Dynamic Path Planning Algorithm for Unmanned Ship Based on Deep Reinforcement Learning

Yue You[1], Ke Chen[1], Xuan Guo[2(✉)], Hao Zhou[3], Guangyu Luo[2], and Rui Wu[2]

[1] Naval Research Institute, Beijing, China
[2] School of Automation, Wuhan University of Technology, Wuhan, China
233980@whut.edu.cn
[3] School of Naval Architecture, Ocean and Energy Power Engineering, Wuhan University of Technology, Wuhan, China

Abstract. In order to enable the unmanned ship to have the ability of autonomous path planning in the complex marine environment, which can avoid obstacles and reach the destination accurately in the unknown environment, this paper combines the ability of deep learning to obtain information with the decision-making ability of reinforcement learning, and proposes an algorithm based on deep reinforcement learning. The simulation results show that after 3000 rounds of training in a given environment, the success rate of sailing to the end point in this environment is 100%. The path planning algorithm based on deep reinforcement learning performs well in the simulation experiment, achieves a very high accuracy after a large number of training times, and meets the needs of the operation and use of the unmanned ship on the water. This algorithm can be carried in the actual environment to realize the autonomous navigation of the unmanned ship on the water.

Keywords: Unmanned ship · Path planning · Deep learning · Reinforcement learning

1 Introduction

The exploration of marine resources promotes the application of path planning algorithm in unmanned ships. Path planning means that a ship finds the best or suboptimal path from the beginning to the target location in a complex environment with obstacles under certain constraints [10]. Compared to global static path planning, Local dynamic path planning is more suitable for the uncertain environment at sea, and it can realize real-time navigation more easily. At present, many scholars have developed different local dynamic path planning algorithms, including non-intelligent and intelligent algorithms:

Inspired by the concept of "field" in physics, Khatib et al. [6] proposed the concept of artificial potential field method in 1986. Chen Huiwei [2] applied the

L. Pan et al. (Eds.): BIC-TA 2021, CCIS 1566, pp. 373–384, 2022.
https://doi.org/10.1007/978-981-19-1253-5_28

artificial potential field method to the path navigation of unmanned ships by changing the models of gravitational potential field and repulsive potential field. Zadeh [8] and others put forward the concept of dynamic window method in 1997 and applied it to local obstacle avoidance of robots. Traditional non-intelligent algorithms also include A* algorithm [7], Theata* [1] algorithm and so on.

Non-intelligent algorithm is simple to apply, but it requires accurate modeling of agents, which leads to the fact that real-time and generalization cannot be well guaranteed. In contrast, the intelligent algorithm does not need an accurate model, and has a stronger sense of the environment and a stronger generalization ability. In 1991, DorigoM and others [4] put forward the ant colony algorithm to solve the travel agency problem. Zhao Feng [11] aims at combining traditional ant colony algorithm with greedy algorithm to meet the requirements of path planning in different dynamic environments. Chen Huiwei [3] applied ant colony algorithm to the path planning of unmanned ships by improving heuristic function and updating pheromones.

The aforementioned algorithm may encounter dimension disaster in complex environment, which makes the algorithm unable to converge. Combining deep learning with intensive learning can solve the problem. The deep neural network directly processes the high-dimensional original information, while the reinforcement learning algorithm interacts with environment.

At present, the commonly used deep reinforcement learning algorithms, such as DQN algorithm [9] and DDQN algorithm [5], are all proposed by combining neural network with Q-Learning algorithm. When these methods are iterative in strategy updating, agents only consider the influence of obtaining the reward values of two adjacent states on the Q value of state-action. Because the $Q(\lambda)$ algorithm can make the reward obtained by the agent affect the state-action Q value of the adjacent multi-step state. This is equivalent to making the agent gain the ability to perceive the development of the previous situation in advance. Therefore, we adopt the idea of $Q(\lambda)$ algorithm to improve DDQN algorithm, and propose a λ-Step DDQN mechanism-oriented DDQN algorithm (λS-DDQN).

2 Algorithm Design

2.1 λS-DDQN Algorithm

In the process of training, λS-DDQN enables the reward obtained by the unmanned ship to spread the estimated value of the state value of the λ-step interval state. In this way, it can better guide the unmanned ship to learn quickly, and at the same time, it can also make the unmanned ship aware of the changes in the future state in advance. The output of the λS-DDQN target value network is:

$$y_{t\arg et}^{\lambda S-DDQN} = \sum_{i=0}^{i=\lambda-1} \gamma^i r_{i+1} + \gamma^\lambda Q_{t\arg et}(s_{t+\gamma}, \arg\max_a Q(s_{t+\lambda}, a, \theta), \theta') \tag{1}$$

The loss function of λS-DDQN network is:

$$L(\theta) = E[y_{t\arg et}^{\lambda S-DDQN} - Q(s,a,\theta))^2] \tag{2}$$

For the training and learning using λS-DDQN algorithm, it is different from the conventional DDQN algorithm in that the data stored in the experience pool is different. In the conventional DDQN algorithm, the data stored in the experience pool is $[s_t, a_t, r_t, s_{t+1}]$, but it becomes $[s_t, a_t, \sum_{i=0}^{i=\lambda-1} \gamma^i r_{t+i}, s_{t+1}]$ in λS-DDQN, so the training methods of the λS-DDQN algorithm are as follows:

Algorithm 1. Training Steps of λS-DDQN

```
1: Randomly initialize the weight $\theta$ of the current network $Q(s_t, a; \theta)$ and the weight
   $\theta'$ of the target network $Q'(s_t, a; \theta)$;
2: Initialize experience pool $D$ and set hyperparameters $\lambda$;
3:   **For** *episode* = 1, M **Do**
4:     Reset the simulation environment and obtain the initial observation state
   $s_t$,$T \leftarrow \infty$
5:     Initialize three empty arrays $S_t, A, R, S_{t+1}$
6:     Select action $a_t = \arg\max Q(s_t, a; \theta)$ according to current policy
7:     Perform action $a_t$ and return the reward value $r_t$ and the new status $s_{t+1}$
8:     Put $s_t$ into $S_t$, $r_t$ into $R$, $a_t$ into $A$, $s_{t+1}$ into $S_{t+1}$
9:     **If** $s_{t+1}$ is the terminal state **Then**
10:      $T \leftarrow t + 1$
11:    $\tau \leftarrow t - \lambda + 1$
12:    **If** $\tau \geq 0$ **Then**
13:      **If** $\tau + \lambda < T$ **Then**
14:      $r_t = \sum_{i=\tau}^{i=\tau+\lambda-1} \gamma^{i-\tau} r_i \quad r_i \in R_t$
15:    **Else**
16:      $r_t = \sum_{i=\tau}^{i=T} \gamma^{i-\tau} r_i \quad r_i \in R_t$
17:    Put $(s_\tau, a_\tau r_\tau s_{\tau+\lambda})$ into experience pool $D$
18:    Random sampling of sample data mini-batch from $D$
19:    Let $y_i = r_i + \gamma^\lambda Q_{t\arg et}(s_{t+\lambda}, \arg\max_a Q(s_{t+\lambda}, a, \theta), \theta')$
20:    Update the weight $\theta$ of the current value network by gradient descent using
   the loss function $L(\theta) = E[(y_i - Q(s, a, \theta))^2]$
21:    **Util** $\tau = T - 1$
22: **End**
```

2.2 Reward Function

The reward function is the reward value that the unmanned ship takes one action in the current state and reaches the next state, indicating the good or bad degree of taking a certain action in the current state. Because the positive reward for unmanned ships is very sparse, a new continuous combined reward function is designed in order to speed up the convergence of the algorithm. Among them, one is the reward obtained after finishing a certain round of training, such as reaching the target point or colliding, which we call terminal reward. The other is the reward obtained when the training round is not finished, which we call non-terminal reward.

Terminal Reward Design

1. When the unmanned ship reaches the target point, it gets a positive reward:

$$r_{arr} = 100; if\ d_{r-t} \leq d_{win} \tag{3}$$

   where d_{r-t} is the Euclidean distance from the unmanned ship to the target point, d_{win} is the threshold for the unmanned ship to reach the target point.
2. When the unmanned ship collides with an obstacle, it gets a negative reward:

$$r_{col} = -100; if\ d_{r-o} \leq d_{col} \tag{4}$$

   where d_{r-o} is the Euclidean distance of the unmanned ship from the nearest obstacle, d_{col} is the threshold of collision between the unmanned ship and the obstacle.

Non-terminal Reward Design

1. When the unmanned ship moves towards the target point, it gets a positive reward, otherwise it gets a negative reward:

$$r_{t\_goal} = c_r[d_{r-t}(t) - d_{r-t}(t-1)] \tag{5}$$

   where the coefficient $c_r \in (0, 1]$, is set to 1 in this paper. The unmanned ship is guided towards the target point by this reward.
2. When the minimum distance of an unmanned ship from an obstacle continuously decreases, so does the dangerous reward $r_{dang} \in [0, 1]$ earned:

$$r_{dang} = \beta * 2^{d_{\min}} \quad 0 \leq r_{dang} \leq 1 \tag{6}$$

   where $d_{\min}$ is the minimum distance of an unmanned ship from an obstacle and β is the coefficient such that the space of values of r_{dang} is $(0, 1)$.

In addition, we designed a direction reward, which is to give the corresponding reward to the unmanned ship through the angle between the forward direction vector of the unmanned ship and the direction vector of the current coordinates of the unmanned ship. When the angle is less than $\pm 18°$, the reward is 1, when the reward is greater than $\pm 18°$ and less than $\pm 72°$, the reward is 0.3, and in other cases, the reward is 0.

$$r_{ori} = \begin{cases} 1\ if\ a_{ori} \leq\ \pm 18° \\ 0.3\ if\ 18° < a_{ori} \leq 72° \\ 0\ otherwise \end{cases} \tag{7}$$

where a_{ori} is the angle between the forward direction vector of the unmanned ship and the direction vector of the unmanned ship reaching the target position in the current coordinates.

$r_{t\_goal} + r_{dang} + r_{ori}$ is defined as the final non-terminal award. This kind of reward combination solves the problem of sparse reward, so that the unmanned ship can get the corresponding reward at every step of the training process. Moving toward the target point, the unmanned ship will receive a positive reward, while away from the target point or in a collision it will receive a negative reward. In this way, the unmanned ship can better learn the corresponding strategies. On the other hand, the combined reward function enables the unmanned ship to learn strategies that can reach the target position more quickly in a shorter path.

2.3 Modular Neural Network

We divided the navigation task into two sub-tasks, namely the local obstacle avoidance module and the global navigation module. The local obstacle avoidance module is mainly used to guide the unmanned ship away from obstacles, and the global navigation is mainly used to guide the unmanned ship to a shorter path to the target position. The deep neural network module of local obstacle avoidance function and the deep neural network module of global navigation are designed respectively. The input state information of the local obstacle avoidance neural network includes the environment information detected by the distance sensor and the relative position information of the unmanned ship which becomes the control instructions output after propagating forward. The input state information of the global obstacle avoidance neural network is the relative position information of the unmanned ship, and the control command is output after the processing of the input information.

Because both the local obstacle avoidance deep neural network and the global navigation neural network output the corresponding instructions to control the unmanned ship, this paper designs an instruction selection module to determine which network's output action instructions to execute. We set a threshold $d_{to\_obs} = 40$ of distance from the ship to the nearest obstacle to determine which module's instructions to use. When the distance is less than 40, the instructions output by the local obstacle avoidance deep neural network will be executed, otherwise the instructions by the global navigation neural network should be carried out to reach the target position at a faster speed. Therefore, the system architecture of this deep learning method is shown in Fig. 1:

Based on the modular neural network framework, the unmanned ship can adopt different strategies in different environmental states. When the unmanned ship approaches the obstacle, the main task of the unmanned ship is to avoid the obstacle, and the global navigation task becomes a secondary task. When the unmanned ship is far away from the obstacle, the global navigation task becomes the main task to help the unmanned ship to reach the target position with a shorter path.

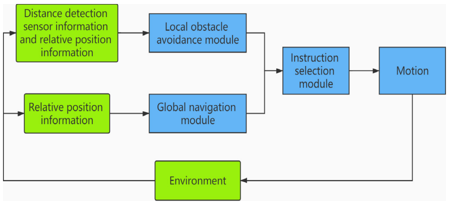

Distance detection sensor information and relative position information
Local obstacle avoidance module
Instruction selection module
Motion
Relative position information
Global navigation module
Environment

Fig. 1. The systematic framework of deep learning method.

3 Experiment

3.1 Dynamic Environment Implementation

A 500-inch 500 window size is defined as the simulation environment, and different obstacles, boundary walls and target locations are added in this window. The environment model established in this article is shown in Fig. 2.

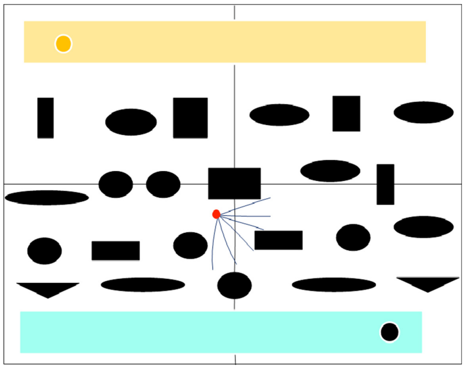


Fig. 2. Simulation environment model.

It is assumed that the starting position of the unmanned ship is P_{start} in the higher left corner of the simulation environment, the coordinate is (x_{start}, y_{start}), the moving speed is $v = 0.5$, and the current direction of the unmanned ship is *angle*. In the current state, the unmanned ship chooses an action to perform, that is, the unmanned ship chooses a steering action, the angle steering is $angle_{tran}$, the steering angle is $angle_{tran} \in (15^{\circ}_{turn\_left}, 30^{\circ}_{turn\_left}, 0^{\circ}, 15^{\circ}_{turn\_right}, 30^{\circ}_{turn\_right})$, and formula (12) is the angle of the unmanned ship after performing the action:

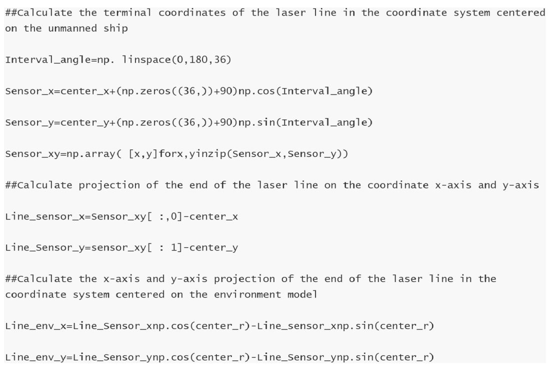
$$angle \leftarrow angle + angle + angle_{train} \tag{8}$$

Combined with formula (12), the coordinates of the unmanned ship can be changed to:

$$x_{next} = x_{start} + \cos(angle) * v \tag{9}$$

$$y_{next} = y_{start} + \sin(angle) * v \tag{10}$$

Before calculating the distance between the lidar simulator and the obstacle, the projection length of the x-axis and y-axis of each laser line in the unmanned ship as the center coordinate system is calculated, and then the projection length of each laser line segment in the environment model is calculated. The transformation from the unmanned ship as the center coordinate system to the environment model as the center coordinate system is based on the middle-two-dimensional coordinate system transformation process. Assuming that the coordinate of the unmanned ship is $(center\_x, center\_y)$, the code for solving the coordinates at the end of the laser segment centered on the environment model is shown in Fig. 3:



```
##Calculate the terminal coordinates of the laser line in the coordinate system centered
on the unmanned ship

Interval_angle=np. linspace(0,180,36)

Sensor_x=center_x+(np.zeros((36,))+90)np.cos(Interval_angle)

Sensor_y=center_y+(np.zeros((36,))+90)np.sin(Interval_angle)

Sensor_xy=np.array( [x,y]forx,yinzip(Sensor_x,Sensor_y))

##Calculate projection of the end of the laser line on the coordinate x-axis and y-axis

Line_sensor_x=Sensor_xy[ :,0]-center_x

Line_Sensor_y=sensor_xy[ : 1]-center_y

##Calculate the x-axis and y-axis projection of the end of the laser line in the
coordinate system centered on the environment model

Line_env_x=Line_Sensor_xnp.cos(center_r)-Line_sensor_xnp.sin(center_r)

Line_env_y=Line_Sensor_ynp.cos(center_r)-Line_Sensor_ynp.sin(center_r)
```

Fig. 3. Coordinate conversion code.

The distance between the unmanned ship and the obstacle can be solved when the coordinate projection of the laser line segment in the coordinate system centered on the environment model is obtained. Then each edge vector of the obstacle, the laser line segment vector, and the vector from the position coordinates of the unmanned ship to each vertex of the obstacle are constructed. By solving a relative relationship between these vectors in real time, the position information between the unmanned ship and the obstacle can be obtained, and the length of each laser line detected by lidar can be obtained.

The rules for determining the collision of the unmanned ship: first, set $d_{\min}$ as the minimum safe distance between the unmanned ship and the obstacle, and when the minimum laser line detected by the lidar is less than the set $d_{\min}$, it will be determined that the collision occurs and the training of this round ends, re-assign a new starting position to the unmanned ship. Otherwise, there is no collision, and the unmanned ship chooses the action to execute according to the relevant strategy.

The rules for determining the unmanned ship's arrival at the target position: first, $d_{Arrivals}$ is defined as the maximum distance for the unmanned ship to reach the target position. In the operation of the unmanned ship, the Euclidean distance between the current position coordinates of the unmanned ship and the target position coordinates is calculated. If the distance is less than or equal to $d_{Arrivals}$, it indicates that the unmanned ship reaches the target position.

3.2 Training of Unmanned Ship Navigation Model

The above simulation environment model is taken as the unmanned ship training environment, and the environment is set as the training environment 1(Env-1). In order to verify the effectiveness of the λS-DDQN method, we test the navigation ability of the unmanned ship in the training Env-1, and compare the λS-DDQN algorithm with the DDQN, PrioritizedDQN and PrioritizedDDQN algorithms. In order to ensure the fairness of the experiment, the same network structure and the same software and hardware platform are used for model training. Before starting the training, we set the relevant hyperparameters in deep reinforcement learning, as shown in Table 1:

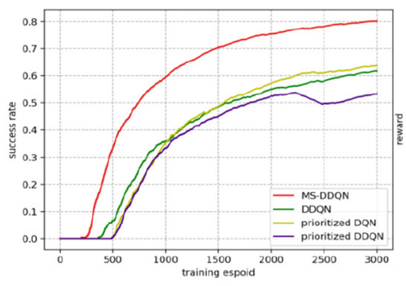
Table 1. Setting of super parameter.

Hyper parameter	Value
Learning rate	0.001
Discount factor	0.9
Maximum capacity of experience pool	15000
Number of training samples	32
Number of steps λ	5
Number of steps between replication parameters θ	300

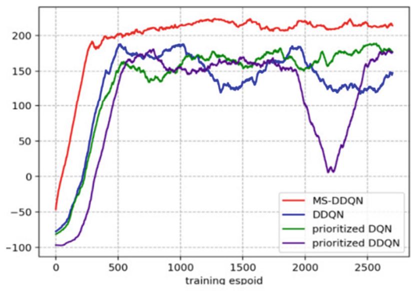
In order to evaluate the performance of each algorithm quantitatively, we use three indicators to evaluate the navigation model. The first is the success rate, which indicates the proportion of the unmanned ship's successful arrival at the target position to the total number of training from the beginning of the training; the second is the reward value curve, which represents the sum of the reward values received by the unmanned ship in each round during the training process. In order to smooth the reward curve, we use the moving average method to process the curve, and the sliding window size is 300. The third is the average value of the reward, which represents the sum of the rewards received by the unmanned ship during the training process and divided by the number of training rounds.

Based on λS-DDQN algorithm, DDQN algorithm, Prioritized DQN algorithm and Prioritized DDQN algorithm, the autonomous navigation ability of the unmanned ship in Env-1 is trained. The training results are shown in Fig. 4:

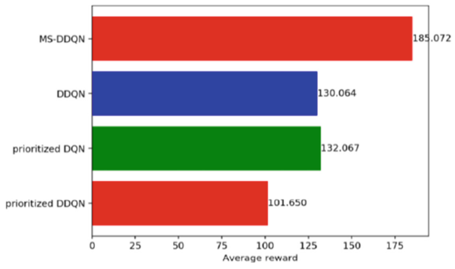
As shown in Fig. 4(a), we can see that the success rate curve of λS-DDQN rises faster than the other three methods, which indicates that the learning efficiency of λS-DDQN algorithm is higher. The reward curve in Fig. 4(b) also proves this view. After 3000 times of training, the success rate of λS-DDQN



(a) Success Rate Curve



(b) Reward Value Curve Per Round



(c) Average Reward

Fig. 4. Training result.

reaching the target location is 80.133%, and the success rate of DDQN is 61.7%. The success rate of Prioritized DQN is 63.633%, and the success rate of Prioritized DDQN is 53.366%, we can see that the success rate of λS-DDQN is much higher than that of other algorithms. This shows that the unmanned ship based on λS-DDQN has carried out more collision-free and target-free training during the training process, indicating that it has stronger obstacle avoidance and navigation functions. In Fig. 4(b), we can see that the reward curve obtained through λS-DDQN is stable at more than 200 after 500 times' training, while the curves of the other three algorithms fluctuate greatly, which indicates that the navigation model based on λS-DDQN has higher stability. In Fig. 4(c), the average reward value of λS-DDQN is 185.072, that of DDQN is 130.064, that of Prioritized DQN is 132.067 and that of Prioritized DDQN is 101.650, which also proves that the unmanned ship based on λS-DDQN has stronger navigation ability. Because the lower reward value means a lot of negative reward, it means that the unmanned ship has more collisions.

By analyzing the success rate curve, reward value curve and average reward value curve of the navigation model based on different algorithms in the training process, it can be concluded that the unmanned ship based on λS-DDQN algorithm has stronger learning efficiency and higher stability than the other three algorithms in the training process.

3.3 Testing the Accuracy of Dynamic Path Planning Model

After 3000 rounds of training in Env-1, the navigation model based on λS-DDQN algorithm, DDQN algorithm, Prioritized DQN algorithm and Prioritized DDQN algorithm is obtained. In this section, we first test these navigation models 200 times in Env-1, and analyze the proportion of successful reaching the target location. In 200 tests, the starting position and target position of the unmanned ship were randomly assigned. By comparing the success rate of the unmanned ship reaching the target position in 200 tests and getting the average reward, the superiority of the navigation model based on different algorithms is measured. The higher the success rate, the higher the average reward, indicating that the navigation model is a better strategy. The results show as shown in Table 2: after 3000 times of training, the unmanned ships trained based on these four algorithms have basically learned how to avoid obstacles and reach the target position in Env-1. According to the test results, the λS-DDQN algorithm is the best, with a success rate of 100% and the highest average reward. The results show that the unmanned ship based on λS-DDQN algorithm has higher obstacle avoidance ability and better navigation strategy. The navigation trajectory of the unmanned ship based on λS-DDQN training in the environment is shown in Fig. 5.

Table 2. Setting of super parameter.

Environment	Algorithm	Success rate	Average reward value
Env-1	λS-DDQN	100.0%	243.996
Env-1	DDQN	88.0%	203.829
Env-1	Prioritized DQN	94.0%	228.829
Env-1	Prioritized DDQN	91.0%	215.293

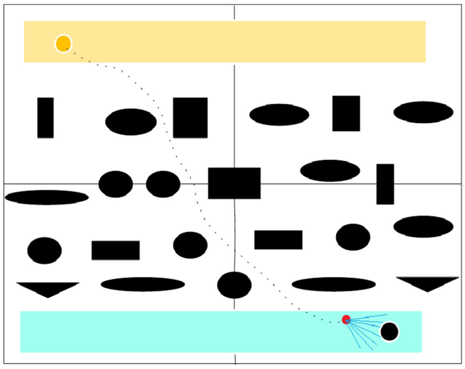


Fig. 5. Dynamic trajectory planning of underwater vehicle.

4 Conclusions

We divided the navigation task of the unmanned ship into two sub-tasks: global navigation and local obstacle avoidance. Through task decomposition, each neural network is more clear about the strategies and methods that need to be learned, so that the unmanned ship can better avoid obstacles and reach the target position with a shorter path.

By combining the idea of $Q(\lambda)$ algorithm with the traditional DDQN algorithm, a λ-step DDQN algorithm is proposed, which makes the current state of the unmanned ship obtain the influence of reward value to extend the action-state Q value of several states. This is equivalent to giving the unmanned ship the ability to perceive the future and facilitate it to evade obstacles in advance. In addition, in the process of training, we propose to use a continuous combined reward function to solve the problems of low accuracy and slow convergence caused by the sparse state of the unmanned ship.

Through the statistics of 3000 rounds of training data in a certain environment, we find that compared to the other three algorithms our algorithm have higher success rate and reward value, which shows that our algorithm has learned more information in the training process. Then the trained algorithm model is tested 200 times in this environment, and the test results show that the success rate of navigation is up to 100% under this model. The accuracy of this algorithm is enough to meet the autonomous navigation needs of unmanned ships in most marine environments, and can be encapsulated in the navigation system of unmanned ships.

References

1. Alexopoulos, C., Griffin, P.M.: Path planning for a mobile robot. IEEE Trans. Syst. Man Cybern. **22**(2), 318–322 (1992)
2. Chen, H., Chen, Y., Feng, F.: Research status of unmanned ship route planning based on artificial potential field method. Sci. Technol. Innov. **000**(17), 23–25 (2020)
3. Chen, H., Liu, P., Liu, S.: Research on path planning method of unmanned ship based on improved ant colony algorithm. Eng. Technol. Dev. **1**(8), 3 (2019)
4. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for discrete optimization. Artif. Life **5**(2), 137–172 (1999)
5. Hasselt, H.V., Guez, A., Silver, D.: Deep reinforcement learning with double Q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, AAAI 2016, pp. 2094–2100. AAAI Press (2016)
6. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous Robot Vehicles, pp. 396–404. Springer, Cham (1986). https://doi.org/10.1007/978-1-4613-8997-2_29
7. Liu, H., Bao, Y.: Application of a* algorithm in searching optimal path of vector map. Comput. Simul. **25**(4), 253–257 (2008)
8. MahmoudZadeh, S., Powers, D., Sammut, K.: An autonomous reactive architecture for efficient AUV mission time management in realistic dynamic ocean environment. Robot. Auton. Syst. **87**, 81–103 (2017)

9. Mnih, V., et al.: Playing Atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
10. Wu, H., Tian, G.H., Li, Y., Zhou, F.Y., Duan, P.: Spatial semantic hybrid map building and application of mobile service robot. Robot. Auton. Syst. **62**(6), 923–941 (2014)
11. Zhao, F.: Dynamic path planning based on ant colony algorithm and its simulation application in formation. Master's thesis, Kunming University of Science and Technology (2017)