2021 International Conference on Networking, Communications and Information Technology (NetCIT) | 978-1-6654-0070-1/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetCIT54147.2021.00040

The USV Path Planning Based on an Improved DQN Algorithm

Zhijian Huang*
Lab of Intelligent Control and Computation
Shanghai Maritime University
Shanghai 201306, China
zjhuang@shmtu.edu.cn

Huiqun Lin
Lab of Intelligent Control and Computation
Shanghai Maritime University
Shanghai 201306, China
202030110135@stu.shmtu.edu.cn

Guichen Zhang*
Lab of Intelligent Control and Computation
Shanghai Maritime University
Shanghai 201306, China
gczhang@shmtu.edu.cn

***Abstract*—Research on the path planning algorithm of USV (unmanned surface vessel) has received widespread attention. Among them, the algorithm based on the combination of deep learning and reinforcement learning achieves better performance. However, in the complex water environment, this algorithm is prone to overestimate the action value, which will lead to path planning deviation. In this paper, we propose a Double DQN algorithm to optimize obstacle avoidance and path planning, which based on deep Q network (DQN) algorithm. The Double DQN algorithm will decouple the target Q value's action selection and action evaluation, and index the action value corresponding to the maximum Q value through the current network, then input the selected action value into the target network to calculate the target Q value. As a result, the overestimation is reduced, and path deviations are corrected. The simulation results show that the Double DQN algorithm has a good performance in path planning compared with the DQN algorithm, and indicate the Double DQN algorithm can effectively process complex environmental information and make optimal path planning.**

Keywords—USV (unmanned surface vessel), Double DQN (deep Q network), reinforcement learning, path planning

I. INTRODUCTION

In recent years, the research of unmanned surface vessel (USV) has achieved a series of achievements include environmental sensing technology, intelligent navigation control technology and path planning technology. Among them, Path planning technology has a significant impact on the mission execution efficiency and navigation safety of USV. Path planning is the process of finding a path between two points in an area while avoiding collisions with objects in the area. Traditional path planning algorithms are difficult to satisfy efficient path planning in various complex environments. Therefore, a reinforcement learning path planning algorithm combined with deep learning is proposed in this case.

The path planning algorithm of USV can be divided into global and local path planning. The idea of the global path planning algorithm is to plan the optimal obstacle avoidance path based on the distribution of obstacles in the known environment [1, 2]. A* algorithm and Dijkstra algorithm are classic algorithms for solving static path planning optimization problems [3, 4]. However, in the complex and changeable water environment, the above algorithms can hardly obtain the optimal performance of the planned path. Intelligent bionic algorithms are used to deal with these highly nonlinear path planning problems, which can effectively reduce path redundancy and loss, and obtain the global optimal path, but the real time ability is still insufficient [5, 6].

Local path planning is usually modeled based on the spatial geometric relationship of surrounding obstacle [7, 8]. However, The complex model structure and huge amount of calculation reduce the efficiency of path planning. The artificial potential field method is combined with other algorithms to avoid the shortcomings of falling into local minimums and improve the flexibility of traditional path planning algorithms[9, 10]. The local path planning algorithm has good real time performance. Since only local information can be obtained from ship equipment, the lack of global environment related information leads to poor path planning performance.

The path planning algorithm that combines the advantages of the above two types of algorithms has outstanding performance, such as obtaining environmental information through electronic chart and AIS information for global path planning, and then using particle swarm optimization or ant colony algorithm for local path planning [11], [12] and [13]. In recent years, with the development of deep learning and reinforcement learning, USV path planning algorithm has achieved a major breakthrough. Deep Q network [14] replaces Q network with neural network, which deals with the Q value table capacity limitation of Q-learning algorithm[15] in complex environment, and improves the performance of the algorithm.

II. DQN ALGORITHM

Compared with the Q-learning algorithm, the DQN algorithm mainly includes three main improvements: 1)the use of neural networks to approximate the value function. 2) Learning by memory replay. 3)A separate neural network is used to handle the deviation in the difference.

A. Target network

The deep neural network in the DQN algorithm approximates $Q(s,a)$ in a high-dimensional state. the predictive network estimates the value of the current state. Before learning the approximation of action value function, it is necessary to determine the optimization objective of the network, and then use the existing parameter learning methods to update the weight parameters of the model, so as to obtain the approximate value function.

978-1-6654-0070-1/21/$31.00 ©2021 IEEE
DOI 10.1109/NetCIT54147.2021.00040

The DQN algorithm builds an optimizable loss function of the neural network based on the Q learning algorithm, and the update formula is as follow:

$$Q(S,A)=Q(S,A)+\alpha\left(R+\gamma\max_{a}Q(S',a)-Q(S,A)\right) \quad (1)$$

According to formula (1), the loss function of DQN algorithm is defined as:

$$L(\theta)=E\left[\left(T\text{arget}Q-Q(\text{s},a,\theta)\right)^2\right] \quad (2)$$

Among them, θ is the pre-update parameter, and the target Q value can be described as follows:

$$T\text{arget}Q=r+Vmax_{a'}Q(s',a',\theta) \quad (3)$$

Since the loss function in the DQN algorithm is determined by the update formula which based on the Q-learning algorithm, formula (1) has the same meaning as formula (3). They are all based on the current Q value to approximate the target Q value, and finally update the weight parameters based on the gradient descent algorithm.

The DON algorithm includes a predictive network and a target network: the predictive network $Q(s,a,\theta)$ estimates the action value in the current state; the target network (s,a,θ^{-}) calculates the target value. The algorithm updates the parameters θ in the predictive network according to the loss function from the formula(2), and then after every iteration, the new network parameter θ is assigned to the parameters θ' in the old network.

DQN algorithm stabilizes the Q value in a constant state by introducing the target network, and at the same time reduces the correlation between the predicted Q value and the target Q value, so that the loss value during training tends to a stable state, and ultimately speeds up the convergence speed of the training algorithm.

B. *Memory replay*

In the reinforcement learning algorithm, most of the experience bars are highly correlated, which will lead to difficulty in the convergence of the algorithm model.

The DQN algorithm introduces memory replay mechanism: store the experience tuples in the memory bank, then, a random number of memories are extracted each time and input into the neural network for learning. With the introduction of memory replay, on the one hand, reward values can be stored; On the other hand, The correlation between experience bars is reduced, which is conducive to accelerating the convergence of the model network and evaluating the value function more accurately .

The DQN algorithm contains many historical memory samples, and each memory is stored in the form of a five-tuple (s,a,r,s',T). Each element in the tuple indicates that the USV indexes and executes behaviors a in the state s, and then enters the state s'. At the same time, the environment rewards *r*. Among them, T judges whether the new state s' is terminated.

After the environment executes a step, the agent stores the experience information obtained by executing the step in the memory bank. Randomly extract a certain amount of experience tuples from the memory bank every few steps, and based on the extracted experience samples, execute formula (3) to update the Q function.

C. *Neural network fitting Q value table*

In order to adapt the algorithm to different complex environments, the DQN algorithm introduces the neural network on the basis of the Q-learning network, and uses the neural network to approximate the Q function. The DQN architecture can be seen from Fig.1, in the early stage, DQN algorithm interacts with the environment through agents to explore the environment and obtain experience, so as to build an experience pool. The creation process of the experience pool is also the process of approximating the Q value through the neural network. Based on a certain strategy, the neural network is trained to fit the corresponding value function.

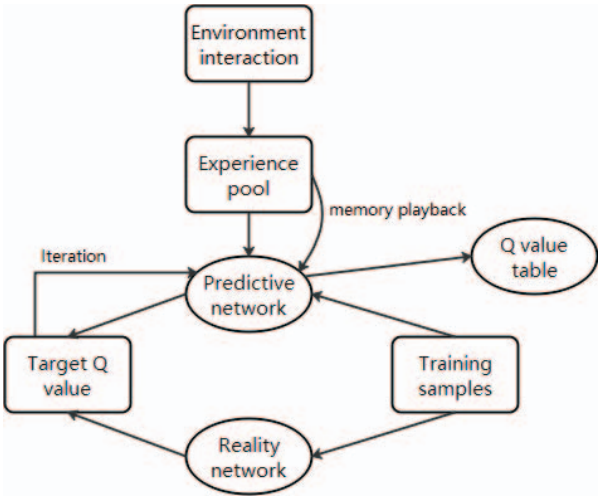


Fig.1. DQN architecture

III. DOUBLE DQN ALGORITHM

A. *Classical DQN algorithm*

After the combination of reinforcement learning and deep neural network, there are some problems of instability in the training process. In order to solve this problem, two methods are proposed. One is memory bank storage. when the agent moves each time, the transition tuple $e_t=(s_t,a_t,r_t,s_{t+1})$ will be stored in the experience pool $D=\{e_1,e_2,\ldots,e_t\}$, and extract a random number of samples from the memory bank for training. The other one is to use two kinds of Q networks, namely $Q(s,a;\theta)$ and $Q(s,a;\theta^{-})$, where θ represents the current parameter and θ^{-} represents the historical parameter. Algorithm convergence is accomplished by minimizing the following cost function:

$$L_i(\theta_i)=E\left[\left(r+\gamma\max_{a'}Q(s',a';\theta_i^{-})-Q(s,a;\theta_i)\right)^2\right] \quad (4)$$

By differentiating this loss function, we will get the following gradient formula:

$$\nabla_{\theta_i} L_i(\theta_i) = \left(r + \gamma \max_{a'} Q\left(s', a'; \theta_i^-\right) - Q\left(s, a; \theta_i\right)\right) \nabla_{\theta_i} Q(s, a; \theta) \quad (5)$$

The DQN algorithm uses a greedy strategy to estimate the Q value, and the selection of actions through the greedy strategy can effectively balance the exploration and utilization of the state. When the probability is $1-\varepsilon$, the greedy strategy is adopted, and when the probability is ε, the random index action is taken.

B. *Q value overestimation*

In the Q learning algorithm training process, the random error of the action value is evenly distributed in the interval $[-\varepsilon, \varepsilon]$,Then each target Q value will be overestimated to $\gamma\varepsilon\frac{m-1}{m+1}$,where m is the number of actions, the DQN algorithm uses the same Q value to index and predict actions, which is likely to result in Q value overestimation.

The greedy strategy is used in Q learning and deep Q learning to obtain the corresponding Q value. Even if different neural networks are used to estimate the Q value, the target Q value of the J-th memory sampling is still obtained through the greedy strategy. The calculation is as follows:

$$y_j = \begin{cases} R_j & is\_end_j\ is\ true \\ R_j + \gamma \max_{a'} Q'(\phi(S'_j), A'_j, w') & is\_end_j\ is\ false \end{cases} \quad (6)$$

Relying on the maximum Q value estimated by the neural network to index the corresponding action that can improve the training speed, and the Q value can be closer to the target value faster, but there will be an overestimation phenomenon. In order to deal with this phenomenon, The Double DQN algorithm will decouple the target Q value's action selection and action evaluation to reduce overestimation. The calculation method for the predicted Q value when it is not over is as follows:

$$y_j = R_j + \gamma \max_{a'} Q'(\phi(S'_j), A'_j, w') \quad (7)$$

In the Double DQN algorithm, index the action value corresponding to the maximum Q value through the current network. the action value can be described as follows:

$$a^{\max}(S'_j, w) = \arg\max_{a'} Q(\phi(S'_j), a, w) \quad (8)$$

Therefore, another neural network in Double DQN is used to reduce the impact of errors. The estimation network is used to estimate the maximum action value of the target network, and the Q value in reality is selected based on this. The update formula is as follows:

$$y_j = R_j + \gamma Q'(\phi(S'_j), \arg\max_{a'} Q(\phi(S'_j), a, w), w') \quad (9)$$

C. *Double DQN architecture*

Based on the above algorithm model, Double DQN algorithm is suitable for USV path planning. Input the USV location information, and then perceive the location information through the neural network, and thereby select the action corresponding to the maximum Q value and execute it, so as to obtain the largest cumulative rewards, which is the optimal strategy to realize USV autonomous obstacle avoidance and path planning. This algorithm architecture is shown in the Fig.2.

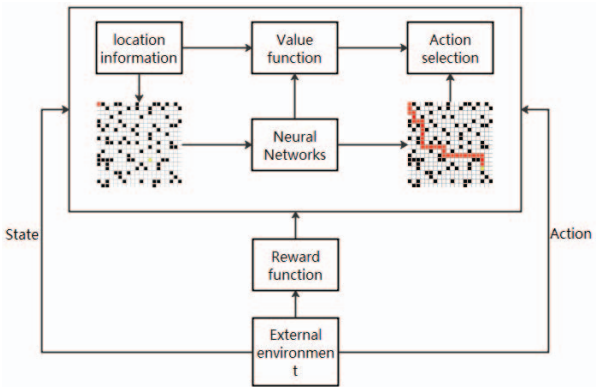


Fig.2. Double DQN architecture

IV. SIMULATION

The simulation environment is realized by the Python's TK-inter module. TK-inter is the interface of Python's standard TK GUI toolkit. It contains several modules for the development of graphical interfaces. The simplified USV model and map environment used in simulation. the USV is approximately equivalent to a 1x1 pixel block. The action space of USV agent is classified into four basic actions: up, down, left and right. As shown in Fig.3.

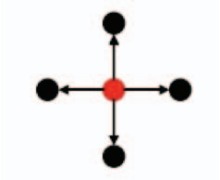


Fig .3 .USV action distribution

The USV agent takes related actions through coordinate transformation. Assuming that the current location information of the USV agent is S(x, y), the corresponding execution behavior can be written as:

$$S_{i+1} = \begin{cases} \left(x_i - 1,\ y_i\right) & \left(up\right) \\ \left(x_i + 1,\ y_i\right) & \left(down\right) \\ \left(x_i,\ y_i - 1\right) & \left(left\right) \\ \left(x_i,\ y_i + 1\right) & \left(right\right) \end{cases} \quad (10)$$

The action selection in the training process adopts the ε greedy strategy, and the exploration mechanism can be defined as:

$$\begin{cases} \alpha = \arg\max_a Q(s, a) & (\varepsilon) \\ exploration & (1-\varepsilon) \end{cases} \quad (11)$$

The action space in this experiment is limited and discretized, so the reward function R can be generalized: When the USV touch with the black block, it means that the USV failed to effectively avoid the obstacle. The reward value is set to -10; the reward value for not reaching the

target point and not touching with the black block is set to 0, when the USV reaches the destination, the reward value is set to 10. The function can be described as follows:

$$R=\begin{cases}10 & (reaching)\\ 0 & (not\ reaching,\ not\ touching)\\ -10 & (touching)\end{cases} \quad (12)$$

In order to verify the effectiveness of the proposed algorithm in USV path planning, a two-dimensional grid is built as shown in Fig.4. Red is USV agent, yellow is target point and black is obstacle. Four different target points are set to observe the cumulative reward and Q value of the two algorithms. In order to ensure the accuracy of the experiment, all hyper-parameters are consistent with the DQN algorithm. A five layer neural network with three hidden layers is built. The input is the location information of the current USV, which is a 1x4 matrix. The number of neurons in the hidden layer is 10, 20 and 10 respectively, and the output is four units, representing the Q values of the four actions respectively. RMSProp algorithm is used to train the network. The learning rate of neural network is set to 0.0001, the attenuation coefficient is set to 0.9, the batch processing capacity is 64, the initial setting is 0.5, and the memory capacity is set to 3000.

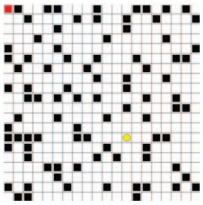


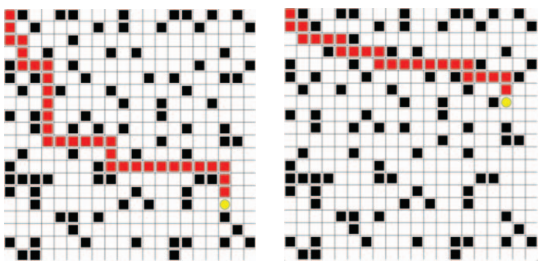
Fig.4. Path planning environment

V. RESULTS

Experimental configuration: Intel Core i9 (CPU frequency 2.3Ghz, RAM 16GB) and two graphics cards NVIDIA GTX1080 Ti (3584cores, 11GB memories), and Ubuntu 16.04 operating system.

A. Path planning

The experiment builds a two-dimensional grid to simulate the path planning of the USV. The USV agent starts from the initial point, explores the environment, obtains rewards and punishments in the process of interacting with the environment. At the same time, the neural network is updated iteratively. Finally, the outcome of path planning is optimal. The path planning of Double DQN algorithm in four random environments is shown in Fig.5.



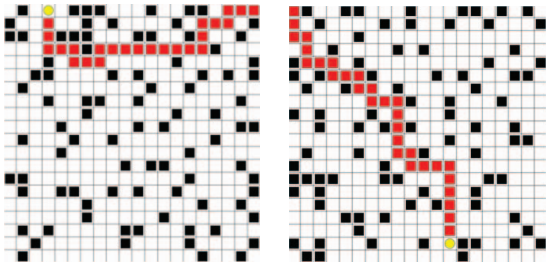


Fig.5. Path planning under different target points

B. Trainging Data

At the beginning of the training, the USV touches with obstacles frequently and cannot effectively avoid it; when training 8000 times, the algorithm can gradually plan a safe path, but the path is not the shortest at this time, and the time required is relatively long; when training 12000 times, the USV obstacles can basically be avoided, the algorithm tends to converge and can plan an effective path, and the time-consuming is gradually shortened; after training 20,000 times, the USV can effectively avoid obstacles and plan the optimal path at the same time. Table 1 shows the average number of steps and average time required in the DQN algorithm training process. It can be seen from the Table 1 that the average number of steps to reach the target point in the four different target point path planning experiments using the DQN algorithm are 189.2, 176.6, 200.3, and 192.8. The average time consumption are $31.7s$, $28.2s$, $34.4s$ and $33.2s$. Table 2 shows the average steps and average time required by Double DQN algorithm in the above path planning tasks. It can be seen from the table that the average steps required by Double DQN algorithm in the experiment are 181.4, 104.7, 112.2 and 128.1. The average time consumption are $24.3s$,$18.9s$, $22.3s$ and $23.5s$.

TABLE I. DQN ALGORITHM RELATED DATA

DQN	ROUTE			
	1	2	3	4
Step	189.2	176.6	200.3	192.8
Time/s	31.7	28.2	34.4	33.2

TABLE II. DOUBLE DQN ALGORITHM RELATED DATA

DOUBLE DQN	ROUTE			
	1	2	3	4
Step	181.4	104.7	112.2	128.1
Time/s	24.3	18.9	22.3	23.5

C. Q value and Cumulative Rewards

The Double DQN algorithm and the DQN algorithm are applied to the path planning of the above four different target points for comparison. Fig.6 shows the comparison of the Q value of the training process of the two algorithms, and Fig.7 shows the comparison of the cumulative reward value of the two algorithms. It can be seen from the figure that the Double DQN algorithm starts to converge at 15000, 12000, 20000, and 18000 steps respectively, while the DQN

algorithm needs 22000, 12000, 25000, and 23000 steps to start to converge respectively.

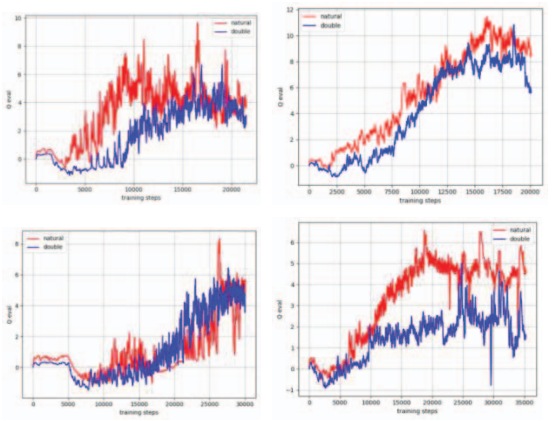


Fig.6. Comparison of the Q value of the training process of the two algorithms

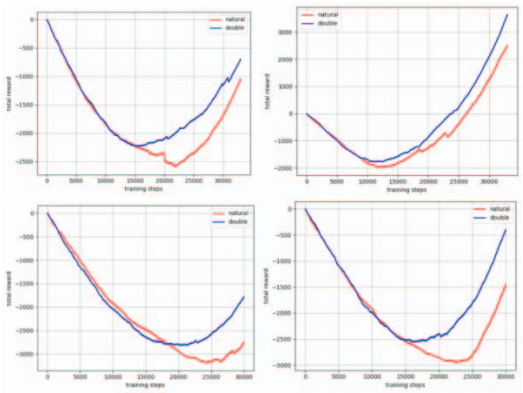


Fig.7. The cumulative reward value of the training process of the two algorithms

VI. CONCLUSION

The USV did not adapt to the environment at the beginning of the interaction with the environment, and could not plan an effective path, resulting in multiple collisions between the USV and obstacles. With continuous training and iteration of network parameters, the decision system has accumulated learning experience, which can effectively plan the path and reach the target point safely. Through comparative experiments, it can be known that path planning the Double DQN algorithm can effectively reduce the deviation of the estimated value and accelerate the convergence speed of the neural network. Tested on Path Planning, the Double DQN algorithm has achieved better performance than DQN algorithm.

In the simulation, the simulation environment used is static rather than dynamic. In practice, the USV needs to deal with many uncertain factors brought about by the complex water environment. Therefore, in order to improve the practical application ability of the algorithm, it is necessary to build a map that can restore the real ocean environment, and then perform obstacle avoidance and path planning algorithm on the map.

This article uses the TK-inter module in python to build the environment. In order to match the algorithm with the environment, the action output of the algorithm is discretized. Since the USV is continuously controlled in reality, the proposed algorithm cannot be applied in real objects. The continuous control algorithms include AC (actor-critic) algorithm, PPO algorithm, multi-threaded PPO (DPPO) algorithm, DDPG algorithm and TD3 algorithm. The improved algorithm based on the above algorithm architecture will be the research focus of USV path planning in the future.

ACKNOWLEDGMENT

This work is supported by the NSFC of China under grant No.51779136, and NSFC of Shanghai under grant No. 21ZR1426600, No. 21DZ1201004.

REFERENCES

[1] J. Hong, K. Park. A new mobile robot navigation using a turning point searching algorithm with the consideration of obstacle avoidance. Adv. Manuf. Technol, 2011,52: 763-775.

[2] P. Yao, H. Wang, H. Ji. Gaussian mixture model and receding horizon control for multiple UAV search in complex environment. Nonlinear Dyn., 2017,88(2):903-919.

[3] R. Song, Y. Liu, R. Bucknall. Smoothed A* algorithm for practical unmanned surface vehicle path planning. Appl. Oceans Res, 2019,83(920).

[4] Y. Singh, S. Sharma, R. Sutton, D. Hatton, A. Khan. Feasibility study of a constrained Dijkstra approach for optimal path planning of an unmanned surface vehicle in a dynamic maritime environment. 18th IEEE International Conference on Autonomous Robot Systems and Competitions, 2018:117-122.

[5] B. Song, Z. Wang, L. Zou, L. Xu, F. E. Alsaadi. A new approach to smooth global path planning of mobile robots with kinematic constraints. Mach. Learn. Cybern., 2019,10(1):107-119.

[6] M. Duguleana, G. Mogan. Neural networks based reinforcement learning for mobile robots obstacle avoidance. Expert Syst. Appl, 2016,62:104-115.

[7] Y. Dai, Y. Kim, s. Wee, D. Lee, S. Lee. A switching formation strategy for obstacle avoidance of a multi-robot system based on robot priority model. ISA Trans, 2015,56:123-1

[8] H. Kim, D. Kim, J.-U. Shin, H. Kim, H. Myung. Angular rate-constrained path planning algorithm for unmanned surface vehicles. Ocean Eng,2014,84(6):37-44.

[9] O. Montiel, U. Orozco-Rosas, R. Sepulveda. Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. Expert Syst, 2015,42(12):5177-5191.

[10] H. Mousazadeh, H. Jafarbiglu, H. Abdolmaleki, et al. Developing a navigation, guidance and obstacle avoidance algorithm for an unmanned surface vehicle (USV) by algorithms fusion. Ocean Eng, 2018,159(6):56-65.

[11] X. Fei. Path planning and obstacle avoidance problem research based on the unmanned ship. Harbin Eng. Univ, 2017.

[12] M.-C. Tsou. Multi-target collision avoidance route planning under an ECDIS framework. Ocean Eng,2016,121:268-278.

[13] S. MahmouZadeh, A. M. Yazdani, K. Sammut, D. M. W. Powers. Online path planning for AUV rendezvous in dynamic cluttered undersea environment using evolutionary algorithms. Appl. Soft Comput, 2018,70:929-945.

[14] Mnih, V., Kavukcuoglu, K., Silver, D. et al. Human-level control through deep reinforcement learning. Nature 518, 529–533 (2015).

[15] Low E S, Ong P, Cheah K C. Solving the optimal path planning of a mobile robot using improved Q-learning. Robotics and Autonomous Systems, 2019, 115: 143-161.