drift angle.

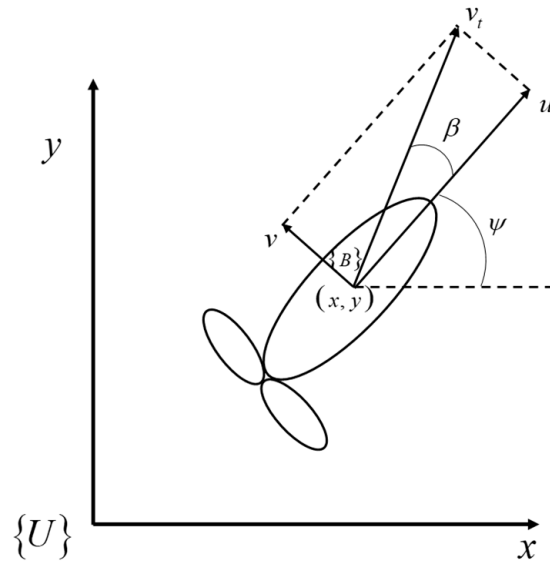


Figure 2. Schematic diagram of horizontal movement of UUV.

In the absence of interference, the kinematic model of the UUV in the horizontal plane is as follows:

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

The corresponding dynamic equations are as follows:

$$\begin{cases} \dot{u} = \frac{F}{m_u} - \frac{d_u}{m_u} \\ \dot{v} = -\frac{m_{ur}ur}{m_v} - \frac{d_v}{m_v} \\ \dot{r} = \frac{T}{m_r} - \frac{d_r}{m_r} \end{cases} \quad (2)$$

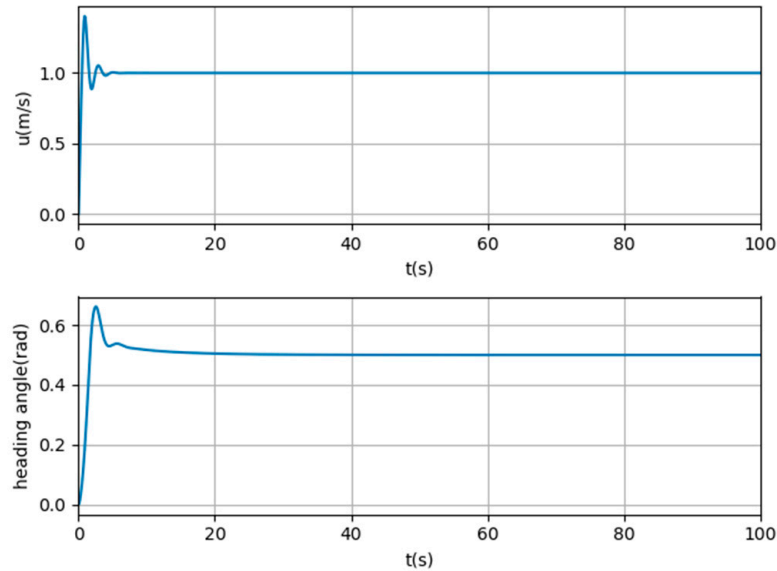
where  $m_u = m - X_{\dot{u}}$ ,  $m_v = m - Y_{\dot{v}}$ ,  $m_r = I_z - N_{\dot{r}}$ ,  $m_{ur} = m - Y_r$ ,  $m_r = I_z - N_{\dot{r}}$ ,  $m_{ur} = m - Y_r$ ,  $d_u = -X_{uu}u^2 - X_{vv}v^2$ ,  $d_v = -Y_{uv}uv - Y_{v|v|}v|v|$ ,  $d_r = -N_{vv}v - N_{v|v|}v|v| - N_rur$ .  $m$  is the mass of the UUV, with a value of 2234.5 kg;  $I_z$  is the mass moment of inertia with regard to the Z axis of the body-fixed frame;  $X_{\{\cdot\}}$ ,  $Y_{\{\cdot\}}$  and  $N_{\{\cdot\}}$  are the hydrodynamic coefficients; and  $F$  and  $T$  are the longitudinal thrust and turning moment of the UUV, respectively.

The INFANTE AUV model is the research object of this article [26]. The mass of the UUV is 2234.5 kg and the relevant hydrodynamic coefficients are shown in the Table 1.

When controlling the UUV, the traditional PID control algorithm was used, with heading angle  $\psi$  and longitudinal velocity  $u$  as the controlled variables. Taking the expected longitudinal speed  $u_{com} = 1$  m/s and the expected heading angle  $\psi_{com} = 0.5$  rad as an example, the PID control effect is shown in the Figure 3.

**Table 1.** Hydrodynamic coefficient.

hydrodynamic parameter	$X_{ii}/Kg$	$Y_{\dot{v}}/Kg$	$N_r/N \cdot m^2$	$X_{uu}/Kg \cdot m^{-1}$	$X_{vv}/Kg \cdot m^{-1}$	$Y_v/Kg \cdot m^{-1}$
parameter value	-142	-1715	-1350	-35.4	-128.4	-346
hydrodynamic parameter	$Y_{v v }/Kg \cdot m^{-1}$	$Y_r/Kg$	$N_r/Kg \cdot m$	$N_{v v }/Kg$	$I_z/N \cdot m^2$	$N_v/Kg$
parameter value	-667	435	-1427	443	2000	-686



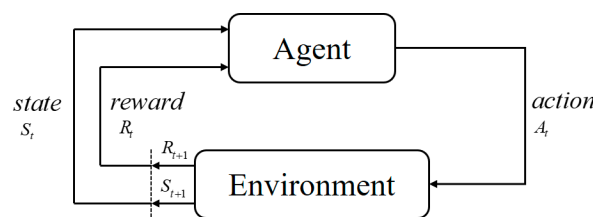
**Figure 3.** PID control effect diagram.

As shown in the above figure, the PID controller can adjust the controlled quantity to the given value within 10 s.

2.2. A3C

2.2.1. Reinforcement Learning

The principle of reinforcement learning is shown in Figure 4. When an agent completes a task, it first generates action  $A_t$ , which will have an impact on the environment, causing the agent to reach a new state  $S_t$ . At this point, the agent will receive an immediate reward  $R_t$  from the environment. If repeated like this, the continuous interaction between the intelligent agent and the environment will generate a large amount of data, including status, rewards, and actions. The reinforcement learning algorithm uses the generated data to modify its own action strategy. The agent interacts with the environment again based on the new strategy, generating a new round of data, and improving its own strategy again using the new round of data. After multiple iterations, the strategy gradually converges, and the agent can learn the optimal strategy to complete the corresponding task.



**Figure 4.** The schematic diagram of reinforcement learning.

In a broad sense, reinforcement learning can be considered a sequential decision problem, with the goal of finding the decision sequence that maximizes the expected cumulative return. Sequential decision problems have a lot of content, and, specifically, they should be incorporated into the framework of the Markov Decision Process (MDP) [27].

The goal of reinforcement learning is to find a strategy that can achieve the maximum expected cumulative return. The strategy can be understood as a mapping relationship from state  $S_t$  to action  $A_t$ , which means that the probability of action in the state can be determined based on the strategy. The strategy can usually be represented by the symbol  $\pi$ .

$$\pi(a|s) = P[A_t = a|S_t = s] \tag{3}$$

The overall process of the MDP is summarized as follows: set the initial state of the intelligent agent to  $s_0$ ; obtain the current action  $a_0$  according to the designed strategy  $\pi(a|s)$ ; then obtain the state  $s_1$  at the next moment based on the probability of state transition  $P(s_{t+1}|s_t, a_t)$ ; at the same time, the intelligent experience receives the reward feedback  $r_1$  from environment. The entire process continues to loop until the final state  $s_T$ , ultimately resulting in a trajectory sequence  $\tau = (s_0, a_0, s_1, \dots, s_T)$ . After obtaining the complete trajectory sequence, the cumulative return can be calculated, which is  $R = \sum_{t=0}^{T-1} \gamma^t r_t$ . The expected cumulative return is shown in the following equation:

$$E_R = E[\sum_{t=0}^{T-1} \gamma^t r_t] \tag{4}$$

The strategy that the agent ultimately seeks is the one that maximizes the expected cumulative return:

$$\pi^* = \max_{\pi} E^{\pi}[\sum_{t=0}^{T-1} \gamma^t r_t] \tag{5}$$

### 2.2.2. A3C

Deep learning belongs to the connectionist school, which mainly focuses on intelligent perception and optimization objectives are mostly continuous functions. Reinforcement learning is an intelligent decision-making approach that belongs to the behaviorist school of thought, with optimization objectives being discrete functions. Although deep learning and reinforcement learning both belong to machine learning, reinforcement learning has certain advantages compared to supervised and unsupervised learning in deep learning [28]. The deep reinforcement learning method combines the powerful perception ability of deep learning with the decision-making ability of reinforcement learning. This algorithm is more intelligent and can solve more complex tasks, making it a more human-like artificial intelligence method [29].

As a type of deep reinforcement learning, the A3C algorithm can effectively utilize computer resources and improve training efficiency by synchronously training Actor-Critic in multiple threads. The A3C algorithm is a reinforcement learning algorithm based on the Actor-Critic framework [30]. The Actor-Critic framework can be divided into two different systems, Actor and Critic, which can be replaced by different neural networks [31]. The Critic can be used to learn the reward and punishment mechanism. After learning, the Actor uses the strategy function to generate actions according to the state. Then, the parameter values of the policy function are updated according to the knowledge learned by the Critic, and the Critic network updates its own parameters while evaluating the Actor. The schematic diagram of the Actor-Critic framework is shown in Figure 5.

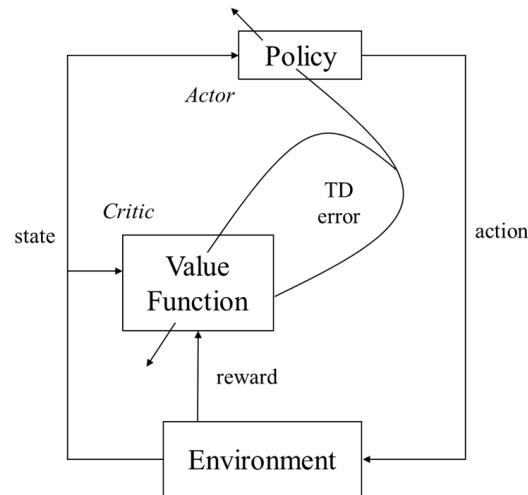


Figure 5. Actor-Critic framework.

The Actor also can be called strategy  $\pi$ . The Actor can be parameterized directly, represented by  $\theta$  here. Then the action  $a$  of the agent can be obtained by sampling the strategy  $\pi$ , which can be expressed in the following form:

$$a \sim \pi(s, a, \theta) \tag{6}$$

In the Actor-Critic framework, in order to evaluate the performance of the agent, the concept of Q value in reinforcement learning can be used. At this point, the role of the Critic is to fit the Q value and evaluate the score of the actor’s behavior, which can be approximated by using a neural network. The gradient expression of the Actor is as follows:

$$\nabla \bar{R}_\theta = \frac{1}{m} \sum_{n=1}^m \sum_{t=1}^{T_n} Q^{\pi_\theta}(s_t^n, a_t^n) \nabla \log p_\theta(a_t^n | s_t^n) \tag{7}$$

In order to update the strategy by the Actors, it is necessary for the Critic to fit the Q value. The updated method of the Critic is achieved through the Temporal Difference (TD) error, and its loss function is as follows:

$$Loss = \frac{1}{m} \sum_{n=1}^m \sum_{t=1}^{T_n} (r_t^n + \max_{a_{t+1}^\pi} Q^{\pi_\theta}(s_{t+1}^n, a_{t+1}^\pi) - Q^{\pi_\theta}(s_t^n, a_t^n))^2 \tag{8}$$

The A3C is a new lightweight parallel algorithm. In order to break the correlation of data, an asynchronous method was adopted, which can ensure that the data is not generated synchronously. The A3C [32] framework sets up a Global Network (Global Neural Network Model) at the top level of the algorithm. The Global Neural Network Model is an Actor-Critic structure, mainly used to collect the experience learned by each sub Actor-Critic model and update its parameters for the sub Actor-Critic. The sub Actor-Critic model can be seen as a local model under multi-threading, where each sub thread corresponds to a set of algorithm models and training environments. Each sub thread does not interfere with each other, and in order to make the model converge faster, some method can be used to make the training environment of each sub thread different, so that the sub threads can interact with different environments as much as possible, learn different experiences, and accelerate the learning of the global model. After interacting with the environment, the sub thread updates the learned experience to the global neural network. The structure diagram of the A3C algorithm is shown in Figure 6.

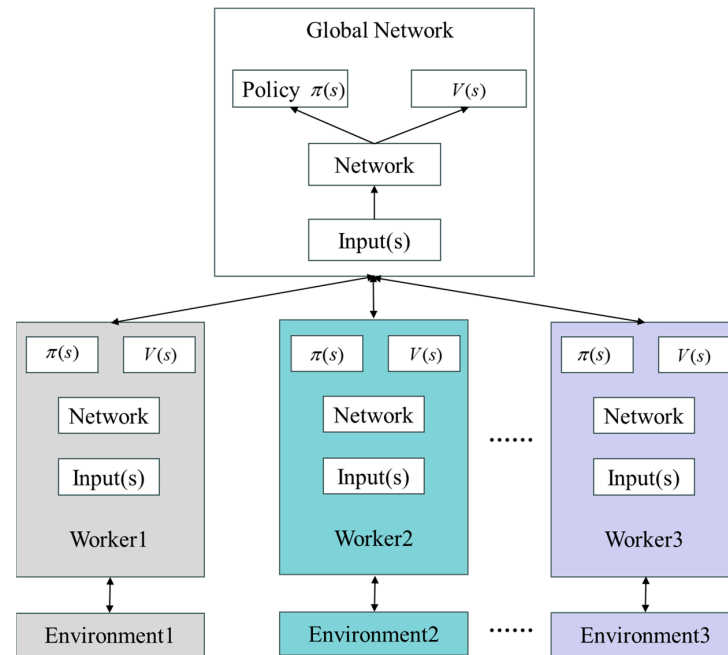


Figure 6. A3C algorithm structure diagram.

The A3C algorithm process is shown in Algorithm 1:

**Algorithm 1** A3C

```

// Assume global shared parameter vectors  $\theta$  and  $\theta_v$  and global shared counter  $T = 0$ 
// Assume thread-specific parameter vectors  $\theta'$  and  $\theta'_v$ 
Initialize thread step counter  $t \leftarrow 1$ 
repeat
  Reset gradients:  $d\theta \leftarrow 0$  and  $d\theta_v \leftarrow 0$ .
  Synchronize thread-specific parameters  $\theta' = \theta$  and  $\theta'_v = \theta_v$ 
   $t_{start} = t$ 
  Get state  $s_t$ 
  repeat
    Perform  $a_t$  according to policy  $\pi(a_t|s_t; \theta')$ 
    Receive reward  $r_t$  and new state  $s_{t+1}$ 
     $t \leftarrow t + 1$ 
     $T \leftarrow T + 1$ 
  until terminal  $s_t$  or  $t - t_{start} == t_{max}$ 
   $R = \begin{cases} 0 & \text{for terminal } s_t \\ V(s_t, \theta'_v) & \text{for non-terminal } s_t \end{cases}$ 
  for  $i \in \{t - 1, \dots, t_{start}\}$  do
     $R \leftarrow r_i + \gamma R$ 
    Accumulate gradients wrt  $\theta'$ :  $d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v))$ 
    Accumulate gradients wrt  $\theta'_v$ :  $d\theta_v \leftarrow d\theta_v + \partial(R - V(s_i; \theta'_v))^2 / \partial \theta'_v$ 
  end for
  Perform asynchronous update of  $\theta$  using  $d\theta$  and of  $\theta_v$  using  $d\theta_v$ 
until  $T > T_{max}$ 

```

**3. The A3C Collision Avoidance Planning Algorithm**

In order to achieve autonomous obstacle avoidance for the UUV, this paper uses the A3C reinforcement learning algorithm. This section introduces the selection of state space, the design of reward function, the construction of the environment, and the design of action space.



### 3.1. State Space

The state information includes observation information obtained from the environment and target point information, which has a significant impact on the final effect. The UUV carries the SeaBat 8125-H forward-looking sonar model from RESON company, which can detect a horizontal 120° fan-shaped area with a vertical opening angle of 17°. The state information selected in this article includes the distance between the UUV and the obstacle, the angle at which the robot's heading deviates from the obstacle, the distance between the UUV and the target point, and the angle at which the robot's heading deviates from the target. Because the obstacle data returned by the forward-looking sonar have a higher dimension, and the A3C algorithm needs to spend more time learning the data. In reactive collision avoidance planning algorithms, the high-dimensional obstacle data are not necessary, so obstacle data are simplified through partitioning. The 120° fan-shaped area is divided at intervals of 10°. The expressions of distance  $\rho_t$  between the UUV and target point, and the angle  $\theta_t$  at which the robot's heading deviates from the target are as follows.

$$\rho_t = \sqrt{(x_r - x_g)^2 + (y_r - y_g)^2} \tag{9}$$

$$\theta_t = \theta_r - \arctan \frac{y_g - y_r}{x_g - x_r} \tag{10}$$

where  $(x_r, y_r)$  are the coordinates of the UUV in the global coordinate system, and  $(x_g, y_g)$  are the coordinates of the target point in the global coordinate system.

### 3.2. Action Space

The action strategy selection used in this section is designed as a continuous action space. If the action space includes the linear velocity and angular velocity, which are both continuous, the deep reinforcement learning algorithm is difficult to converge. Therefore, the A3C collision avoidance planning algorithm keeps the linear velocity unchanged and only changes the heading of the UUV, that is, the action space includes the angular velocity item. Considering the actual situation, the angular velocity of the UUV is limited between  $-5^\circ/s$  and  $5^\circ/s$ .

### 3.3. Reward Function

The design of the reward function is of utmost importance in reinforcement learning, and the quality of reinforcement learning training results is closely related to the reward function. The purpose of reinforcement learning is to obtain the maximum long-term cumulative reward. The reward function evaluates each action based on its interaction with the environment, so the quality of the reward function affects the quality of the final strategy.

The reward function settings in this section are as follows:

$$R = R_r + R_d R_c + R_s \tag{11}$$

$R_r$  represents the reward received by the agent when it reaches the final target point;  $R_c$  represents the reward received by the agent when colliding with obstacles in the environment;  $R_d = k_{dis} \cdot dis_{obs}$ , and  $dis_{obs}$  indicate the distance between the UUV and the nearest obstacle, and  $k_{dis}$  is the obstacle reward coefficient. The main purpose of the  $R_d$  is to successfully avoid obstacles.  $R_s = k_{step} \cdot step\_count$  represents the penalty of the reward function on the number of steps and can ensure that the number of steps for the UUV to reach the target point is minimized as much as possible,  $k_{step}$  is the penalty coefficient for the number of steps, and  $step\_count$  is the current cumulative number of steps.

## 4. Experiments and Results

The experiments are conducted to verify the A3C collision avoidance planning algorithm. The experiments take the UUV model in Section 2 as the research object. The

obstacle environments include dense irregular obstacle environments and narrow passage obstacle environments. The performance of the algorithms is evaluated from the aspects of path length, average solution time, and whether the target point has been reached.

In order to demonstrate the advantages of the algorithm in this article, this section uses the artificial potential field method and genetic algorithm as comparisons. In the experiment, the longitudinal velocity  $u$  of the UUV remains fixed at 1 m/s, and only the heading angle of the UUV is changed to achieve obstacle avoidance. The experiments use the PID controller to control the heading angle and the longitudinal speed of the UUV.

The simulation results are as follows:

Figure 7 shows the reward function curve, from which it can be seen that after a certain number of iterations, the reward function converges, proving the feasibility of the algorithm proposed in this article. In the Figure 8, it can be seen that the A3C collision avoidance planning algorithm can plan a safe and collision-free path in a dense and irregular obstacle environment, while the APF algorithm cannot adapt to such obstacle environments due to its lack of learning ability to the environment. In Figure 9, it can be seen that three algorithms can successfully reach the target point in narrow channel obstacle environments. However, due to the addition of turning restrictions, the A3C algorithm has a relatively smooth change in heading angle and trajectory, while the APF algorithm and GA algorithm do not have turning restrictions, resulting in a significant difference between the expected heading angle obtained and the current heading angle. On the one hand, it puts higher requirements on the UUV controller, and on the other hand, it will cause the UUV to perform a significant turning action, which puts higher requirements on the performance of the UUV. Tables 2 and 3 show the performance comparison of three algorithms. From the perspective of path length, the three algorithms are equally matched. From the perspective of average execution time, the execution time of the A3C algorithm and APF algorithm is similar, while the execution time of the GA algorithm is longer.

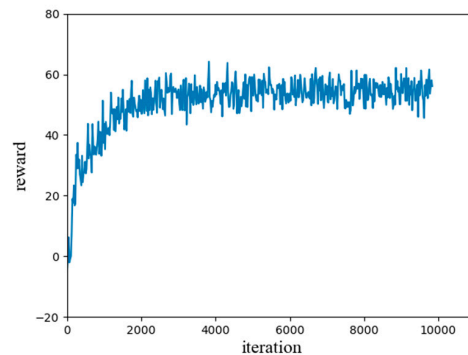


Figure 7. Reward function curve.

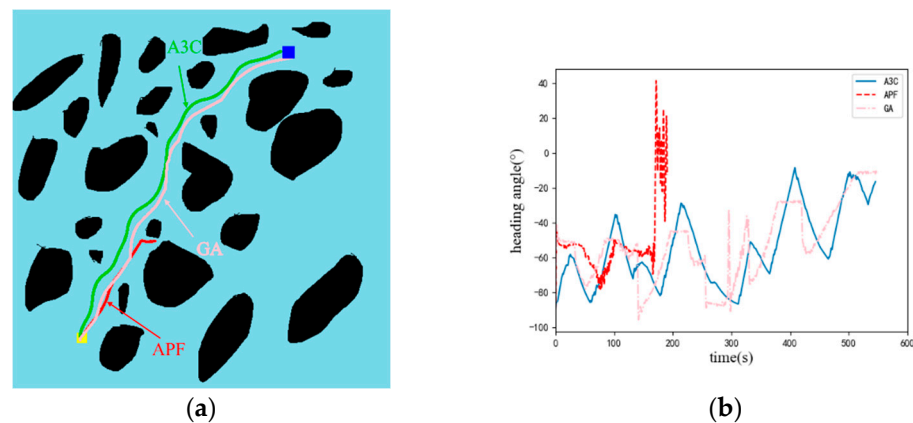
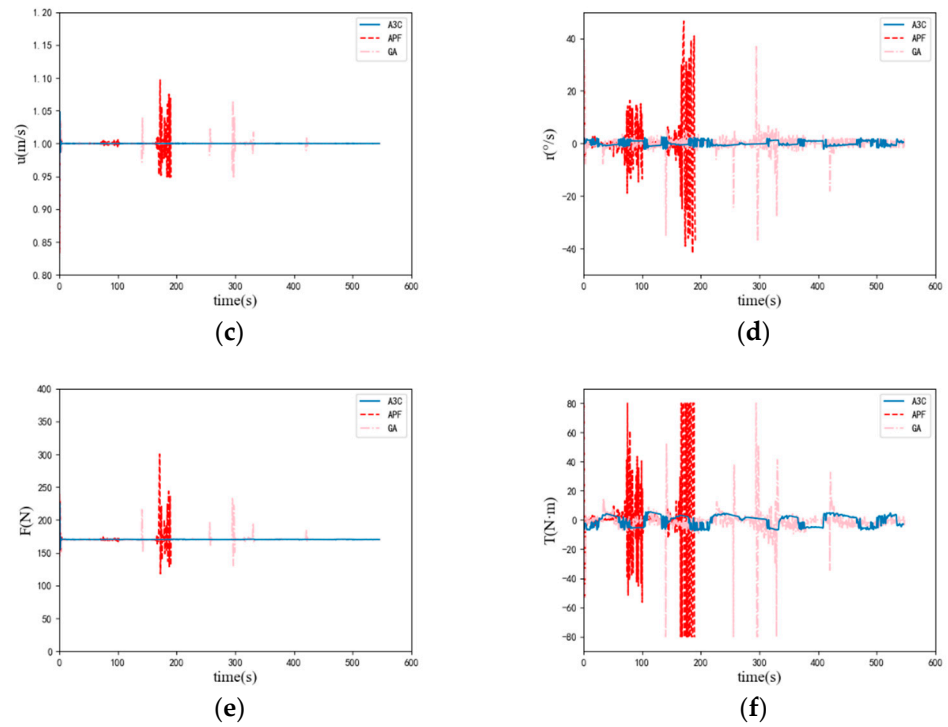
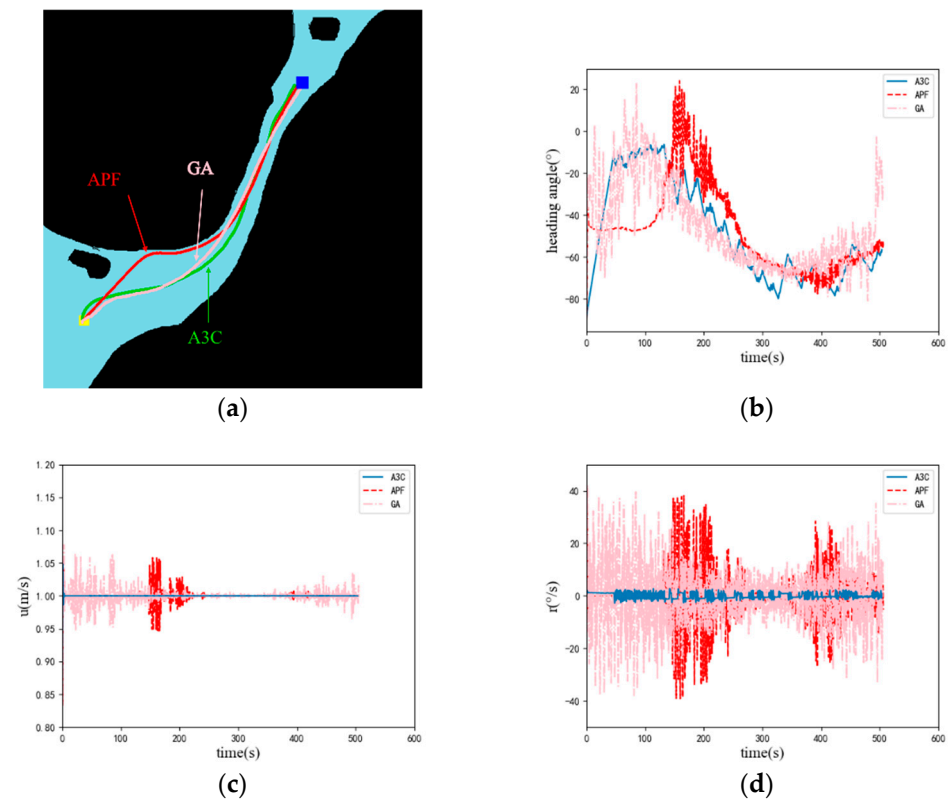


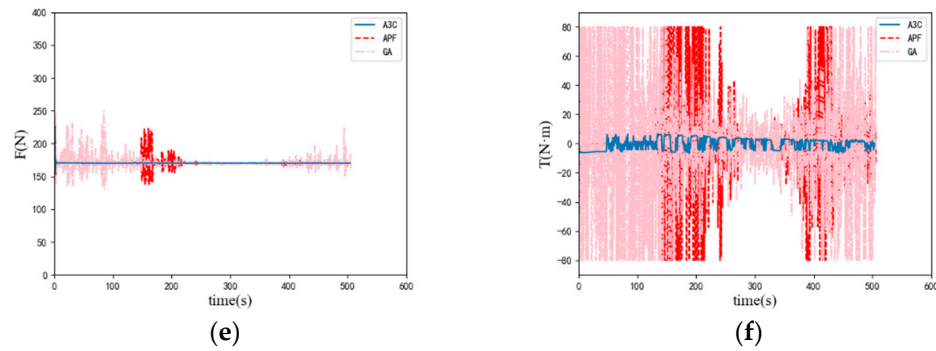
Figure 8. Cont.



**Figure 8.** The experiment results in the dense irregular obstacle environment. (a) comparison of the collision avoidance trajectory; (b) comparison curve of the heading angle  $\psi$ ; (c) comparison curve of the longitudinal speed  $u$ ; (d) comparison curve of the yaw rate  $r$ ; (e) comparison curve of the longitudinal thrust  $F$ ; and (f) comparison curve of the turning moment  $T$ .



**Figure 9.** Cont.



**Figure 9.** The simulation results in the narrow passage obstacle environment. (a) comparison of the collision avoidance trajectory; (b) comparison curve of the heading angle  $\psi$ ; (c) comparison curve of the longitudinal speed  $u$ ; (d) comparison curve of the yaw rate  $r$ ; (e) comparison curve of the longitudinal thrust  $F$ ; and (f) comparison curve of the turning moment  $T$ .

**Table 2.** The performance in the dense irregular obstacle environment.

Algorithm	Path Length/m	Execution Time/ms	Whether the Target Point Has Been Reached
A3C	545	1.2	✓
APF	/	1.1	×
GA	546	311	✓

**Table 3.** The performance in the narrow passage obstacle environment.

Algorithm	Path Length/m	Execution Time/ms	Whether the Target Point Has Been Reached
A3C	503	1.3	✓
APF	506	1.3	✓
GA	505	310	✓

### 5. Conclusions

This article proposes the A3C local collision avoidance planning algorithm for UUVs, which maintains the fixed longitudinal velocity of the UUV and only changes its heading. For the problem of collision avoidance planning, this article designs state space, action space, and reward function. In order to smooth the collision avoidance trajectory of the UUV and reduce the difficulty of controlling the UUV, this article applies yaw restrictions to the final output of the algorithm, controlling the angular velocity between  $-5^\circ/s$  and  $5^\circ/s$ . In order to verify the effectiveness of the algorithm proposed in this article, simulation experiments were conducted and compared with the artificial potential field method and genetic algorithm. The simulation results show that the A3C collision avoidance planning algorithm can plan smooth collision avoidance paths in dense and irregular obstacle environments, as well as narrow channel obstacle environments. The execution time of the A3C collision avoidance planning algorithm is in milliseconds, and the planned yaw rate is small, which can ensure the smoothness of the trajectory.

**Author Contributions:** H.W.: conceptualization, supervision, funding, and acquisition. W.G.: methodology, investigation, software, and writing—original and draft. Z.W.: writing—original and draft, and software. K.Z.: formal analysis and data curation. J.R.: conceptualization, and writing—reviewing and editing. L.D.: methodology, software, and investigation. S.H. validation, investigation, and writing—original and draft. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research work is supported by National Science and Technology Innovation Special Zone Project (21-163-05-ZT-002-005-03), the National Key Laboratory of Underwater Robot Technology Fund (No. JCKYS2022SXJQR-09), and a special program to guide high-level scientific research (No. 3072022QBZ0403).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are available on request from the corresponding author.

**Acknowledgments:** The authors would like to thank the anonymous reviewers and the handling editors for their constructive comments that greatly improved this article from its original form.

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

## References

- Zhu, D.; Yang, S.X. Bio-Inspired Neural Network-Based Optimal Path Planning for UUVs Under the Effect of Ocean Currents. *IEEE Trans. Intell. Veh.* **2021**, *7*, 231–239. [\[CrossRef\]](#)
- Yue, Y.; Hao, W.; Guanjie, H.; Yao, Y. UUV Target Tracking Path Planning Algorithm Based on Deep Reinforcement Learning. In Proceedings of the 2023 8th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS), Xi'an, China, 7–9 July 2023; pp. 65–71.
- Li, D.; Wang, P.; Du, L. Path Planning Technologies for Autonomous Underwater Vehicles-A Review. *IEEE Access* **2019**, *7*, 9745–9768. [\[CrossRef\]](#)
- Cai, Y.; Zhang, E.; Qi, Y.; Lu, L. A Review of Research on the Application of Deep Reinforcement Learning in Unmanned Aerial Vehicle Resource Allocation and Trajectory Planning. In Proceedings of the 2022 4th International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI), Shanghai, China, 28–30 October 2022; pp. 238–241.
- Zhu, K.; Zhang, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **2021**, *26*, 674–691. [\[CrossRef\]](#)
- Lample, G.; Chaplot, D.S. Playing FPS Games with Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1609.05521. [\[CrossRef\]](#)
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *Comput. Sci.* **2013**, 201–220. [\[CrossRef\]](#)
- Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [\[CrossRef\]](#) [\[PubMed\]](#)
- Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. *Proc. Mach. Learn. Res.* **2015**, *48*, 1995–2003.
- Hasselt, H.V.; Guez, A.; Hessel, M.; Mnih, V.; Silver, D. Learning functions across many orders of magnitudes. *arXiv* **2016**, arXiv:1602.07714.
- Lillicrap, T.; Hunt, J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
- Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.P.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous Methods for Deep Reinforcement Learning. *arXiv* **2016**, arXiv:1602.01783.
- Dobrevski, M.; Skočaj, D. Adaptive Dynamic Window Approach for Local Navigation. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 24 October 2020–24 January 2021; pp. 6930–6936.
- Rodriguez, S.; Tang, X.; Lien, J.-M.; Amato, N.M. An Obstacle-based Rapidly-exploring Random Tree. In Proceedings of the 2006 IEEE International Conference on Robotics and Automation, Orlando, FL, USA, 15–19 May 2006; pp. 895–900.
- Igarashi, H.; Kakikura, M. Path and Posture Planning for Walking Robots by Artificial Potential Field Method. In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004.
- Hu, Y.; Yang, S.X. A Knowledge Based Genetic Algorithm for Path Planning of a Mobile Robot. In Proceedings of the IEEE International Conference on Robotics and Automation, New Orleans, LA, USA, 26 April–1 May 2004.
- Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the 1995 IEEE International Conference, Perth, WA, Australia, 27 November–1 December 1995; pp. 1942–1948.
- Li, S.; Su, W.; Huang, R.; Zhang, S. Mobile Robot Navigation Algorithm Based on Ant Colony Algorithm with A\* Heuristic Method. In Proceedings of the 2020 4th International Conference on Robotics and Automation Sciences, Wuhan, China, 12–14 June 2020; pp. 28–33.
- Tang, B.; Zhanxia, Z.; Luo, J. A Convergence-guaranteed Particle Swarm Optimization Method for Mobile Robot Global Path Planning. *Assem. Autom.* **2017**, *37*, 114–129. [\[CrossRef\]](#)
- Lin, C.; Wang, H.; Yuan, J.; Yu, D.; Li, C. An Improved Recurrent Neural Network for Unmanned Underwater Vehicle Online Obstacle Avoidance. *Ocean Eng.* **2019**, *189*, 106327. [\[CrossRef\]](#)

21. Bhopale, P.; Kazi, F.; Singh, N. Reinforcement Learning Based Obstacle Avoidance for Autonomous Underwater Vehicle. *J. Mar. Sci. Appl.* **2019**, *18*, 228–238. [[CrossRef](#)]
22. Wang, J.; Lei, G.; Zhang, J. Study of UAV Path Planning Problem Based on DQN and Artificial Potential Field Method. In Proceedings of the 2023 4th International Symposium on Computer Engineering and Intelligent Communications, Nanjing, China, 18–20 August 2023; pp. 175–182.
23. Bodaragama, J.; Rajapaksha, U.U.S. Path Planning for Moving Robots in an Unknown Dynamic Area Using RND-Based Deep Reinforcement Learning. In Proceedings of the 2023 3rd International Conference on Advanced Research in Computing (ICARC), Belihuloya, Sri Lanka, 23–24 February 2023; pp. 13–18.
24. Sasaki, Y.; Matsuo, S.; Kanezaki, A.; Takemura, H. A3C Based Motion Learning for an Autonomous Mobile Robot in Crowds. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019.
25. Zhou, Z.; Zheng, Y.; Liu, K.; He, X.; Qu, C. A Real-time Algorithm for USV Navigation Based on Deep Reinforcement Learning. In Proceedings of the 2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP), Chongqing, China, 11–13 December 2019; pp. 1–4.
26. Lapiere, L.; Soetanto, D. Nonlinear path-following control of an AUV. *Ocean Eng.* **2007**, *34*, 1734–1744. [[CrossRef](#)]
27. White III, C.C.; White, D.J. Markov Decision Process. *Eur. J. Oper. Res.* **1989**, *39*, 1–16. [[CrossRef](#)]
28. Siraskar, R.; Kumar, S.; Patil, S.; Bongale, A.; Kotecha, K. Reinforcement learning for predictive maintenance: A systematic technical review. *Artif. Intell. Rev.* **2023**, *56*, 12885–12947. [[CrossRef](#)]
29. Yu, K.; Jin, K.; Deng, X. Review of Deep Reinforcement Learning. In Proceedings of the 2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), Chongqing, China, 16–18 December 2022; pp. 41–48.
30. Peters, J.; Schaal, S. Natural actor-critic. *Neurocomputing* **2008**, *71*, 1180–1190. [[CrossRef](#)]
31. Bhatnagar, S.; Sutton, R.S.; Ghavamzadeh, M.; Lee, M. Natural actor-critic algorithms. *Automatica* **2009**, *45*, 2471–2482. [[CrossRef](#)]
32. Chen, T.; Liu, J.Q.; Li, H.; Wang, S.R.; Niu, W.J.; Tong, E.D.; Chang, L.; Chen, Q.A.; Li, G. Robustness Assessment of Asynchronous Advantage Actor-Critic Based on Dynamic Skewness and Sparseness Computation: A Parallel Computing View. *J. Comput. Sci. Technol.* **2021**, *36*, 1002–1021. [[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.