Proceedings of The 2020 IEEE International
Conference on Real-time Computing and Robotics
September 28-29, 2020

Research on Virtual Path Planning Based on Improved DQN

Cheng Yi[1], Meng Qi[2]

Abstract— **An end-to-end approach based on the theory of Deep Reinforcement Learning has been proven to be able to meet or exceed human-level strategic capabilities. Applying this learning algorithm to path planning methods can make robots self-contained learning ability and environment interaction ability, and increased generalization ability. In this paper, Deep Q Network (DQN) as the typical Deep Reinforcement Learning method is improved. Improvement points can be divided into two steps. Firstly, the two steps of the selection of actions in the current network and how to calculate the target Q value are decoupled to eliminate overestimation caused by the rapid optimization of Q value in the possible direction. Then, considering that the action value function can bring benefits in addition to the action with the greatest value made by the agent, the static environment also has certain influence, the final result is a linear combination of two parts, which is to estimate the value functions of the upper, lower, left and right actions of the neural network output and the value of the environment state itself. Under the same experimental conditions, the improved DQN network is compared with the original DQN network, the result shows that the estimated final target value function of improved DQN network is more accurate and effective for virtual path planning tasks.**

I. Introduction

As an important part of robot visual navigation, path path planning has been a hot topic in robotics research for a long time. It refers to the intelligent robot in a static or dynamic environment select an optimal path from the specified point to the target point, at the same time not touch any obstacles in this way [1]. According to the path planning in the known environment or the unknown environment, the robot path planning is divided into global path planning [2] and local path planning [3]. Global path planning can find the optimal solution, but it needs to know the accurate information of the environment in advance. When the environment changes, or suddenly unknown obstacles are emerged, it can't deal with the sudden event, so the real-time ability of global path planning is poor [4]. Local path planning is planned when the environmental information is completely unknown or partially known [5]. The local environmental information of the robot is relatively important, it needs to collect environmental data, detect the working environment of the robot through various laser sensors or depth sensors, etc. to obtain information such as the position geometric properties of the obstacle and the dynamic environmental model can be updated [6]. The local planning method requires the robot system has high-speed information processing, computing power and high robustness to environmental errors and noise, providing real-time feedback and correction of the planning results [7].

In the direction of global path planning, the typical methods are Artificial Potential Field Method, Grid Method [8], A* algorithm [9], Visibility Graph Method [10], Free Space Method, Topological Method, Rapidly-Exploring Random Tree (RRT) Algorithm, Genetic Algorithm, and so on. For local path planning, there are intelligent algorithms, such as Artificial Potential Field Method, Fuzzy Rules and Simulated Annealing, Genetic Algorithm [11], Ant Colony Algorithm [12] and Artificial Neural Network. The path planning algorithm should have the ability to make quick decisions under the constantly changing or unknown complex environment.

The classic approaches have limitations, such as slow convergence speed, as well as falling into local optimization. New methods or combined classical methods with human intelligence methods have been proposed. Pfeiffer and Mark proposed an end-to-end learning model [13] that uses convolutional neural networks (CNN) to analyze laser information, then uses the A* algorithm as the marker information to conduct supervised learning, but this method is strongly dependent on the marking algorithm, and the generalization ability for environmental changes is weak. In recent years, with the rapid development of Deep Learning and Reinforcement Learning, applying the powerful perception ability of Deep Learning and the decision-making ability of Reinforcement Learning to path planning has become a new research hot topic [14], and gradually become the main technology in path planning research. Reinforcement Learning is the Q-learning method, which enables agent to express Q (s, a) by trial and error in the environment without any prior knowledge, and then establish a table by value iteration and other methods to store the mapping of state (s) and action (a). In this way, when the robot is in any position, the action of the next moment can be quickly determined, but it is very weak to build such a table in an unknown environment or complex environment for taking up a lot of

*This work is supported by Tianjin Natural Science Foundation Project (18JCYBJC88400); Tianjin Natural Science Foundation Project (18JCYBJC88300); Tianjin University Innovation Team Training Program Funding Project (TD13-5036).

Cheng Yi is with the School of Electrical Engineering and Automation, Tiangong University, Tianjin 300387 CHINA(E-mail:chengyi@tjpu.edu.cn).

Meng Qi is with the School of Electrical Engineering and Automation, Tiangong University, Tianjin 300387 CHINA(corresponding author to provide phone: 13752705837; E-mail:1281440938@qq.com).

memory [15]. Therefore, a deep neural network can be used to represent a function to approximate Q(s, a). It can acquire knowledge and adjust its own actions autonomously, and realize obstacle avoidance in path planning. The advantage of Deep Reinforcement Learning is that it does not rely on the trajectory of manual marking, it only needs to set the target, for example, in order to achieve the optimal strategy, let the robot keep trying to update the iterative network before reaching the destination.

When the Reinforcement Learning is used in the complex situation, the used states will become a discrete infinite set of states. As the size of the state set of the problem increases, a feasible modeling method is the approximate representation of the value function, which respectively introduces a state value function and an action value function. It means that the value function is described by a parameter, and the value of the state and the action is calculated, that is, the expectation. Among all the methods used to express the function approximation, such as Linear Representation, Decision Tree, Nearest Neighbor, Fourier Transform, and Neural Network, the most widely used expression method is Neural Network.

Typical Deep Reinforcement Learning methods are Deep Q Network (DQN), Actor Critic, Sarsa, etc. and their variants. In recent years, some people have tried to apply DQN algorithm to path planning task, and found that DQN can solve this kind of problem very well, which shows great potential.

The DQN algorithm has good function approximation ability and unknown state learning ability, but in function approximation of the Q value are obtained by neural network to the optimization of the best action in fast inevitably exists the disadvantage of overestimate, and the absence of an action of static environment also has a great influence on the final action value function. In view of the above situation, the DQN algorithm is improved in this paper. From the aspects of optimize the network architecture and optimize target Q value calculation, the main contributions of this paper are described as follows. Firstly, the two steps of the selection of actions in the current network and how to calculate the target Q value are decoupled. The action selected by the maximum Q value calculated in the current Q network is used to calculate the target Q value in the target network. Secondly, in order to fully consider the impact of environmental factors on the final direction selection, divide each network into the maximum value of actions agent and the benefits of environment static state the final result is a linear combination of two parts. In addition, the other overall structure of this paper is as follows. In Section II, the theories related to traditional deep reinforcement learning and typical algorithm DQN are introduced. In Section III, an effective improved algorithm and network structure are proposed. The fourth section verifies the advantages of the proposed method through a simple virtual path experiment. The last part of the article summarizes this article and discusses the tasks that need to be done next.

II. Basic Theory

A. Reinforcement Learning

In particular, robots are regarded as agents in Reinforcement Learning to perceive the state of unfamiliar environment and reward feedback, and to learn and make decisions. Decision making is the state that the agent perceives in the environment and then makes an action. Learning is a strategy of constantly updating based on rewards. The interaction with the agent is the environment, which creates a new state during the interaction, while the environment gives rewards. In the constant interaction between the agent and the environment. New data will be obtained, and the strategy will continue to be iterated. Finally, the agent will learn the action strategy that meets people's requirements.

B. Markov Decision Process

The path planning problem of an intelligent robot is a typical Markov Decision Process (MDP). The sequence $s_0, s_1, s_2, \cdots s_t \in S$ of random variables with Markov property, the state s_{t+1} of the next moment depends only on the current state s_t. Adding a variable to the Markov process, action a, the next state s_{t+1} is related to the current state s_t and action a_t

$$p(s_{t+1} \mid s_t, a_t, \cdots, s_0, a_0) = p(s_{t+1} \mid s_t, a_t) \tag{1}$$

In the formula (1), $p(s_{t+1} \mid s_t, a_t)$ is the state transition probability, which is the probability that the agent will change to state s_{t+1} in the next moment after making an action a_t according to state s_t. If the given strategy is $\pi(a \mid s)$, then a trajectory is formed in the Markov Decision Process

$$\tau = s_0, a_0, s_1, r_1, a_1, \cdots s_{T-1}, a_{T-1}, s_T, r_T \tag{2}$$

The probability of formula (2) is

$$p(\tau) = p(s_0) \prod_{t=0}^{T-1} \pi(a_t \mid t_t) p(s_{t+1} \mid s_t, a_t) \tag{3}$$

The cumulative reward received by track τ, that is, the discount received is

$$G(\tau) = \sum_{t=0}^{T-1} \gamma^t r_{t+1} \tag{4}$$

In the formula (4), $\gamma \in [0,1]$ is the discount factor used to reduce the impact of the return of the subsequent state on the current state measurement.

C. Q-function

For a strategy π, starting from state s, the expected total return from executing the strategy is

$$\mathrm{V}^{\pi}(s) = E_{a\sim\pi(a|s)}E_{s'\sim p(s'|s,a)} [r(s,a,s') + \gamma V^{\pi}(s')] \tag{5}$$

The above formula is the Bellman Equation, also called the Dynamic Programming Equation, where $\mathrm{V}^{\pi}(s)$ is the evaluation function, and the above equation shows that the evaluation function of the state s can be derived from the estimation function of s'.

Another expectation in the formula (5) is the state-action value function, which is the Q function

$$\mathrm{Q}^{\pi}(s,a) = E_{s'\sim p(s'|s,a)}[r(s,a,s') + \gamma V^{\pi}(s')] \tag{6}$$

Combining the above formulas (5) and formulas (6) to get the Q function can also be written as

$$\mathrm{Q}^{\pi}(s,a) = E_{s'\sim p(s'|s,a)}[r(s,a,s') + \gamma E_{a'\sim\pi(a'|s')}[Q^{\pi}(s',a')]] \tag{7}$$

If in state s, an action a is performed to make $\mathrm{Q}^{\pi}(s,a) > V^{\pi}(s)$, it means that the effect of performing action a is better than the strategy $\pi(a\,|\,s)$, and the parameters need to be adjusted to increase the probability of the strategy.

D. DQN

The simplest case when calculating the Q function is that all states are discrete, so that all states and actions can be enumerated to list the Q matrix. However, in dealing with the complex problem of path planning, the information that the agent perceives according to various sensors is continuous and unknown, so it is difficult to list a complete Q matrix, so Mnih V et al. proposed combining the convolutional neural network with the Q-learning algorithm in traditional Reinforcement Learning [16]. The function of deep neural network is approximate representative estimation function, the function of Q-learning is to provide target values for deep neurons, so that the network is continuously updated until convergence, so that the agent can perceive complex unknown state of the environment and build more complex strategies. Deep Q Network (DQN) is a Deep Reinforce Learning algorithm that is currently mainstream and widely used.

The neural network has the advantage of taking the action and state as input, and then obtaining the Q value of the action after analysis by the neural network, or only taking the state value as input, outputting all actions, and then using the Q function to select the maximum value. The action will be executed as the next step. The network update process is shown in Fig.1.

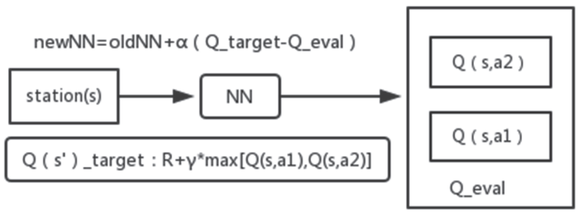
newNN=oldNN+α (Q_target-Q_eval)
station(s)
NN
Q (s') _target : R+γ*max[Q(s,a1),Q(s,a2)]
Q (s,a2)
Q (s,a1)
Q_eval

Figure 1. DQN network update process

In addition to the network update, DQN has two key factors: the experience pool and the fixed target value network. The experience pool is used to randomly extract the previous experience, the fixed target value networks are also used to disrupt data correlation, which breaks the correlation between experiences and makes neural network updates more efficient.

III. Improved Algorithm Based on DQN

At present, various algorithms based on Deep Reinforcement Learning have been proved to be human-achievable at the Atari 2600 game and can even surpass humans [17]. This paper uses Tkinter module in Python to design a virtual map with starting point, target point and obstacles as the environment of verification algorithm, the agent has four actions of up, down, left and right. The agent takes its own state observation as the input of the network, and selects the optimal strategy based on the detection strategy, and makes the action to reach the next state observation. There are two important steps in the algorithm to break the correlation of interval training. Firstly, the agent stores the observations, actions, rewards and the next observation value $<s,a,r,s'>$ in the memory when performing the action, and stores the new $<s,a,r,s'>$ in the memory in a certain number of steps, the data is randomly selected from a fixed batch of samples for training with a certain probability, another method of breaking the correlation is to design another identical fixed target value network, but the output Q value of this network is updated more slowly than the previous one, and it is necessary to use the predicted Q value of the previous network, then uses the previous neural network to estimate and select the maximum Q value in the network. This approach improves the stability of the training and interrupts the correlation between the data, making the model easier to converge.

In the DQN network, there are two Q networks with the same structure. On this basis, this paper also eliminates the problem of overestimation by decoupling the two steps of the selection of actions in the current network and target Q value calculation, as follows.In the previous algorithm, it only needs in the target Q network to find the corresponding Q value of all actions. Here, we do two steps, the first is to find the corresponding action of the maximum Q value in the current Q network, that is

$$\mathrm{a}^{\max}(S_j', \omega) = \arg\max_{a'} Q(\theta(S_j'), a, \omega) \tag{8}$$

Then use this selected action to calculate the target Q value in the target network, that is

$$y_j = R_j + \gamma Q'(\theta(S'_j), a^{\max}(S'_j,\omega),\omega') \tag{9}$$

The predicted Q value is calculated by the following formula (10)

$$Q(s,a|\theta) = V(s,a) + (A(s,a|\theta) - \arg\_a(A(s,a'|\theta))) \tag{10}$$

That is, the value corresponding to each state plus the advantage value of each action in the current state.

The target Q value is calculated as follows

$$q\_next = r' + \gamma\ Q(s', \arg\max_a Q(s',a|\theta')|\theta'^{-}) \tag{11}$$

In the formula (11), θ' is applied to choose proper actions of the Q-value, The corresponding θ'^{-} is applied to evaluate the Q-value of the optimal movement. These two parameters are derived from the predictive network and the target value network. The purpose of adding these two parameters is to calculate the movement selection and policy assessment separately, which helps to improve the over estimation problem.

Calculating the mean squared loss function of the target Q value and the predicted Q value

$$Loss = E[(r' + \gamma \max Q_{\text{target}}(s', \arg\max_a(Q(s',a))) - Q(s,a|\theta))^2] \tag{12}$$

In the formula (12), r' is the reward, γ is the learning rate, and θ is the network parameter, and all parameters in the network are updated by the gradient back propagation of the neural network. Making the predicted Q value keep approaching q_target. The overall training model is shown in Fig.2.

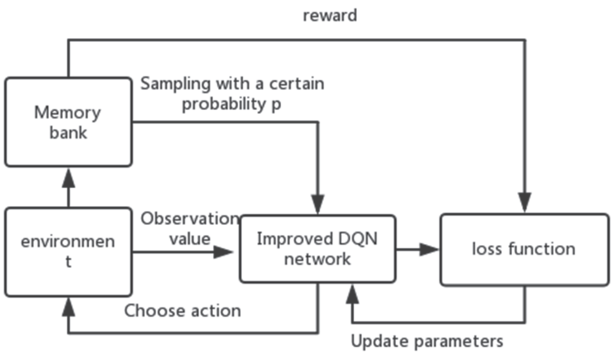

Figure 2. Training model of this paper

Improved DQN network partial structure is shown in Fig.3.

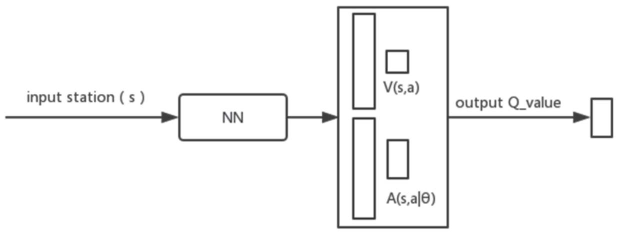

Figure 3. Improve the structure of DQN network

IV. Analysis of Results

A. Algorithm Parameters and Rewards Settings

The experimental environment is the Windows system CPU server, tensorflow, Python 3.5. The tkinter module of Python is used to design a research environment with a two-dimensional grid as a virtual path, and the environment of the robot is decomposed into small grids, each grid representing a state. The size of the grid model is 250*250, the minimum moving unit is 50, and the state space is 5*5. The state of the experimental environment is shown in Fig.4. Some experimental algorithm parameters and reward number settings are shown in table I and table II.

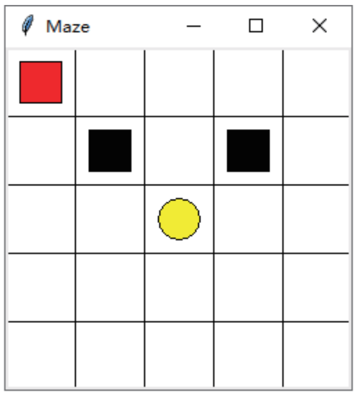

Figure 4. Simulation environment

The intelligent agent is represented by a red square, and its actions are up, down, left and right. The black square is the obstacle, and the target point is a yellow circle.

TABLE I. Algorithm Parameters

Parameter	Meaning	Size
n_actions	Number of actions	4
n_features	Number of features	2
learning_rate	Learning rate	0.9
gamma	Reward attenuation rate	0.9
memory_size	Memory size	500
replace_memory	Replace memory steps	200
batch_size	Mini-batch size	32

TABLE II. REWARDS SETTINGS

Scene	Numerical value
Yellow	+1
Black	-1
Space	0

Fig.5 is a state of the agent in the environment iteration process, and Fig.6 is the final point reached by the agent. It shows that the agent can learn from itself in the process of continuous interaction with the environment, and finally make the agent reach the final position.

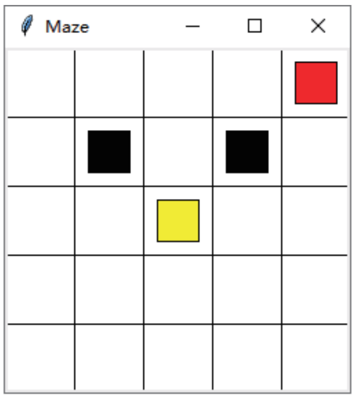

Figure 5. Simulation process

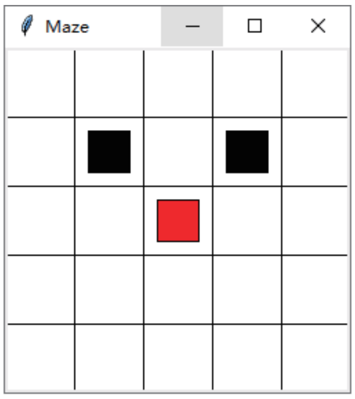

Figure 6. Experiment results

B. *Result Analysis*

The cost curve in the learning process is shown in Fig.7 and Fig.8. Since this is different from the general supervised learning, as the training process progresses, the agent continuously explores and obtains new data, and the error may suddenly appear high, therefore, the curve in the graph does not fall smoothly. It can be seen from the figure that as the number of iteration steps gradually increases, the network gradually converges, which indicates that the path planned by the agent tends to be more and more optimal in the process of environment iteration. The convergence rate of the method used in this paper is obviously better than that of the original DQN algorithm

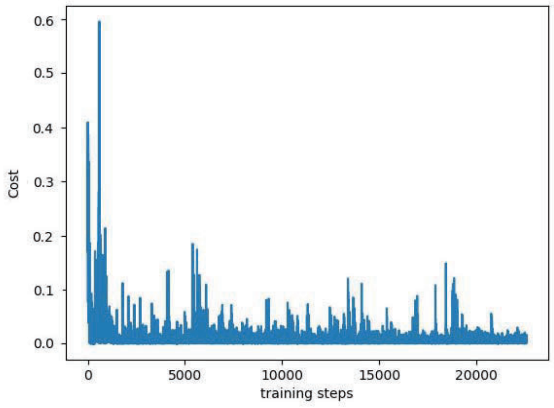

Figure 7. Cost curve of DQN algorithm

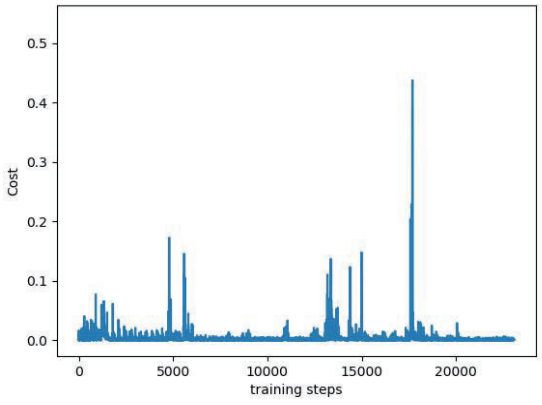

Figure 8. Cost curve of Improved DQN algorithm

Comparing the values of the original DQN network with the Q estimate of the improved DQN network, as shown in Fig.9. It can be seen from the whole that the improved network improves the original DQN overestimated Q value, the improved approach has more obvious advantages. Therefore, the self-learning ability and algorithm adaptability of the method described in this paper are more effective.

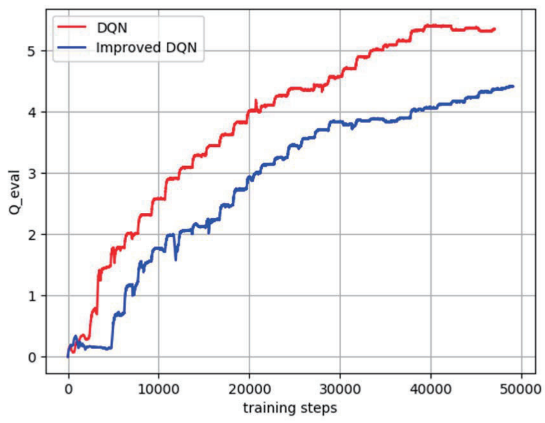

Figure 9. Q estimate value comparison

V. CONCLUSION

In this paper, the DQN algorithm of Deep Reinforcement Learning is improved. The advantages of the DQN algorithm experience pool and the fixed target network are combined, the method of preserving two networks with the same structure of the original DQN network, and the calculation method of the target network and the output of the network Q value are improved on the basis of making the network converge faster and reduce over-estimated Q value. The previous research only focused on the value of the action obtained after the response to the choice. Here, the benefits of the static environment without the selected action for the final action value function were added. The effectiveness and adaptability of this method compared with the original DQN algorithm are verified under the same experimental simulation environment. The future work is to apply this algorithm to the field of robotic autonomous navigation to achieve autonomous control of robots in complex environments.

ACKNOWLEDGMENT

At the completion of this thesis, I thank the experimental environment provided by visual navigation lab and the teachers and visual navigation team who have guided me for a long time. Thank you for your contributions and efforts. I sincerely thank all the people who care and help me.

REFERENCES

[1] Koubaa A, Bennaceur H, Chaari I, et al. Introduction to Mobile Robot Path Planning[M]//Robot Path Planning and Cooperation. Springer , Cham , 2018: 3-12.

[2] Mac T T, Copot C, Tran D T, et al. A Hierarchical Global Path Planning Approach for Mobile Robots based on Multi-Objective Particle Swarm Optimization[J]. Applied Soft Computing, 2017.59:68-76.

[3] HAN J, SEO Y. mobile robot path planning with surrounding point set and path improvement[J]. Applied Soft Computing, 2017,57:35-47.

[4] Shi E, Huang Y, Zhu C, et al. A Novel Method of Planning Path for DDWMR[J]. China Mechanical Engineering, 2012, 23(23):2805-2809.

[5] Henkel C, Bubeck A, Xu W. Energy Efficient Dynamic Window Approach for Local Path Planning in Mobile Service Robotics *[J]. 2016, 49(15):32-37.

[6] Chu K , Lee M , Sunwoo M . Local Path Planning for Off-Road Autonomous Driving With Avoidance of Static Obstacles[J]. Intelligent Transportation Systems, IEEE Transactions on, 2012, 13(4):p.1599-1616.

[7] Wu Y, Qu X J. Path planning for taxi of carrier aircraft launching[J]. Science China. Technological Sciences, 2013, 56(6):1561-1570.

[8] Wang D. Indoor mobile-robot path planning based on an improved A* algorithm[J]. Journal of Tsinghua University (Science and Technology), 2012, 52(8):1085-1089.

[9] Knuth D E. A generalization of Dijkstra\"s algorithm[J]. 1977, 6(1):1-5.

[10] Tran N, Nguyen D T, Vu D L, et al. Global path planning for autonomous robots using modified visibility-graph[C]// 2013 International Conference on Control, Automation and Information Sciences (ICCAIS). IEEE, 2013:317-321.

[11] Tuncer A, Yildirim M. Dynamic path planning of mobile robots with improved genetic algorithm[J]. Computers & Electrical Engineering, 2012, 38(6):1564-1572.

[12] Fatemeh Khosravi Purian, Ehsan Sadeghian. Mobile robots path planning using ant colony optimization and Fuzzy Logic algorithms in unknown dynamic environments[C]// 2013 International Conference on Control, Automation, Robotics and Embedded Systems (CARE). IEEE, 2013:1-6.

[13] Pfeiffer M, Schaeuble M, Nieto J, et al. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots[C]// 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017.

[14] Tai L, Paolo G, Liu M. Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation[J]. 2017:31-36.

[15] Jaradat M A K, Al-Rousan M, Quadan L. Reinforcement based mobile robot navigation in dynamic environment[J]. Robotics and Computer-Integrated Manufacturing, 2011, 27(1):135-149.

[16] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning.[J]. Nature, 2015, 518(7540):529.

[17] Van H H, Guez A, Silver D. Deep reinforcement learning With double Q-learning[J]. Computer Science，2015.