UAV Path Planning Based on Deep Reinforcement Learning

Yifan Guo
School of Computer Science & Engineering
Xi'an Technological University
Xi'an, China
E-mail: 958782423@qq.com

Zhiping Liu
School of Computer Science & Engineering
Xi'an Technological University
Xi'an, China
E-mail: leopard@xatu.edu.cn

***Abstract*—Path planning is one of the very important aspects of UAV navigation control, which refers to the UAV searching for an optimal or near-optimal route from the starting point to the end point according to the performance indexes such as time, distance, et al. The path planning problem has a long history and has more abundant algorithms. The path planning problem has a long history and a rich set of algorithms, but most of the current algorithms require a known environment, however, in most cases, the environment model is difficult to describe and obtain, and the algorithms perform less satisfactorily. To address the above problems, this paper proposes a UAV path planning method based on deep reinforcement learning algorithm. Based on the OpenAI-GYM architecture, a 3D map environment model is constructed, with the map grid as the state set and 26 actions as the action set, which does not need an environment model and relies on its own interaction with the environment to complete the path planning task. The algorithm is based on stochastic process theory, modeling the path planning problem as a Markov Decision Process (MDP), fitting the UAV path planning decision function and state-action function, and designing the DQN algorithm model according to the state space, action space and network structure. The algorithm enables the intelligences to carry out strategy iteration efficiently. Through simulation, the DQN algorithm is verified to avoid obstacles and complete the path planning task in only about 160 rounds, which validates the effectiveness of the proposed path planning algorithm.**

Keywords-Component; UAV; Path Planning; DQN; Deep Reinforcement Learning

I. INTRODUCTION

In recent years, unmanned aerial vehicles (UAVs) have been used in a wide variety of fields and nowadays, UAVs are always used to collect data for humans in dangerous places. Quadrotor UAVs, as a type of UAV, have a wide range of applications (due to their hovering capabilities and low condition takeoff and landing capabilities) such as search and rescue, aerial photography, environmental monitoring, industrial inspection, et al [1]. Regardless of the mission, autonomous path planning is always the key to accomplish the task.

Traditional path planning methods are relatively mature, such as Dijkstra's method, A* algorithm, D* algorithm, and artificial potential field method, et al [2]. These methods have been widely used in some scenarios, but as the difficulty of the task to be performed increases, especially in unknown environments, the path planning problem becomes more complex and increases the uncertainty, from the existing path planning methods, each method has its own advantages and shortcomings, and none of them has the ability to learn, and also does not have the ability to cope with the changes in the environment and the uncertainty.

In recent years, artificial intelligence and machine learning (ML) have been widely used in the field of robotics, and reinforcement learning (RL) is even more with the improvement of algorithms and theories, with the ability to apply its latest theoretical achievements to the actual control of robotic systems. Based on the theoretical foundation of reinforcement learning, this paper proposes a deep reinforcement learning (DRL)-based UAV path planning method different from the traditional method, which can make the UAV obtain human-like learning ability, and in

DOI: 10.2478/ijanmc-2023-0068

the task with high difficulty coefficient, unknown environment, complex and uncertain factors, it can cope with unexpected situations and complex terrain to a considerable extent and complete the task. A 3D map environment model is constructed based on the OpenAI-GYM architecture, with the map grid as the state set and 26 actions as the action set, which does not need an environment model and relies on its own interaction with the environment to accomplish the path planning task. The algorithm is based on stochastic process theory, modeling the path planning problem as a MDP, fitting the UAV path planning decision function and state-action function, and designing a DQN algorithm model based on the state space, action space and network structure. The algorithm enables the intelligences to carry out strategy iteration efficiently. Simulation experiments demonstrate that the DQN algorithm can avoid obstacles and realize the path planning task in only a small number of rounds, proving the effectiveness of the proposed algorithm.

II. RELATED WORKS

The path planning problem has a long history, and scholars at home and abroad have made a large number of research results. Path planning is one of the very important aspects of UAV navigation control, which means that the UAV searches for an optimal or near-optimal route from the starting point to the end point according to time, distance, and other performance metrics. Its essence can be treated as a conditional optimization problem, and the optimization index will be different in the face of different demands. For different task requirements, scholars have proposed a large number of methods from different angles and different fields.

Traditional path planning methods are relatively mature and have been widely used, with the continuous completion of reinforcement learning theory and the vigorous development of methods, many traditional problems can be solved under the framework of reinforcement learning, and the path planning problem is one of them [3]. RL, as an important research direction in the areas of ML, optimizes action strategies based on the interaction between an intelligent body and its environment, and scholars have been paying more and more attention to this direction in recent years. Different from traditional methods, reinforcement learning enables agents to learn optimal strategies autonomously and maximize cumulative rewards through trial-and-error interactions with the environment [4]. As for deep reinforcement learning, which is the use of deep neural networks to solve the problem model of reinforcement learning, the integration of the two techniques can solve the problems of high storage capacity requirements for high-dimensional state and action space and complex model training, which were difficult to deal with in the past. Deep reinforcement learning is a technique that empowers intelligences to learn by themselves, so this method is very suitable for tackling the path planning problem of UAVs.

In terms of solving the path planning problem, Faust et al [5] used preference estimation to realize the path planning of robots in moving obstacle environments while solving the "dimensional catastrophe" problem, and Jaradat et al [6] proposed a new method to define the state space based on the Q-learning algorithm to reduce the number of states in dynamic environments, which effectively reduces the number of dimensions of Q-tables and improves the system learning efficiency. Method to reduce the number of states in a dynamic environment, which effectively reduces the dimension of the Q-table, accelerates the convergence speed, and improves the learning efficiency of the system. Shammah et al [7] combined deep learning with reinforcement learning methods and applied the policy gradient method, which successfully solved the path planning problem in automated driving, and improved the efficiency of the path planning problem. Wang et al [8] combined the Q-learning algorithm with Sarsa's algorithm and proposed a reverse Q-learning algorithm, which can improve the convergence speed and learning efficiency of the system. Bianchi et al [9] proposed an accelerated Q-learning algorithm based on a heuristic strategy, i.e., a heuristic function is used to help decision-making during action selection, and experiments show that a simple heuristic function can improve the convergence speed and

computational overhead of the reinforcement learning algorithm.

III. Related theoretical studies

A. Definition of path planning

Trajectory planning is one of the very important technical aspects of UAV navigation control. It is a trajectory from the start point to the end point that can avoid obstacles and is short enough based on the map environment information within the mission area and satisfies the aircraft's own physical performance constraints [10].

There are two methods for UAVs to perform path planning. The first is pre-flight planning; this method also needs to scan the surrounding environment and establish the initial model in advance when considering the mission target point, and to fully consider the impact of factors such as surrounding obstacles and weather on the trajectory. The second is real-time planning of the flight path. When using this method for planning, although there are fewer integrated factors to consider, satisfying the real-time nature of the path, it is necessary for the aircraft load to have a powerful chip to process the surrounding environment and update the path at any time. Therefore it will greatly increase the cost of flight cost. In this paper, the first UAV trajectory design method using pre-flight planning is chosen. At present, the theoretical technology about the trajectory planning in the two-dimensional plane is relatively mature and there are many related literature, but many of these algorithms degree cannot meet the requirements of three-dimensional space and do not have the actual flight. Because path planning in three-dimensional space is closer to real-life applications, the problem will definitely become a focus of research in the future, but when the space is expanded to three-dimensional space, it is necessary to consider not only the theoretical planning algorithm of the trajectory, but also involves GPS, sensor technology, et al. The requirements for path planning for UAVs are as follows:

- Flight paths that achieve mission requirements.
- The trajectory is a continuous curve or line from the starting point of the aircraft to the target point.
- Try to ensure that the path of the UAV is optimal, that is, to achieve the purpose of the algorithm to plan the path with the least amount of time or the shortest designed path.

B. Markov Decision Process

In DRL, the main core still lies in the foundation of RL, through the ability of deep learning to extract features and complex mapping of functions, some pain points in reinforcement learning will be solved. In reinforcement learning, it is mainly expected that an intelligence learns a behavioral strategy and is able to output a behavior corresponding to the current environmental information:

$$a = \pi(s) \tag{1}$$

Where a is the action and s is the state of the intelligent body in the environment. Depending on the environment, the actions and states may be discrete or continuous.

The problem of reinforcement learning can be modeled as a MDP, which is characterized by a system in which the state at the next moment is related to the current state only and is independent of the previous state, and by a MDP in which the state at the next moment is determined by the current state and the action at the current moment. The MDP can be defined by a five-tuple $< S, A, R, P, \gamma >$. The tuple $< S, A, R, P, \gamma >$ consists of the set of states S, the set of actions A, the reward function $R(s,a)$, the transfer function $P(s' \mid s,a)$, and the discount factor $\gamma \in [0,1]$. In each state $s \in S$, the intelligence takes an action $a \in A$. After performing the action a in the environment, the intelligence receives the reward $R(s,a)$ and arrives at the new state s', determined by the probability distribution $P(s' \mid s,a)$ [11].

Solutions to dynamic programming problems with finite states and action spaces can be obtained by a variety of methods, especially with a given

transfer probability. However, in most MDPs, transfer probabilities or reward functions are not available. In this case, the intelligence needs to interact with the environment to obtain some internal information to solve the MDP, which is achieved by reinforcement learning methods. The core problem to be solved by the Markov decision process is the strategy of choosing an action, which is mathematically represented using the π function as follows:

$$\pi(a|s_0) = P(a_t = a|s_t = s_0) \quad (2)$$

C. Reinforcement learning algorithms

RL, also known as augmented learning, is a branch of ML [12]. The method differs from the common supervised learning and from unsupervised learning. RL is trained without explicit supervised data, i.e., the environment does not give explicit feedback that the action made by the intelligence at the current moment must be correct or incorrect; the environment simply helps the intelligence to learn by giving it a reward signal. Alternatively, instead of directly determining whether a state or action is good or bad, the environment assists the learning by giving a reward for a state or action. In the case of Go, for example, the current move is not directly judged as a correct or incorrect move, but a reward feedback is generated at the end of the whole game [13].

While the training data used in the traditional supervised and unsupervised learning training process are independent of each other, in RL the data are time-series related. The state generated by an intelligent body at each interaction with the environment influences the next move and the subsequent state.

RL is based on the principle that if an action taken by an intelligence in a state returns a positive reward to the environment, the tendency of the intelligence to take that action in that state will increase in the future. The basic model of RL, which views learning as a process in which an intelligence explores the learning environment, is shown in Figure 1. In RL, the intelligence selects an action a based on the current policy and passes it to the environment by observing the environment at the current moment, and the new state is generated based on the state transfer matrix after the action is executed in the environment. At the same time, the environment also transmits a reward signal r to the intelligent body, which adjusts its strategy by using the reward signal r to generate the next action according to its strategy and the current state of the environment [14].

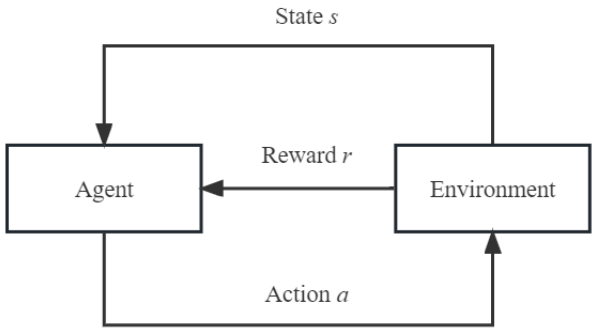

Figure 1. Reinforcement learning model

Reward r, is a scalar quantity that reflects the performance of the action produced by the intelligence in the environment performance in the environment, and the aim of the intelligence is to maximize the cumulative reward. RL is based on this hypothesis: the goal of all the goal of all problems can be expressed as maximizing the cumulative reward.

The intelligence and the environment, the RL problem can be described from both the perspective of the intelligence and the environment. At moment t, the perspective of the intelligent body is that it observes the state s_t of the environment, needs to make an action a_t at based on this state, and gets a reward signal r_t returned by the environment after executing this action in the environment. The perspective of the environment is that it receives the action a_t of the intelligent body to update the environment information, generates a new state, and gives a reward signal r_t back to the intelligent body.

State s, which can be divided into environment state, intelligences state, and information state. The environment state, which is the detailed description of the environment, including the data

of the next observation needed by the intelligence and the data of the reward signal r_t generated, et al., is usually not fully visible to the intelligence, that is, the intelligence sometimes does not know all the detailed data of the environment state, and even if sometimes the environment state is fully visible to the individual, these data may contain some data that are not needed by the intelligence. The intelligences state, which is the internal structure and parameters of the intelligence, includes all the data that the intelligence uses to determine future actions, such as network parameters, policy information, et al. The information state is all the useful data in the history, also known as Markov state.

From the above description it is shown that the state space S, action space A, the reward function R and the state transfer probability matrix P need to be designed in the RL algorithm. The state space is a characterization of the environment other than the intelligent body and is the basis of information provided to the intelligent body for selecting actions. The action space describes the decision details of the intelligent body. The reward function is the benefit of performing an action in a certain state, which is the evaluation of the goodness of the performed action, and is the most central part of RL. A good reward function often helps RL algorithms converge quickly and achieve good results and learn the best strategies.

D. Deep Reinforcement Learning

Deep reinforcement learning (DRL) was proposed because RL deals with problems with small sample spaces and discrete actions, while it does not work well for the kind of problems with huge state spaces and continuous actions [15]. Therefore, it is difficult to find good strategies in large state spaces using RL. DRL introduces neural networks in deep learning based on RL, combining both the technique of giving rewards based on actions in RL and the idea of using neural networks to learn feature representations by processing some high-dimensional state data through neural networks, training deep neural networks and learning the underlying features of the input data to approximate any nonlinear function. Its powerful representation capability allows another breakthrough in the integration of RL with deep neural networks.

The basic model of DRL is shown in Figure 2.

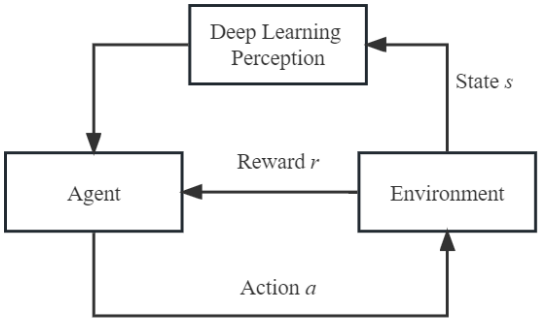

Figure 2. Basic model of deep reinforcement learning

IV. METHODS AND MATERIAL

The DQN algorithm is based on deep learning and Q-learning algorithms, i.e., it combines the advantages of both the feature-awareness capability of deep learning and the trial-and-error-learning capability of Q-learning algorithms. Based on the Q-learning algorithm, in order to overcome the shortcomings of using Q-tables that occupy a large amount of space and the inefficiency of updating in a high-dimensional state space, DQN is able to compute the action value function $Q(s,a)$ in a huge state and action space, and the DQN algorithm uses $Q(s,a,\theta)$ to approximate the optimal action value function as follows:

$$Q(s,a,\theta)=Q(s,a) \tag{3}$$

Where $Q(s,a,\theta)$ is a deep neural network (DNN) with parameters θ, called Q-network.

When using the temporal difference (TD) method, $Q(s,a,\theta)$ is used to approximate $E\left[r+\gamma \max_{a'} Q(s',a',\theta)\right]$, and in the DQN algorithm, the set of loss functions of Eq. (4) is used as the optimization objective of the current network, and the gradient descent method is used to solve for its weights θ.

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

However, the current Q value and the target Q value in Eq. (5) both use parameters of the same shape θ and are updated simultaneously, making the model training unstable and difficult to converge. To solve this situation, the DQN algorithm uses the old network parameter θ' to evaluate the state Q value at the next time step, and updates the parameter θ' every certain time step, providing a clearer reference target to the current Q network to be fitted in this way, and adjusting the optimization target to that shown in Equation (5).

$$L(\theta)=E\left[\left(r+\gamma\max_{a'}Q(s',a',\theta')-Q(s,a,\theta)\right)^2\right] \quad (5)$$

Therefore, the DQN algorithm contains two neural networks. As shown in Figure 3, the current network $Q(s,a,\theta)$ is used to evaluate the value function of the current state and action, and the target network $Q(s,a)$ is used to compute the temporal difference target. The algorithm updates the parameters θ for N rounds and then replicates them directly to θ'.

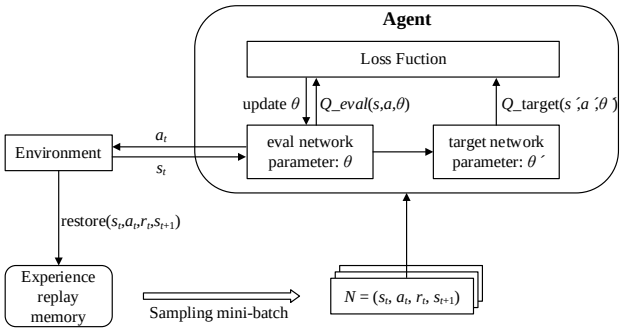

Figure 3. DQN algorithm structure diagram

In the training process of deep learning, it is usually required that the training data must satisfy the nature of independent identical distribution, and if the correlated data of RL is trained directly, it will lead to the difficulty of convergence of the model or the continuous riot of loss values. Based on this, the DQN algorithm proposes an experience replay mechanism, where the algorithm saves the current moment's state s_t , the action a_t generated by the intelligence, and the new state body s_{t+1} and reward r_t generated by the environment performing the action in the form of a tuple (s_t,a_t,r_t,a_{t+1}) into the experience pool, and randomly draws a specific batch of experience data from the experience pool for training during training, effectively removing the correlation and dependence between samples.

The detailed process of the DQN algorithm is shown in Algorithm 1. It can be seen that the algorithm mainly consists of two loop operations, the first loop is responsible for the replay of the empirical trajectory, which is executed M times; the second loop is responsible for iteratively traversing the empirical trajectory with time steps, T being the termination time step, while updating the parameters of the prediction model using the gradient descent method based on small batch sample data. In the second loop, the algorithm copies the model parameters p of the current network to the target network k at every C steps to achieve the update of the target network model parameters. Accordingly, the algorithm performs continuous loop computation to continuously update the network parameters and learn the update strategy from historical experience until convergence to a relatively stable state.

Algorithm 1 DQN.

Input: replay memory D, initial current network parameters θ , initial target network parameters θ' , Update Frequency C.

1. **for** episode = 1 to M **do**
2. Initialize the state s_0
3. **for** t = 1 to T **do**
4. Select $a_t=\max_a Q(s_t,a,\theta)$
5. Execute the action a
6. Get the next state s_{t+1}
7. Get the reward value r
8. Store D = $(s_t,a_t,r_t,s_{t+1},is\_end_t)$
9. Sampling mini-batch in D
10. Calculate y_j
11. **if** is_end_t = true **then** $y_j=r_j$
12. **else** $y_j=r_j+\gamma\max_{a'}Q^*(s_{j+1},a_j,\theta')$

13. **end if**
14. Update $\theta = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - Q\left(s_j, a_j, \theta\right)\right)^2$
15. Update $s_t = s_{t+1}$
16. Every C step copy θ to θ'
17. **end for**
18. **end for**

V. RESULTS AND DISCUSSION

In the path planning process, the DQN algorithm is studied and analyzed using the raster method to define the environment model. The simulation study in this paper is conducted in a 3D simulation environment, and to ensure the planning accuracy and at the same time ensure the safety of the UAV's body, the raster size is set to be exactly the same as the length of the UAV's outer contour side, and the following assumption is made: when the quadrotor UAV performs diagonal motion, its collision with the obstacle boundary is not considered. The path planning results using the DQN method are shown in Figure 4.

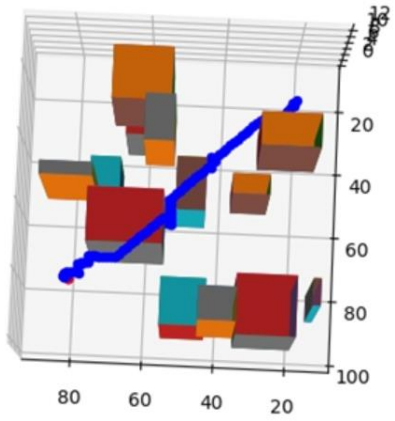

Figure 4. DQN algorithm learning results

The number of neurons in the input layer of the deep neural networks (DNN) is consistent with the size of the state of each time slot, and the number of neurons in the output layer of the network is consistent with the size of the action space. The parameters used in the DQN algorithm in this chapter are shown in Table 1. As can be seen in Figure 5, after 160 experiments, the DQN method was able to solve the path from the starting point to the end point and successfully bypassed all obstacles with a time step of 84.

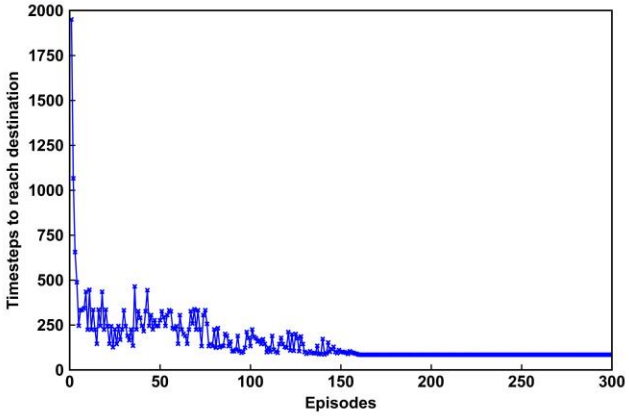

Figure 5. DQN algorithm learning results

TABLE I. DQN SIMULATION PARAMETERS

Parameters	Value
Size of the replay memory pool D	5000
Size of Mini-batch	64
Discount Factor γ	0.9
Initial exploration rate ε	0.9
Learning Rate	0.0002
Neural network activation function	ReLU

VI. CONCLUSIONS

In recent years, ML has been widely used in the field of robotics, and with the improvement of RL algorithms and theories, it has the possibility of being applied to robotic systems. Distinguished from traditional methods, RL theory applied to the field of robot path planning, can enable the robot to obtain human-like learning ability, in the higher difficulty coefficient, the environment is unknown, complex and uncertain factors in the task, to a considerable extent, can cope with the unexpected situation and complex terrain, to complete the task. And DRL that combines DNN to solve the problem model of RL, after the fusion of the two technologies can cover the traditional algorithms are difficult to deal with the high-dimensional state and action space under the high storage capacity requirements, the model training complexity and other issues. DRL is a technology that empowers intelligences with self-learning capabilities, so such methods are very suitable for solving the problems faced by traditional UAV path planning algorithms. Therefore, a UAV path planning

method based on DQN algorithm is proposed in this paper.

This paper systematically introduces the model, principle and its main components of the RL method, and also includes the Markov decision process, an important description method of the RL method. The path planning task is realized based on the deep reinforcement learning method. Based on the OpenAI-GYM architecture, the 3D environment is described by the raster method and a raster-based map is built. The DQN algorithm is used to solve the optimal policy by taking each raster as a state, the movements in 26 directions as action sets, and the action value function composed of state-action pairs as the objective function. Through simulation, the DQN algorithm is verified to avoid obstacles and complete the path planning task in only about 160 rounds, which validates the effectiveness of the proposed path planning algorithm.

In this paper, although the DQN method is used to realize the path planning task, but there are still some problems and shortcomings when applied to the actual system. The maps in this paper are discrete maps based on grids, which means that the state space and action space are discrete and finite, and the discrete states and actions determine the upper limit of the shortest paths planned, i.e., they cannot be moved at any angle within the map, and the absolute shortest paths cannot be obtained. At the same time, after simulation experiments, it appears that the sample efficiency of this model algorithm is poor, and the next step is to improve the sample efficiency by combining the off-policy method.

REFERENCES

[1] Cui Z, Wang Y. UAV Path Planning Based on Multi-Layer Reinforcement Learning Technique [J]. IEEE Access, 2021: 59486-59497.

[2] Qadir Z, Zafar M H, MOOSAVI S K R, et al. Autonomous UAV path planning optimization using Metaheuristic approach for pre-disaster assessment [J]. IEEE Internet of Things Journal, 2022: 12505-12514.

[3] Wu R, Gu F, Liu H L, et al. UAV Path Planning Based on Multicritic-Delayed Deep Deterministic Policy Gradient [J]. Wireless Communications and Mobile Computing, 2022: 1-12.

[4] Yan C, Xiang X, Wang C. Towards Real-Time Path Planning through Deep Reinforcement Learning for a UAV in Dynamic Environments [J]. Journal of Intelligent & Robotic Systems, 2020: 297-309.

[5] Faust A, Chiang H T, Rackley N, et al. Avoiding moving obstacles with stochastic hybrid dynamics using PEARL: PrEference Appraisal Reinforcement Learning [C]//2016 IEEE International Conference on Robotics and Automation (ICRA), Stockholm, Sweden. 2016.

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

[7] Shalev-Shwartz S, Shammah S, Shashua A. Safe, Multi-Agent, Reinforcement Learning for Autonomous Driving [J]. arXiv:1610.03295v1, 2016.

[8] Wang Y H, Li T H S, Lin C J. Backward Q-learning: The combination of Sarsa algorithm and Q-learning [J]. Engineering Applications of Artificial Intelligence, 2013, 26(9): 2184–2193.

[9] Bianchi R A, Martins M F, Ribeiro C H, et al. Heuristically-accelerated multiagent reinforcement learning.[J]. IEEE Transactions on Cybernetics, 2014, 44(2): 252–265.

[10] Roberge V, Tarbouchi M, Labonte G. Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning [J]. IEEE Transactions on Industrial Informatics, 2013: 132-141.

[11] Smolyanskiy N, Kamenev A, Smith J, et al. Toward low-flying autonomous MAV trail navigation using deep neural networks for environmental awareness [C]//2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC. 2017.

[12] Walker O, Vanegas F, Gonzalez F, et al. A Deep Reinforcement Learning Framework for UAV Navigation in Indoor Environments [C]//2019 IEEE Aerospace Conference, Big Sky, MT, USA. 2019.

[13] Walker O, Vanegas F, Gonzalez F, et al. A Deep Reinforcement Learning Framework for UAV Navigation in Indoor Environments [C]//2019 IEEE Aerospace Conference, Big Sky, MT, USA. 2019.

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

[15] Chen P, Pei J, Lu W, et al. A Deep Reinforcement Learning Based Method for Real-Time Path Planning and Dynamic Obstacle Avoidance [J]. Neurocomputing, 2022, 497: 64-75.