IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 9, 2021, accepted February 2, 2021, date of publication February 5, 2021, date of current version February 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3057485

Unmanned Aerial Vehicle Path Planning Algorithm Based on Deep Reinforcement Learning in Large-Scale and Dynamic Environments

RONGLEI XIE, ZHIJUN MENG, LIFENG WANG, HAOCHEN LI, KAIPENG WANG, AND ZHE WU

School of Aeronautic Science and Engineering, Beihang University, Beijing 100191, China

Corresponding author: Zhijun Meng (mengzhijun@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61702023 and Grant 61976014, and in part by the Fundamental Research Funds for the Central Universities.

ABSTRACT Path planning is one of the key technologies for autonomous flight of Unmanned Aerial Vehicle. Traditional path planning algorithms have some limitations and deficiencies in the complex and dynamic environment. In this article, we propose a deep reinforcement learning approach for three-dimensional path planning by utilizing the local information and relative distance without global information. UAV can obtain the limited environmental information nearby in the actual scenario with limited sensor capabilities. Therefore, path planning can be formulated as a Partially Observable Markov Decision Process. The recurrent neural network with temporal memory is constructed to address the partial observability problem by extracting crucial information from historical state-action sequences. We develop an action selection strategy that combines the current reward value and the state-action value to reduce the meaningless exploration. In addition, we construct two sample memory pools and propose an adaptive experience replay mechanism based on the frequency of failure. The simulation experiment results show that our method has significant improvements over Deep Q-Network and Deep Recurrent Q-Network in terms of stability and learning efficiency. Our approach successfully plans a reasonable three-dimensional path in the large-scale and complex environment, and has the perfect ability to avoid obstacles.in the unknown environment.

INDEX TERMS Deep reinforcement learning, path planning, recurrent neural network.

I. INTRODUCTION

The unmanned aerial vehicle (UAV) has attracted wide attention in both military and civilian fields because of low cost, flexibility and small size, et [1], [2]. The reliable and reasonable path planning is the basis for ensuring its own safety and mission success due to the increasingly complex environment. Some missions have restrictions on flight altitude and fuel consumption, etc. In addition, there are many threats such as hills, buildings at low flighting altitude. However, the UAV can only rely on their own limited sensors to obtain partial threat information [3]. Furthermore, the path planning of the UAV is more complicated and difficult than the path planning of robots because of the huge state space and action space. Path planning for the UAV has become a research hotspot and received wide attention from researchers in recent years. In this article we propose an approach of autonomous planning of available paths from starting point to destination in complex environment.

In previous work, there are many classical methods such as A* algorithm [4] and artificial potential field method [5], [6], Voronoi diagrams [7], which have been proven to be effective for two-dimensional environment. For the path planning of complex environment with large state space, the intelligent path planning methods have attracted a lot of attention. Typical intelligent methods include genetic algorithm [8], [9], ant colony algorithm [10], [11], artificial bee colony [12], particle swarm optimization [13], [14], and so on. The combination

The associate editor coordinating the review of this manuscript and approving it for publication was Junxiu Liu.

of different methods can make use of the advantages of each algorithm [15]–[17]. Fast Marching [18], [19] is an efficient three-dimensional path planning method for keeping a fixed flight height in an open field with non-uniform terrain, which fully considers the smoothness and safety of the generated path. Compared with the classical path planning algorithm, Fast Marching method has better computational efficiency and practical value. The paper addresses the UAV three-dimensional path planning in a local dynamic environment, while the classic A* and RRT algorithms are suitable for static path planning with a known global environment. Intelligent algorithms represented by genetic algorithm and ant colony algorithm have environmental adaptability and system robustness. However, these algorithms require a lot of search and iteration, resulting in inefficient planning. The artificial potential field method has good real-time performance due to its simple calculation. It would be easy to fall into a local optimal solution if the gravity and repulsion are the same at certain points. If the target is near an obstacle, the large repulsion would make it difficult for the UAV to approach the target point.

The classic path planning algorithms perform well for obstacle avoidance in simple environments. However, reinforcement learning can show its advantages in the dynamic and complex environment. First of all, reinforcement learning can dynamically adjust parameters through training by interacting with the environment, which has positive adaptability and robustness compared with classic algorithms. Secondly, reinforcement learning essentially obtains the mapping relationship from state to action, which does not involve a complex search process in the decision-making process, so it is suitable for UAV path planning that requires real-time decision-making. Thirdly, reinforcement learning defines reward function according to distance, fuel consumption, which can achieve multi-objective optimization ability without a lot of manual parameter adjustment. Fourthly, reinforcement learning does not depend on the prior information of the environment, so it is suitable for dynamic path planning with limited information.

Deep neural network has excellent feature learning ability, which can extract concise effective feature information from complex high-dimensional state. Reinforcement learning algorithm can learn an optimal policy by interacting with environment. Deep reinforcement learning algorithm combining deep learning and reinforcement learning has achieved remarkable achievements in the fields of industrial control and gaming games [20]. Lange and Riedmiller [21] propose the combination of deep learning and reinforcement learning. Mnih *et al.* [22] propose Deep Q-Network (DQN) algorithm combining Convolutional Neural Network (CNN) with Q learning algorithm, and the training results show that DQN has surpassed human players in the Atari2600 game platform. Schaul *et al.* [23] propose an experience replay mechanism to accelerates the learning speed of DQN by breaking the correlation of sample sequence. Hausknecht and Stone [24] develop Deep Recurrent Q-Network (DRQN) algorithm combining Recurrent Neural Network (RNN) with Q learning algorithm. The performance of DRQN is much better than DQN in long-term strategy game with delayed reward, but DRQN has the disadvantage of unstable convergence.

In recent years, researchers have tried to solve the problem of path planning by deep reinforcement learning algorithm. Pfeiffer *et al.* [25] propose an end-to-end learning model that analyzes laser information by CNN and then uses A* algorithm as label information for supervised learning. The robot movement instruction is directly obtained through the network according to the laser ranging information and the target position. Chen *et al.* [26] control the autonomous vehicles by obtaining the semantic information extracted from images. Kretzschmar *et al.* [27] develop an inverse reinforcement learning approach to solve the robot navigation problem by making use of the human experience to speed up the learning speed. Lei Tai *et al.* [28] develop a robot motion planner based on deep reinforcement learning, which utilizes laser ranging sensors to obtain information about nearby obstacles. Navigation problems without global information in complex environments can be modeled as Partially Observable Markov Decision Process (POMDP), and recurrent neural network can be used to memorize past observation to gain more knowledge [29], [30].

Cui *et al.* [31] propose a navigation system that makes UAV take advantage of 2D laser rangefinder to navigate in the foliage environment without GPS. Simultaneous Localization and Mapping (SLAM) is a classic method, which can estimate the position of the UAV and obtain information about nearby obstacles by constructing a map of the surroundings [32], [33]. Gageik *et al.* [34] combine multiple low-cost sensors such as infrared and ultra-sonic to achieve obstacle avoidance flight of UAV with low computational burden and economic cost.

The deep reinforcement learning can learn available path planning policy by interacting with the environment without manually setting complex rules. However, there are some challenges of path planning in the large-scale and dynamic environment:

1) The enormous number of states makes the neural network learning slowly and converging difficultly.

2) The UAV can only obtain local environment information due to limited sensor capabilities, making it difficult to select the correct action.

3) The reward has the character of delay, because the agent may have to make hundreds of steps to reach the target point in the large-scale environment.

In this article, we propose a deep reinforcement learning approach to solve the problem of the UAV path planning in the complex and dynamic environment. The main contributions of this article are summarized as follows:

1) We propose a new action selection strategy by combining the current reward R value and the Q value, which addresses the problem of inaccurate prediction of the neural network at the early stage of training. The purpose of new action selection strategy is to reduce meaningless

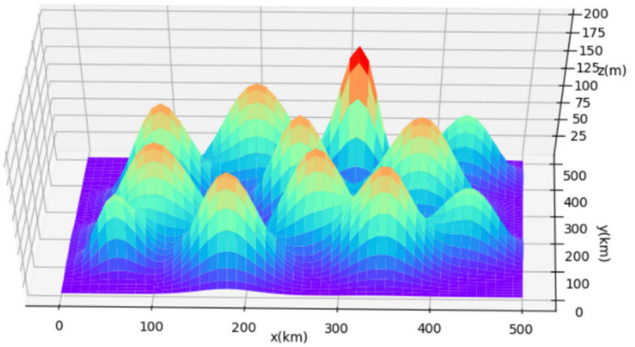


FIGURE 1. The three-dimensional topographic map.

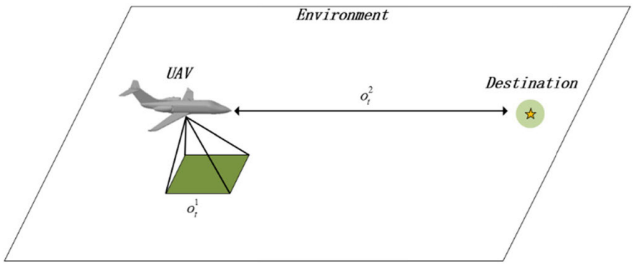


FIGURE 2. The observation o_t consists of two parts: o_t^1 represents the relative terrain altitude information near the UAV, and o_t^2 represents the distance to the destination.

exploration, especially to reduce repeated meaningless collision.

2) We propose an adaptive sampling mechanism. Two kinds of memory pools are constructed according to the average reward value per round. Samples are taken from the important memory pool according to the frequency of task failure. This approach improves the learning efficiency and convergence stability of the algorithm with low computational cost.

The rest of the paper is organized as follows: Section II introduces the framework of the path planning algorithm. Section III points out some methods to accelerate the learning speed of algorithm. Section IV describes the experimental results and analyzes it. Section V presents conclusion and future work.

II. CONSTRUCTION OF THE ALGORITHM

In this section, we define the state space and action space, and formulate the path planning problem as POMDP. Then we propose a path planning algorithm based on deep reinforcement learning and present the network structure including CNN and RNN.

A. DEFINITION OF OBSERVATION AND ACTION

We establish a three-dimensional topographic map for the UAV to evaluate path planning algorithm. The UAV should avoid collision with the obstacle, and the height of obstacle can be calculated by:

$$H(X, Y) = \sum_{i=1}^{N} h_i e^{-\left(\frac{X-a_i}{c_i}\right)^2 - \left(\frac{Y-b_i}{c_i}\right)^2} \tag{1}$$

where $H(X, Y)$ represents the height of obstacle at the horizontal position (X, Y). h_i controls the height of the obstacle. c_i determines the size of the obstacle. (a_i, b_i) is the horizontal coordinate of the obstacle.

The three-dimensional topographic map based on (1) is shown as Fig. 1. The UAV cannot acquire global information due to sensor limitations in the real environment. And the limited information perceived by the UAV sensor is called the observation o_t. As depicted in Fig. 2, o_t consists of two parts: the local environment information o_t^1 and the relative position o_t^2 from the current point to the target point.

Assume that the three-dimensional coordinate of a point is (X_i, Y_i, Z_i), and the height of the UAV relative to the ground o_{si} (the probabilistic safety) can be defined as (2). H_{uav} represents the absolute height of the UAV.

$$o_{si} = H_{uav} - H(X_i, Y_i) \tag{2}$$

The sensor on the UAV can sense the environmental information of surrounding area, so the observation o_{t1} is composed of the probabilistic safety of k points $o_t^1 = [o_{s1}, o_{s2} \cdots o_{sk}]$.

The UAV can obtain its own location information (X_O, Y_O, Z_O) by the inertial measurement unit or GPS. The target position (X_T, Y_T, Z_T) can be known, so o_t^2 is calculated by:

$$o_t^2 = (X_T - X_O, Y_T - Y_O, Z_T - Z_O) \tag{3}$$

We have considered some aircraft maneuvering restrictions in defining the action space of the UAV, such as minimum turning radius and maximum climbing angle. As shown in Fig. 3, if the UAV enters the "saddle-shaped" area between the two obstacles, the probability of failure could be very high. It is difficult for the UAV to detect the above dangerous areas in time due to the limited sensor detection capability. If the UAV takes the action of direct flight A_0 or A_{45}, the probability of obstacle avoidance failure could be high. If the UAV chooses to turn left A_{63}, it can ensure flight safety. Therefore, the stronger the horizontal maneuvering ability is, the greater the probability of UAV escaping safely. The action includes horizontal action a_t^H and vertical action a_t^V. As shown in Fig. 4, the red squares represent optional actions. Horizontal action a_t^H includes turning left 45/63 degrees, direct flight, turning right 45/63 degrees. Vertical action a_t^V includes climbing 45 degrees, direct flight and descending 45 degrees. There are 11 actions in the action space of the UAV.

All maneuvers and spatial coordinates are set to integers for ease of calculation. Discretization also helps to reduce the scale of state space.

B. MARKOV MODEL

As shown in Fig. 5, it is a simplified model of obstacles that the UAV may encounter in the real environment. It is assumed that the UAV is at the point A at the beginning. Since the sensor does not perceive the "inverted V-shaped" obstacle in front due to the limited measuring range of the sensor, point B may be chosen as the desired position point in order to

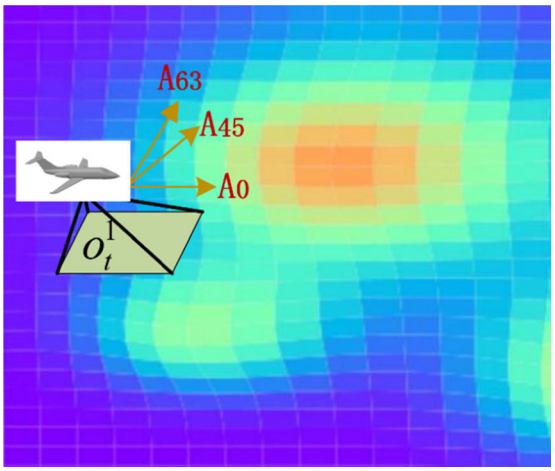


FIGURE 3. The strong horizontal maneuverability is helpful to improve the success rate of UAV obstacle avoidance.

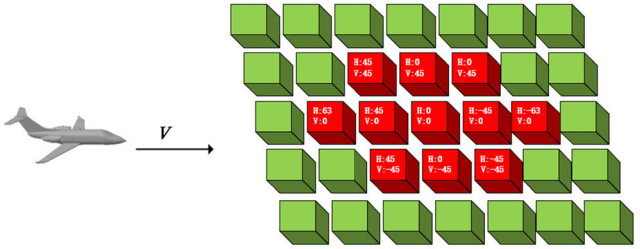


FIGURE 4. The action space includes horizontal action and vertical 2 action.

approach the target point. The UAV finds out that there is an obstacle in front of point B, and could choose to turn to point C or point D. Assuming that the UAV chooses to reach point C, the UAV can still choose point B in the next step. As a result, the UAV may fall into a loop trap and cannot escape the ''inverted V-shaped'' obstacle. If the algorithm has the ability to memorize historical information, it can effectively avoid the above problem. The true state of the environment can be more accurately estimated through a piece of historical trajectory information. The observation-action sequence does not have Markov property $p\left(o_{t+1}|o_0, a_0, o_1, a_1 \cdots o_t, a_t\right) \neq p(o_{t+1}|o_t, a_t)$. Therefore, we can model the path planning problem as POMDP.

POMDP can be represented by $(S, A, P, R, \Omega, O, \gamma)$. S is the set of true state s_t in the environment. A is the set of all available actions of the UAV, and $a_t \in A$ represents the action that the UAV executes at time t. P is the probability distribution that the UAV moves to other states after executing action a_t at the state s_t, $P\left(s_{t+1}|s_t, a_t\right) \rightarrow (0, 1)$. R is the reward that the UAV executes action a_t at the state s_t and transferring to state s_{t+1}, and can be defined as $R\left(s_t, a_t\right) = r_t$. Ω is the set of observations. $o_t \in \Omega$ is the local environmental information obtained by the sensor. $O(o_t|s_t, a_t)$ is the conditional observation probability distribution that the UAV executes action a_t at the state s_t. γ is the discount factor.

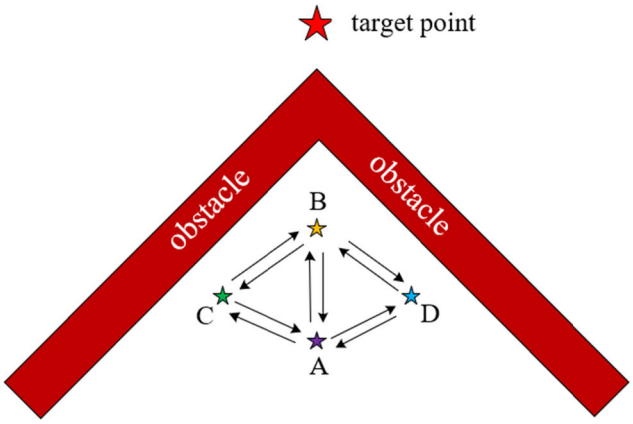


FIGURE 5. The UAV encounters an inverted V-shaped obstacle. A∼D 28 indicates possible location points..

The reinforcement learning algorithm learns a policy π that the UAV chooses an action a_t based on a piece of historical trajectory φ_t=$(o_{t-L+1}, o_{t-L+2}, \cdots o_t)$ of length L, which can be denoted as $a_t \sim \pi(\varphi_t)$. Historical trajectory φ_t is beneficial to estimate true environmental state s_t information more accurately. The goal of algorithm is to learn an optimal policy π^*, which the UAV can obtain the maximum accumulated discounted reward.

The state value function $V^\pi\left(\varphi_t\right)$ is defined as the expected sum of discounted rewards in accordance with policy π. $V^\pi\left(\varphi_t\right)$ is defined as:

$$V^\pi\left(\varphi_t\right) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \varphi_t\right] \tag{4}$$

The state-action value function $Q^\pi\left(\varphi_t, a_t\right)$ is the expected discounted reward obtained by taking the action a_t in the state φ_t according to the policy π. $Q^\pi\left(\varphi_t, a_t\right)$ can be defined as:

$$Q^\pi\left(\varphi_t, a_t\right) = E\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | \varphi_t, a_t\right] \tag{5}$$

Q-learning is a model-free reinforcement learning algorithm. The $Q_{k+1}^\pi\left(\varphi_t, a_t\right)$ is updated iteratively by Bellman formula:

$$Q_{k+1}^\pi\left(\varphi_t, a_t\right) = Q_k^\pi\left(\varphi_t, a_t\right) + \alpha\left(r_t + \gamma max_{Q_k^\pi\left(\varphi_{t+1}, a_{t+1}\right) - Q_k^\pi\left(\varphi_t, a_t\right)}\right) \tag{6}$$

where $\alpha \in [0, 1]$ is learning rate, which controls parameter updating speed.

Traditional Q-learning utilizes tables to record the learned state-action values, which limits the ability of the algorithm to solve the problem of large-scale planning state space. Therefore, we use the neural network to approximate the state-action value. The neural network can theoretically fit any non-linear model with enough training data and suitable learning parameters.

C. NETWORK STRUCTURE

As illustrated in Fig. 6, the network structure contains three neural networks. The input of the model is a sequence of historical trajectory φ_t. We adopt the CNN module to capture

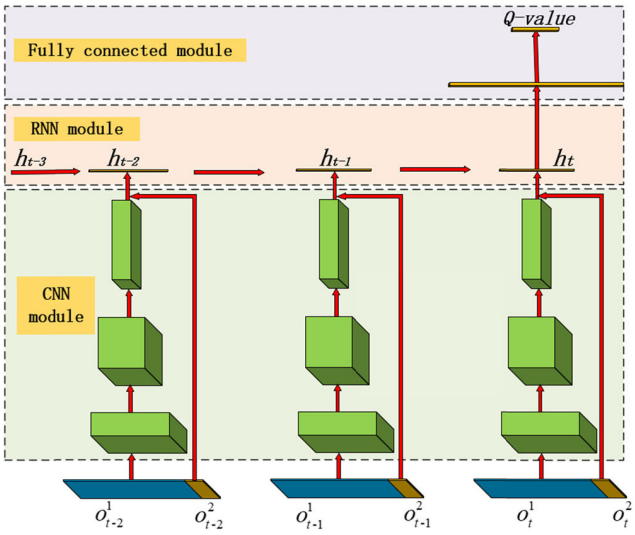


FIGURE 6. Neural network architecture.

spatial feature information from the observation o_t^1, then construct the RNN module to extract temporal feature information. Finally, the estimated state-action value is predicted by fully connected neural network. The CNN is easy to lose information when extracting features. The relative position o_t^2 is an important three-dimensional information, so it bypasses the CNN module and directly inputs to the RNN module. The CNN module removes the pooling layer, because the pooling layer can result in information loss.

If o_t^1 directly inputs to the RNN module, the amount of calculation could become very huge because of its high-dimensional and redundant information. Thus, the function of CNN module is to extract low-dimensional and effective spatial features. RNN module can extract the temporal feature information of the sample sequence, and fundamentally solve the memory problem of sequence samples. The hidden layer h_t not only receives the feature c_t from the CNN module, but also receives the hidden layer state h_{t-1} at the previous time. The calculation of RNN is given by:

$$h_t = f\left(w_{xh}c_t + w_{hh}h_{t-1}\right) \tag{7}$$

where f is a nonlinear activation function. w_{xh} and w_{hh} represent the connection weights of neurons of different layers.

We adopt the Long Short-Term Memory (LSTM) to alleviate the problem of gradient disappearance [35]. The neural network has the disadvantage of being difficult to converge due to correlation of sample sequences [36]. We adopt the fixed Q-target network and experience replay mechanism to speed up the convergence of the algorithm [22].

Since RNN has the ability of remembering temporal information, the purpose of experience replay mechanism is convenient for the training of the algorithm rather than breaking the correlation of the samples in this article. The sample $e_t = (\varphi_t, a_t, r_t, \varphi_{t+1})$ generated by the interaction between the UAV and environment is stored in the memory pool $D = \{e_1, e_2, \cdots, e_V\}$ with fixed-size V. L is the length of observation sequence. The network randomly takes a certain number of sample sequences from the memory pool for training.

There are two independent neural networks to predict the action value: the current value network $Q(\varphi_t, a_t|\theta_i)$ and the target value network $Q\left(\varphi_{t+1}, a_{t+1}|\theta_i'\right)$. The structure of the two networks is the same. The current network uses gradient descent to update the network weights, and the updated parameters are copied to the target network at a regular interval. Since the two networks are not updated synchronously, the stability of the algorithm is improved. The loss function $L(\theta_i)$ is given as:

$$L(\theta_i) = E_{(\varphi,a,r,\varphi')\sim D}\left[\frac{1}{2}\left(r_t + \gamma max\right.\right. \\ \left.\left. Q\left(\varphi_{t+1}, a_{t+1}|\theta_i'\right) - Q\left(\varphi_t, a_t|\theta_i\right)\right)^2\right] \tag{8}$$

Current value network is updated by:

$$\Delta\theta_i = E_{(\varphi,a,r,\varphi')\sim D}[r_t + \gamma maxQ\left(\varphi_{t+1}, a_{t+1}|\theta_i'\right) \\ -Q\left(\varphi_t, a_t|\theta_i\right)]\frac{\partial Q\left(\varphi_t, a_t|\theta_i\right)}{\partial\theta_i} \tag{9}$$

$$\theta_{i+1} = \theta_i + \alpha\Delta\theta_i \tag{10}$$

III. APPROACHES TO SPEED UP ALGORITHM

In this section, we propose several methods to speed up the learning of the algorithm.

A. REWARD DESIGN: INCORPORATING DOMAIN KNOWLEDGE

The reward function evaluates the agent executing actions in a certain state. The sparse reward function is a common method, which the agent only gets rewards when it succeeds or fails. However, the state space is too large to obtain valuable reward in a large-scale environment, which undoubtedly leads to low learning efficiency of the algorithm. In addition, the UAV is not only to avoid obstacles to reach the target point, but also to ensure the path to be as short as possible. And some tasks require the flight altitude to be as low as possible. Therefore, by incorporating domain knowledge such as fuel consumption and distance to the target point, a heuristic reward function r_t is constructed:

$$r_t = \omega_1 p_1 + \omega_2 p_2 + \omega_3 p_3 + \omega_4 p_4 \tag{11}$$

where $\omega_1 \sim \omega_4$ are coefficients, and $p_1 \sim p_4$ are the evaluation factors of path performance. The concrete meaning and calculation method are as follows:

1) p_1 represents the maneuvering cost of UAV executing action a_t. The UAV should keep the flight state as straight as possible to reduce fuel consumption. The maneuvering cost consists of horizontal maneuvering cost p_1^H and vertical maneuvering cost p_1^V. The p_1 is given by:

$$p_1 = \omega_1^H p_1^H + \omega_1^V p_1^V \tag{12}$$

where ω_1^H and ω_1^V are coefficients.

2) p_2 is related to the distance from the current position to the target point. In artificial potential field method, the target point would exert gravitation on the UAV, which provides direction guidance for UAV's action selection policy. p_2 is given by:

$$p_2 = \log_{10}(|o_{t2}| + \delta) \tag{13}$$

where δ is positive constant to avoid 0.

3) p_3 represents the reward of UAV deviating from cruise altitude Z_0, which is designed to ensure UAV flying at cruise altitude. The p_3 is given by:

$$p_3 = (Z_t - Z_{t+1})(Z_t - Z_0) \tag{14}$$

4) p_4 is the traditional sparse reward function, which includes three cases. The p_4 is given by:

$$p_4 = \begin{cases} R_0 & success \\ -R_0 & failure \\ 0 & otherwise \end{cases} \tag{15}$$

where R_0 is positive constant. If the distance between UAV and target is less than a certain threshold, the mission can be considered as successful. If the UAV collides with obstacles or exceeds the height limit, it would be set as mission failure.

B. IMPROVED ACTION SELECTION STATEGY

The common action selection strategy for reinforcement learning is ε-greedy. It selects an action based on maximum state-action value $Q(\varphi_t, a_t|\theta_i)$ with the probability ε, and randomly selects an action with the probability 1-ε. The ε-greedy is given by:

$$\pi(a_t|\varphi_t) = \begin{cases} random\ choice & 1-\varepsilon \\ \underset{a\in A}{\operatorname{argmax}} Q(\varphi_t, a\,|\,\theta_i) & \varepsilon \end{cases} \tag{16}$$

The weight parameters of $Q(\varphi_t, a_t|\theta_i)$ and $Q\left(\varphi_t, a_t|\theta_i^{'}\right)$ is randomly set at the beginning, so the state-action value predicted by the neural network is very inaccurate. The reward r_t is the only valuable parameter in the target value $r_t + \gamma maxQ_t\left(\varphi_{t+1}, a_{t+1}|\theta_i^{'}\right)$ at the beginning. Therefore, it can basically be considered that the action is randomly selected in the early stage of algorithm training. Song *et al.* [37] increase the learning efficiency by taking advantage of prior knowledge to initialize the weight of $Q(\varphi_t, a_t|\theta_i)$, but it is difficult to obtain enough prior knowledge. In addition, the learning rate of the algorithm decreases as the number of training steps increases. In the early stage of training, there are a large number of repeated collisions because it cannot effectively avoid obstacles. The algorithm has a high learning rate when exploring the first part of the environment. After a large number of training steps, the learning rate is already very low when learning the latter part of the environment.

To solve the above problem, we propose a new action selection strategy $\pi^{'}(a_t|\varphi_t)$ by combining the reward R and the Q value, referred to as RQ action selection strategy. The approach utilizes the reward function to evaluate all the executable actions according to the current state φ_t. The evaluation value only focuses on the current reward. The corresponding set of rewards rs_t is given by:

$$rs_t = [R(\varphi_t, a_1), R(\varphi_t, a_2), \ldots R(\varphi_t, a_n)] \tag{17}$$

We expect to pay more attention to the role of rewards rather than Q value in the early stages of training, so as to avoid a lot of repetitive and meaningless failures due to collisions. The prediction of the neural network $Q(\varphi_t, a_t|\theta_i)$ could be more accurate as the training steps increase. The effect of rs_t on the overall selection strategy is reduced due to the increasing of $Q(\varphi_t, a_t\,|\,\theta_i)$. The RQ action selection strategy is given by:

$$\pi'(a_t|\varphi_t) = \begin{cases} random\ choice & 1-\varepsilon \\ \underset{a\in A}{\operatorname{argmax}}(Q(\varphi_t, a|\theta_i) + rs_t g^{\frac{1}{n}\sum_{k=1}^{n} Q(\varphi_t, a_k|\theta_i)}) & \varepsilon \end{cases} \tag{18}$$

where $g \in (0,1)$ is constant, and n represents the number of actions in the action space.

The purpose of RQ action selection strategy is to reduce meaningless exploration, especially to reduce repeated meaningless collision. RQ method cannot guarantee that the algorithm can learn an effective strategy quickly. Therefore, UAV can still explore the environment in the learning process so as to avoid falling into local optimum. In addition, ε-greedy is used for action selection to ensure the UAV to explore the environment.

The RQ action selection strategy utilizes the reward value set rs_t to correct neural network prediction errors. The proposed method focuses on short-term reward in the early stage of training, which can reduce the collision probability between UAV and obstacles. In the later stage of network training, since $Q(\varphi_t, a_t|\theta_i)$ becomes more accurate, the prediction of the neural network is more emphasized in the action selection. In addition, RQ action selection strategy can make the UAV move as long as possible in the initial training stage, and obtain more environmental information to speed up the learning speed of the algorithm.

C. ADAPTIVE SAMPING MECHANISM

It is necessary to take some measures to reduce number of UAV-environment interactions and make efficient use of existing samples. The transfer learning can transfer the existing knowledge from the source task to the target task [38]. However, it is difficult to measure the similarity between the source task and the target task, which makes the application difficult. The samples can be utilized to model the environment, but this method requires higher accuracy of the model [39]. The prioritized experience replay mechanism [23] is proved to be an effective method, which determines the sample priority based on the TD-error. But this method needs to continuously sort the samples in the memory pool according to the priority, which greatly increases the calculation amount of the algorithm.

We construct two memory pools: one for important samples and the other for normal samples. There are two important requirements for path planning problems: avoiding obstacles and reaching the destination. The samples colliding with obstacles can help the algorithm learn to avoid collisions quickly. And samples with a larger reward value indicates that the UAV is closer to the target point. The above two types of samples can be used as important sample to train network. If only two types of samples are utilized for training, insufficient sample diversity may cause the algorithm to fall into a local optimal solution. Furthermore, the absolute value of the reward for the two samples is large, which may make it difficult for the algorithm to converge. Therefore, it is necessary to establish a normal memory pool for training [40].

Based on the above analysis, this article proposes an Adaptive Sampling (ADSA) mechanism. First of all, we construct a memory pool I to store important samples and a memory pool D to store normal samples, as follows:

1) Saving all samples of the p round into $er_p = \{\varphi_0, a_0, r_0, \varphi_1, a_1, r_1, \cdots, \varphi_N\}$.
2) If the round ends, calculating the round average reward value R_p^{ave}:

$$R_p^{ave} = \frac{1}{N-1}\sum_{i=0}^{N-1} r_i \tag{19}$$

3) Finding the minimum value of average reward value R_{min}^{ave} in the memory pool I. The memory pool I is updated by strategy $\mu(er_p)$:

$$\mu(er_p) = \begin{cases} er_p \to D & R_p^{ave} < R_{min}^{ave} \\ er_p \to I & R_p^{ave} \geq R_{min}^{ave} \end{cases} \tag{20}$$

4) Repeating 1)- 3).

Secondly, the algorithm adaptively samples from two memory pools based on round failure rate. It indicates that the current action selection policy is incorrect when the round failure rate is high. Valuable samples from the important memory pool should be taken to quickly correct the policy. The specific method is as follows:

1) If the round ends, go to 2), otherwise go to 4).
2) Setting a fixed sampling probability $\tau \in [0, 1]$, randomly generating a number $v \in [0, 1]$. when $v < \tau$, go to 3), otherwise go to 4).
3) Randomly selecting m samples from the important memory pool I.
4) Randomly selecting m samples from the normal memory pool D.

The ADSA method has the two advantages. The calculation amount is very small, and there are only average calculation and minimum calculation for each round. The method adaptively selects important samples and normal samples according to the failure rate of the round, which helps to correct the wrong strategy.

This article designs a reward function based on the mission environment and the characteristics of the UAV. Aiming at the problems of algorithms, we propose the RQ_ADSA_DRQN approach which uses RQ method and ADSA method based on DRQN. The specific flow of the algorithm is shown in pseudocode and Fig. 7.

Algorithm 1 Improved DRQN for UAV path planning

Input: observation $\boldsymbol{\varphi_t}$
Output: action $\boldsymbol{a_t}$
Episodes $\boldsymbol{M_e}$, Steps per episode $\boldsymbol{N_e}$
Initialization parameters: current value network weight $\boldsymbol{\theta_i}$, target value network weight $\boldsymbol{\theta_i'}$
$\boldsymbol{Mark} = 0$
REPEAT
$\boldsymbol{M_e} = \boldsymbol{M_e} - 1$.
REPEAT
$\boldsymbol{N_e} = \boldsymbol{N_e} - 1$.
Put $\boldsymbol{\varphi_t}$ into the current value network to get $\boldsymbol{Q}(\boldsymbol{\varphi_t}, \boldsymbol{a}|\boldsymbol{\theta_i})$.
Calculate the reward $\boldsymbol{rs_t}$.
Choose action $\boldsymbol{a_t}$ according to (18).
Execute action $\boldsymbol{a_t}$ to get a new state $\boldsymbol{\varphi_{t+1}}$ and calculate the reward $\boldsymbol{r_t}$.
Store $(\boldsymbol{\varphi_t}, \boldsymbol{a_t}, \boldsymbol{r_t}, \boldsymbol{\varphi_{t+1}})$ into $\boldsymbol{er_p}$.
IF the round ends:
Calculate $\boldsymbol{R_{min}^{ave}}$ and $\boldsymbol{R_p^{ave}}$.
IF $\boldsymbol{R_{min}^{ave}} < \boldsymbol{R_p^{ave}}$:
$\boldsymbol{er_p} \to \boldsymbol{I}$.
Else:
$\boldsymbol{er_p} \to \boldsymbol{D}$.
IF $\boldsymbol{v} < \tau$:
Sample a minibatch from $\boldsymbol{A}$.
$Mark = 1$.
IF $\boldsymbol{Mark} = 0$:
Sample a minibatch from $\boldsymbol{D}$.
Else:
$Mark = 0$.
Update the weight $\boldsymbol{\theta_i}$ according to (8)-(10).
Update the target network per certain steps $\boldsymbol{\theta_i'} \leftarrow \boldsymbol{\theta_i}$.
$\boldsymbol{t} \leftarrow t + 1$.
UNTIL $\boldsymbol{N_e} = 0$
UNTIL $\boldsymbol{M_e} = 0$

IV. SIMULATION EXPERIMENT AND RESULT

A. EXPERIMENTAL SETTING

The algorithm needs a lot of interaction with the environment to learn an effective collision avoidance policy. The learning cost is very high in a real environment, so we construct a simulation environment according to (1). Assuming that the mission environment area is 500 km*500 km*200 m. The starting point is (0 km, 300 km, 30 m), and the target point is (500 km, 200 km, 30 m).

We have done a lot of experiments on parameter setting. Different parameters are tested for their impact on the performance of the RQ_ADSA_DRQN algorithm. The number of training steps is 100,000, and the evaluation indicators is the

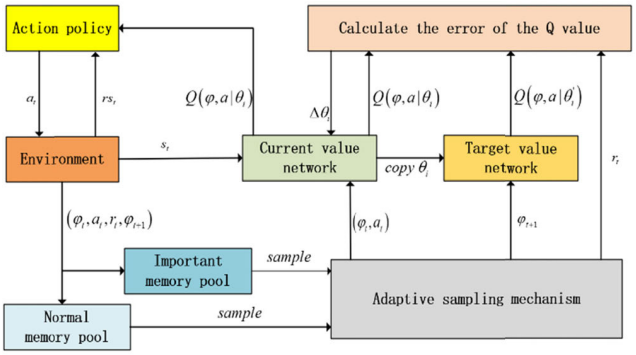


FIGURE 7. Improved DRQN for UAV path planning structure.

TABLE 1. The influence of parameter values on algorithm performance.

learning rate α	value	0.01	0.001	0.0001
	times of success	0	32	20
learning rate decay	value	0.7	0.8	0.9
	times of success	7	3	32
exploratory factor ε	value	0.4	0.5	0.6
	times of success	20	32	35
size of normal memory pool	value	6000	8000	10000
	times of success	28	32	30
size of key memory pool	value	20	30	40
	times of success	16	32	31
batch size	value	16	32	64
	times of success	27	32	35
network update interval C	value	600	800	1000
	times of success	29	32	32
neuron number of LSTM	value	256	512	1024
	times of success	20	32	12
length of sequence L	value	6	7	8
	times of success	5	32	27
Weight coefficient ω_1	value	$-6.25*10^{-4}$	$-1.25*10^{-3}$	$-2.5*10^{-3}$
	times of success	26	32	18
Weight coefficient ω_2	value	0.075	0.15	0.3
	times of success	9	32	30
Weight coefficient ω_3	value	$7.5*10^{-4}$	$1.5*10^{-3}$	$3*10^{-3}$
	times of success	15	32	20
Weight coefficient ω_4	value	0.5	1	2
	times of success	31	32	12
sampling probability τ	value	0.3	0.4	0.5
	times of success	29	32	30

total number of times that the UAV reaches the target point. The results are shown in Table 1. It can be seen that learning rate and the reward function have a relatively large impact on the result, and some parameters have little impact on the results, such as the experience library size V. According to the test results, the parameters are set as follows: The initial learning rate α is 0.001, multiplied by the learning rate decay 0.9 every 5000 steps. The exploratory factor ε of action selection is initially 0.5, and decreases linearly to 0.001 after 70,000 steps. Because the UAV would not fall into local traps and reach the target point after training 70000 steps. The sampling probability τ is 0.4. The size of the normal memory pool is $V = 8000$, while the size of important memory pool is $U = 30$. The batchsize is $m = 32$. The length of observation sequence is $L = 7$. The parameters of the current value network are copied to the target network every $C = 800$ steps. The neuron number of LSTM is 512. The weight coefficients in reward function are set as follows: $\omega_1 = -0.00125, \omega_2 = 0.15, \omega_3 = 0.0015, \omega_4 = 1$. The cruise altitude $z_0 = 10$ to ensure the flight height is as low as possible. There are 100 episodes in training to compare the performance of different algorithms, and each episode has 2000 steps.

B. COMPARISON OF ALGORITHM PERFORMANCE IN A STATIC SCENARIO

In this article, DQN, DRQN, RQ_DRQN, ADSA_DRQN and RQ_ADSA_DRQN are selected for comparison. The DQN has three convolution layers and two fully connected layers. DRQN replaces a fully connected layer of DQN with LSTM. RQ_DRQN utilizes RQ method to accelerate DRQN learning. ADSA_DRQN uses adaptive sampling to increase the stability of DRQN. The two improved methods are all used for DRQN is RQ_ADSA_DRQN. We assess the performance of algorithms based on five indicators: The average reward value in an episode *avg_rewards*, the average state-action value in an episode *avg_qs*, the rounds in an episode *num_round*, the number of times to reach the destination *num_success*, and the average loss of model in an episode *avg_loss*.

avg_rewards is the average of all reward values in an episode, and it is given by:

$$avg\_rewards = \frac{1}{N_e} \sum_{t=1}^{N_e} r_t \tag{21}$$

avg_qs is the average of all $Q(\varphi, a)$ in an episode, and it is given by:

$$avg\_qs = \frac{1}{N_e} \sum_{t=1}^{N_e} \frac{1}{M} \sum_{i=1}^{M} Q(\varphi_t, a_i) \tag{22}$$

where N_e is the number of steps in an episode. M is the size of action space.

1) THE AVERAGE REWARD *avg_rewards*

The goal of reinforcement learning is to learn a policy that enables the agent to obtain the maximum long-term average reward. Therefore, the average reward is a crucial indicator for evaluating various algorithms. As shown in the Fig. 8, the average reward curve is volatile and noisy, because small changes in network weights can cause the choice of different actions. Algorithms would be analyzed based on the value of reward value, convergence stability and convergence speed. It's clearly seen that the average reward of DQN is far worse than other algorithms, because DQN cannot remember previous historical experience information and makes decisions based only on its perception of the current environment. Other four algorithms can get high rewards, because LSTM can remember more information about the local structure of the environment based on their historical observations. The average reward of RQ_ADSA_DRQN and ADSA_DRQN

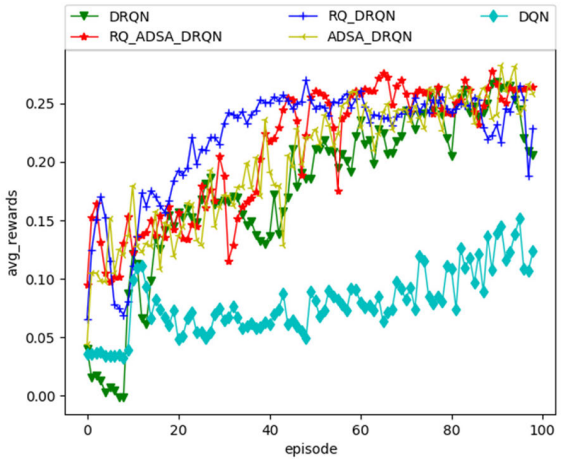


FIGURE 8. The average reward in a static scenario.

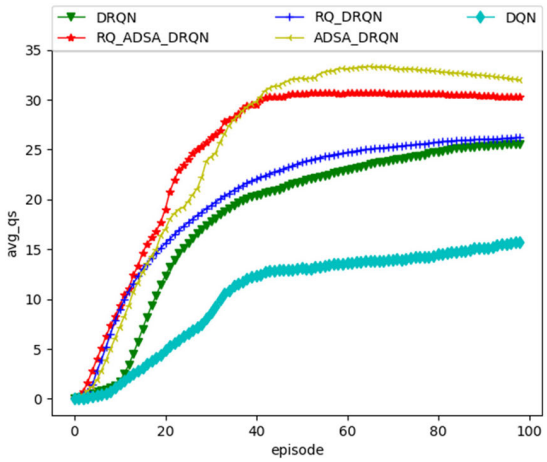


FIGURE 9. The average action value in a static scenario.

can basically reach a relatively stable value after 70 episodes, while DRQN and RQ_DRQN are unstable at the later stage of training. Instability and divergence are inherent disadvantages of reinforcement learning, as the distribution of sample data in training is always changing. The ADSA mechanism proposed in this article can effectively improve the stability of the algorithm. If the round failure rate is higher, the probability of sampling from important memory pool is higher. The update frequency of important memory pool is slow, so its sample distribution is relatively fixed. The result shows that adaptive sampling mechanism can quickly correct the wrong strategy and effectively improve the stability of the algorithm. The ADSA mechanism also has the effect of accelerating convergence. The convergence speed of RQ_DRQN is the fastest in all algorithms. The RQ method introduces the reward set rs_t to correct $Q(\varphi, a|\theta_i)$, which can compensate for the inaccurate output of the neural network in the initial stage of training. It is worth noting that RQ_ADSA_DRQN also adopts the RQ method, but its convergence speed is lower than RQ_DRQN. The reason can be found by analyzing the average action value *avg_qs* in the following.

2) THE AVERAGE STATE-ACTION VALUE *avg_qs*

avg_qs is a more stable indicator compared with *avg_rewards*. As shown in Fig. 9, the performance of DQN is still the worst in all models from the view of increasing speed and absolute value. According to Bellman formula, the update of state-action value is closely related to the reward value. The reward of DQN is always much smaller than that of other models, which also leads to the smaller *avg_qs*. Because the reward value of the samples in the important memory pool is higher than that of normal sample, the average action values of the two algorithms using the adaptive sampling mechanism are greater than those of other algorithms. The state-action value of ADSA_DRQN is the largest, which does not mean that this algorithm is superior to other algorithms. The reward value is basically positive except for the collision reward, which means that the target value $r + \gamma maxQ\left(\varphi_{t+1}, a_{t+1}|\theta_i'\right)$ is likely to be greater than the current value $Q(\varphi_t, a_t|\theta_i)$. Therefore, a large loss value causes the current value $Q(\varphi_t, a_t|\theta_i)$ to increase. As shown in Fig. 10, the loss of ADSA_DRQN is always greater than that of RQ_ADSA_DRQN in 40-60 episodes, which causes the state-action value of ADSA_DRQN to continue to increase. The state-action value of RQ_ADSA_DRQN is larger than that of RQ_DRQN at the beginning of training. According to (18), the first item increases while the second item decreases, which is equivalent to reducing the influence of rs_t. It explains that the convergence speed of RQ_ADSA_DRQN is lower than RQ_DRQN.

3) THE AVERAGE LOSS *avg_loss*

As shown in Fig. 10, The loss value of RQ_DRQN is larger than that of other models in the initial stage of training. Because RQ method chooses an action by considering two factors: $Q(\varphi, a\,|\,\theta_i)$ and rs_t. For example, $Q(\varphi, a_1\,|\,\theta_i)$ is bigger than $Q(\varphi, a_2\,|\,\theta_i)$, but the reward of action a_1 is bigger than that of action a_2. It is possible that the final choice of action is a_2, which would cause large loss when $Q(\varphi, a_2|\theta_i)$ is updated. The large loss value can produce a large gradient, which accelerates the updating speed of the parameters. The learning speed of RQ_DRQN is much faster than other models at the beginning. However, the sharp increase of loss value at 90 episodes shows that the stability of the algorithm is insufficient. It is difficult to judge the convergence of ADSA_DRQN and RQ_ADSA_DRQN from the trend of the average reward value, but it can be clearly seen from the *avg_loss* that the convergence of the latter is better than the former. The performance of DQN is still the worst, but the loss value gradually decreases from 60 episode, indicating a convergence trend.

4) THE NUMBER OF ROUNDS *num_round*

It reflects the ability of UAV to avoid obstacles. As shown in Fig. 11, DQN can learn an effective policy to avoid obstacles, but lack of temporal information restricts the ability

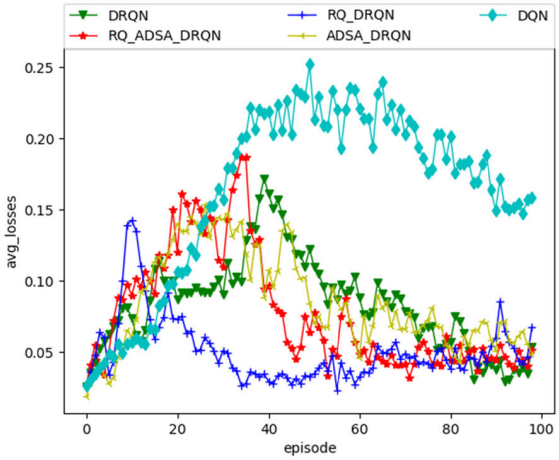


FIGURE 10. The average loss in a static scenario.

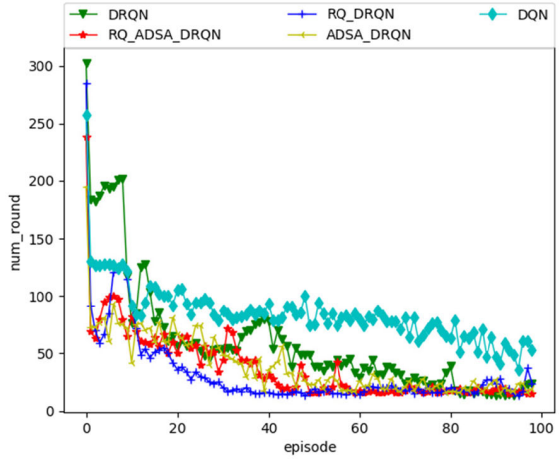


FIGURE 11. The number of rounds in a static scenario.

of obstacle avoidance. Although four algorithms based on DRQN have different convergence speeds, they can learn a good obstacle avoidance policy.

5) NUMBER OF TIMES TO REACH TARGET POINT

num_success

The most important requirement for path planning is to reach the target point. As shown in Fig. 12, the *num_success* reflects the cumulative number of times to reach the target point. The slope of curve indicates the number of times to reach the target point per episode, reflecting the stability of the algorithm convergence. DQN never reaches the target point throughout the training process, which indicates that this algorithm is not suitable for path planning in a large-scale environment. The DRQN reaches the target point at 70 episodes, indicating that the learning of algorithm is relatively slow. The small slope value indicates that the algorithm cannot learn the policy to reach the target point well. RQ_DRQN starts to arrive the target point at 20 episodes, which indicates that the learning efficiency is relatively high. The UAV cannot

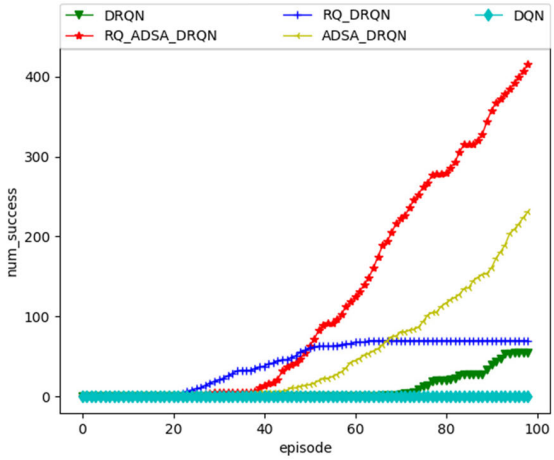


FIGURE 12. The number of times to reach the target point in a static scenario.

arrive the target point from 50 episodes, which indicates that RQ_DRQN learns a good obstacle avoidance strategy rather than a strategy to reach the target point. The convergence speed of RQ_DRQN is the fastest. If the instability phenomenon at the end of training is ignored, it can be said that it is the best performing algorithm. But *num_success* shows that this algorithm has a major flaw. After 100 episodes, the number of times to reach the target point exceeds 200 for ADSA_DRQN. The learning speed of RQ_ADSA_DRQN is second only to RQ_DRQN. The slope is much larger than the other four algorithms. The number of times to reach the target point exceeds 400, which is better than other algorithms.

The five algorithms are compared by the above indicators. The convergence speed of RQ_ADSA_DRQN is lower than that of RQ_DRQN, and other indicators are far superior to these four algorithms. The performance of ADSA_DRQN is second only to RQ_ADSA_DRQN. RQ_DRQN has a very fast learning speed and learns good obstacle avoidance policy. However, its success rate is relatively low, and there are problems of instability and divergence in the final stage of training. Although the indicators are not as good as the above three improved algorithms, DRQN also can successfully plan the available path. DQN still can't reach the target point after 100 episodes training, which is the worst performance among all algorithms.

In this article, we choose the best performing algorithm RQ_ADSA_DRQN to be used for path planning in a fixed and large-scale scenario, and the result of path planning is shown in Fig. 13. When there is an obstacle in front, the UAV turns to find a relatively low saddle between the two obstacles and climbs over it. From the topographic map, we can see that the UAV needs to climb to cross three saddles. If there are no obstacles ahead, the UAV would drop to a lower altitude because a high penalty term is set in the reward function. When the UAV detects an obstacle in front, it tends to take a climbing action instead of a turning action to avoid the

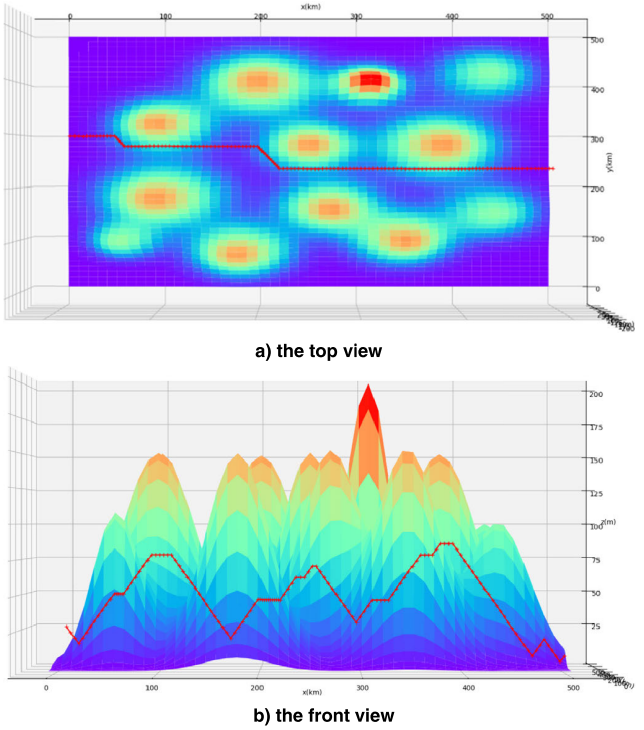


FIGURE 13. Path planning by RQ_ADSA_DRQN in a static scenario.

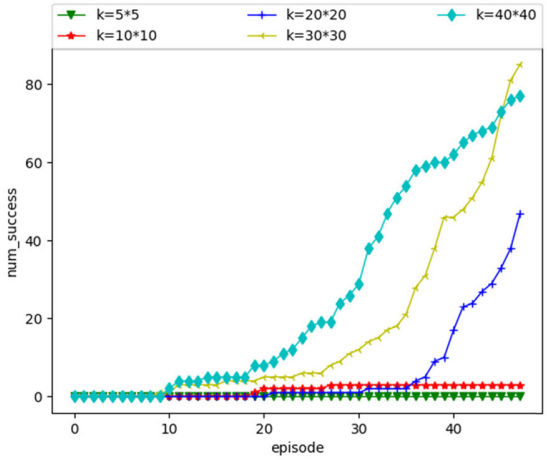


FIGURE 14. Influence of different k values on RQ_ADSA_DRQN.

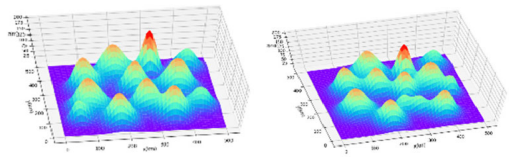


FIGURE 15. The different three-dimensional topographic maps.

obstacle. This phenomenon is closely related to the setting of the reward function.

The size of probabilistic safety points k is a very important parameter. The larger the k value means the more environmental information the algorithm obtains, which is theoretically beneficial to UAV obstacle avoidance. However, a large k value brings about an increase in the amount of calculation. Therefore, the choice of k value needs to consider the balance of efficiency and profit. After 100,000 training steps, we choose the number of successes to reach the target point as the evaluation of the impact of different k values. As shown in Fig. 14, we can clearly find that the performance of the algorithm increases as the value of k increases. The performance of the algorithm is obviously the best at k =30*30 in the later stage of training. The algorithm performs poorly when k is less than 10*10. We have found the possible reasons by analyzing the UAV motion and input sequence state length. The number of actions that can be executed is 11, which the distance of x-direction movement of 9 actions is 4 km. The length of the state sequence is 7. Then we can easily conclude that the distance in the x-direction of the state is likely to be 28 (4 * 7) km. If the k value exceeds the scale of the input state, it would cause information redundancy, which is not conducive to the learning of the algorithm.

As shown in the Fig. 15, there are two different three-dimensional topographic maps. After 100000 training steps, the performance of the three algorithms is shown in the Table 2. RQ_ ADSA_ DRQN is much better than the other

TABLE 2. The performance of DQN, DRQN, RQ_ADSA_DRQN in different three-dimensional maps.

Result \ *Environment*		ENV 1	ENV 2	Average
Avg_rewards	DQN	0.057	0.052	0.0545
	DRQN	0.196	0.202	0.199
	RQ_ADSA_DRQN	**0.246**	**0.257**	**0.2515**
Num_success	DQN	0	0	0
	DRQN	6	0	3
	RQ_ADSA_DRQN	**32**	**47**	**40.5**

two algorithms in terms of average reward value and times of reaching the target point.

The starting point of UAV is fixed in the above simulation test. We would evaluate the performance of the three algorithms at different starting points. There are 100000 training steps, and the starting point $[X_S, Y_S, Z_S]$ of UAV is set randomly. When the height of the random starting point is less than the height of the three-dimensional terrain, the random starting point is generated again.

$$X_S \in [0, 100], Y_S \in [200, 300], Z_S \in [20, 60]$$

The performance of the three algorithms at different starting points in the ENV 1 is shown in Table 3. Because the starting point of random generation is closer to the target point, the three algorithms obtain higher reward values. RQ_ ADSA_ DRQN is still the best performing algorithm.

We propose a new action selection strategy that combines the Q value and current reward value, where $g \in (0,1)$ controls the impact of the current reward value on the action selection.

TABLE 3. The performance of DQN, DRQN, RQ_ADSA_DRQN at different starting points.

Result \ Algorithm	DQN	DRQN	RQ_ADSA_DRQN
Avg_rewards	0.103	0.211	**0.248**
Num_success	1	16	**34**

TABLE 4. The performance of RQ_ADSA_DRQN at different g values.

g value \ Result	*Avg_reward*	*Num_success*
0	0.228	25
0.3	0.245	28
0.5	**0.257**	**47**
0.7	0.199	16

TABLE 5. The performance of RQ_ADSA_DRQN at different τ values.

τ value \ Result	*Avg_reward*	*Num_success*
0	0.248	54
0.2	0.243	12
0.4	0.257	47
0.6	**0.272**	**110**

The value in the second term of the (18) tends to zero as the Q value increases, and the g value controls the rate of change. There are 100000 training steps for RQ_ADSA _DRQN in the ENV 2. Table 4 shows the effect of different g values on the performance of the RQ_ADSA_DRQN algorithm in the ENV 2. The average reward of the algorithm is the lowest as $g = 0.7$. Therefore, the g value needs to consider the balance between the current reward and the long-term reward.

If the round ends, the ADSA method randomly would generate a number $v \in [0,1]$. If $v < \tau$, the ADSA method would sample from the important pool. The probability of the algorithm sampling from the important pool increases with the increase of τ. There are 100000 training steps for RQ_ADSA _DRQN in the ENV 2. As shown in Table 5, a large τ value is helpful to improve the performance of the algorithm.

C. COMPARISON OF ALGORITHM PERFORMANCE IN A DYNAMIC SCENARIO

We train five algorithms in different scenarios. There are four obstacles in the environment, which the y-direction position of obstacle randomly samples with a range of [200, 400] per round. In the above comparison of the algorithms, it is found that the following three indicators are more discriminative: *avg_rewards*, *num_round*, *num_success*.

As illustrated in Fig. 16, two algorithms based on adaptive sampling can effectively increase the reward value. RQ_ADSA_DRQN has the best performance in terms of convergence speed, stability, and absolute value. RQ_DRQN has obvious instability as it appears in the dynamic environment. It is important to note that the average reward of DQN is greater than that of DRQN. The main reason is that the environment changes randomly every round. Perception of the environment constructed by extracting historical trajectory

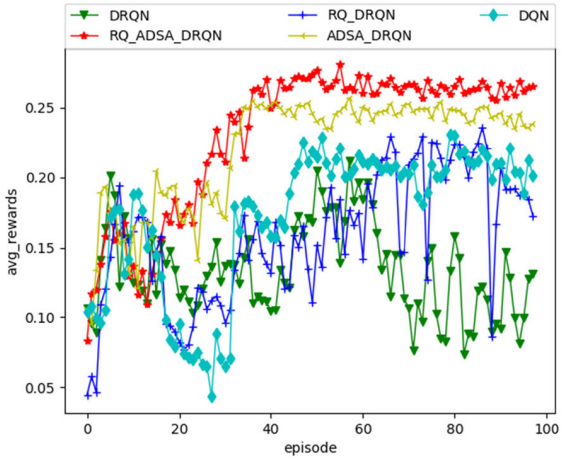


FIGURE 16. The average reward in different scenarios.

information becomes inaccurate as the environment changes randomly. DQN only needs to pay attention to the obstacle information currently perceived. In addition, few obstacles are beneficial to DQN for higher reward compared to the static complex scenario. As depicted in Fig. 17, adaptive sampling exhibit significant improvements over other algorithms. RQ_ADSA_DRQN and ADSA_DRQN can stably avoid all obstacles in the late training period. As shown in Fig.18, RQ_DRQN reaches the target point the most times, but it doesn't learn a good obstacle avoidance strategy. By contrast, two algorithms using adaptive sampling have learned a good obstacle avoidance strategy, but the success rate is not high. We think the reason comes from two aspects. First of all, since obstacles are set randomly in the environment, the difficulty of the UAV reaching the target point is also relatively random. Secondly, it is still possible for the UAV to obtain a large reward value if it fails to reach the target point. Even compared with the rewards of mission success, the reward value approaching the target point is not small. To solve the above problem, we try to reduce the reward related to the distance to the target, but the experimental results show that the learning efficiency becomes very slow. We have to choose a large reward related to the target distance. Therefore, the strategy learned by ADSA_DRQN is to approach the target without collision.

To validate the effectiveness of our proposed approaches, a well-trained RQ_ADSA_DRQN is dispatched to execute path planning in a dynamic scenario, and the result of path planning is shown in Fig. 19. The positions of the four obstacles in the environment are always changing with the time steps. There are four moving obstacles in the environment, and the results show that the UAV can easily avoid obstacles.

D. COMPARISON OF ALGORITHM PERFORMANCE IN AN INDOOR SCENARIO

In order to evaluate the practical application value of our approach, we build a simulation indoor scenario based on

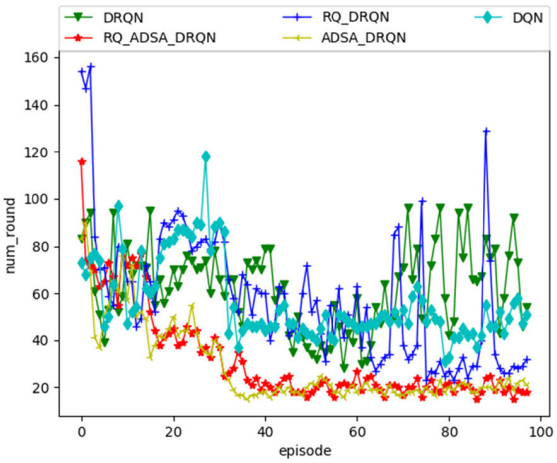


FIGURE 17. The number of rounds in different scenarios.

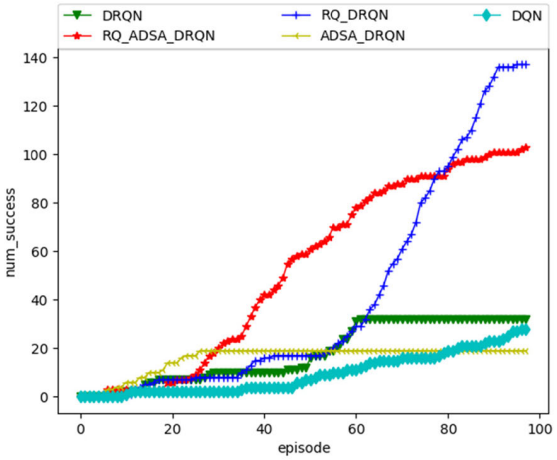


FIGURE 18. The number of times to reach the target point in different scenarios.

V-REP which is more similar to the real environment. As shown in Fig. 20, the new simulation scenario is basically the same as the real indoor environment, including sofa, wardrobe, shelf, pedestrian, window, etc. The UAV starts from the left side, avoiding the wardrobe, shelf and other obstacles, passes through the window to the right side.

ACTION SPACE: The UAV can get abundant surrounding information in the scenario, and the forward distance of each step is set relatively small. Therefore, the indoor simulation scenario does not require a strong horizontal maneuver capability of the UAV. There are 9 actions in the action space without 63 degrees of horizontal turning, compared with the actions in the previous three-dimensional topographic map.

INPUT STATE: The status input of the UAV includes vision sensor and range sensor to obtain environmental information in front of the movement. As shown in Fig. 21, the green box represents the image obtained by the vision sensor, and the red points represent the distance information obtained

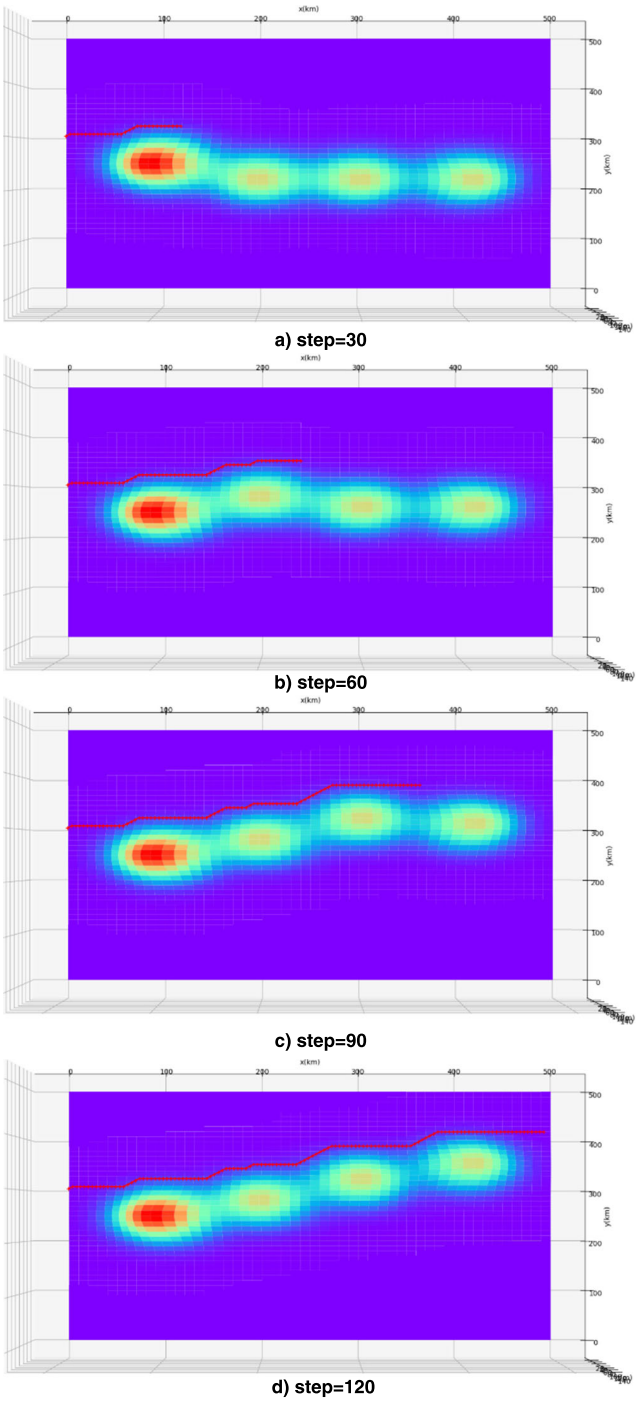


FIGURE 19. Path planning by RQ_ADSA_DRQN in a dynamic scenario, the y-coordinate of the obstacle changes with time.

by the range sensor. The input image of vision sensor is $256 * 256 * 3$, and we convert the color image into a gray image. Because color information has little effect on obstacle avoidance decision-making, and the gray image compared with the color image is less different between the real and simulated environments. The range sensor obtains the distance information of 16 points in front of the UAV. In addition, the UAV also needs to obtain position information relative to the target point.

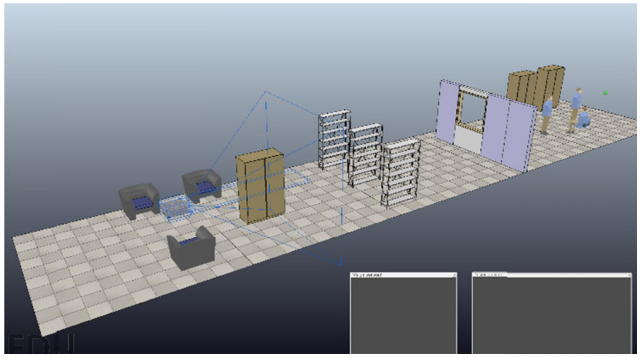


FIGURE 20. The indoor simulation scenario based on V-REP.

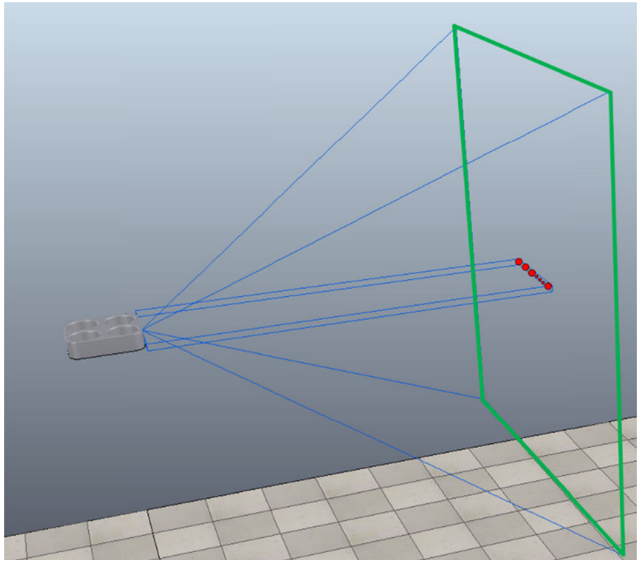


FIGURE 21. The UAV is equipped with the range sensor and the vision sensor.

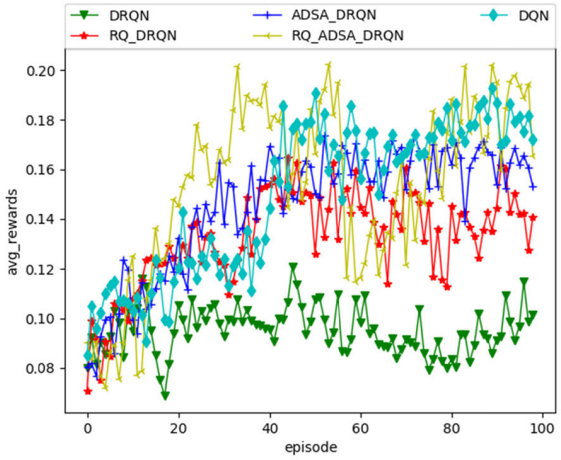


FIGURE 22. The average reward in an indoor scenario.

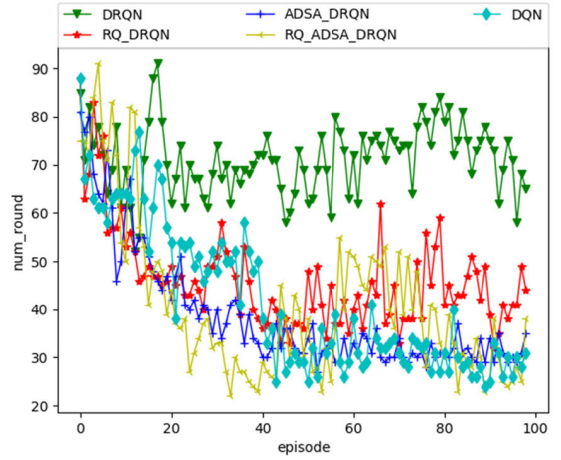


FIGURE 23. The number of rounds in an indoor scenario.

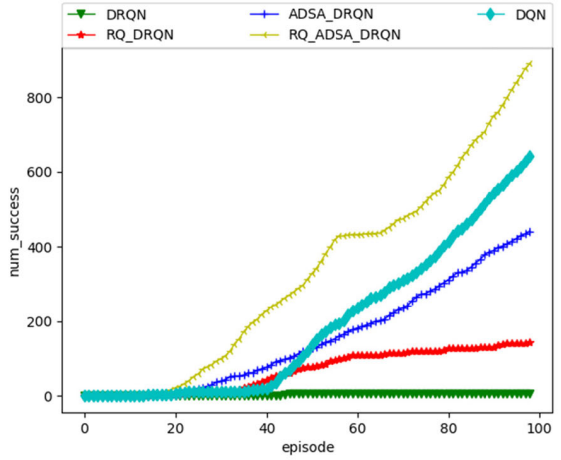
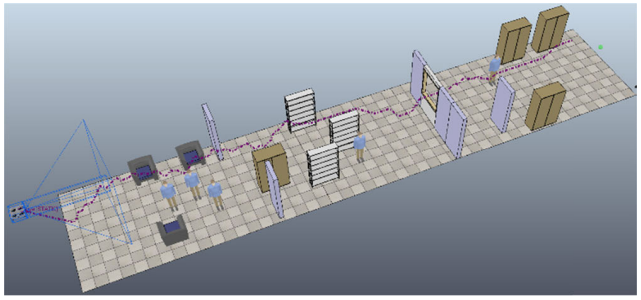


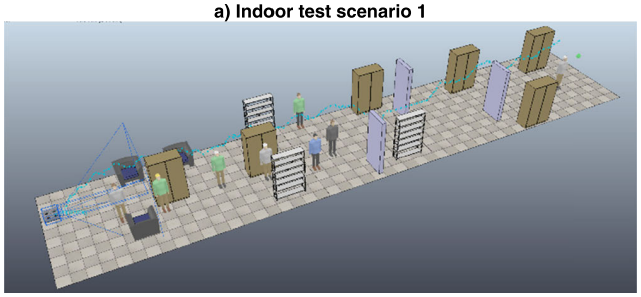
FIGURE 24. The number of times to reach the target point in an indoor scenario.

There are 100 episodes in training to compare the performance of different algorithms, and each episode has 5000 steps. We choose three more discriminative indicators to evaluate the performance of different algorithms, including: *avg_rewards*, *num_round*, *num_success*.

As shown in Fig. 22, the performance of DQN is far better than that of DRQN, and even partially exceeds the improved algorithm for DRQN. Because the UAV can easily reach the destination as long as successfully avoiding obstacles. The UAV needs to pay attention to the current obstacles instead of remembering the previous states and actions. The indoor obstacle avoidance scenario can be described as MDP rather than POMDP. There is a similar phenomenon in Atari's game. DRQN performs worse than DQN for Beam Rider, and we don't expect DRQN to outperform DQN for MDPs [24]. The improved methods we propose can increase the reward value compared with DRQN. The convergence stability of ADSA_DRQN is better than other algorithms, which further verifies the effectiveness of the adaptive sampling mechanism. RQ_ADSA_DRQN has the fastest learning speed and the highest reward value in the early training period, but it is unstable in 60-80 rounds. It is speculated that the instability caused by RQ method, and similar situation occurs in both



a) Indoor test scenario 1



b) Indoor test scenario 2

FIGURE 25. Indoor unknown test scenarios.

the static and dynamic three-dimensional maps. As shown in Fig. 23, The small number of rounds shows that the agent can stably avoid obstacles, and ADSA_DRQN still performs well in stability. As shown in Fig. 24, The number of times that DRQN reaches the target point is 0, which is far behind DQN. The performance of RQ_ADSA_DRQN even exceeds DQN. The two methods we propose greatly improve the performance of the DRQN algorithm.

Our method relies on local information rather than global information. The obstacle avoidance ability of the algorithm is tested in two new unknown scenarios. As shown in Fig. 25, the UAV can successfully avoid obstacles in new environment. The flight trajectory is drawn with purple and blue lines respectively. The results show that UAV can successfully complete obstacle avoidance task in unknown environment.

V. CONCLUSION

This article develops a deep reinforcement learning algorithm for the UAV path planning in the complex and dynamic environment, and proposes two methods to speed up algorithm learning. First of all, we propose a RQ method to address the problem of inaccurate network prediction at the initial stage of training, which can reduce the blindness of action selection and the probability of collision. Secondly, we propose a simple and effective adaptive sampling mechanism. It Constructs two memory pools based on the average round reward value, and then adaptively samples based on the failure rate with low computational complexity. To validate the effectiveness of our methods, we built a three-dimensional map simulation environment based on (1) and a more realistic indoor simulation environment based on the V-REP simulation platform. The results show that the RQ method can greatly improve the convergence speed of the algorithm, and ADSA mechanism can improve the speed and stability of convergence. It is verified that our method has effective obstacle avoidance ability in a dynamic environment.

It is worth noting that the performance of DQN and DRQN are not the same in the two simulation environments. The performance of DRQN in the three-dimensional complex environment is much better than that of DQN, and the opposite results appear in the V-REP simulation environment. For the POMDP model, DRQN performs better than DQN. The method we proposed can greatly improve the performance of DRQN for MDP model.

The input information of the algorithm contains the relative location information of the UAV and the nearby environment information detected by the sensor. Simulation results show that the algorithm can ensure UAV to avoid obstacles successfully in dynamic and complex environment. This research proves that UAV only rely on limited range sensors and image information to complete obstacle avoidance tasks in indoor environments with many obstacles. Deep reinforcement learning can complete the fusion of various sensor information and make reasonable decisions. The method we propose can improve the performance of existing algorithms, which has application value for the development of autonomous driving technology. Therefore, the research in this article has practical significance.

Reinforcement learning plays an important role in autonomous decision of UAV and automobile. It is an important research direction that reinforcement learning uses multi-sensor to achieve more reasonable decision-making. Different types of sensors have different information redundancy. It is a potential research hotspot to make use of attention mechanism to process sensor information in different channels. In addition, the reliability of decision-making in reinforcement learning method is an urgent problem to be solved. Some traditional planning methods can be used as auxiliary decision-making to improve the reliability of reinforcement learning.

REFERENCES

[1] L. D. Tran, C. D. Cross, M. A. Motter, J. H. Neilan, G. Qualls, P. M. Rothhaar, A. Trujillo, and B. D. Allen, "Reinforcement learning with autonomous small unmanned aerial vehicles cluttered environments," in *Proc. AIAA Aviation Technol., Integr., Oper. Conf.*, Dallas, TX, USA, 2015, p. 2899. [Online]. Available: https://arc.aiaa.org/doi/pdf/10.2514/6.2015-2899

[2] S. Scherer, J. Rehder, S. Achar, H. Cover, A. Chambers, S. Nuske, and S. Singh, "River mapping from a flying robot: State estimation, river detection, and obstacle mapping," *Auto. Robots*, vol. 33, nos. 1–2, pp. 189–214, Aug. 2012.

[3] B. Zhang, Z. Mao, W. Liu, and J. Liu, "Geometric reinforcement learning for path planning of UAVs," *J. Intell. Robot. Syst.*, vol. 77, no. 2, pp. 391–409, Feb. 2015.

[4] G. Nannicini, D. Delling, D. Schultes, and L. Liberti, "Bidirectional A* search on time-dependent road networks," *Networks*, vol. 59, no. 2, pp. 240–251, Mar. 2012.

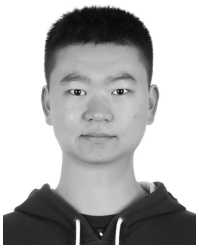
[5] R. A. Conn and M. Kam, "On the moving-obstacle path-planning algorithm of shih, lee, and gruver," *IEEE Trans. Syst., Man, Cybern. B. Cybern.*, vol. 27, no. 1, pp. 136–138, Feb. 1997.

[6] M. Gyu Park, J. Hyun Jeon, and M. Cheol Lee, "Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing," in *Proc. ISIE . IEEE Int. Symp. Ind. Electron. Process.*, Pusan, South Korea, 2001, pp. 1530–1535.
[7] F. Aurenhammer, "Voronoi diagrams—A survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, Sep. 1991.
[8] X. Meng, Y. Zhao, and Q. Xue, "Application of genetic algorithm in path planning," *Comp. Eng.*, vol. 34, no. 16, pp. 215–220, 2008.
[9] Y. Chen, J. Yu, Y. Mei, S. Zhang, X. Ai, and Z. Jia, "Trajectory optimization of multiple quad-rotor UAVs in collaborative assembling task," *Chin. J. Aeronaut.*, vol. 29, no. 1, pp. 184–201, Feb. 2016.
[10] C. Zhang, Z. Zhen, D. Wang, and M. Li, "UAV path planning method based on ant colony optimization," in *Proc. Chin. Control Decis. Conf.*, Xuzhou, China, 2010, pp. 3790–3792.
[11] M. Chen, Q. Wu, and C. Jiang, "A modified ant optimization algorithm for path planning of UCAV," *Appl Soft Comput.*, vol. 8, no. 4, pp. 1712–1718, 2008.
[12] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.
[13] Y. Liu, X. Zhang, Y. Zhang, and X. Guan, "Collision free 4D path planning for multiple UAVs based on spatial refined voting mechanism and PSO approach," *Chin. J. Aeronaut.*, vol. 32, no. 6, pp. 1504–1519, Jun. 2019.
[14] Y. Hao and Z. Zhao, "Real-time obstacle avoidance method based on polar coordination particle swarm optimization in dynamic environment," in *Proc. Conf. Ind. Electron. Appl.*, Harbin, China, 2007, pp. 1612–1617.
[15] H. Duan, Y. Yu, X. Zhang, and S. Shao, "Three-dimension path planning for UCAV using hybrid meta-heuristic ACO-DE algorithm," *Simul. Model. Pract. Theory*, vol. 18, no. 8, pp. 1104–1115, Sep. 2010.
[16] M. Darrah, W. Niland, and B. Stolarik, "Increasing UAV task assignment performance through parallelized genetic algorithms," in *Proc. AIAA Infotech Aerosp. Conf. Exhib.*, May 2007, pp. 7–10.
[17] Q. Yang and S.-J. Yoo, "Optimal UAV path planning: Sensing data acquisition over IoT sensor networks using multi-objective bio-inspired algorithms," *IEEE Access*, vol. 6, pp. 13671–13684, 2018.
[18] S. Garrido, M. Malfaz, and D. Blanco, "Application of the fast marching method for outdoor motion planning in robotics," *Robot. Auto. Syst.*, vol. 61, no. 2, pp. 106–114, Feb. 2013.
[19] V. González, C. A. Monje, L. Moreno, and C. Balaguer, "UAVs mission planning with flight level constraint using fast marching square method," *Robot. Auton. Syst.*, vol. 94, pp. 162–171, Apr. 2017.
[20] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.
[21] S. Lange and M. Riedmiller, "Deep auto-encoder neural networks in reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Barcelona, Spain, Jul. 2010, pp. 1–8.
[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," 2013, *arXiv:1312.5602*. [Online]. Available: http://arxiv.org/abs/1312.5602
[23] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," 2015, *arXiv:1511.05952*. [Online]. Available: http://arxiv.org/abs/1511.05952
[24] M. Hausknecht and P. Stone, "Deep recurrent Q-Learning for partially observable MDPs," 2015, *arXiv:1507.06527*. [Online]. Available: http://arxiv.org/abs/1507.06527
[25] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 1527–1533.
[26] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deep driving: Learning affordance for direct perception in autonomous driving," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 2722–2730.
[27] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *Int. J. Robot. Res.*, vol. 35, no. 11, pp. 1289–1307, Sep. 2016.
[28] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Shanghai China, Sep. 2017, pp. 1529–1538.
[29] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Solving deep memory POMDPs with recurrent policy gradients," in *Proc. Int. Conf. Artif. Neural Netw.*, Porto, Portugal, 2007, pp. 697–706.
[30] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.
[31] J. Q. Cui, S. Lai, X. Dong, and B. M. Chen, "Autonomous navigation of UAV in foliage environment," *J. Intell. Robotic Syst.*, vol. 84, nos. 1–4, pp. 259–276, Dec. 2016.
[32] J. Li, Y. Bi, M. Lan, H. Qin, M. Shan, F. Lin, and B. M. Chen, "Realtime simultaneous localization and mapping for UAV: A survey," in *Proc. Int. Micro Air Vehicle Conf. Competition (IMAV)*, 2016, p. 237. [Online]. Available: https://pdfs.sema-nticscholar.org/e6fb/502ca4dda8fef8e4bfe075396281f326e2dd.pdf
[33] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular visual-inertial SLAM-based collision avoidance strategy for fail-safe UAV using fuzzy logic controllers," *J. Intell. Robotic Syst.*, vol. 73, nos. 1–4, pp. 513–533, Jan. 2014.
[34] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.
[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.
[37] Y. Song, Y.-B. Li, C.-H. Li, and G.-F. Zhang, "An efficient initialization approach of Q-learning for mobile robots," *Int. J. Control, Autom. Syst.*, vol. 10, no. 1, pp. 166–172, Feb. 2012.
[38] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.
[39] M. Deisenroth and C. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, Bellevue, WA, USA, 2011, pp. 465–472.
[40] L. Lin, "Self-improvement based on reinforcement learning," *Planning Teaching. Mach. Learn. Proc.*, vol. 6, pp. 323–327, Sep. 1991.

RONGLEI XIE received the B.Eng. degree from Northwestern Polytechnical University, in 2015. He is currently pursuing the Ph.D. degree with the School of Aeronautic Science and Engineering, Beihang University, China. His current research interests include UAV intelligent mission planning and machine learning.

ZHIJUN MENG received the B.Eng. and Ph.D. degrees from the School of Aeronautic Science and Engineering, Beihang University, China. He is currently an Associate Professor with the Department. He has published more than 30 SCI articles and has authored more than 20 invention patents. His research interests include vision and machine learning-based autonomous target recognition/tracking/obstacle avoidance technology research.

LIFENG WANG received the Ph.D. degree from Beihang University, China, in 2008. He is currently an Associate Professor with the School of Aeronautic Science and Engineering, Beihang University. His current research interests include aircraft design and obstacle avoidance.



HAOCHEN LI received the B.Eng. degree from Beihang University, China, in 2017, where he is currently pursuing the master's degree with the School of Aeronautic Science and Engineering. His current research interests include UAV intelligent mission planning and image identification.

KAIPENG WANG received the B.Eng. degree from Beihang University, China, in 2012, where he is currently pursuing the Ph.D. degree with the School of Aeronautic Science and Engineering. His current research interests include UAV intelligent control and mission planning.

ZHE WU received the Ph.D. degree from the Harbin Institute of Technology, China, in 1988.

He is currently a Professor with the School of Aeronautic Science and Engineering, Beihang University. He used to be the Director of the Department of Aircraft Design and Applied Mechanics, and the Vice President of Beihang University. His current research interests include aircraft design and aircraft stealth design. He was selected as a Young and Middle-Aged Expert with outstanding contributions at the national level, in 1998. He was appointed as a Yangtze River Scholar Distinguished Professor in 2005.