Research on USV path planning method based on CW-RNN framework

Jian Gao, Ziwen Wu*, Dawei Zhao, Xiaogong Lin
College of Intelligent Systems Science and Engineering, Harbin Engineering University, Harbin 150001, Heilongjiang, China

ABSTRACT

This paper presents an Unmanned Surface Vehicle (USV) path planning algorithm based on a Clockwork-RNN (CW-RNN) framework. Compared with RNN, CW-RNN divides the hidden state matrix into several small modules and uses a method similar to clock frequency mask to divide the memory of RNN into several parts, so that each part of CW-RNN memory matrix can process data at different times and enhance the memory effect. The existence of ocean current will cause drifting motion in the navigation track of unmanned ship, and make the ship deviate from the planned path and direction. In view of this interference factor, ocean current vectors of different sizes and directions are added in the simulation environment to make the environmental model closer to the actual sea surface. As the environment becomes more complex, reinforcement learning takes a long time to train in the complex environment and is not easy to converge. Therefore, this paper combines reinforcement learning method with the traditional path planning method Dijkstra algorithm, inputs the local map information detected by unmanned ship into Dijkstra algorithm to give task sub targets, and uses these sub targets to guide unmanned ship and improve the search efficiency of neural network. Finally, the simulation results and analysis show that the training algorithm is effective in the current environment.

Keywords: Path planning, reinforcement learning, CW-RNN, sub targets, Dijkstra algorithm

1. INTRODUCTION

The path planning of surface unmanned ship means that the navigation system of unmanned ship continuously generates or modifies smooth and effective route trajectory according to the needs of its operation task through a variety of signals provided by sensors, so that the unmanned ship can safely and accurately sail from the starting point to the destination under various constraints to complete the path planning task[1]. There are many challenges in autonomous navigation of unmanned ship, including uncertain interference factors such as wind, wave and current in the environment, lack of kinetic energy, steering constraints and so on. Therefore, the path planning of unmanned ship is very complex[2]. It is the key to improve the autonomous ability of unmanned ship to automatically respond to route trajectory, which has been an important technology continuously studied and optimized by relevant researchers[3]. Improving the autonomous ability of unmanned ship will help to reduce its dependence on manual operation, reduce the safety problems caused by human misoperation, and reduce the cost of human using.

Path planning is widely used in many fields, such as robots[4] and unmanned ships[5]. Traditional path planning algorithms, such as artificial potential field method[6], A-star Algorithm[7], Genetic Algorithm[8], Ant Colony Algorithm[9], cannot handle complex high-dimensional environmental information in complex environment, and are easy to fall into local optimization. Reinforcement learning is an intelligent algorithm that does not require prior knowledge and interacts with the environment through trial and error iterations to obtain feedback reward information to optimize strategies. It has gradually become the research focus of path planning for unmanned ship in unknown environments[10]. The traditional reinforcement learning method is limited by the dimension of action space and sample space, and it is also difficult to deal with complex situations closer to the actual environment. In recent years, reinforcement learning using neural networks as function approximators has developed rapidly. The latest deep reinforcement learning framework has shown excellent performance in various tasks from game[11] to continuous robot control[12]. Most deep reinforcement learning studies use feedforward neural networks to solve tasks that can be modeled with Markov models[13]. Su et al.[14] put forward the theory of adding reinforcement learning method to path planning. Tan et al.[15] proposed the theory of reinforcement learning path planning based on Dijkstra algorithm. At present, in the decision-making process, the reinforcement learning framework using recurrent neural network has attracted more and more attention[16]. One of the advantages of recurrent neural network is its ability to solve time series problems. For reinforcement learning, the data

* 3073631669@qq.com

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 125060B
 · 0277-786X · doi: 10.1117/12.2662175

generated by the interaction between the agent and the environment is obviously a time-dependent data sequence. In order to better train the neural network and make it converge as much as possible, memory playback technology is generally used to eliminate the time correlation between the data. The recurrent neural network is very suitable for the processing of this kind of time related data series. However, the traditional recurrent neural network adopts the time-varying back-propagation algorithm for gradient derivation, which will cause the problem of gradient disappearance, so it is unable to obtain the long-term historical information in the past. In order to solve this problem, network structures such as LSTM[17], GRU[18] and CW-RNN[19] are proposed.

Aiming at the disadvantages of unbalanced neural network training and long training duration in reinforcement learning in large state space, this paper proposes a path planning algorithm based on the combination of actor critic and Dijkstra in CW-RNN framework. The algorithm uses the traditional algorithm to generate subtasks, reduces the state search space of reinforcement learning, generates the optimal route for the explored map, reduces invalid exploration, and improves the training speed. In the simulation environment, the influence of ocean current is considered. By setting a constant ocean current vector field, simulation experiments are carried out on the same map with and without ocean current.

2. BACKGROUND

2.1 CW-RNN network

CW-RNN is an RNN with clock frequency. For time series input, RNN uses hidden layer for memory, and then calculates output. In view of the poor memory effect of the original RNN on long sequences, CW-RNN designed a method to divide the hidden state matrix (memory mechanism) into several small modules and use a method similar to the clock frequency mask to divide the RNN memory, so that each part of the CW-RNN memory matrix can process data at different times to enhance the memory effect. As shown in Figure 1, the CW-RNN network structure has an input layer, a hidden layer and an output layer. The hidden layer is divided into G modules, each module has *k* units, and each module is given a clock frequency manually according to $T_i = 2^{i-1}$. from left to right, the clock frequency increases in turn, that is, the module on the left updates faster, and the module on the right updates slower. The internal units of the modules are fully connected. The modules that update slowly are connected to the modules that update faster in turn. Such an update mechanism and connection mechanism enable the network to save the contents of short-term memory and long-term memory at the same time. The input and output formula of CW-RNN is as follows:

$$y_H^{(t)} = f_h(W_H \mathrm{g} y_H^{(t-1)} + W_I \mathrm{g} x^{(t)}) \tag{1}$$

$$y_O^{(t)} = f_O(W_O \mathrm{g} y_H^{(t)}) \tag{2}$$

In order to add the clock frequency, W_H and W_I are processed in blocks, that is:

$$W_H = \begin{bmatrix} W_{H_1} \\ \mathrm{M} \\ W_{H_g} \end{bmatrix} \text{ and } W_I = \begin{bmatrix} W_{I_1} \\ \mathrm{M} \\ W_{I_g} \end{bmatrix}$$

During the operation, some modules will be selected to participate in the operation, and the modules that do not participate will be set to 0. The specific operation is determined by the following formula:

$$W_{H_i} = \begin{cases} W_{H_i} & if\ (t \bmod T_i) = 0 \\ 0 & otherwise \end{cases} \tag{3}$$

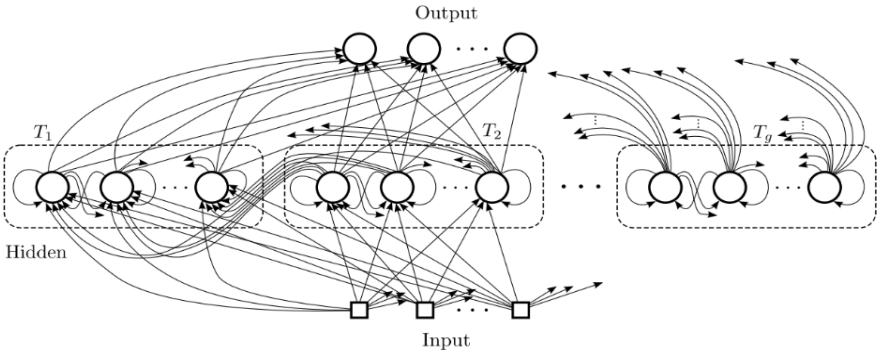

Figure 1. Network structure of CW-RNN.

As shown in Figure 2, the clock frequency in the vertical direction is arranged from high frequency to low frequency. The trigonometric matrix is set so that the relatively low frequency does not extract the memory information of the high frequency, but the high frequency extracts all the memory information of the low frequency. Therefore, the G module with high frequency update is equivalent to short-term memory, and the low-frequency *g* module is equivalent to long-term memory. For example, to predict a time series with a length of *L*=16, when using CW-RNN, we can manually set the memory sequence by setting the clock frequency. We can set CW to [1, 2, 4, 8, 16], which is equivalent to setting multiple memory points in the series with a length of 16. Each memory point carries out Abstract memory based on the input value before this memory point. Not all modules are updated at the same time at time *t*, but only the module *i* that meets the integer division of T_i is updated at time *t*. When we want to process the 6th (*t*=6) element in the sequence, we can find that only the first two modules will participate in the operation after calculating the remainder between *t* and the clock cycle of each module. So the values of all the elements of the W_H and W_I matrices except the above two rows are 0. After calculation, only the first two modules of $y_H^{(t)}$ have values. Therefore, we can regard the CW-RNN process as selecting different hidden layer nodes to work through some manual intervention.

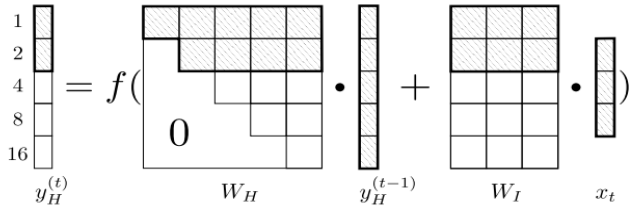

Figure 2. Calculation rules of CW-RNN hidden layer.

Observe the calculation process of the second module of the hidden layer, which is obtained by inner product of the second line of W_H and the vector of $y_H^{(t-1)}$. Since $\mathrm{W_H}$ is artificially set as the upper triangular matrix, and the value corresponding to the first module in the second row of W_H is 0, the value in the first module of $y_H^{(t-1)}$ is set to 0 during the inner product operation with $y_H^{(t-1)}$. So the value of the second module of $y_H^{(t)}$ is the weighted sum of all modules after the second module of $y_H^{(t-1)}$. Therefore, the upper triangular matrix can transfer the hidden layer module of the high-frequency clock cycle at the previous time to the hidden layer module of the low-frequency clock cycle at the current time.

2.2 Dijkstra algorithm

Dijkstra algorithm[20] is a typical geometric search method, which deals with the shortest path problem in weighted digraph by breadth first retrieval. At first, Dijkstra algorithm was only suitable for finding the shortest route between two

points. Later, researchers improved the algorithm to find the shortest route from a vertex to any other node in the graph, thus forming a shortest route search tree. The algorithm takes out the point with the smallest distance among the non visited vertices each time, and updates the distance to other vertices with the vertex. Most Dijkstra algorithms cannot effectively deal with graphs with negative weight edges. The weight edge of the path planning task in this paper is distance, which is a non negative value. A weighted digraph can be described by a triple: G= (V, E, W), where V is the set of vertices, E is the set of edges, and W is the set of weights. Figure 3 below is an example of a weighted digraph:

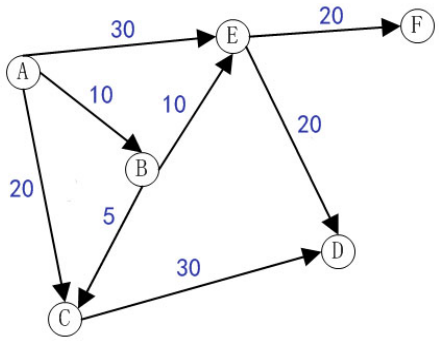

Figure 3. Digraph with weight.

Given a point in a directed graph as the starting point, we can use Dijkstra algorithm to get the shortest route from that point to other points in the graph. Calculation process is as follows: for a weighted digraph G, the vertex set *V* is divided into two parts. One part is the vertex set that has obtained the shortest route length (represented by *S*, there is only one source starting point *s* in *S* at the beginning. For each shortest route, the route vertex is added to the *S* set until *S*=*V* is calculated). The other part is the other vertex set that has not determined the shortest route, and then we will add all vertices of this part to the *s* set according to the order of the shortest alignment length.

3. ALGORITHM

3.1 Design of reward function

The state of the unmanned ship is defined as a triple (x, y, α), where x and y are the real-time coordinates of the unmanned ship in the simulation map, and α is the yaw angle between the unmanned ship and the target point. If the initial coordinate of the unmanned ship is (x_0, y_0), then the actual heading angle is initialized to $\theta = \arctan(x_0, y_0)$. If the current coordinates are (x, y) and the target coordinates are (p, q), then the calculation formula of the expected heading β is: $\beta = \arctan(x-p, y-q) - \arctan(x, y)$. The actual heading angle θ of the unmanned ship changes with each action. If the corresponding direction angle of action a is set to a_θ, then the update formula of θ is: $\theta_{new} = \theta_{old} + a_\theta$. Yaw angle α is defined as: $\alpha = \beta - \theta$. Steering angle $\Delta\theta$ is defined as: $\Delta\theta = \theta_{new} - \theta_{old}$. Obviously, the value of steering angle $\Delta\theta$ at time t is the action angle taken by the unmanned ship at this time is a_θ. The reward rules for interaction between unmanned ship and environment are set as follows:

(1) When the unmanned ship reaches the final target, it will be rewarded with +100. The purpose is to give the unmanned ship a large positive incentive and make the unmanned ship tend to move to the target position.

(2) If the unmanned ship encounters an obstacle or boundary, reward -2. Each time it collides with an obstacle or boundary, it is an unexpected action. Therefore, a negative penalty will be given to the unmanned ship, so that the unmanned ship tends to avoid collision with the obstacle or boundary when sailing.

(3) Reducing (increasing) the distance between the unmanned ship and the target, we set +1 (-1) reward, and we encourage the unmanned ship to move close to the target position to avoid the unmanned ship to move away from the target position.

(4) The smaller the yaw angle α of the unmanned ship, the greater the reward. This is to encourage the unmanned ship to reduce the deviation between the actual course and the desired course, make the course of the unmanned ship relatively flat, avoid meaningless tracks, and reduce the length of the final planned path.

(5) The smaller the steering angle $\Delta\theta$ of the unmanned ship, the greater the reward. This is to encourage the unmanned ship to reduce the steering angle. If the steering angle is too large, the rotation torque and the inclination angle between the unmanned ship and the water surface will increase, which will consume more energy of the unmanned ship, reduce its endurance, and increase the risk of the unmanned ship capsizing.

In order to meet the requirements of 4 and 5, we will take the cosine value of the corresponding angle as the reward.

3.2 Influence of ocean current factors on reward design

During USV navigation, not only the threat of obstacles such as islands and reefs to the route safety, but also the influence of ocean currents on the track deviation of unmanned ship should be considered. In the marine environment, small disturbances will affect the navigation attitude of the unmanned ship, resulting in the unmanned ship deviating from the scheduled route. Therefore, it is necessary to adjust the rotating speed of the propeller blade to cooperate with the actions of various rudders to reduce the impact caused by disturbance. The simplified dynamics model of the unmanned ship is shown in the following equations (4) and (5):

$$M\dot{v}+C(v)v+D(v)v+g(\eta)=g_0+w+\tau \tag{4}$$

$$\dot{\eta}=J(\eta)v \tag{5}$$

Among them, M represents inertia force matrix, $C(v)$ represents Colorado force matrix, $D(v)$ represents damping force matrix, and $g(\eta)$ represents buoyancy moment; g_0 represents the equilibrium moment provided by the ship's ballast water, w represents the marine environment moment, here we believe that the ship is only affected by the ocean current, and τ represents the ship's propulsion moment.

The influence of the angle between the current and the bow direction of the unmanned ship is also different. When the angle between the unmanned ship and the current is right angle, the current forms the maximum deflection moment to the unmanned ship; When the sailing direction is opposite to the current direction, the power of the unmanned ship has to overcome the current resistance to do work, and the power loss is the largest; In order to minimize the interference of ocean current to the unmanned ship, the course should be at an acute angle with the direction of ocean current as far as possible. When selecting actions in the action space, considering the ocean current interference, the unmanned ship can make more effective use of its own power and improve its range and path planning ability.

Due to the influence of random current, the reward function needs to be improved to adapt to the new environment. The current will make the course of the unmanned ship deviate. In order to make full use of the current and reduce the power loss of the unmanned ship, we add the influence of the current to the reward function. In state s, the included angle between the current direction and the heading direction of the unmanned ship is set to be σ, and the range of angle σ is -180 degrees to +180 degrees. Therefore, the additional reward under the influence of current can be defined as $r_d=\cos(\sigma)$. If the reward without considering current is r_c, the final reward can be expressed as $r=r_c+\eta r_d$, where η is an adjustment coefficient. In this paper, the length of the planned path is mainly considered. In order to make the model less complex, the energy loss under the influence of ocean currents is not considered too much, so $\eta=0.5$ is taken. In addition, we will use the Dijkstra algorithm to generate the shortest path to the final target point, evenly extract 16 points from the path coordinates (excluding the start point and end point) as sub task targets, and number the sub targets according to the direction from the start point to the end point. Each sub target will be rewarded only when it arrives for the first time. At the same time, if the number of sub targets encountered is less than the sub targets encountered before. No reward will be given regardless of whether it is encountered for the first time. This is to prevent the agent from repeatedly moving between the target sequences in order to obtain high rewards and ignoring that our final goal is to reach the destination.

3.3 Path planning algorithm based on CW-RNN framework

The network structure we use is actor critic, which replaces the state feature extraction part of the environment with CW-RNN. The simulation map of the experiment is 400*400, and the unmanned ship initializes a 400*400 matrix map to record the explored map information. We set the exploration radius of the unmanned ship as 5. With the continuous exploration of the unmanned ship, the map will be updated continuously, and then the Dijkstra algorithm will be used to generate sub task targets. During the initial exploration, the map does not contain the end point target goal. The unmanned ship will calculate the non obstacle point subgoal in the explored map that is closest to the straight line of the end point goal as the temporary target point, and the Dijkstra algorithm will calculate the shortest path from the start point start to the temporary target point subgoal. When the explored map contains destination target goal, we update the temporary target point: $subgoal \leftarrow goal$ and calculate the shortest path from the starting point start to the ending point goal. The ratio of the number of proved map units to the total number of map units is called the map exploration rate μ. When $\mu > 0.9$, we think that the map has been basically proved and will not update the shortest path L. In order to reduce the computational complexity of Dijkstra algorithm, when generating the shortest path, we combine 5*5 units, and reduce the map size to 80*80.

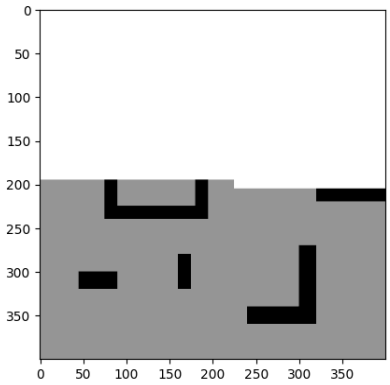

Figure 4. Unmanned ship explored area.

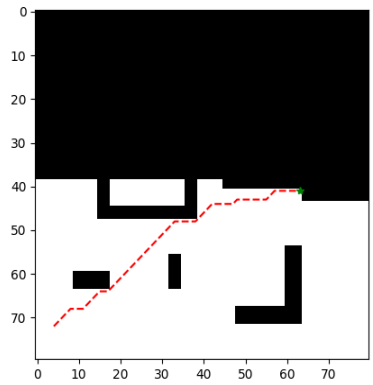

Figure 5. Generate path l based on explored area.

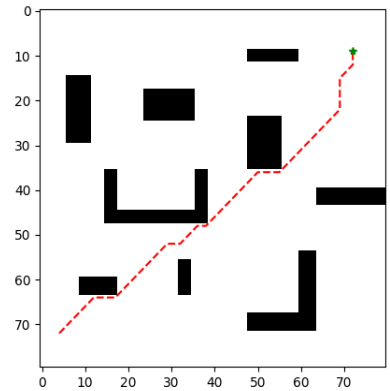

Figure 6. Final path L.

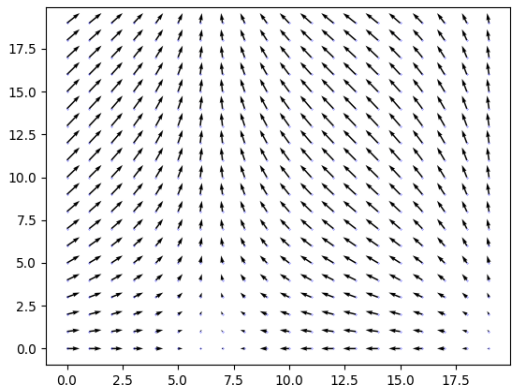

Figure 7. Current vector field.

The unmanned ship will incline when turning. We use the heel angle to describe the inclination angle between the unmanned ship and the water surface. If the heel angle is too large, the unmanned ship will overturn. Therefore, the design of the action space must ensure that the unmanned ship can keep its course stable as much as possible or reduce the number of large angles turns as much as possible during navigation and obstacle avoidance, so as to ensure that the

planned route path is relatively smooth. This can not only ensure the navigation safety of unmanned ship, but also reduce the power loss of unmanned ship during navigation. In order to make the angle of the unmanned ship more in line with its motion characteristics in actual navigation, the angle is designed according to the principle of small angle steering. The action space is set to advance a unit distance in the specified direction, in which the direction is set to plus or minus 15 degrees, plus or minus 30 degrees and 0 degrees. We remove the start point and the end point on the shortest path L, and uniformly select the coordinates of the points on the 16 paths as the task sub goals. The process of unmanned ship exploring the map and generating the shortest path L is shown in Figures 4-6. The explored area is represented by gray, and the unexplored area in the map generating the path is represented by black by default.

In order not to make the ocean current too complex, the 20*20 unit is taken as the basic range, within which the ocean current is of constant direction and size. In this way, 20*20=400 ocean current vectors will be added to the simulation map. The current vector field is shown in Figure 7. The construction of current vector uses trigonometric function. If the coordinates of any point on the map are (x, y), the current vector divided by point (x, y) is expressed as:

$$\begin{cases} S_x = \cos(0.24x) \\ S_y = \sin(0.12y) \end{cases} \tag{6}$$

3.4 Algorithm steps

The specific implementation steps of USV path planning algorithm are as follows:

(1) We initialize parameters ω and θ of critic network and actor network; counter count, explored map information matrix, map exploration rate μ.

(2) We initialize the input sequence input and start state S_0.

(3) Actor value network outputs action a and executes action a to obtain environmental feedback: next state s', immediate reward r, completing signal done, collision signal collision.

(4) We judge whether done is true. If yes, we will end the training and jump to Step 2.

(5) We judge whether it is $\mu > 0.9$. If yes, we jump to Step 6. Otherwise, we update the explored map. The network generates the shortest path using Dijkstra algorithm, and evenly extracts the path coordinates as the subtask target.

(6) We update network parameters ω and θ:

$$\omega \leftarrow \omega + \beta\delta\nabla_w \hat{v}(s_t, \omega) \tag{7}$$

$$\theta \leftarrow \theta + \alpha\delta\nabla_\theta \log\pi(a_t|\ s_t, \theta) \tag{8}$$

(7) The agent synchronizes the critic value network and critic target network parameters every 20 steps.

(8) We judge whether the path generated by Dijkstra algorithm contains the final target. If yes, we set the completing flag to true.

(9) We judge whether collision is true. If yes, we return to the previous state and jump to Step 3.

(10) We judge whether the count reaches the maximum value, whether to end the training or jump to Step 2.

4. SIMULATION EXPERIMENT

We will conduct two simulation experiments on the same map. In the first experiment, the random current factor is not added, and in the second experiment, the influence of random current is added. The map size is 400*400, each unit represents 5 meters, and the control group is the actor critic algorithm built using the fully connected network. The training frequency is 2000 times, and the upper limit of each training is set at 500 steps. If the final goal is not reached after 500 steps, it will be regarded as failure. The comparison indicators include average steps, average rewards, success rate, path length and turning times. Among them, the turning times refer to the turning times of unmanned ship at large

angles, which in this paper refers to plus or minus 30 degrees. The parameter settings of the simulation experiment are shown in Table 1 below:

Table 1. Simulation experiment parameter setting.

Parameter	Value
Maximum single iteration steps	500
Maximum number of iterations	2000
Initial exploration rate ε_0	0.9
Lower limit of exploration rate $\varepsilon_{\min}$	0.05
Decreasing exploration rate η	0.0005
Subtask target quantity	8
Number of hidden layer neurons in each group of CW-RNN	32
Number of CW-RNN clock frequency modules	6
Number of fully connected network neurons	16
Actor network learning rate α	0.001
Critic network learning rate β	0.001
Critic network iteration discount rate γ	0.99

The simulation experiment is shown in Figures 8 and 9 below. Simulation Experiment 1 does not include the influence of ocean current, and Simulation Experiment 2 includes the influence of ocean current. After 2000 times of training, the path planning trajectories of the two algorithms are given. The solid line trajectory represents the algorithm based on CW-RNN, and the dotted line trajectory represents the actor critic algorithm. For simplicity, the legend and the following description refer to the algorithm based on CW-RNN as CW-RNN algorithm for short.

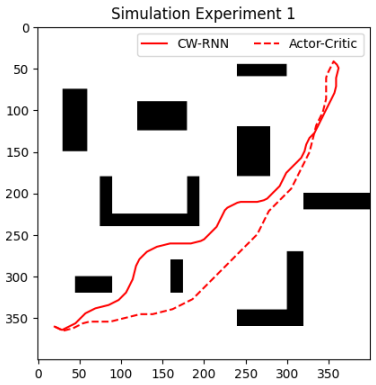

Figure 8. Simulation experiment 1.

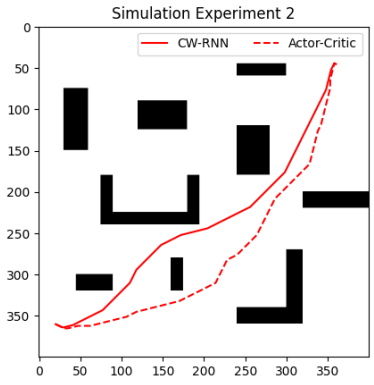

Figure 9. Simulation experiment 2.

Figure 8 depicts the path planning trajectories of two algorithms. The trajectory planned by CW-RNN algorithm is relatively flat, with a length of 2543 meters. the trajectory planned by actor critic algorithm has relatively more turns, with a length of 2615 meters. Figure 9 shows that the track length planned by CW-RNN algorithm is 2472 meters, while the track planned by actor critic algorithm has many turns, and the track length is 2603 meters. Two simulation

experiments show that CW-RNN algorithm has obvious advantages over actor critic algorithm in path length. Figures 10-13 show the reward data of the simulation experiment:

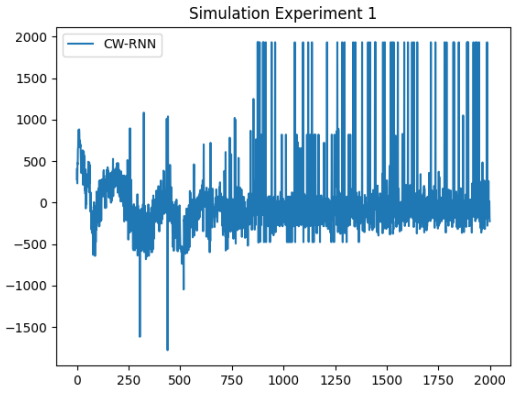

Figure 10. Experiment 1 CW-RNN reward.

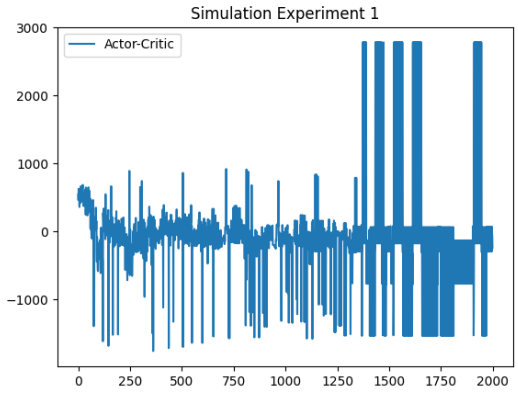

Figure 11. Experiment 1 Actor-Critic reward.

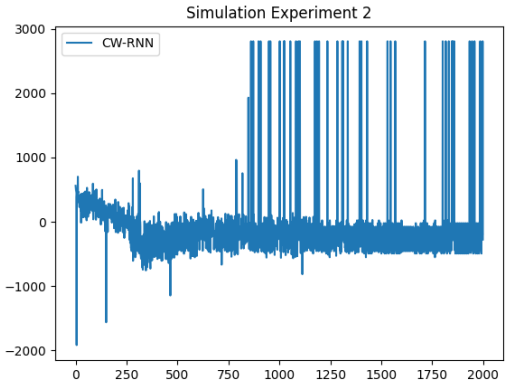

Figure 12. Experiment 2 CW-RNN reward.

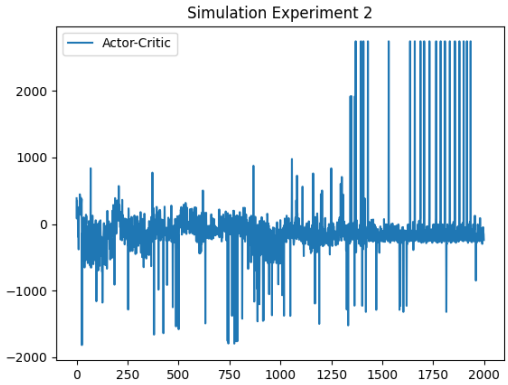

Figure 13. Experiment 2 Actor-Critic reward.

Tables 2 and 3 list the statistical data of the two simulation experiments:

Table 2. Simulation experiment 1.

Experiment 1	Average reward	Average step	Success	Path length	Turns
CW-RNN	-13.93	355.44	71.06	2543m	2
Actor-Critic	-46.47	381.87	39.47	2615m	7

Table 3. Simulation experiment 2.

Experiment 2	Average reward	Average step	Success	Path length	Turns
CW-RNN	-114.05	351.48	73.65	2472 m	1
Actor-Critic	-123.62	394.93	38.12	2603	5

5. CONCLUSION

This paper presents a path planning method for unmanned ship based on CW-RNN framework. There is one simulation map. The influence of ocean current is not included in simulation experiment 1, and the influence of ocean current is included in simulation experiment 2. We focus on the length of the path, and consider the current impact as a secondary factor. Compared with actor critic algorithm, CW-RNN algorithm has higher average reward, smaller average steps, fewer turns, shorter and smoother path trajectory. In the simulation experiment, the success rate of CW-RNN algorithm is almost twice that of actor critic algorithm. In simulation experiment 2, ocean current influence is added. It can be seen that the trajectory of the two algorithms on the right of the map is not consistent with the direction of ocean current vector field, so the average reward of the two algorithms decreases significantly. For the CW-RNN algorithm, the sub target has a greater impact on it (achieved by setting a larger reward value), so the ocean current has little impact. The average number of steps increases slightly compared with simulation experiment 1, while the actor critic algorithm receives the ocean current impact, and the average number of steps decreases. In general, from the simulation experiments, we can see that the CW-RNN algorithm has obvious advantages over the traditional actor critic algorithm of fully connected network, with small average steps, high average reward, high success rate, shorter and smoother planned path, which indicates that the path planning algorithm based on the CW-RNN framework proposed in this paper has better achieved the expected goal.

REFERENCES

[1] Zhang, W. J. and Xiao, W., "Development of the world surface unmanned vehicle (USV), a new weapon at sea in the future," Light Weapons (12), 13 (2017).

[2] Li, J. L., "Development and application of unmanned surface vehicle," Fire Control & Command Control 37(06), 205-209 (2012).

[3] Li, K., [Automatic Route Design Based on Ant Colony Algorithm], Dalian Maritime University, Master's Thesis, (2018).

[4] Wong, C. C., Chien, S. Y., Feng, H. M., et al., "Motion planning for dual-arm robot based on soft actor-critic," IEEE Access 9(2), 26871-26885 (2021).

[5] Gasparetto, A., Boscario, P., Lanzutti, A., et al., "Path planning and trajectory planning algorithms: A general overview," Motion and Operation Planning of Robotic Systems 29(3), 3-27 (2015).

[6] Khatibk, O., "Real-time obstacle avoidance system for manipulators and mobile robots," The International Journal of Robotics Research 5(1), 90-98 (1986).

[7] Hotel, R., Perez, M., Zimmer, R., et al., "Hierarchical A*: Searching abstraction hierarchies efficiently," 1996 AAAI Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference. Portland: AAAI, 530-535 (1996).

[8] Karami, A. H. and Hasanzadeh, M., "An adaptive genetic algorithm for robot motion planning in 2D complex environments," Computers & Electrical Engineering 43(4), 317-329 (2020).

[9] Mirjalili, S., Dong, J. S. and Lewis, A., "Ant colony optimizer: Theory, literature review, and application in AUV path planning: Methods and applications," Studies in Computational Intelligence 811(2), 7-21 (2020).

[10] Polydoros, A. S. and Nalpantidis, L., "Survey of model based reinforcement learning: Applications on robotics," Journal of Intelligent & Robotic Systems 86(2), 153-173 (2017).

[11] Mnih, V., Kavukcuoglu, K., Silver, D., et al., "Human-level control through deep reinforcement learning," Nature 518(7540), 529-533 (2019).

[12] Haarnoja, T., Zhou, A., Abbeel, P., et al., "Soft Actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," arXiv:1801.01290v2, (2018).

[13] Bellman, R., "A Markovian decision process," Journal of Mathematics and Mechanics 6(6), 679-684 (1957).

[14] Su, M. C., Huang, D. Y., Chou, C. H., et al., "A reinforcement learning approach to robot navigation," 2004 IEEE International Conference on Networking, Sensing and Control, 665-669 (2004).

[15] Tan, G. Z., Huan, H. E., et al., "Global optimal path planning for mobile robot based on improved Dijkstra algorithm and ant system algorithm," Journal of Central South University of Technology 13(1), 80-86 (2006).

[16] Al-Shedivat, M., Bansal, T., Burda, Y., et al., "Continuous adaptation via meta-learning in nonstationary and competitive environments," arXiv:1710.03641v2, (2018).

[17] Hochreiter, S. and Schmidhuber, J., "Long short-term memory," Neural Computation 9(8), 1735-1780 (1997).

[18] Cho, K., Merrienboer, B. V., Gulcehre, C., et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," arXiv:1406.1078v3, (2014).

[19] Koutnik, J., Greff, K., Gomez, F., et al., "A clockwork RNN," Proc. of the 31st International Conference on Machine Learning, 1863-1871 (2014).

[20] Dijkstra, E. W., "A note on two problems in connexion with graphs," Numerische Mathematik 1(1), 269-271 (1959).