2021 2nd International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT) | 978-1-6654-1296-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/AINIT54228.2021.00024

Research on PPO algorithm in solving AUV path planning problems

Zheng Wang
School of Electrical Engineering
Naval University of Engineering
Wuhan, China
marchy618@163.com

Zhiyuan Hu
School of Electrical Engineering
Naval University of Engineering
Wuhan, China
757308793@qq.com

Yang Yang
School of Electrical Engineering
Naval University of Engineering
Wuhan, China
yyang319@126.com

Yang Yin*
School of Electrical Engineering
Naval University of Engineering
Wuhan, China
reeyan@163.com

***Abstract*—In the face of a complex three-dimensional environment, the computational complexity of the traditional path planning algorithms is extremely increased. But the performance of path planning results is relatively poor. Deep reinforcement learning (DRL) algorithms don't rely on accurate environmental models, and its overall efficiency is much higher than traditional algorithms. Aiming at the problem of AUV path planning in a three-dimensional environment, proximal policy optimization algorithm (PPO) is selected. The PPO algorithm with improved network structure is used to train the model based on the established AUV model and gym environment. Through simulation experiments, the accuracy and effectiveness of the algorithm is verified.**

Keywords- deep reinforcement learning; path planning; proximal policy optimization

I. INTRODUCTION

Automatic underwater vehicle (AUV) is a kind of self-propelled vehicle for performing underwater tasks. AUV equipped with its own energy, autonomous navigation and control system, has the characteristic of small size, high degree of intelligence, good stealth performance, and low operational risk [1]. Nowadays, with the continuous development of the ocean, AUV has been widely used. As one of the core technologies of AUV, path planning technology is an important support for its safety and reliability.

Traditional path planning algorithms include A* algorithm, artificial potential field method, ant colony algorithm, genetic algorithm and so on [2]. Document [3] optimized the ant colony algorithm's pheromone representation, pheromone update rules, and heuristic function. Path planning of the ant colony algorithm in a three-dimensional environment was realized. Document [4] used the idea of genetic algorithm to propose an algorithm that simplifies the two-dimensional code of the AUV path into one-dimensional code. The computer simulation results of AUV dynamic collision avoidance in two-dimensional and three-dimensional space show that the obtained path curve is smooth and the algorithm's convergence speed was greatly improved. However, the actual three-dimensional marine environment is relatively complex. The marine environment is full of natural phenomena such as ocean currents, climatology, internal waves and other nonlinear changes. The generalization ability of traditional path planning algorithms is poor, and it is difficult to achieve better path planning effects. With the development of artificial intelligence technology, new ideas and methods are provided for AUV path planning.

Reinforcement learning uses the interaction between the agent and the environment to maximize the reward value and can be used for path planning tasks. Deep Reinforcement Learning (DRL), which combines reinforcement learning with deep neural networks, doesn't rely on accurate environmental models. DRL has strong learning capabilities, and is suitable for solving problems with complex environmental models [5]. Document [6] proposed an end-to-end deep Q network for the problem of robot path planning. Through feature extraction and feature matching of the incoming images, corresponding action strategies are generated to guide the robot to dynamically avoid obstacles. Document [7] applied the DRL algorithm to the research on the depth control of AUV. Compared with the traditional PID algorithm, DRL can improve the accuracy of depth control.

In this paper, based on OpenAI gym platform a simulation environment to solve path planning problem is established. PPO algorithm is chosen and optimized for model training and prediction. The effectiveness of the algorithm is verified by simulation.

II. AUV MODEL ESTABLISHMENT

A. Kinematics and dynamics model

AUV can perform 3 linear motions along the axis and 3 rotational motions around the axis, including a 6-degree-of-freedom motion space. As shown in Fig. 1, the fixed coordinate system E-$\xi\eta\zeta$ and the motion coordinate system O-xyz are usually used to indicate the movement status of the AUV.

978-1-6654-1296-4/21/$31.00 ©2021 IEEE
DOI 10.1109/AINIT54228.2021.00024

The fixed coordinate system represents the position coordinates and posture of the AUV. The motion coordinate system represents the translation and rotation speed of the AUV. The parameters related to the two coordinate systems are shown in the Table 1 and Table 2.

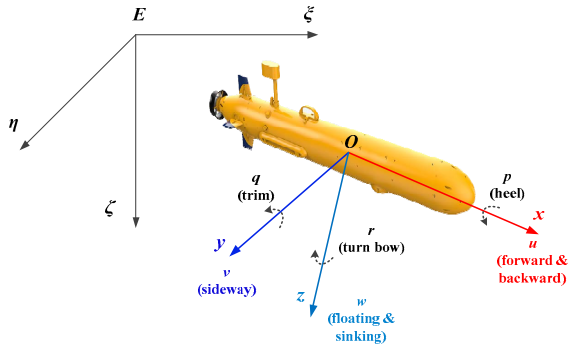

Figure 1. Schematic diagram of AUV coordinate system

TABLE I. RELATED PARAMETERS IN THE FIXED COORDINATE SYSTEM

Axis	ξ axis	η axis	ζ axis
Position	x	y	z
Posture	φ (heel angle)	θ (trim angle)	ψ (heading angle)

TABLE II. RELATED PARAMETERS IN THE MOTION COORDINATE SYSTEM

Axis	x axis	y axis	z axis
Line speed	u	v	w
Posture	p	q	r
external force	X	Y	Z
external moment	K	M	N

Respectively record the above parameters as vectors: $\vec{\eta}_1=[x,y,z]^T$, $\vec{\eta}_2=[\phi,\theta,\psi]^T$, $\vec{v}_1=[u,v,w]^T$, $\vec{v}_2=[p,q,r]^T$. On this basis, let $\vec{\eta}=[\vec{\eta}_1,\vec{\eta}_2]^T$, $\vec{v}=[\vec{v}_1,\vec{v}_2]^T$. The kinematics and dynamics equations of AUV can be expressed as [8]:

$$\dot{\vec{\eta}}=\begin{bmatrix}\mathbf{J_1}(\vec{\eta}_2) & \mathbf{O}_{3\times3}\\ \mathbf{O}_{3\times3} & \mathbf{J_2}(\vec{\eta}_2)\end{bmatrix}\vec{v} \tag{1}$$

$$M\dot{\vec{v}}+C(\vec{v})\vec{v}+D(\vec{v})\vec{v}=\tau \tag{2}$$

where, $\mathbf{J_1}(\vec{\eta}_2)$ and $\mathbf{J}_2(\vec{\eta}_2)$ are the conversion matrix, can be calculated by Equation (3) and (4). M represents the inertial matrix, $C(\vec{v})$ is the coriolis force matrix caused by inertial hydrodynamic forces, $D(\vec{v})$ is the damping matrix, and τ is the sum of the external forces on the AUV.

$$\mathbf{J_1}(\vec{\eta}_2)=\begin{bmatrix}\cos\psi\cos\theta & \cos\psi\sin\theta\sin\varphi-\sin\psi\cos\varphi & \cos\psi\sin\theta\cos\varphi+\sin\psi\sin\varphi\\ \sin\psi\cos\theta & \sin\psi\sin\theta\sin\varphi+\cos\psi\cos\varphi & \sin\psi\sin\theta\cos\varphi-\cos\psi\sin\varphi\\ -\sin\theta & \cos\theta\sin\varphi & \cos\theta\cos\varphi\end{bmatrix} \tag{3}$$

$$\mathbf{J}_2(\vec{\eta}_2)=\begin{bmatrix}1 & \sin\varphi\tan\theta & \cos\varphi\tan\theta\\ 0 & \cos\varphi & -\sin\varphi\\ 0 & \sin\varphi/\cos\theta & \cos\varphi/\cos\theta\end{bmatrix} \tag{4}$$

B. Collision avoidance detection model

The collision avoidance detector can ensure the AUV detecting and perceiving the surrounding environment in time. It can be seen as the basis for AUV path planning. During the navigation, AUV judges whether obstacles exist and whether the obstacles affect navigation through the data obtained by the sensors in the detector.

Due to the characteristic of simple principle and low cost, ranging sonar is the most widely used underwater detector [9]. The ranging sonar can actively emit a single beam. When the beam encounters an obstacle, it produces a reflected beam and returns to the ranging sonar. The ranging sonar calculates the distance to the obstacle by calculating the time difference between the emitted beam and the received beam. Multi-beam sonar integrates multiple ranging sonars, which can send out multiple beams at the same time to realize the joint detection of a sector angle area directly in front of the AUV. Multi-beam sonar is mostly placed in the bow of the AUV. The collision avoidance detection schematic diagram is shown in Fig. 2, where *sonar_range* represents the maximum detection distance of the ranging sonar.

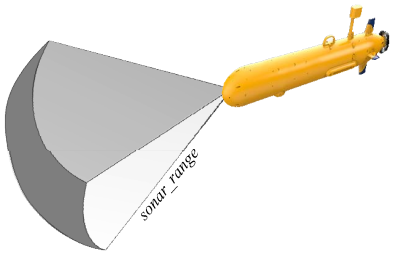

Figure 2. Collision avoidance detection model

This paper assumed that at the bow of the AUV, arranges 15*15 ranging sonars with an opening angle of 10°. The multi-beam sonar works together to realize detection in the horizontal and vertical directions within the range of -70°~70°. When there is no obstacle within the opening angle of the beam, the detected distance information is *sonar_range*. Otherwise, the distance from the AUV to the obstacle is detected. Record the detected distance information according to the position distribution of the beams, the corresponding detection matrix $\mathbb{R}$ can be obtained. The matrix size is 15*15, contains 225 distance information. Considering that the amount of data of the detection matrix is relatively large and the amount of useful information is small, $\mathbb{R}$ is processed by normalization and max pooling methods as follow.

(1) Normalization

The detection distance in the matrix is normalized using *sonar_range*, and the processing equation is:

$$\mathbb{R}'(i,j) = 1 - \frac{\mathbb{R}(i,j)}{\text{sonar\_range}} \quad i \in [1,15] \ \ j \in [1,15] \tag{5}$$

When the detection distance is *sonar_range*, there are no obstacles in the detection range at this time, and the corresponding value in the matrix is 0; when the detection distance is less than *sonar_range*, there are obstacles in the detection range at this time, and the corresponding value of the detection matrix is greater than 0 and gradually increased to 1 within the distance decreased.

(2) Max pooling

In convolutional neural networks, the pooling layer is usually used to compress the image matrix information and extract its main features, which simplifies the computational complexity of the network. Max pooling can be used to reduce the dimension of the detection matrix to achieve the purpose of reducing the amount of data. The max pooling operation uses the maximum value in the filter to represent the characteristics of the region, as shown in Fig. 3. The size of the filter and the stride together determine the pooling effect. Assuming the size of the filter is 2*2 and stride is 2. The size of the detection matrix □″ after pooling is 8*8, and the amount of data is greatly reduced to 64.

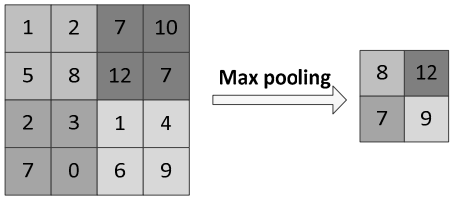

Figure 3. Max pooling

III. GYM ENVIRONMENT ESTABLISHMENT

Based on the AUV model, a DRL environment for path planning can be established through the Gym platform. In the gym environment, state space, action space, and related functions need to be defined.

A. State space and action space definition

The AUV's observations of the environment constitute the state space. The defined state space includes: the linear velocity and angular velocity in the three-degree-of-freedom motion space, the error angle between the heading and the target guidance direction, and the detection matrix after normalization and max pooling. The specific settings of the state space are shown in Table 3. The action space contains the actions that AUV can perform. The specific settings of the action space are shown in Table 4.

TABLE III. STATE SPACE DEFINITION

Num	*Symbol*	*Meaning*	*Category*	*Quantity*
1	*u*	forward & backward speed	speed information	3
2	*v*	sideway speed		
3	*w*	floating & sinking speed		
4	*p*	heel	angular velocity	3
5	*q*	trim		
6	*r*	turn bow	information	
7	*error_1*	angle error in the horizontal plane	angle error	2
8	*error_2*	angle error in the vertical plane		
9	□	processed detection matrix	environmental detection	8*8
Total				72

TABLE IV. ACTION SPACE DEFINITION

Num	*Symbol*	*Meaning*	*Quantity*
1	τ_T	thrust	1
2	*rudder*	rudder control angle	1
3	*elevator*	elevator control angle	1
Total			3

B. Related function definition

The core interface of gym is **Env**. As a unified environment interface, its main architecture includes functions such as **Step**, **Reset**, and **Render** [10]. Among them, the **Step** function is used to write the step logic for the interaction between the Agent and the environment; the **Reset** function is used to reset the state of the Agent before the start of each epoch; the **Render** function is mainly a visual operation for drawing pictures.

(1) **Reset** function

The **Reset** function resets the initial position and attitude information of the AUV, updates the position information of obstacles and target points, resets the speed and direction of the ocean current, and returns to the reset state observation observe. In the environment, all obstacles are set as fixed spherical obstacles. In order to avoid the generation of infinite loops, after each obstacle is generated, it is necessary to determine the positional relationship with the generated obstacle and the target point. When an overlap is found, the obstacles are randomly generated again until the obstacles that meet the number requirements are generated.

(2) **Step** function

The **Step** function executes the input action, and performs a time step simulation of ocean current and AUV movement. Among them, the motion of the ocean current is set as a noise signal whose speed and direction are in accordance with the normal distribution; the motion of the AUV is solved by the fourth-order Runge-Kutta method of its dynamic equation.

After a time step of updating, we can get the new state observation, the reward value, and the flag information of whether the current epoch is over. The reward value is set by the reward function. In order to avoid the sparseness of the reward function and speed up the training speed of the algorithm, this paper divides the reward function into two parts, epoch reward and single-step reward. The low limit of reward is set as *min_reward*.

At the end of the epoch, according to the positional relationship between the AUV, obstacles and the target point, the corresponding epoch reward can be set as:

$$R_{\text{episode}}=\begin{cases}+100, & \text{reach the target}\\ -100, & \text{collision with obstacles}\\ 0\ , & \text{otherwise}\end{cases} \quad (6)$$

During each epoch, rewards obtained by the AUV are single-step rewards. The single-step reward is composed by the accumulation of obstacles penalty, speed penalty, and distance reward:

- Obstacles penalty

The azimuth angles measured by different ranging beams are different, and the severity of the impact of the measured obstacle information on the AUV is also different. According to offset angle, the degree of influence of a single beam is defined as $\omega(i,j)$, calculated by:

$$\omega(i,j)=\left(1-\frac{\left|\text{angle}_H(i)\right|}{\text{angle}_H(\max)}\right)*\left(1-\frac{\left|\text{angle}_V(i)\right|}{\text{angle}_V(\max)}\right)+\xi \quad (7)$$

where, $\text{angle}_H(i)$ and $\text{angle}_V(i)$ are respectively indicate the horizontal and vertical ranging angles of the current beam. $\text{angle}_H(\max)$ and $\text{angle}_V(\max)$ are maximum offset angle, set as 70°. $\xi=0.05$, is the correction coefficient, which avoids the problem of weight disappears due to the excessively large offset angle. The heat map distribution of the beam influence degree is shown in Fig. 4.

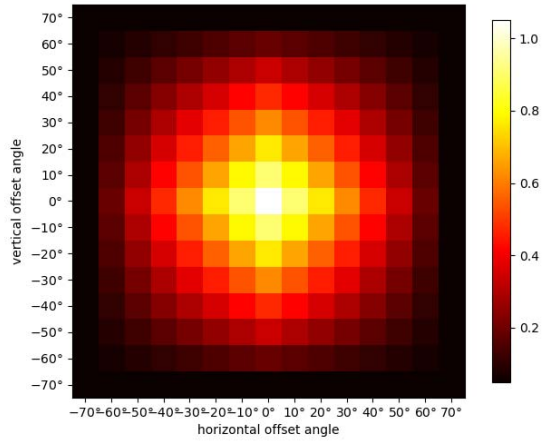

Figure 4. Heat map of beam influence

According to the obstacle information in the normalized detection matrix $\mathbb{R}'$, combined with the degree of beam influence, the obstacle penalty information is defined as:

$$R_{\text{obs}}=\sum_{i=1}^{15}\sum_{j=1}^{15}\left(\omega(i,j)*\mathbb{R}'(i,j)\right)*r_1 \quad (8)$$

where, r_1 is the penalty coefficient for obstacles.

- Speed penalty

In each epoch, after the AUV executes the corresponding action, the speed will change to a certain extent, which deviates from its own cruising speed. After normalizing the difference between the current AUV speed v and the cruise speed v_{cruise}, it is added to the reward function in the form of penalty information. The calculation equation is:

$$R_{\text{speed}}=\text{abs}\left(\frac{v-v_{\text{cruise}}}{v_{\max}}\right)*r_2 \quad (9)$$

where, r_2 is the penalty coefficient for speed, $v_{\max}$ is the maximum speed of the AUV.

- Distance reward

Record the distance between the AUV and the target point in the previous time step as d_{last}. After a time step updated, the distance between the AUV and the target is recorded as d_{new}. In a single time step Δt, the distance between the AUV and the target changes can be expressed as: d_{new}-d_{last}. Under the current time step, the cruise distance of the AUV is $v_{\text{cruise}}*\Delta t$. The distance reward can be expressed as:

$$R_{\text{distance}}=\text{clip}\left(\frac{d_{\text{new}}-d_{\text{last}}}{v_{\text{criuse}}*\Delta t},-1,1\right)*r_3 \quad (10)$$

where, r_3 is the reward coefficient for distance. The clip function mainly limits the ratio to the range of [-1,1] to prevent the distance reward from being too large.

C. *Other elements setting*

After a single time step, the state space is updated, and the generated actions can continue to act on the AUV for the next time step. Whether the current round is finished can be judged by the epoch completion flag. In the gym environment, three evaluation indicators are set: the minimum reward value, the maximum distance limit, and the maximum number of time steps. When any of the following conditions is met, the completion flag is set to the completed state, and the epoch ends.

(1) The cumulative reward value generated is less than the minimum reward value;

(2) The distance between the position of the AUV and the target point exceeds the maximum distance limit;

(3) The cumulative running time step of the round is greater than the limit of the maximum number of time steps;

(4) AUV reaches the target point;

(5) AUV collision with obstacles.

So far, the gym environment for AUV path planning has been established. The overall framework process of the gym environment is shown in Fig. 5.

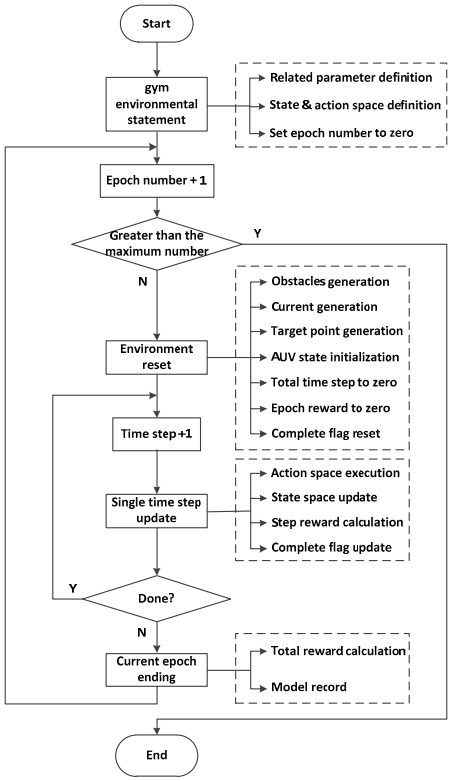

Figure 5. gym environment framework

IV. PPO ALGORITHM DESIGN

A. Related theoretical basis

Proximal Policy Optimization (PPO) algorithm as a representative algorithm under the Actor-Critic framework, it optimizes the step size selection mechanism and can be well applied to path planning problems including high-dimensional state space and continuous action space.

In the PPO algorithm, in order to prevent the performance collapse caused by the selection of the step size, the ratio of the transition probability under the current strategy $\pi_\theta(a_t|s_t)$ to the transition probability under the original strategy $\pi_{\theta old}(a_t|s_t)$ is introduced into the objective function, thereby modifying the objective function to:

$$J^{CPI}(\theta)=E_t\left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}A_t^{\pi_{\theta old}}\right]=E_t\left[r_t(\theta)A_t\right] \quad (11)$$

In order to avoid the sudden change of the strategy caused by the large gap between the current strategy and the original strategy, Schulman [11] proposed the PPO algorithm with clip method. The ratio $r_t(\theta)$ is artificially set within a fixed cut-off range to prevent fluctuations caused by the large gap between strategies. The objective function of the PPO algorithm in clip truncation mode is:

$$J^{clip}(\theta)=E_t\left[\min\left(r_t(\theta)A_t,\text{clip}\left(r_t(\theta),1-\varepsilon,1+\varepsilon\right)A_t\right)\right] \quad (12)$$

where, ε is the cut-off coefficient set manually, usually set to 0.1~0.2.

B. Optimized PPO

Generally, the Actor and Critic networks of the PPO algorithm contain 2 fully connected layers, and the number of neurons is 64 and 64. When solving the path planning problem, the performance is not good. Therefore, the network structure needs to be optimized. The optimized network structure is shown in the Fig. 6. Both the Actor and Critic networks contain 3 fully connected layers, the numbers of neurons are 128, 128, and 64 respectively. The activation function selects the tanh function.

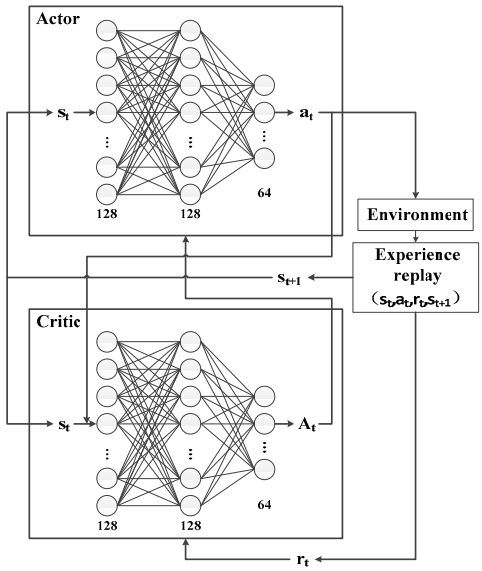

Figure 6. Algorithm network structure diagram

In the Actor network, the observation state of the environment is taken as input, the corresponding strategy is generated. The corresponding action is generated at the same time. The Critic network evaluates the current strategy through the advantage function. Assuming that the parameters of the Actor network is θ_A, the learning rate is α_A, the original network parameters can be expressed as θ_{Aold}. The parameters of the Critic network is θ_C, and the learning rate is α_C. The advantage function can be expressed as:

$$A_t=Q_{\text{target}}-Q_{\theta_C}(s_t,a_t) \quad (13)$$

where, Q_{target} is the target Q value of the state-action pair, which can be expressed as:

$$Q_{\text{target}}=r_t+\gamma Q_{\theta_C}\left(s_t,\pi_{\theta_A}(s_{t+1})\right) \quad (14)$$

The objective function and parameter update formulas of Actor and Critic networks are:

$$r_t(\theta_A)=\frac{\pi_{\theta_A}(a_t|s_t)}{\pi_{\theta_{Aold}}(a_t|s_t)} \tag{15}$$

$$J^{clip}(\theta_A)=E_t\left[\min\left(r_t(\theta_A)A_t, clip(r_t(\theta_A),1-\varepsilon,1+\varepsilon)A_t\right)\right] \tag{16}$$

$$\theta_A=\theta_A+\alpha_A\nabla_{\theta_A}J^{clip}(\theta_A) \tag{17}$$

$$L(\theta_C)=E\left[A_t^2\right] \tag{18}$$

$$\theta_C=\theta_C+\alpha_C\nabla_{\theta_C}L(\theta_C) \tag{9}$$

In the PPO algorithm, it is necessary to generate a state transition sequence under the Actor network with a network parameter of, and extract a small sample *minibatch* from it for training. At fixed time steps *n_steps*, the corresponding objective function is calculated, and the parameters are updated through the gradient of the strategy and *MSE*. The parameter settings of the PPO algorithm are shown in Table 5.

TABLE V. ALGORITHM PARAMETER SETTING

Num	*Symbol*	*Meaning*	*Value*
1	α_A	learning rate in Actor network	10^{-5}
2	α_C	learning rate in Critic network	10^{-5}
3	ε	elevator control angle	0.2
4	γ	cut-off coefficient	0.99
5	*n_steps*	interval time step	1024
6	*batch_size*	batch size	256

V. PATH PLANNING SIMULATION VERIFICATION

A. Simulation parameter setting

The compiled gym environment needs register. The related parameters need to be set at the same time. The parameter settings are shown in Table 6.

TABLE VI. ENVIRONMENT RELATED PARAMETER SETTINGS

Num	*Symbol*	*Meaning*	*Value*	*Measure*
1	α_A	thrust range	0~14	N
2	*rudder*	maximum rudder angle	30	°
3	*elevator*	maximum elevator angle	30	°
4	v_{max}	maximum speed	2	m/s
5	v_{cruise}	cruise speed	1.5	m/s
6	r_1	obstacle penalty coefficient	-0.5	
7	r_2	speed penalty coefficient	-0.08	
8	r_3	distance reward coefficient	1	
9	*sonar_range*	maximum detection distance	40	m
10	*n_obstacles*	number of obstacles	10	
11	*goal_dist*	target distance	400	m
12	*min_reward*	reward value lower limit	-500	
13	$\triangle t$	single time step	0.1	s

In the DRL training process, when the completion flag of the gym environment is true, it means that the current epoch is over. The generated epoch reward and the single-step reward are added together to form the total reward value of the current epoch. After recording the total reward value of the current epoch, call the **Reset** function to initialize the gym environment and restart the next epoch of training.

B. Simulation result record

Python is used for simulation. Based on the corresponding basic DRL algorithm in the *Stable_baselines* 3 [12], the programming and training of the PPO algorithm are carried out. The training lasted for 5000 epochs in total, and the corresponding reward function curve is shown in Fig. 7.

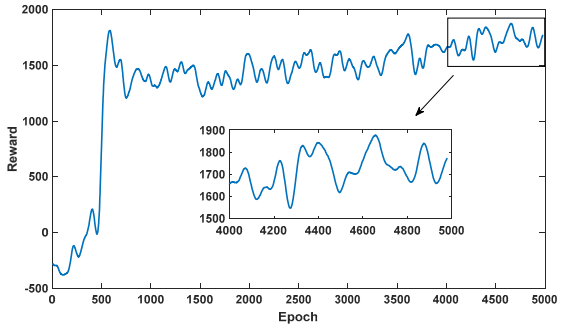

Figure 7. Reward value change trend

This paper randomly generates 20 environments, uses the trained model to make predictions. The total reward value and the number of time steps are recorded in Fig. 8. Fig. 9 shows the AUV path planning results in one test environment.

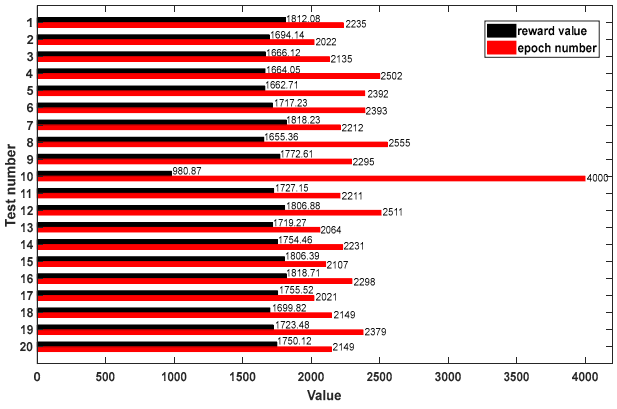

Figure 8. Model prediction result record

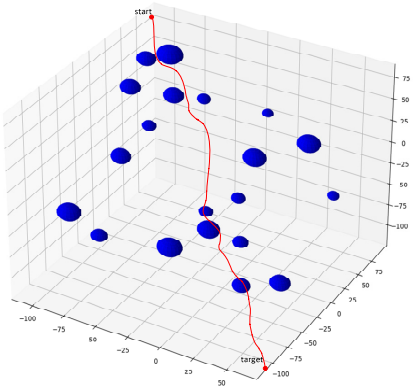

Figure 9. Path planning results

C. *Result analysis and conclusion*

Through Fig. 7, it can be seen that at the beginning of the epoch iteration, the reward value is increased. Around 600 epoch, the reward value increases significantly, and the algorithm converges faster. At this time, the Agent begins to generate autonomous awareness of path planning. Around 4000 epoch, the reward value function tends to stabilize. The reward value from 4000 to 5000 epoch is stable at 1550 to 1880. Through calculation, the average reward value is 1721.96. Since the reward value is stable within a fixed interval, it can be considered that the PPO algorithm has been trained and the corresponding training model has been output.

Through Fig. 8, it can be seen that the predicted reward value of the tenth environment is 980.87, and the number of execution time steps is 4000, which exceeds the set maximum time step limit. In this verification environment, the AUV did not reach the target point. As shown in Fig. 9, in the remaining 19 environments, the AUV has completed the local path planning well.

Through the analysis of the results, the conclusions are as follows:

(1) The gym environment established for path planning is practical and effective;

(2) The optimized PPO algorithm has a faster convergence rate and can complete DRL training faster;

(3) The trained model can be used to quickly plan the path. The success rate of the model is 95%, and the generalization ability is strong.

VI. Conclusions

In this paper, in order to solve the AUV path planning problem, the AUV models including the kinematics and dynamics model, the obstacle avoidance model are established. Base on the models above, the gym simulation environment for path planning is established. The network structure has been optimized to improve the performance of PPO algorithm. Simulation was carried out to verify the effective.

The generalization ability of the model obtained through 5000 times of training is strong. The path results obtained are effectively. The convergence speed and robustness of the algorithm are verified. In the follow-up research, the path tracing problem will be studied.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (41771487) and Hubei Outstanding Youth Science Foundation Project (2019CFA086).

References

[1] Fan, Frank. "Summary of AUV Yumeiruka sea trial results." OCEANS 2016-Shanghai. IEEE, 2016, pp.1-7.

[2] Petres, Clment, et al. "Path planning for autonomous underwater vehicles." IEEE Transactions on Robotics , vol. 23(2), 2007, pp. 331-341.

[3] Liqiang Liu, Fei Yu, Yuntao Dai. “Path Planning of Underwater Vehicle in 3D Space Based on Ant Colony Algorithm.” Journal of System Simulation, vol. 20(14), 2008, pp. 3712-3716.

[4] Zheping Yan, Yufeng Huang, Feng Li. “Research on the application of genetic algorithm in local path planning for AUV.” Applied Science and Technology, 36(2), 2009, pp. 46-50.

[5] Quan Liu, Jianwei Zhai, Zongzhang Zhang, et al. “A Survey on Deep Reinforcement Learning.” Chinese Journal of Computers, 41(1), 2018, pp. 1-27.

[6] J. Xin, H. Zhao, D. Liu and M. Li, "Application of deep reinforcement learning in mobile robot path planning," 2017 Chinese Automation Congress (CAC), 2017, pp. 7112-7116, doi: 10.1109/CAC.2017.8244061.

[7] Rizhong Wang, Huiping Li, Di Cui, Demin Xu. “Depth control of autonomous underwater vehicle using deep reinforcement learning.” Chinese Journal of Intelligent Science and Technology, 2(4), 2020, pp. 354-360.

[8] Xin Pan, Guoli Feng, Xinguo Hou. “Research on AUV path tracking technology based on hierarchical reinforcement learning.” Journal of navel university of engineering, 33(3), 2021, pp. 106-112.

[9] Ishiguro, H., et al. "Behavior selection and environment recognition methods for humanoids based on sensor history." 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2006, pp. 3468-3473.

[10] Brockman, Greg, et al. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).

[11] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[12] Raffin, Antonin, et al. "Stable baselines3." GitHub repository (2019).