Received 5 April 2023, accepted 27 April 2023, date of publication 4 May 2023, date of current version 10 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3273164

RESEARCH ARTICLE

B-APFDQN: A UAV Path Planning Algorithm Based on Deep Q-Network and Artificial Potential Field

FUCHEN KONG, QI WANG, (Member, IEEE), SHANG GAO, AND HUALONG YU
School of Computer, Jiangsu University of Science and Technology, Zhenjiang 212003, China

Corresponding author: Qi Wang (wangqi@just.edu.cn)

ABSTRACT Deep Q-network (DQN) is one of the standard methods to solve the Unmanned Aerial Vehicle(UAV) path planning problem. However, the way agent deepens its cognition of the environment through frequent random trial-and-error leads to slow convergence. This paper proposes an optimized DQN with Artificial Potential Field (APF) as prior knowledge called B-APFDQN for path planning. Replacing the traditional neural network which has only one Q-value output with a multi-output neural network to promote the training process in combination with APF. Furthermore, a SA-ε-greedy algorithm that can automatically adjust the stochastic exploration frequency with steps and successes is proposed in order to prevent the agent from falling into local optimum. We remove the nodes that do not affect the path connectivity and apply the B-spline algorithm to make the path shorter and smoother. Simulation experiments show that the proposed B-APFDQN algorithm performs better than the classical DQN, has a strong ability to avoid falling into local optimum, and the obtained paths are smooth and shorter, which proves the effectiveness of B-APFDQN in the UAV path planning problem.

INDEX TERMS Reinforcement learning, deep Q-network, artificial potential field, path planning.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are easier to integrate with information technology due to their tiny size, convenience, flexibility, and numerous other advantages. Networked information transmission, digitization of combat space, and intelligent flight platforms will become the main trends in the evolution of the UAV technique [1]. UAVs have been widely applied in multitudinous fields, especially in communication, cloud computing, and the development of big data. In the sphere of communication, the combination of communication technology and UAV technology makes it an essential means of auxiliary communication [2], [3], [4]; in the field of cloud computing, cloud computing technology is combined with UAV technology to invite it a tool for information acquisition, extract valid information from the data collected by the UAV and interact with the consumers in real-time with cloud computing. The most typical examples are in agriculture and forestry [5], [6] and industrial production [7], [8], [9], [10] applications; in the domain of big data, the massive amount of data collected by the sensor nodes onboard drones is fully utilized, and the potential value contained in the information is explored by big data technology [11], [12], [13], [14].

Whether in communications, cloud computing, big data or other fields, reasonable and optimal mission planning is a necessary prerequisite for UAVs to play their role fully and complete their tasks efficiently. Mission planning for UAVs consists of task assignment and path planning [15], [16]. The focus of this paper is on the UAV path planning problem. UAV path planning refers to the design of the optimal flight path to ensure that the UAV completes a specific flight mission based on the energy limitation and the actual mission requirements, avoiding obstacles and threat areas in the process of completing the mission. However, energy limitation caused by UAV loads and special mission existence limit the broader application of UAVs [17], [18].

The associate editor coordinating the review of this manuscript and approving it for publication was Guillermo Valencia-Palomo.

The UAV path planning needs to have the properties of feasibility, safety, and the shortest path. Feasibility means that the planned trajectory must be executable, and the path must meet the dynamic constraints so that specific performance indicators meet the requirements, such as the limit of the longest flight time and the limit of the maximum energy consumption. Safety means that the path must ensure that the aircraft avoids obstacles, no-fly zones, and other potentially dangerous areas. The shortest path means that the flight distance is as short as possible on the premise of completing the set task. Reinforcement learning (RL) is one of the standard methods to solve UAV path planning problems [19], [20], [21]. RL deepens cognition of the environment through continuous interaction between the agent and the environment, and updates the optimal policy guided by the feedback from the environment to obtain a model that can fulfill our expectations. Deep reinforcement learning (DRL) combines the robust feature extraction and representation capabilities of deep neural networks with the powerful decision-making capabilities of RL. Nevertheless, for UAVs, we can only gather partial information about the entire environment due to the limitation of sensor perception range. Therefore, in the practice of path planning for UAVs with RL, it is necessary to implement a Partially Observable Markovian Decision Process (POMDPs) [22], [23], [24] with this limited information. In this process, the frequent random trial-and-error performed by the agent to overcome the lack of cognition of the environment leads to slow convergence of the DRL algorithm.

In contrast, the trajectories obtained by classical path planning algorithms are difficult to balance safety and achieve the shortest path. This paper proposes a path planning algorithm based on Artificial Potential Field (APF) to adapt Deep Q-network (DQN) to solve the above-mentioned problem. The main contributions of this paper are as follows:

1) We adopt APF as assistant for DQN action selection and initiated a new network structure that can output both Q-values and action distribution instead of the classical DQN network structure that can only output Q-values, which speeds up the convergence of the algorithm.

2)The proposed design of SA-ε-greedy with the agent allowing some deviation from the guidance of APF speeds up the convergence and can overcome the disadvantage of unreachable target points brought by APF.

3) Removing redundant nodes and adopting B-spline for further optimization makes the path smoother and shorter than other similar methods.

The rest of this article is organized as follows. Section II introduces some recent research progress related to UAV path planning and focuses on APF and DQN adopted in this paper; section III describes how this paper combines the DQN and APF. A new network structure is designed for fusion, and a path planning policy is generated; section IV compares the path planning model proposed in this paper with classical DQN in various grid environments and compares it with the recent path planning algorithm. Comparisons and comprehensive evaluations are presented; section V summarizes the fusion algorithms and experimental results.

II. RELETED WORK

Reasonably planning a path that meets the mission requirements is one of the crucial reasons why UAVs can be successfully applied in various fields. We can roughly divide UAV-related path planning algorithms into the following five types: algorithms based on graph search technology, such as Voronoi diagram [25], [26], [27], A-Star [28], [29], [30], [31], Dijkstra [32], [33], etc. These algorithms are easy to implement and fast to construct. However, they are often based on the environment estimation in the search process, lacking the correct information from the environment feedback. Therefore, they risk falling into a local optimum.

Probability-based methods, such as Probabilistic Roadmap (PRM) [34], [35], Rapid-exploration Random Tree (RRT) [36], [37], [38], etc., generate paths by randomly sampling the task space to be planned and connecting the sampled points according to some rules. In theory, both PRM and RRT have probabilistic completeness [39], [40]. If there is a path solution between the starting point and the target point, then a solution must be found by continuously increasing the number of sampling points to satisfy the search duration. Nevertheless, because it is based on a random sampling of the task space, the quality of the obtained path cannot be guaranteed, and the result is often sub-optimal.

Heuristic-based methods, such as Ant Colony Algorithm (ACO) [41], [42], Particle Swarm Algorithm (PSO) [43], [44], Simulated Annealing (SA) [45], [46], etc. Most heuristic algorithms are stochastic optimization algorithms that imitate the laws of nature, so the initial conditions can significantly affect the quality of the optimized results. For example, in the initial PSO particles setting, the initial attribute of the particles determines whether the particles have the potential to be optimized to the global optimum. In addition, the heuristic-based algorithms suffer from drawbacks such as a slow convergence rate, complex operators, long computational time, the need to tune many parameters, and design for only real or binary search space [47], [48].

The other two are the RL-based methods and the field-based methods. RL requires the agent to interact continuously with the environment, and the agent deepens its cognition of the environment according to the reward function obtained by the environment feedback. The purpose is to let the agent learn an optimal policy that maximizes rewards or other user-provided reinforcement signals accumulated from immediate rewards. The most representative ones are Q-Learning [49], [50] and DQN [51], [52]. Q-learning is a model-free tabular approach [53], [54]; that is, Q-Learning can be applied to situations where the environment is unknown. The table is updated by interacting with the environment to find the optimal policy. DQN is based on Q-Learning and applies neural networks as function approximators instead of Q-tables to solve the problem that tables are challenging to store continuous state space with continuous actions, or the state dimension is too large with the action dimension. Furthermore, the target network technique with an experience replay mechanism is introduced.

The potential field-based method assumes that the agent is affected by a potential field in space [55], [56], [57], the target point will have a gravitational force on the agent, and the obstacles will have a repulsive force on the agent. Under the combined effect of the target point's gravitational force and the obstacle's repulsive force, the agent avoids the obstacle and gradually approaches the target point.

Research on solving path-planning problems using the above algorithms has become increasingly sophisticated. Wang et al. [58] describe and thoroughly classify the above algorithm on the shipboard aircraft scheduling problem. In the study of graph-based methods for path planning problems, Luo et al. [59] proposed an extended Dijkstra algorithm that employs triangulation to model the surface environment and equivalently converts the triangle mesh of a surface to a triangle on a two-dimensional plane, thereby solving the surface path planning of a mobile robot. Wang et al. [60] employ a hybrid A* algorithm to generate coarse paths and then implement safe dispatch corridors along the coarse paths by resampling to solve the autonomous dispatch trajectory planning problem on the flight deck. In the study of probability-based methods for path planning problems, Ravankar et al. [61] propose an improved sampling-based path planning method for mobile robot navigation, which uses a layered hybrid PRM and the APF method for global planning. Wang et al. [62] proposed an improved artificial potential field (IAPF) and model predictive control (MPC) techniques for solving autonomous navigation for unmanned surface vessels (USVs) in unknown environments. In the study of heuristic-based methods for path-planning problems, Song et al. [63] proposed an improved PSO algorithm combined with continuous high-degree Bessel curves to solve the problem of planning smooth paths for mobile robots. Li et al. [64] proposed an improved ant colony optimization-artificial potential field method for path planning using improved ACO to search for the globally optimal path and then using an improved APF to avoid unknown obstacles for unmanned ground vehicles (UGVs). In the study of RL-based methods for path planning problems, maw et al. [65] proposed a hybrid path planning algorithm using an anytime graph-based path planning algorithm for global planning and DRL for local planning. Li et al. [66] proposed a path planning algorithm that combines global path planning with improved A* and local path planning with improved Q-learning to solve the path planning problem of UAVs. In the study of potential field-based methods for path planning problems, Wang et al. [67] proposed a UAV path planning algorithm combining convolutional neural networks with the RRT* algorithm, using the A* algorithm to generate map information and optimal paths to form a training data set. Yafei et al. [68] proposed a combination of APF and RRT to enable UAVs to quickly and effectively avoid threat areas.

The following parts will focus on the RL-based DQN algorithm and APF.

A. DEEP Q-NETWORK

RL problems discuss how one or more agents learn optimal policy based on their observations and the feedback provided by the environment [69], [70], [71]. The policy expresses a mapping relationship from state space S to action space A. In each time step, the agent takes action according to its state through the learned policy and observes the reward $r(s_t, a_t)$ from the environment. We want to maximize the balance between the current and the cumulative reward of the discount coefficient γ of future returns $R_t = \sum_{i=t}^{T} r^{i-t} r_i$.

For policy π, the value of an action in a particular state can be expressed as $Q^\pi(s, a) = \mathbb{E}_{s,a,\pi}[\sum_{t=1}^{\infty} \gamma^t r_i]$, this expectation indicates the initial state s, the initial action is the policy π selects the subsequent actions, and the optimal value is $Q^*(s, a) = \max_\pi Q^\pi(s, a)$, then the optimal policy is

$$\pi^* = \arg\max_a Q^*(s, a) \tag{1}$$

which the policy that maximizes the Q function is the optimal policy. The state space and action space faced by RL agents in practical robotics applications are often continuous, or the number of observable states and optional actions is large, which makes optimization problems very difficult, such as the same tabular encoding as Q-Learning is no longer applicable, so a function approximator $Q(s, a; \theta) \approx Q^*(s, a)$ is often applied to estimate the Q function. As we know, the deep neural network can automatically learn features from different forms of data and has a solid, expressive ability. It is better for RL agents to apply a deep neural network instead of a tabular encoding method.

Unlike Q-Learning, DQN learns from empirical sampling transitions rather than complete online learning, and DQN applies a target network with a parameter of θ_t^-, where the value of θ_t^- comes from copying the parameters of the learning network in stages θ_t^-.

For a deep Q-network, the loss can be defined as

$$L(\theta_t) = \mathbb{E}_{s,a,\pi}[(Y_t - Q(s_t, a_t; \theta_t))^2] \tag{2}$$

where Y_t is the target value generated by the target network as

$$Y_t = R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta_t^-) \tag{3}$$

The DQN is very effective in dealing with the problem of UAV path planning. However, due to the unfamiliarity of the unknown environment in the early stage of training, the agent often needs to spend more time exploring and determining the optimal policy through continuous trial-and-error. It leads to the fact that algorithms adopting DQN often take a long time to train and consume many computing resources.

B. ARTIFICIAL POTENTIAL FIELD

The APF algorithm was first proposed by Oussama Khatib et al. in their 1986 paper [55]. The APF method assumes that the target point exerts a gravitational force on the agent to guide the agent toward it, while the obstacle exerts a repulsive force on the agent to avoid the collision with it. The force on the agent's path is equal to the sum of the repulsive force and the gravitational force at this point.

For the gravitational field function:

$$U_{att}(q) = \frac{1}{2}\xi\rho^2(q, q_{goal}) \tag{4}$$

where ξ is the gravitational gain, and $\rho^2(q, q_{goal})$ is the distance between the current agent's position q and the target point q_{goal}.

Then the gravitational force generated by the gravitational field can be expressed as:

$$F_{att}(q) = -\xi\rho(q, q_{goal}) \tag{5}$$

The closer the agent is to the target point, the weaker the gravitational effect of the target point on the agent, according to (4), and the less the gravitational force on the agent, according to (5).

For the repulsion field function:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, & D(q) \leq Q^* \\ 0, & D(q) \geq Q^* \end{cases} \tag{6}$$

where, η is the repulsion gain, $D(q)$ is the distance between the agent and the nearest obstacle, and Q^* is the threshold for the obstacle to generate repulsion. If this threshold is exceeded, the obstacle will not generate repulsion to the agent. Then the repulsive force generated by the repulsive force field can be expressed as:

$$F_{rep}(q) = \begin{cases} \eta(\frac{1}{D(q)} - \frac{1}{Q^*})(\frac{\nabla D(q)}{D^2(q)}), & D(q) \leq Q^* \\ 0, & D(q) \geq Q^* \end{cases} \tag{7}$$

When the agent is within the range of the effect of the repulsive force of the obstacle, the closer the agent is to the obstacle, the more significant the effect of the repulsive field, according to (6). Therefore the greater is the repulsive force according to (7).

Therefore, the resultant force experienced by the agent is the superposition of gravitation and repulsion:

$$F(q) = F_{att}(q) + F_{rep}(q) \tag{8}$$

The gravitational force generated by the target point will guide the agent towards the target point to complete the task, while the repulsive force generated by the obstacle will keep the agent away from the obstacle. With the superposition of both gravitational and repulsive forces, the agent will safely approach the target point while avoiding the obstacle. It can be seen from (8) that since the resultant force is the superposition of gravitation and repulsion, the gravitation and repulsion may cancel each other, and the resultant force is 0. In addition, when one or more obstacles appear near the target point, according to (8), the repulsive force generated by the obstacle may be greater than the gravitational force generated by the target point. The target point will not be the global minimum point of the entire potential field, so the agent cannot reach the destination [72], [73].

III. B-APFDQN BASED ON APF AND DQN

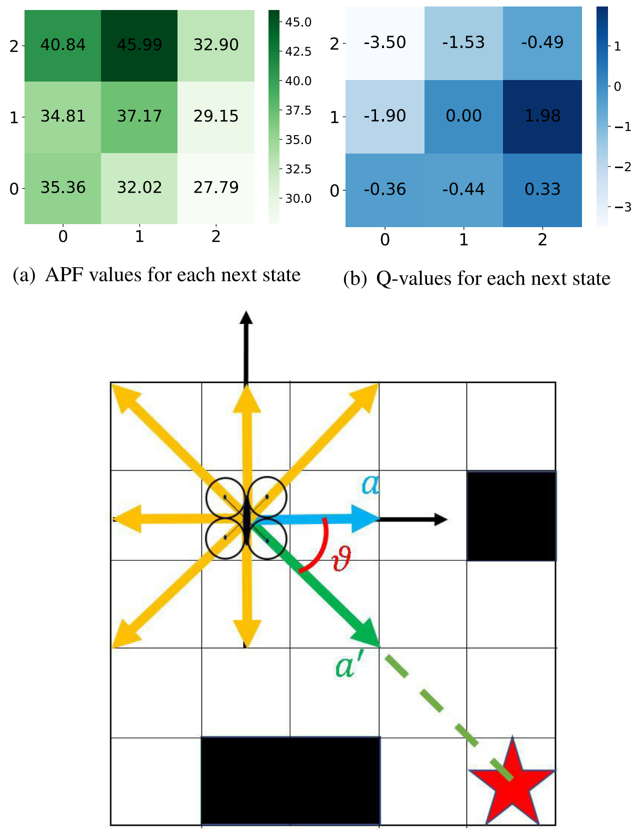
We adopt the APF method as the prior knowledge for the agent of DQN during action selection to reduce the frequency of blind trial-and-error in the search process of the agent. To ensure that the agent can fully explore the environment, we propose a SA-ε-greedy algorithm that varies with the number of search steps and successes. The agent is allowed to perform actions different from the APF generation actions within the error tolerance to avoid the agent from falling into local optimum due to the guidance of APF. In terms of the network employed, we propose a network that outputs both Q-values and action distribution instead of the classic DQN network that outputs only Q-values to accelerate the training process in combination with the APF. We remove the nodes that do not affect the connectivity in the obtained paths and apply the B-spline algorithm to make the paths shorter and smoother.

To simplify the process of guiding the APF to the agent, we set Q^* in (6) to a minimum value so that, according to (7), the obstacle generates a repulsive force only at a very close distance in its vicinity. In the gridded environment, the obstacle generates a considerable repulsive force only in the grids where they exist.

When the input state is s, Q-values and APF values may be shown in Fig.1(a) and Fig.1(b). The agent of classical DQN will choose the action a that maximizes the Q value, while APF will choose the action a' that decreases the potential field the fastest. Since the agent is unfamiliar with the environment at the initial stage of training, the actions performed are not necessarily reasonable, so we apply the APF as a reference for the agent's choice of action, and we measure the difference between these two actions in an angle. Taking the Q value corresponding to each action and the APF value corresponding to each action shown in Fig.1 as an example, the action vector selected according to the Q-values is $a = [1, 0]$, the action vector selected according to the APF values is $a' = [1, -1]$, from which the angle $\varphi = \frac{\pi}{4}$ between action a and action a' can be calculated in Fig.1(c).

$$action = \begin{cases} \text{argmax}_a\, Q(s, a; \theta), & \text{if } \varphi \leq \delta \\ \text{argmax}_{a'}\, F(s, a), & otherwise \end{cases} \tag{9}$$

If φ is smaller than the threshold, it means that the action a is consistent with the expectation of the action a' selected. Assuming that $\delta = \frac{\pi}{2}$, according to (9), the action at this time is a; otherwise, it is the action a'. In particular, when $\delta = 0$, it means that action a needs to be strictly consistent with action a'; when $\delta = \pi$, it means that whatever action a is selected is within the error tolerance with the action a'.



(a) APF values for each next state (b) Q-values for each next state (c) Agent action selection

FIGURE 1. The agent combines APF values and Q-values to select the action.

According to the action selection, the calculation of the target value function should be refined accordingly:

$$Y_t = \begin{cases} R_{t+1} + \gamma \max_a Q(s_{t+1}, a; \theta), & \text{if } \varphi \le \delta \\ R_{t+1} + \gamma Q(s_{t+1}, \max_{a'} F(s_t, a'); \theta), & \textit{otherwise} \end{cases} \tag{10}$$

To further accelerate the training process of the algorithm in combination with APF, this paper proposes a new neural network structure, as shown in Fig.2, which is different from the single output Q-values of the classical DQN.

The first layer of the network is the input layer, which receives the state information sensed by the agent. Two fully connected layers follow the input layer for further information processing. Unlike the classical DQN, which outputs Q-values directly at the output layer, we insert a fully connected layer called "copy layer" before the output layer, and split the output layer into two. One output is a copy of the "copy layer" and the other is the output of the "copy layer" processed by the softmax function $softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$. From the perspective of reinforcement learning, the softmax function enables the mapping from Q-values to strategies and is therefore often used for agent action selection [74]. Therefore, the output layer of the neural network is the Q-values, and the action distribution is obtained according to the Q-values. The loss function is:

$$L(\theta_t) = \alpha * \mathbb{E}_{s,a,\pi}[(Y_t - Q(s_t, a_t; \theta_t))^2] + \beta * KL(p, q, \theta_t) \tag{11}$$

where, α is the weight of the mean square error loss between the target value calculated according to the output of the target network and the target value output by the learning network, and β is the weight of the Kullback-Leibler(KL) divergence loss between the ideal action distribution and the actual output distribution of the learning network. When $\varphi \le \delta$, the ideal action distribution p is obtained by Q-values; otherwise, p is obtained according to APF values.

In order to make the agent perform a certain degree of exploration while reasonably utilizing the existing knowledge and solve the problem of exploration-utilization, we design the ε-greedy ε as a value that changes with the number of search steps and is affected by the number of successful searches:

$$\varepsilon(step) = \varepsilon_{end} + \frac{\varepsilon_{start} - \varepsilon_{end}}{e^{\frac{step}{decay}}} \tag{12}$$

where ε_{start} is the initial value of ε at the beginning of the search, ε_{end} is the final value of ε when we want the algorithm to run, and step is the total number of search steps from the first episode of the algorithm to the current episode. When the agent performs one action, then $step = step + 1$. When the agent completes the search task from the start point to the endpoint, $step = alpha * step$, $alpha$ is the parameter to adjust the step growth amount after the successful search, and decay is to adjust ε through step delay. We set $0 < alpha < 1$ when we consider that the agent may be under-explored, and we set $alpha > 1$ when we consider that the agent does not need to be explored further. Since our proposed algorithm can self-adaptively adjust the eplison according to steps and the completion of the task, we call the algorithm self-adaptive ε-greedy (SA-ε-greedy). Classical ε-greedy performs a random search of the environment with a fixed probability of eplison. If this fixed value is set large, it will cause many unnecessary searches after the agent is fully aware of the environment, and if this value is set small, it will be difficult for the agent to establish a comprehensive knowledge of the environment. Therefore, unlike the classical ε-greedy with fixed eplsion, our SA-ε-greedy algorithm sets ε as the amount that varies with the number of steps and successful searches. The SA-ε-greedy enables the agent to adjust the probability of exploration based on the search to reduce the probability of unnecessary exploration.

We call the entire algorithm for action selection and updating the network regarding the APF above APFDQN.

In performing tasks, the UAV usually needs to adjust its travel direction at certain positions due to obstacle avoidance or the needs of specific tasks. The point where the tangent direction of the path changes discontinuously is called the turning node. In order to reduce the number of turning nodes in the path and make the path more suitable for the actual operation of the UAV, we further optimize the path. We divide

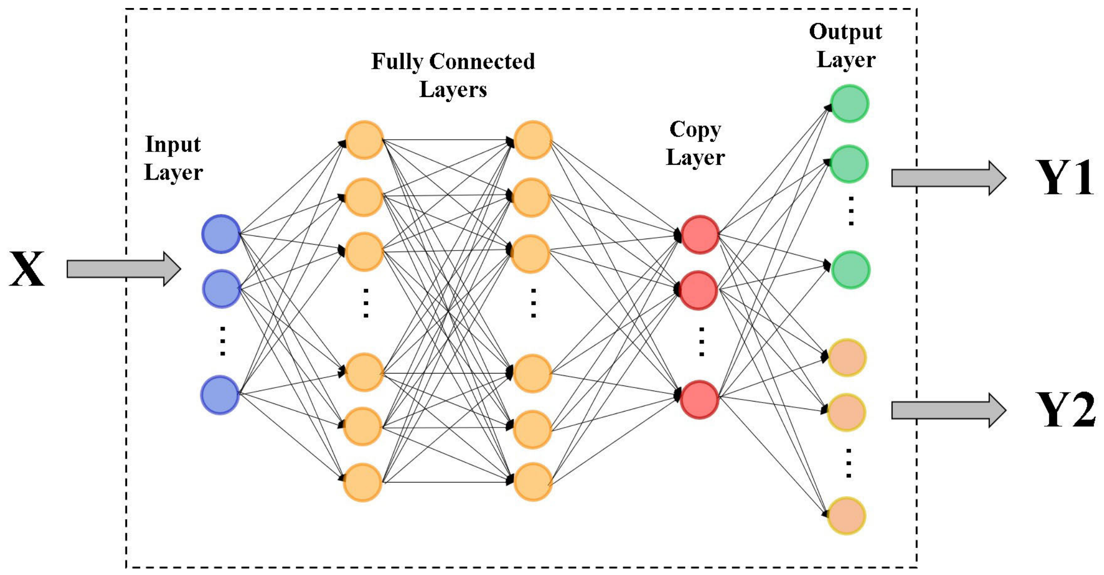


FIGURE 2. Simultaneous output action corresponds to the neural network structure's Q-values and action distribution.

Algorithm 1 APFDQN

```
Input: $\theta$, the neural network weights; $\delta$, the action error allowance threshold; $\alpha$ and $\beta$, the loss function weights; $alpha$, $\varepsilon$ update delay; $step$, the number of steps performed; $p$, the current path sequence $pbest$, the current best path sequence $len(p)$, the length of $p$
1:  for $episode = 1, M$ do
2:      while $s_t$!=endpoint do
3:          Generate a probability $prob$ with uniform distribution at (0, 1)
4:          if $prob > \varepsilon$ then
5:              $a_t$ = a random action;
6:          else
7:              $a_t$ = an action selected according to (9)
8:          end if
9:          Ideal action $a'_t$ selected according to (9)
10:         Execute action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$
11:         $step = step + 1$
12:         Store $< s_t, a_t, r_t, s_{t+1}, a'_t >$ into replay memory
13:         Set $s_t = s_{t+1}$
14:         Calculate the target value $Y_t$ according to (10)
15:         Randomly sampling data from replay memory
16:         Calculate the loss according to (11)
17:         Update $\theta$ with gradient descent
18:         Update $\varepsilon$ according to (12)
19:     end while
20:     if $len(p) < len(pbest)$ then
21:         Set $pbest = p$
22:     end if
23: end for
Output: The path sequence $pbest$
```

this optimization process into two parts. The first is to remove as many points in the path as possible that do not affect the path connectivity. At this time, there may still be many turning points in the path, which is unsuitable for the actual execution of the UAV. Therefore, the second step is to apply B-Spline to remove these turning points and make the path smooth and suitable for UAV execution, as well as to make the path length shorter.

A B-spline approximation algorithm is a powerful tool in computer-aided curve and surface design [75], [76]. A similar algorithm is the Bernstein approximation, which applies Bézier curves. However, compared to the Bernstein approximation, B-spline provides a generalization of Bézier and converges faster by introducing more knots and keeping the degree low. It also has the advantages of simple implementation and high computational efficiency.

We call the APFDQN after removing redundant points and adding B-spline optimization as B-APFDQN.

Assuming that the path obtained by APFDQN consists of n sequence points, starting from the second point of the sequence and ending at the penultimate point, traverse the middle $n-2$ sequence points, i represents the current sequence during the traversal process. Index, connect the $(i-1)th$ sequence point with the $(i+1)th$ sequence point. If this line segment does not intersect with the obstacle, remove the ith sequence point in the path, and repeat this process until there are no sequence points to remove, the new path is smoothed applying the cubic B-spline algorithm.

IV. EXPERIMENT

The experiments approached in this paper all design an $n \times n$ grid environment, the starting point is [0, 0], and the endpoint is $[n, n]$. We set that the UAV can only execute in eight directions, as shown in Fig.3.

Algorithm 2 B-APFDQN

```
1:  Obtain the path sequence $p$ by APFDQN
2:  Set $n$ =the sequence length of $p$
3:  for i=2,n-1 do
4:      $p'$=$p$ removes $ith$ point
5:      if $p'$ does not hit an obstacle then
6:          $p$=$p'$
7:      else
8:          $p$=$p$
9:      end if
10: end for
11: Optimize $p$ with B-spline
Output: The path sequence $p$
```

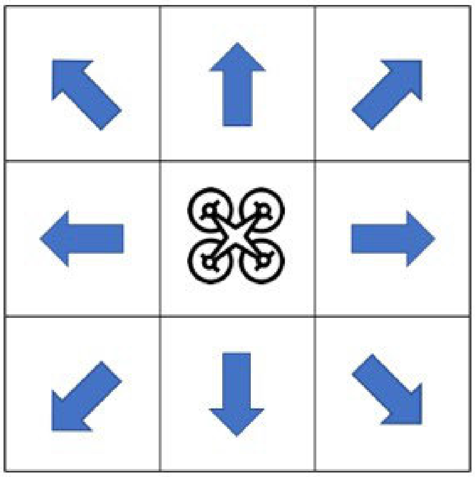


FIGURE 3. Eight optional actions for the agent.

The reward function is designed as

$$Reward(s_t) = beta * (d(s_{t-1}) - d(s_t)) - k \qquad (13)$$

where $d(s_{t-1})$ is the distance between the agent and the target point at the last moment, $d(s_t)$ is the distance between the agent and the target point at the time t, $beta$ is applied to adjust the proportion of $d(s_{t-1}) - d(s_t)$ in the reward, k is a fixed constant.

We compared APFDQN with classical DQN [51], [52] in two different 10 × 10 grid environments as shown in Fig.4, where each cell represents a state, black cells represent obstacles, green circles represent the start point, and the red star represents the endpoint. The agent's task is to move from the start and reach the end point while avoiding all obstacles. The neural network structure used to implement APFDQN, shown in Fig.2, consists of an input layer with 2 neurons, two fully connected layers with 256 neurons each, a copy layer with 8 neurons, and an output layer divided into two parts with 8 neurons each. The α in (11) is 1 and the β is 0.2. The structure of the neural network used to implement the DQN is a neural network with an input layer of 2 neurons, two fully connected layers with 256 neurons each and an output layer with 8 neurons. The learning rate of both neural networks is 0.0001, the discount coefficient of future returns is 0.9, and the batch size of the training data sampled from the replay memory is 32. Its optional action is shown in Fig.3, and the reward function is (13), where the coefficient beta is 1 and the constant k is -1. Fig.4(a) shows the scenario of regular obstacles, and Fig.4(b) shows the scenario of randomly generated irregular obstacles.

In this paper, we call each action performed by the agent taking a "step". When the agent completes searching for a

TABLE 1. Table of symbols mentioned in the paper.

Symbol	Parameters
θ	Neural network parameters
α, β	Loss function parameters
$alpha$	Steps gain coefficient
$beta$, k	Reward function parameters
γ	Reward discount Coefficient
ξ	Repulsive gain coefficient
η	Gravitational gain coefficient
ϕ	Angle between actions
δ	Action error tolerance threshold

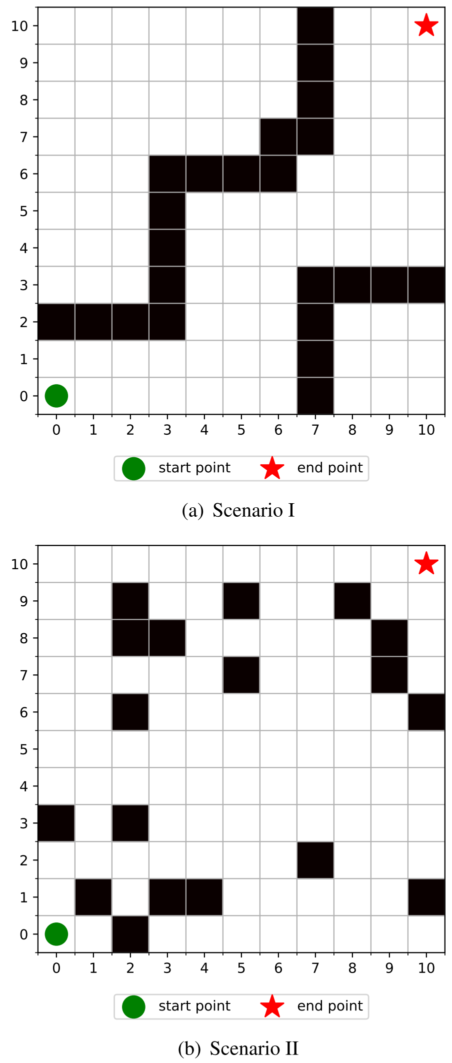


FIGURE 4. Two 10 × 10 grid environments, (a) for regular obstacles and (b) for irregular obstacles.

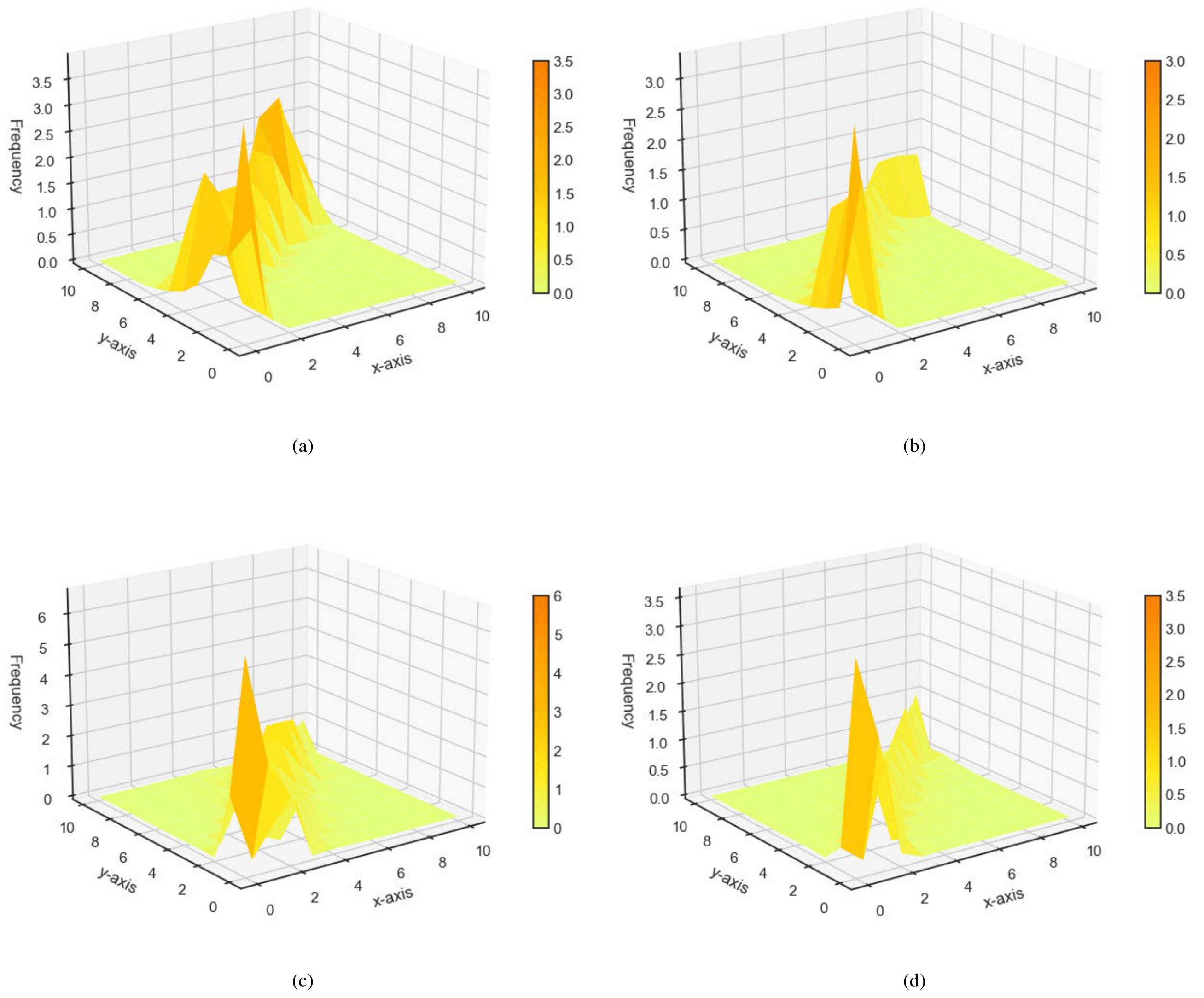


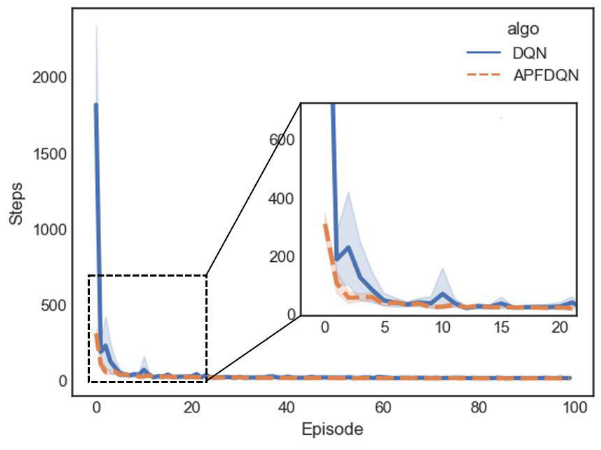
FIGURE 5. Frequency of agent access states of DQN and APFDQN under scenario I and scenario II. (a) and (b) is the frequency of the state being visited under scenario I. (c) and (d) is the frequency of the state being visited under scenario II. Where the left is DQN the right is APFDQN.

path from the start point to the end point, we call the agent completing an ''episode''. We say the agent has completed a ''training session'' when the episode reaches the set point.

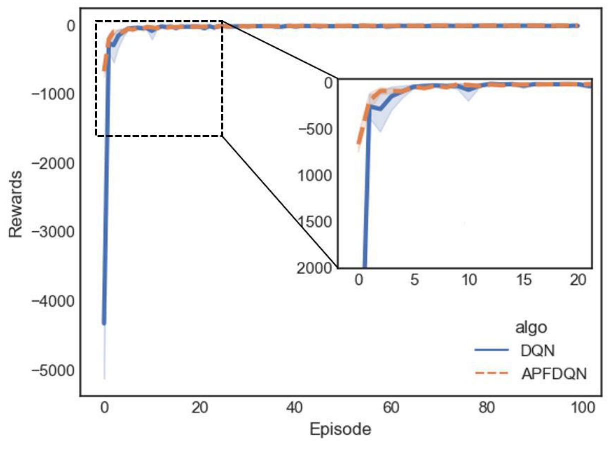
We conducted 100 training sessions in the two 10×10 grid environments shown in Fig.4 and set the size of the episode to 100 for each training session. Consider this training session a failure if the number of steps searched by an agent in an episode exceeds 4000. We set the allowable error threshold $\delta = \frac{\pi}{4}$ in APFDQN and then statistically compare its success rate with the classical DQN algorithm over 100 training sessions in two scenarios and the frequency of agents accessing each state. From Fig.5, the distribution of states visited by the agent of APFDQN is more concentrated than that of DQN in Scenarios I and II. For most of the visited states, the agent of APFDQN access them significantly less frequently than the agent of DQN. It indicates that APFDQN effectively reduces the trial-and-error frequency of classical DQN during training and enhances the agent's search efficiency for unfamiliar environments.

Fig.6 shows the steps and rewards change with episodes for DQN and APFDQN in scenario I and II in training phase. The blue curve and the orange dashed line indicate the average of the cumulative rewards in a single episode and the average of the cumulative steps in a single episode, respectively. In the first 20 episodes, the number of steps the APFDQN agent takes to complete an episode and the range of step variation are always smaller than that of the classical DQN. Therefore, from Fig.6, we can conclude that the stability and convergence speed of the APFDQN agent search process are better than the classical DQN.

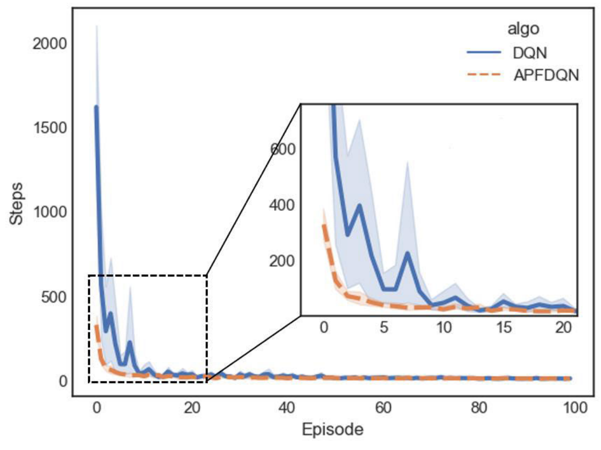
Fig.7 shows the number of successful 100 training sessions between DQN and APFDQN in two different scenarios.



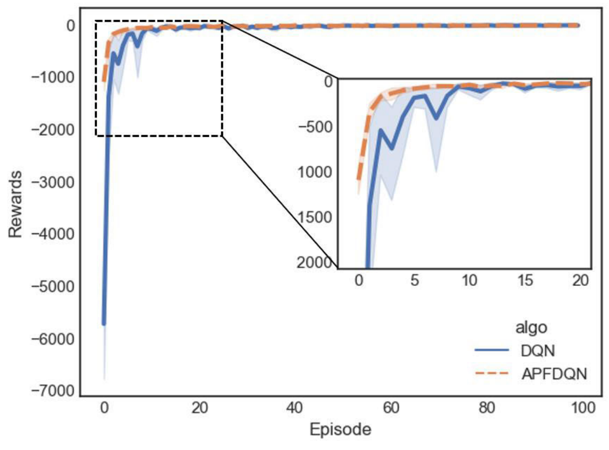
(a)



(b)



(c)



(d)

FIGURE 6. Average of cumulative steps and cumulative rewards for a single episode in 100 training sessions under scenarios I and II. (a)(b) is the change of steps and rewards with episodes for DQN and APFDQN under scenario I. (c)(d) is the change of steps and rewards with episodes for DQN and APFDQN under scenario II.

We set the successful threshold to 4000. In a training session, when the number of search steps for each episode is less than 4000, we consider the agent succeeds this time. Otherwise, it fails. It can be seen that the success rate of DQN in both scenarios is less than 50%, while the success rate of APFDQN in both scenarios can almost reach 100%, which shows that APFDQN has a solid ability to adapt to different environments. It can cope with various disturbances in the environment and has strong robustness.

In order to verify the influence of the allowable error threshold of the action selected according to the APF values and the action selected according to the Q-values on the algorithm, we conducted the tests in the environment shown in Fig.8, respectively, when setting the allowable error thresholds $\delta = \frac{\pi}{4}$, $\delta = \frac{\pi}{2}$ and $\delta = \frac{3\pi}{4}$, and 20 training sessions were performed for each different threshold.

It can be seen from Fig.9 that the smaller the value of δ, the faster and more stable the convergence of the algorithm. Because the δ represents the similarity requirement between the action selected based on the Q-values and the action selected based on the APF values The smaller the value of delta, the more similar the two actions need to be, which also means that the action selection is more dependent on the APF, and the greater the possibility of falling into a local optimum. Since the experiment is implemented in a grid environment, and it is stipulated that the agent can only move in a limited eight directions, the obtained path may have many turning points. With many turning points, this path does not align with the actual application of UAVs. Further optimization of

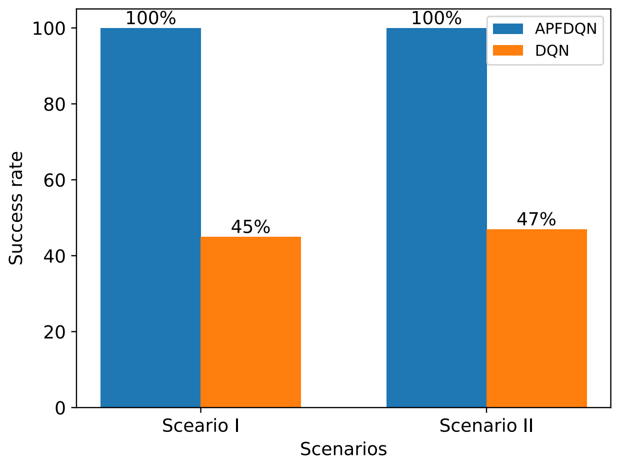


FIGURE 7. The number of successful searches of DQN and APFDQN under 100 training sessions in two different scenarios.

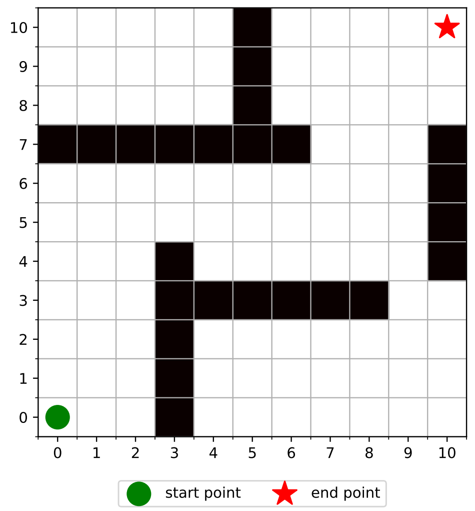
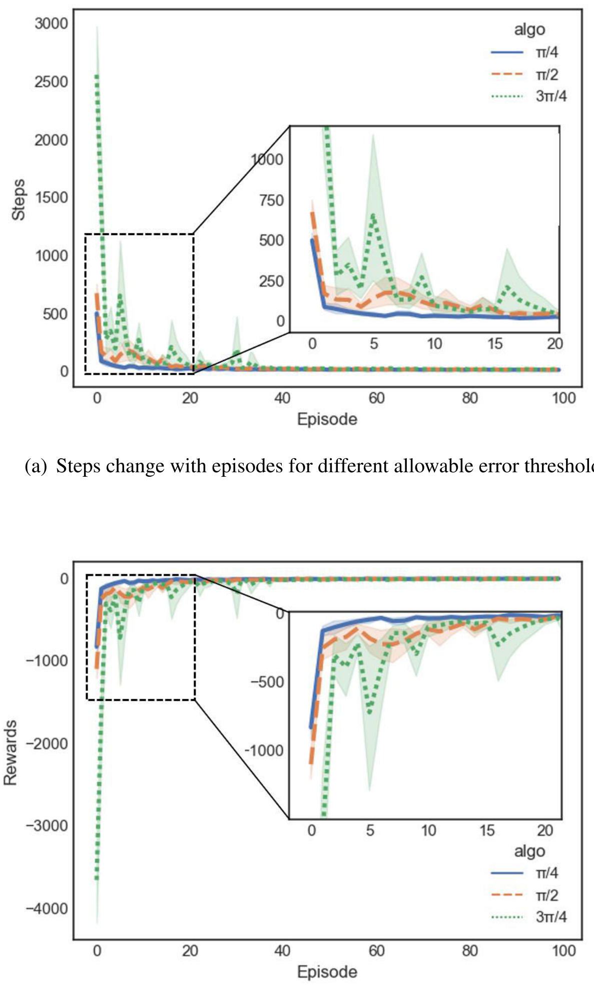


FIGURE 8. 10×10 grid environment designed to test the allowable error threshold of different actions.



(a) Steps change with episodes for different allowable error thresholds.

(b) Rewards change with episodes for different allowable error thresholds.

FIGURE 9. Average of cumulative steps and cumulative rewards in a single episode in 20 training sessions for different allowable error thresholds.

the obtained path is required. The path obtained by APFDQN after optimization is shown in Fig.10.

We divide this optimization process into two parts. First, remove as many nodes in the path as possible that do not affect the connectivity of the path, and then apply the cubic B-spline algorithm to smooth the initially optimized path.

Assume that the path obtained adopting APFDQN consists of n sequence points. First, start from the second point of the sequence and traverse the middle $n-2$ sequence points until the penultimate point position, i denotes the index of the current sequence during the traversal. Connect the $(i-1)st$ sequence point with the $(i+1)st$ sequence point, and if this line segment does not intersect with the obstacle, remove the ith sequence point in the path, repeat this process until there are no sequence points that can be removed from the path, and then smooth the new path with the cubic B-spline algorithm. We call the algorithm after this optimization B-APFDQN.

To further verify the effectiveness of the B-APFDQN algorithm proposed in this paper, we compare our algorithm with heuristic-based SDPSO [77], probability-based PQ-RRT* [78], and graph search-based DFPA [79].

As shown in Fig.11 and Fig.12, B-APFDQN has the shortest path length among all the tested algorithms and the smoothest trajectory without apparent turning nodes. Next is APFDQN, this model obtains a slightly longer path length than the B-APFDQN model and obtains a path with obvious turning nodes, but the length is still smaller than the other algorithms.

The PQ-RRT* algorithm can also obtain a path with a shorter length and fewer turning nodes. However, since the PQ-RRT* algorithm is an improvement based on the RRT algorithm, it does not eliminate the randomness of the search process and the inability to receive feedback from the environment and make adjustments, so the obtained paths are often sub-optimal.

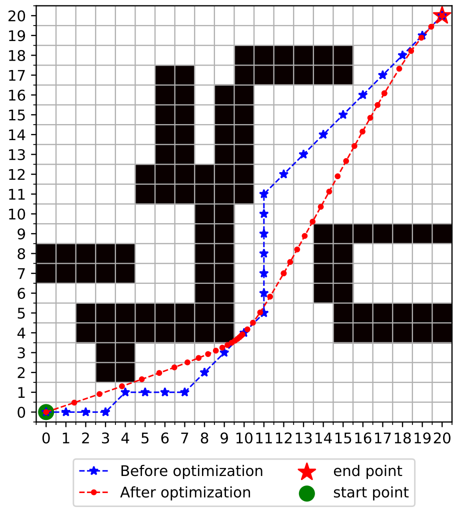


FIGURE 10. Before and after path optimization.

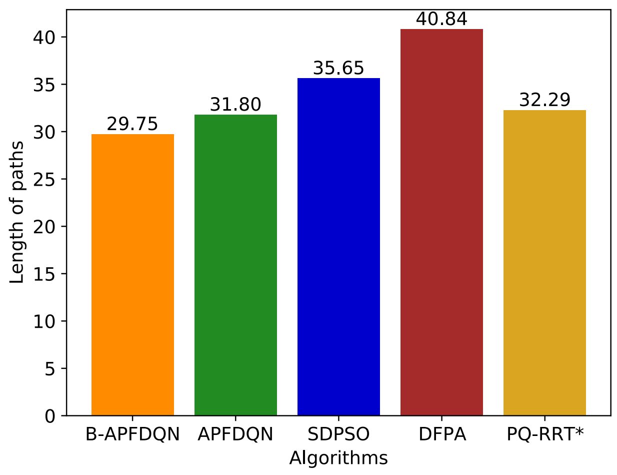


FIGURE 12. Path length comparison of B-APFDQN, APFDQN, SDPSO, DFPA, and PQ-RRT*.

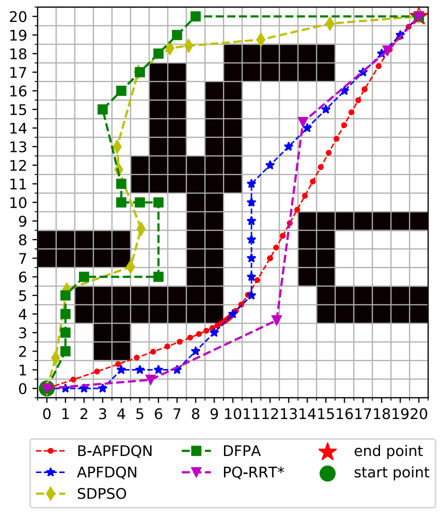


FIGURE 11. Path comparison of SDPSO, PQ-RRT*, DFPA, APFDQN, and B-APFDQN.

SDPSO adjusts the update policy of the optimal global solution in the PSO algorithm by combining the simulated annealing algorithm, which speeds up the convergence of the algorithm and achieves good results. However, the algorithm based on PSO is prone to fall into the local optimum. The upper limit that can be obtained by optimizing the path depends on the quality of the initial particles we can provide. If the particles we provide have the potential to be optimized to the global optimum, it is easier for the optimized path to converge to the global optimum.

Finally, DFPA can also achieve good results. The DFPA algorithm applies Delaunay triangulation to construct the Voronoi diagram corresponding to the environment according to the obstacles, starting points, and ending points. When applying the A-STAR algorithm, the Voronoi diagram nodes are preferentially selected. After the path is obtained, the path is optimized by inserting Steiner points between the nodes. The Voronoi diagram can provide certain global environment information for the A-STAR algorithm during the search process and reduce the possibility of the A-STAR algorithm misestimating the future environment during the search process. However, the introduction of Voronoi can only partially solve this problem. During the STAR search process, it is still possible to miss the optimal global solution because the current optimal solution is selected.

In summary, the B-APFDQN algorithm has a faster convergence rate than the classical DQN algorithm, and we can control the degree of dependence on the potential field method when selecting an action by adjusting the threshold of the allowable action angle error, which significantly reduces the trial-and-error frequency of the agent in the early stages of training. The agent will obtain positive feedback from the environment in the continuous interaction with the environment to update its cognition of the environment, so the probability of falling into the local optimum is minuscule. The path obtained after convergence removes the redundant nodes and conducts the cubic B-spline processing. Experiments show that the path output by B-APFDQN has almost no prominent turning nodes, and the trajectory is smooth. The path length is more than 5% shorter than that obtained by other algorithms, which meets the needs of UAVs in practical applications.

V. CONCLUSION

This paper proposed a path planning model B-APFDQN based on DQN and APF algorithm. The algorithm takes APF as the agent's prior knowledge in selecting actions. An error allowance threshold is set between the actions selected according to the Q-values and those selected according to the APF values. The proposed SA-ε-greedy algorithm that varies with the number of search steps and search successes is adopted to ensure that the agent can fully explore the environment and avoid the search from falling into local optimum. A neural network that outputs both action distribution and Q-values is also applied instead of the traditional DQN that only outputs Q-values to accelerate training in combination with APF. For the obtained paths, we remove the redundant nodes and apply the B-spline algorithm to make the paths shorter and smoother.

The simulation experiments in the grid environment show that the proposed B-APFDQN algorithm can effectively reduce the trial-and-error frequency of the agent in the early training stage and can effectively avoid the agent from falling into local optimum.The experiments also verify that B-APFDQN has a faster convergence speed than the classical DQN algorithm. A smooth path with no obvious turning points can be obtained finally. However, there are still some things that could be improved in this study. Firstly, B-APFDQN needs to grid the target space, which can complicate the problem of planning paths. Secondly, when the task space is complex, the policy obtained by the APF algorithm could be more reasonable and may produce misguidance to the agent. In the future, we need to improve B-APFDQN further so that it can take effect in the continuous space directly and adjust the algorithm for guiding the agent to reduce the impact on the agent due to the error of the guidance algorithm.

REFERENCES

[1] B. Fan, Y. Li, R. Zhang, and Q. Fu, "Review on the technological development and application of UAV systems," *Chin. J. Electron.*, vol. 29, no. 2, pp. 199–207, Mar. 2020.

[2] Y. Zeng and R. Zhang, "Energy-efficient UAV communication with trajectory optimization," *IEEE Trans. Wireless Commun.*, vol. 16, no. 6, pp. 3747–3760, Jun. 2017.

[3] J. Lyu, Y. Zeng, and R. Zhang, "Cyclical multiple access in UAV-aided communications: A throughput-delay tradeoff," *IEEE Wireless Commun. Lett.*, vol. 5, no. 6, pp. 600–603, Dec. 2016.

[4] X. Sun, C. Shen, D. W. K. Ng, and Z. Zhong, "Robust trajectory and resource allocation design for secure UAV-aided communications," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, May 2019, pp. 1–6.

[5] T. Adão, J. Hruska, L. Pádua, J. Bessa, E. Peres, R. Morais, and J. Sousa, "Hyperspectral imaging: A review on UAV-based sensors, data processing and applications for agriculture and forestry," *Remote Sens.*, vol. 9, no. 11, p. 1110, 2017.

[6] P. Tokekar, J. V. Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1498–1511, Dec. 2016.

[7] R. Sharma and R. Arya, "UAV based long range environment monitoring system with industry 5.0 perspectives for smart city infrastructure," *Comput. Ind. Eng.*, vol. 168, Jun. 2022, Art. no. 108066.

[8] R. Carvalho, R. Nascimento, T. D'Angelo, S. Delabrida, A. G. C. Bianchi, R. A. R. Oliveira, H. Azpúrua, and L. G. Uzeda Garcia, "A UAV-based framework for semi-automated thermographic inspection of belt conveyors in the mining industry," *Sensors*, vol. 20, no. 8, p. 2243, Apr. 2020.

[9] S. Qiu, B. Chen, R. Wang, Z. Zhu, Y. Wang, and X. Qiu, "Estimating contaminant source in chemical industry park using UAV-based monitoring platform, artificial neural network and atmospheric dispersion simulation," *RSC Adv.*, vol. 7, no. 63, pp. 39726–39738, 2017.

[10] H. Liu, Y. Sun, J. Cao, S. Chen, N. Pan, Y. Dai, and D. Pan, "Study on UAV parallel planning system for transmission line project acceptance under the background of industry 5.0," *IEEE Trans. Ind. Informat.*, vol. 18, no. 8, pp. 5537–5546, Aug. 2022.

[11] J. Xu, K. Ota, and M. Dong, "Big data on the fly: UAV-mounted mobile edge computing for disaster management," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2620–2630, Oct. 2020.

[12] J. Li, Y. He, X. Zhang, and Q. Wu, "Simultaneous localization of multiple unknown emitters based on UAV monitoring big data," *IEEE Trans. Ind. Informat.*, vol. 17, no. 9, pp. 6303–6313, Sep. 2021.

[13] F. Luo, C. Jiang, S. Yu, J. Wang, Y. Li, and Y. Ren, "Stability of cloud-based UAV systems supporting big data acquisition and processing," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 866–877, Jul./Sep. 2017.

[14] S. Jung, W. J. Yun, J. Kim, and J.-H. Kim, "Coordinated multi-agent deep reinforcement learning for energy-aware UAV-based big-data platforms," *Electronics*, vol. 10, no. 5, p. 543, Feb. 2021.

[15] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV coverage path planning under varying power constraints using deep reinforcement learning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 1444–1449.

[16] C. H. Liu, Z. Chen, J. Tang, J. Xu, and C. Piao, "Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 9, pp. 2059–2070, Sep. 2018.

[17] X. Wang, H. Wang, H. Zhang, M. Wang, L. Wang, K. Cui, C. Lu, and Y. Ding, "A mini review on UAV mission planning," *J. Ind. Manag. Optim.*, vol. 19, no. 5, pp. 3362–3382, 2023.

[18] X. Yu, X. Gao, L. Wang, X. Wang, Y. Ding, C. Lu, and S. Zhang, "Cooperative multi-UAV task assignment in cross-regional joint operations considering ammunition inventory," *Drones*, vol. 6, no. 3, p. 77, Mar. 2022.

[19] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, and G. Casalino, "Drone deep reinforcement learning: A review," *Electronics*, vol. 10, no. 9, p. 999, 2021.

[20] H. Sun, W. Zhang, R. Yu, and Y. Zhang, "Motion planning for mobile robots—Focusing on deep reinforcement learning: A systematic review," *IEEE Access*, vol. 9, pp. 69061–69081, 2021.

[21] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, Oct. 2021.

[22] T. Dam, G. Chalvatzaki, J. Peters, and J. Pajarinen, "Monte–Carlo robot path planning," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11213–11220, Oct. 2022.

[23] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Auton. Agents Multi-Agent Syst.*, vol. 27, no. 1, pp. 1–51, Jul. 2012.

[24] S. Ragi and E. K. P. Chong, "UAV path planning in a dynamic environment via partially observable Markov decision process," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 49, no. 4, pp. 2397–2412, Oct. 2013.

[25] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proc. 4th Int. Symp. Voronoi Diagrams Sci. Eng. (ISVD)*, Jul. 2007, pp. 38–47.

[26] P. Bhattacharya and M. Gavrilova, "Roadmap-based path planning-using the Voronoi diagram for a clearance-based shortest path," *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, Jun. 2008.

[27] S. Garrido, L. Moreno, M. Abderrahim, and F. Martin, "Path planning for mobile robot navigation using Voronoi diagram and fast marching," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2006, pp. 2376–2381.

[28] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Proc. Eng.*, vol. 96, pp. 59–69, 2014.

[29] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei, "An improved A-star based path planning algorithm for autonomous land vehicles," *Int. J. Adv. Robotic Syst.*, vol. 17, no. 5, Sep. 2020, Art. no. 172988142096226.

[30] C. Wang, L. Wang, J. Qin, Z. Wu, L. Duan, Z. Li, M. Cao, X. Ou, X. Su, W. Li, Z. Lu, M. Li, Y. Wang, J. Long, M. Huang, Y. Li, and Q. Wang, "Path planning of automated guided vehicles based on improved A-star algorithm," in *Proc. IEEE Int. Conf. Inf. Autom.*, Aug. 2015, pp. 2071–2076.

[31] D. Mandloi, R. Arya, and A. K. Verma, "Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3D environment," *Int. J. Syst. Assurance Eng. Manage.*, vol. 12, no. 5, pp. 990–1000, 2021.

[32] L.-S. Liu, J.-F. Lin, J.-X. Yao, D.-W. He, J.-S. Zheng, J. Huang, and P. Shi, "Path planning for smart car based on Dijkstra algorithm and dynamic window approach," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–12, Feb. 2021.

[33] H. Wang, Y. Yu, and Q. Yuan, "Application of Dijkstra algorithm in robot path-planning," in *Proc. 2nd Int. Conf. Mechanic Autom. Control Eng.*, Jul. 2011, pp. 1067–1069.

[34] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, Sep. 1996.

[35] L. E. Kavraki, M. N. Kolountzakis, and J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Trans. Robot. Autom.*, vol. 14, no. 1, pp. 166–171, Feb. 1998.

[36] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, Jan. 1998.

[37] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2011, pp. 1478–1483.

[38] K. Naderi, J. Rajamäki, and P. Hämäläinen, "RT-RRT*: A real-time path planning algorithm based on RRT*," in *Proc. 8th ACM SIGGRAPH Conf. Motion Games*, 2015, pp. 113–118.

[39] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *Proc. ICRA. Millennium Conf. IEEE Int. Conf. Robot. Automation. Symposia*, Apr. 2000, pp. 521–528.

[40] G. Sanchez and J.-C. Latombe, "Using a PRM planner to compare centralized and decoupled planning for multi-robot systems," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 2, Apr. 2002, pp. 2112–2119.

[41] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[42] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, nos. 2–3, pp. 243–278, Aug. 2015.

[43] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4, May 1995, pp. 1942–1948.

[44] J. L. Foo, J. Knutzon, V. Kalivarapu, J. Oliver, and E. Winer, "Path planning of unmanned aerial vehicles using B-splines and particle swarm optimization," *J. Aerosp. Comput., Inf., Commun.*, vol. 6, pp. 271–290, Apr. 2009.

[45] H. Miao and Y.-C. Tian, "Dynamic robot path planning using an enhanced simulated annealing approach," *Appl. Math. Comput.*, vol. 222, pp. 420–437, Oct. 2013.

[46] Q. Zhu, Y. Yan, and Z. Xing, "Robot path planning based on artificial potential field approach with simulated annealing," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, vol. 2, Oct. 2006, pp. 622–627.

[47] Z. Beheshti and S. M. H. Shamsuddin, "A review of population-based meta-heuristic algorithms," *Int. J. Adv. Soft Comput. Appl.*, vol. 5, no. 1, pp. 1–35, 2013.

[48] N. Kokash, "An introduction to heuristic algorithms," Dept. Inform. Telecommun., 2005, pp. 1–8.

[49] A. Konar, I. G. Chakraborty, S. J. Singh, L. C. Jain, and A. K. Nagar, "A deterministic improved Q-learning for path planning of a mobile robot," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 43, no. 5, pp. 1141–1153, Sep. 2013.

[50] A. Maoudj and A. Hentout, "Optimal path planning approach based on Q-learning algorithm for mobile robots," *Appl. Soft Comput.*, vol. 97, Dec. 2020, Art. no. 106796.

[51] 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*.

[52] M. Volodymyr, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, Feb. 2015.

[53] S. Qiu, Z. Li, Z. Li, and X. Zhang, "Model-free optimal chiller loading method based on Q-learning," *Sci. Technol. Built Environ.*, vol. 26, no. 8, pp. 1100–1116, Sep. 2020.

[54] K. G. Vamvoudakis, "Q-learning for continuous-time linear systems: A model-free infinite horizon optimal control approach," *Syst. Control Lett.*, vol. 100, pp. 14–20, Feb. 2017.

[55] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, Mar. 1986, pp. 396–404.

[56] J. Sun, J. Tang, and S. Lao, "Collision avoidance for cooperative UAVs with optimized artificial potential field algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.

[57] X. Chen and J. Zhang, "The three-dimension path planning of UAV based on improved artificial potential field in dynamic environment," in *Proc. 5th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Aug. 2013, pp. 114–147.

[58] X. Wang, J. Liu, X. Su, H. Peng, X. Zhao, and C. Lu, "A review on carrier aircraft dispatch path planning and control on deck," *Chin. J. Aeronaut.*, vol. 33, no. 12, pp. 3039–3057, Dec. 2020.

[59] M. Luo, X. Hou, and J. Yang, "Surface optimal path planning using an extended Dijkstra algorithm," *IEEE Access*, vol. 8, pp. 147827–147838, 2020.

[60] X. Wang, B. Li, X. Su, H. Peng, L. Wang, C. Lu, and C. Wang, "Autonomous dispatch trajectory planning on flight deck: A search-resampling-optimization framework," *Eng. Appl. Artif. Intell.*, vol. 119, Mar. 2023, Art. no. 105792.

[61] A. A. Ravankar, A. Ravankar, T. Emaru, and Y. Kobayashi, "HPPRM: Hybrid potential based probabilistic roadmap algorithm for improved dynamic path planning of mobile robots," *IEEE Access*, vol. 8, pp. 221743–221766, 2020.

[62] X. Wang, J. Liu, H. Peng, X. Qie, X. Zhao, and C. Lu, "A simultaneous planning and control method integrating APF and MPC to solve autonomous navigation for USVs in unknown environments," *J. Intell. Robotic Syst.*, vol. 105, no. 2, p. 36, Jun. 2022.

[63] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Appl. Soft Comput.*, vol. 100, Mar. 2021, Art. no. 106960.

[64] J. Li, Y. Chen, X. Zhao, and J. Huang, "An improved DQN path planning algorithm," *J. Supercomput.*, vol. 78, no. 1, pp. 616–639, Jan. 2022.

[65] A. A. Maw, M. Tyan, T. A. Nguyen, and J.-W. Lee, "IADA*-RL: Anytime graph-based path planning with deep reinforcement learning for an autonomous UAV," *Appl. Sci.*, vol. 11, no. 9, p. 3948, Apr. 2021.

[66] D. Li, W. Yin, W. E. Wong, M. Jian, and M. Chau, "Quality-oriented hybrid path planning based on A* and Q-learning for unmanned aerial vehicle," *IEEE Access*, vol. 10, pp. 7664–7674, 2022.

[67] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Trans. Autom. Sci. Eng.*, vol. 17, no. 4, pp. 1748–1758, Oct. 2020.

[68] L. Yafei, W. Anping, C. Qingyang, and W. Yujie, "An improved UAV path planning method based on RRT-APF hybrid strategy," in *Proc. 5th Int. Conf. Autom., Control Robot. Eng. (CACRE)*, Sep. 2020.

[69] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*.

[70] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, no. 1, pp. 237–285, Jan. 1996.

[71] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, 2018.

[72] B. Zhang, W. Chen, and M. Fei, "An optimized method for path planning based on artificial potential field," in *Proc. 6th Int. Conf. Intell. Syst. Design Appl.*, Oct. 2006, pp. 35–39.

[73] Y. Koren and J. Borenstein, "Potential field methods and their inherent limitations for mobile robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 2, Apr. 1991, pp. 1398–1404.

[74] B. Gao and L. Pavel, "On the properties of the softmax function with application in game theory and reinforcement learning," 2017, *arXiv:1704.00805*.

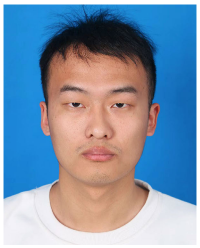
[75] R. Riesenfeld, *Applications of B-Spline Approximation to Geometric: Problems of Computer-Aided Design*. Ann Arbor, MI, USA: Univ. Microfilms International, 1985.

[76] H. Prautzsch, W. Boehm, and M. Paluszny, *Bezier and B-Spline Techniques*. Berlin, Germany: Springer, 2011.

[77] Z. Yu, Z. Si, X. Li, D. Wang, and H. Song, "A novel hybrid particle swarm optimization algorithm for path planning of UAVs," *IEEE Internet Things J.*, vol. 9, no. 22, pp. 22547–22558, Nov. 2022.

[78] Y. Li, W. Wei, Y. Gao, D. Wang, and Z. Fan, "PQ-RRT*: An improved path planning algorithm for mobile robots," *Exp. Syst. Appl.*, vol. 152, Aug. 2020, Art. no. 113425.

[79] Z. Liu, H. Liu, Z. Lu, and Q. Zeng, "A dynamic fusion pathfinding algorithm using Delaunay triangulation and improved A-star for mobile robots," *IEEE Access*, vol. 9, pp. 20602–20621, 2021.



FUCHEN KONG was born in Liaoning, China, in 1997. He received the B.S. degree in the Internet of Things from the Jiangsu University of Science and Technology, Zhenjiang, China, where he is currently pursuing the M.S. degree in software engineering. His research interests include machine learning and reinforcement learning.

QI WANG (Member, IEEE) received the B.S. degree from the Shandong University of Finance and Economics, Jinan, China, in 2005, the M.S. degree from Sungkyunkwan University, Suwon, South Korea, in 2011, and the Ph.D. degree from the University of Trento, Trento, Italy, in 2015. He was a Visiting Scholar with North Carolina State University, Raleigh, NC, USA, from 2013 to 2014. He is currently a Lecturer with the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China. He has authored and coauthored several top journal articles. His research interests include wireless communication, smart cities, and the Internet of Things. He is a reviewer for a couple of journals and conferences.

SHANG GAO received the B.S. degree in system engineering and the M.S. degree in military equipment from Air Force Engineering University, Xi'an, China, in 1993 and 1996, respectively, and the Ph.D. degree in pattern recognition and intelligent systems from the Nanjing University of Science and Technology, Nanjing, China, in 2006. Since 2009, he has been a Professor with the School of Computer, Jiangsu University of Science and Technology, Zhenjiang, China, where he has been the Dean, since 2017. He has published more than 80 research articles on the professional journals and conferences. His research interests include optimization theory, swarm intelligence, and machine learning.

HUALONG YU was born in Harbin, China, in 1982. He received the B.S. degree in computer science from Heilongjiang University, Harbin, in 2005, and the M.S. and Ph.D. degrees in computer science from Harbin Engineering University, Harbin, in 2008 and 2010, respectively. From 2013 to 2017, he was a Postdoctoral Fellow with the School of Automation, Southeast University, Nanjing, China. From 2017 to 2018, he was a Senior Visiting Scholar with the Faculty of Information Technology, Monash University, Melbourne, Australia. Since 2020, he has been a Professor with the School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang, China. He has authored or coauthored more than 90 journals and conference papers, and four monographs. His research interests include machine learning, data mining, and bioinformatics. He is a member of the organizing committee of several international conferences, the ACM, the China Computer Federation (CCF), and the Youth Committee of the Chinese Association of Automation (CAA). He is an active Reviewer of more than 20 high-quality international journals, including IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, IEEE TRANSACTIONS ON COMPUTATIONAL BIOLOGY AND BIOINFORMATICS, and *ACM Transactions on Knowledge Discovery from Data* (TKDD). He is an Associate Editor of IEEE ACCESS.