Hindawi
International Journal of Aerospace Engineering
Volume 2022, Article ID 1360956, 15 pages
https://doi.org/10.1155/2022/1360956

Research Article

Path Planning of Unmanned Helicopter in Complex Environment Based on Heuristic Deep Q-Network

Jiangyi Yao,[1] **Xiongwei Li**,[1] **Yang Zhang**,[1] **Jingyu Ji**,[2] **Yanchao Wang**,[1] **and Yicen Liu**[3]

[1]*Equipment Simulation Training Center, Shijiazhuang Campus, Army Engineering University, Shijiazhuang, Hebei 050003, China*
[2]*Department of UAV Engineering, Shijiazhuang Campus, Army Engineering University, Shijiazhuang, Hebei 050003, China*
[3]*State Key Laboratory of Blind Signal Processing, Chengdu, Sichuan 610000, China*

Correspondence should be addressed to Xiongwei Li; lxw-wys@163.com

Received 17 April 2022; Revised 26 May 2022; Accepted 15 June 2022; Published 28 June 2022

Academic Editor: Erkan Kayacan

Unmanned helicopters (UH) can evade radar detection by flying at ultralow altitudes, so as to conduct raids on targets. Path planning is one of the key technologies to realize UH's autonomous completion of raid missions. Since the probability of UH being detected by radar varies with height, how to accurately identify the radar coverage area to avoid crossing has become a difficult problem in UH path planning. Aiming at this problem, a heuristic deep Q-network (H-DQN) algorithm is proposed. First, as part of the comprehensive reward function, a heuristic reward function is designed. The function can generate dynamic rewards in real time according to the environmental information, so as to guide the UH to move closer to the target and at the same time promote the convergence of the algorithm. Second, in order to smooth the flight path, a smoothing reward function is proposed. This function can evaluate the pros and cons of UH's actions, so as to prompt UH to choose a smoother path for flight. Finally, the heuristic reward function, the smooth reward function, the collision penalty, and the completion reward are weighted and summed to obtain the heuristic comprehensive reward function. Simulation experiments show that the H-DQN algorithm can help UH to effectively avoid the radar coverage area and successfully complete the raid mission.

1. Introduction

Unmanned helicopter (UH) has the characteristics of strong manoeuvrability and good concealment and can avoid radar detection by flying at ultralow altitude. Therefore, UH is widely used to raid important targets on the battlefield. Ordinary UH still needs to be operated by rear personnel to complete a series of tasks, which just transfers the operator from the front to the background, and does not achieve true unmanned operation. The intelligent UH should complete a series of tasks through autonomous decision-making, so as to truly operate completely autonomously without the control of personnel. To achieve this, research related to real-time communication, resource allocation, and path planning needs to be paid more attention [1–3].

As one of the key technologies for unmanned systems to achieve intelligence, path planning technology plays an important role in improving the intelligence, safety, and adaptability of UH [4]. UH needs to complete a series of decisions under the guidance of the safe path to achieve autonomous movement. Therefore, path planning technology is the basis for UH to move towards the target. In order to ensure the safety of the UH movement process, the path planning needs to consider a large number of constraints such as the battlefield environment and the manoeuvrability of the UH. Since many elements in the battlefield environment will pose a serious threat to the safe flight of the UH, the path planning of the UH will face complex constraints.

In recent years, researchers have proposed a series of solutions to solve the path planning problem of unmanned aerial vehicle (UAV). A new metaheuristic grey wolf optimizer (GWO) was proposed to solve the UCAV two-dimensional path planning problem in the literature [5], which fully consider the threats and constraints of the battlefield environment. In the literature [6], an improved pigeon-inspired optimization algorithm (PIOFOA) was proposed to solve problems about path planning in a three-dimensional dynamic environment of oilfields. An improved constrained differential evolution (DE) algorithm was proposed in the literature [7], which combines DE algorithm with the level comparison method, to find the optimal route in feasible regions. An adaptive selection mutation-constrained differential evolution algorithm was proposed in [8]. In this paper, UAV path planning was modelled as the optimization problem, in which fitness functions include traveling distance and risk of UAV; three constraints involve the height of UAV, angle of UAV, and limited UAV slope. On the one hand, the environmental threats in existing research are usually static, and the threat area is completely impassable. This constraint method reduces the difficulty of UAVs avoiding dangerous areas to a certain extent. However, there are usually many dynamically changing threat areas in the battlefield environment, and it is difficult for the algorithms in the above studies to accurately identify and avoid these areas. On the other hand, most of the above studies were conducted on general-purpose UAV, and few studies were conducted on UH alone. There is a big difference between UAV and UH in terms of flight height and usage scenarios. First of all, UAVs usually conduct cruise operations at high altitudes of tens of thousands of meters, while UHs usually operate at low altitudes of thousands or even hundreds of meters. Secondly, UAV is usually used for high-altitude reconnaissance, confrontation, and other tasks on the battlefield, while UH is more used for low-altitude raid missions. Therefore, it is necessary to distinguish UH and UAV for separate research. In summary, the existing path planning algorithms cannot fully meet the path planning requirements of UH in complex battlefield environments.

UH needs to fly in low airspace for a long time during the raid mission, which will face ground obstacles and radar threats. Ground obstacles such as mountains are usually stationary, and it is not difficult to accurately identify and avoid them. Due to factors such as terrain and the curvature of the earth, the probability of UH being detected by radar will vary with the flight height, which means that it will face a dynamically changing threat area for a long time. Traditional path planning algorithms generate optimal paths based on real-time environmental information, which is effective in the face of static threat areas. However, the security status of some locations within the dynamic threat changes in real time, and some locations are passable at one time and not at another. Therefore, the paths planned by traditional path planning algorithms are not absolutely safe in the face of dynamic threat areas. A good solution to this problem is to accurately identify the dynamic threat area and avoid it entirely. Deep Q-network (DQN) algorithm is the product of the combination of neural network and Q-learning algorithm. It can not only process large state space information but also interact with the environment to seek optimal strategies when the environment state is unknown. With appropriate reward settings, the DQN algorithm can accurately identify the dynamic threat area and avoid it by interacting with the environment. Therefore, using the DQN algorithm for path planning can help UH to effectively avoid dynamic threat areas in the battlefield environment. This paper was aimed at providing an effective path planning method based on DQN algorithm, which can help UH to effectively avoid the dynamically changing radar coverage area and successfully complete the raid task in complex environment.

Based on the above analysis, a heuristic deep Q-network (H-DQN) algorithm is proposed in this paper. We study the ability of the proposed algorithm to plan paths for UH in the complex environment and try to make the planned paths smoother, thereby reducing the manoeuvring consumption of UH. Compared with traditional algorithms, the H-DQN algorithm can effectively identify the dynamically changing radar coverage area and help UH plan a safe and effective flight path. The main contributions of this paper are as follows:

(1) A heuristic comprehensive reward function is designed, which mainly includes two parts: heuristic reward function and smooth reward function. The heuristic reward function can promote the rapid convergence of the algorithm and effectively improve the sparse reward problem faced by traditional reinforcement learning. The smooth reward function can constrain the UH's behaviour, prompting the UH to choose a smoother path for flight, thereby reducing flight consumption. The proposed heuristic comprehensive reward function integrates the information of environmental constraints and motion constraints, which can effectively promote the convergence speed of algorithm and further improve the quality of planned path. It has certain versatility and can be combined with other intelligent algorithms

(2) We model the dynamic threat constraints faced by UH in low-altitude raid missions and apply the proposed H-DQN algorithm to UH path planning. The modelling of dynamic constraints fully considers the complexity of the battlefield environment, which can reflect the difficult situations faced by UH in the application of the battlefield. The proposed algorithm embedded in the environment model for path planning is described in detail, which provides a new solution to the path planning problem

The rest of this paper is structured as follows. The related works are presented in next section. In Section 3, numerical analysis and modelling of the complex low airspace environment faced by UH are carried out. Section 4 introduces the deep reinforcement learning methods. In Section 5, the design of the comprehensive reward function and the proposed H-DQN algorithm are explained in detail. In Section 6, the experimental results and comparative analysis are presented. The conclusions are presented in Section 7.

2. Related Works

Path planning technology usually refers to finding the optimal path from the starting position to the target position according to certain evaluation criteria under certain environmental constraints [9]. Path planning algorithms are usually divided into global path planning algorithms and local path planning algorithms [10]. Among them, the global path planning algorithm requires that the environmental model is known, and the algorithm can generate the global optimal path according to the environmental constraints, and the representative one is the A∗ algorithm [11]. The A∗ algorithm was used to compute near-optimal paths in static and dynamic environments with underwater obstacles in the literature [12], which completed path replanning and obstacle avoidance for unmanned underwater vehicles. By using a modified A∗ method in the literature [13], the global path planning problem of a robot was solved by establishing an approximation to the optimal path. In the literature [14], the authors propose a novel decentralized coordination scheme for autonomous ground vehicles to enable map building and path planning with a network of smart overhead cameras, and the A∗ algorithm is used to calculate the path. However, since the A∗ algorithm requires the environment model to be fully known when performing path planning, its scope of application is relatively limited.

The local path planning algorithm can make corresponding decisions according to the local environment information and explore the passable path when the global information is unknown by interacting with the environment. The more representative algorithms include genetic algorithm, dynamic window approach, ant colony algorithm, particle swarm algorithm, and artificial potential field method [15–20]. In the literature [15], the authors find the optimal flight path for the UAV by using an improved genetic algorithm with a new genetic factor on the basis of the probability map. Aiming at the problem that the classical DWA has an unreasonable path in dense obstacles and cannot guarantee speed and safety at the same time, the literature [16] proposes an adaptive DWA algorithm, which is successfully applied to the local path planning of the robot. A heterogeneous UAV coverage path planning algorithm based on ant colony algorithm is proposed in [17]; the author applied ant colony algorithm to cooperative search system to minimize the time consumption of the task. An improved particle swarm algorithm was proposed to solve the problem of path planning for an unmanned aerial vehicle (UAV) in adversarial environments including radar-guided surface-to-air missiles (SAMs) and unknown threats in the literature [18]. In order to efficiently complete the underwater information collection, a heterogeneous AUV-aided information collection system with the aim of maximizing the energy efficiency of IoUT nodes taking into account AUV trajectory, resource allocation, and the Age of Information (AoI) was proposed in the literature [19]. Particle swarm optimization algorithm was used as the method for the trajectory planning of underwater robots. In order to ensure the optimality, rationality, and path continuity of the formation trajectory of unmanned surface vehicles, a deterministic algorithm of multi-subtarget artificial potential field (MTAPF) based on improved APF was proposed in the literature [20]. MTAPF can greatly reduce the probability of USV falling into the local minimum and help USV escape from the local minimum by switching the target point. However, these algorithms generally have shortcomings such as difficult to guarantee convergence and easy to fall into local optimum, so their applicable scenarios are relatively limited. Among these algorithms, the genetic algorithm has been widely used in many fields because of its strong scalability and easy to combine with other algorithms [21]. In the literature [22], an improved cost function for a grid path planning in 2D static environment-based genetic algorithm (GA) was proposed, which was used to reduce the energy consumption of AUVs. A genetic algorithm was used to determine the optimized path with the minimum travel time for a USV under environmental loads in the literature [23]. A new hybrid algorithm which is based on genetic algorithm and firefly algorithm was proposed in the literature [24], which was used to solve the path planning problem of mobile robots. It is worth noting that the parameters of the genetic algorithm are numerous and complex, so its path search is inefficient and time-consuming.

Introducing reinforcement learning technology into path planning is a research hotspot in recent years. Reinforcement learning is a learning method that maps from the environment state to the action. By constructing a Markov decision model, the learner repeatedly interacts and explores with the environment to learn the optimal strategy. Reinforcement learning does not require complete prior knowledge. Since learners can independently obtain optimal behaviour strategies through dynamic interaction with the environment when facing an unfamiliar environment, the application of reinforcement learning to path planning has certain advantages. According to the update method of the policy, reinforcement learning can be classified into value function-based and policy gradient-based [25], among which value function-based reinforcement learning is more widely used. As a kind of value function-based reinforcement learning algorithm, Q-learning algorithm has been widely used in the field of path planning. In order to prove the ability of Q-learning algorithm to interact with the environment, the Q-learning algorithm was used to extract the state of the environment in the literature [26]. The path planning task of a mobile robot in an unknown environment was accomplished by combining the Q-learning algorithm with the dynamic window approximation algorithm. In the literature [27], the Q-learning algorithm is used to complete the autonomous navigation and control of intelligent ships in simulated waterways. The author completed the environmental information modelling during the ship's navigation and set environmental factors such as obstacles and restricted areas as reward and punishment information. By combining the Q-learning algorithm, a multi-AUV collaborative data acquisition algorithm was proposed in the literature [28], which can reduce the data acquisition load of a single AUV and serve as a path planning algorithm for autonomous underwater vehicles. However, as the environment becomes more complex, the state space of the environment is also becoming larger, and the problem of state space explosion occurs at this time, which makes it difficult for the traditional Q-learning algorithm to converge.

Relevant studies have shown that deep reinforcement learning formed by the combination of deep learning and reinforcement learning can effectively improve the state space explosion problem [29]. The DQN algorithm is formed by combining the deep neural network and the Q-learning algorithm. The appearance of DQN algorithm further solves the problem of path planning. In the literature [30], a deep reinforcement learning method ANOA based on dueling deep Q-network was proposed, which tailored design of state and action spaces and the reward function. In the literature [31], a smoothly convergent DRL (SCDRL) method was proposed based on the deep Q-network (DQN) and reinforcement learning, which to solve the path following problem for an underactuated unmanned-surface-vessel (USV). Aiming at the problem of vehicle model tracking error and overdependence in traditional path planning of intelligent driving vehicles, a path planning method of intelligent driving vehicles based on deep reinforcement learning was proposed in the literature [32]. A novel hierarchical framework to achieve real-time path planning and following for a gliding robotic dolphin was proposed in the literature [33], which present a novel hierarchical deep Q-network method to separately plan the collision avoidance path and the approach path and also design different continuous states under the kinematic constraints.

Based on the above analysis, it can be seen that the DQN algorithm has obvious advantages in solving path planning problems in complex environments. In view of the complexity of UH's model for performing raid tasks, a heuristic DQN algorithm for UH's path planning based on the deep reinforcement learning DQN algorithm is designed in this paper.

3. Environment Model

Figure 1 is an illustration of the battlefield environment that UH faces when performing low airspace raid missions. The helicopter flight area $\Omega_{\text{helicopter}}$, the mountain area Ω_{mountain}, and the radar coverage area Ω_{radar} are included in the complex battlefield environment Ω_X. At the same time, any position in Ω_X is represented by $X = (x, y)$, x represents the horizontal position, and y represents the height.

The experimental environment is set as a low airspace with a length of 50 km and a height of 1 km. UH's mission is to raid radar positions 50 km away. It is assumed that the UH is equipped with a radar warning device that can determine whether it is locked by the radar, and the position of the target radar is known. The position coordinates of the radar are expressed as

$$Radar_{(x,y)} = [50, 0]. \tag{1}$$

UH avoids colliding with mountains during flight. Assuming that the height of the mountain is 0.15 km, its position is expressed as

$$\text{Mountain}_{(x,y)} = [20, 0.15]. \tag{2}$$

As shown in Figure 2, due to the powerful manoeuvrability of UH, it can do 8 degrees of freedom within Ω_X. The flight

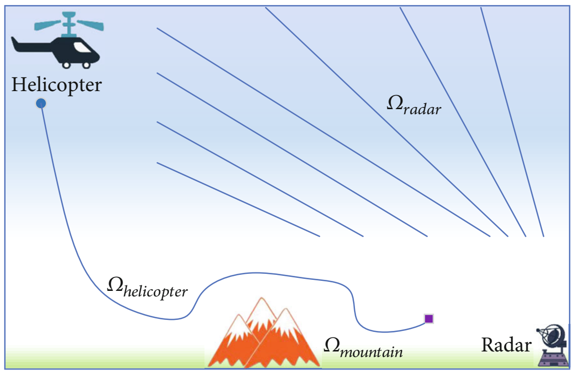


Figure 1: Illustration of the complex battlefield environment.

speed of UH can be decomposed into two directions of x-axis and y-axis, which can be represented by $\overrightarrow{v_x}$ and $\overrightarrow{v_y}$.

$$\begin{cases} \overrightarrow{v_x} = 360\text{km/h}, \\ \overrightarrow{v_y} = 36\text{km/h}. \end{cases} \tag{3}$$

Then, the position of UH in Ω_X can be expressed as

$$\text{UH}_{(x,y)}(t) = \left[\text{UH}_{(x)}(t), \text{UH}_{(x)}(t)\right] = \left[\overrightarrow{v_x}t, \overrightarrow{v_y}t\right]. \tag{4}$$

In the process of UH flight, the distance from the obstacle is greater than the safety radius R_{safe}, which is the premise of its own safety. The safety radius R_{safe} is determined by the following factors:

$$\Omega_{R_{\text{safe}}} = \left\{ \left\| \text{UH}_{(x,y)}(t) - \text{Mountain}_{(x,y)} \right\| \bigcup \left\| \text{UH}_{(x,y)}(t) - \Omega_{\text{radar}} \right\| \geq R \right\}. \tag{5}$$

Due to the large difference between the horizontal and vertical speeds of UH, its safe radius R_{safe} should also be decomposed into two directions of x-axis and y-axis. The UH safety radius is

$$R_{\text{safe}} = \begin{cases} x = 1\text{km}, \\ y = 0.05\text{km}. \end{cases} \tag{6}$$

The maximum attack distance of UH is 8 km. Assuming that each attack of UH is regarded as a hit, the condition for completing the task is that the distance d_t from the radar is less than 8 km, namely,

$$d_t = \left| \text{UH}_{(x,y)}(t) - \text{Radar}_{(x,y)} \right| = \sqrt{\left(\text{UH}_{(x,y)}(t) - 50\right)^2 + \left(\text{UH}_{(x,y)}(t) - 0\right)^2} \leq 8\text{km}. \tag{7}$$

The maximum detection range of the radar is 45 km. Due to factors such as ground reflection clutter and the curvature of the earth, it is usually difficult for radars to detect low-

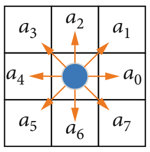


FIGURE 2: Illustration of UH movement direction.

flying targets. The radar detection probability is expressed as

$$i = \begin{cases} 0, & d > 45\text{km}, \\ 1, & d \le 45\text{km}, h \ge 1\text{km}, \\ \dfrac{1}{1 + e^{(-(20h-7))}}, & d \le 45\text{km}, 0.2\text{km} < h < 1\text{km}, \\ 0, & h \le 0.2\text{km}. \end{cases} \quad (8)$$

In equation (8), d represents distance information, and h represents height information. Equation (8) is a simplification of the real radar detection probability, which can represent the distribution of the radar detection probability, but is not the real data. The probability model of UH flight process detected by radar can be obtained from Figure 3:

In Figure 3, the d-axis is the distance between the helicopter and the radar position, the h-axis is the flying height of the helicopter, and the i-axis is the probability of being detected by the radar.

Based on the above information, the passable condition C_m and impassable condition C_{um} of UH can be obtained as

$$\begin{aligned} C_m &= \{\Omega_{\text{movable}} | f(\Omega_h) = 0\}, \\ C_{um} &= \{\Omega_{\text{umovable}} | f(\Omega_{\text{mountain}}, \Omega_{\text{radar}} = 1) = 1\}, \end{aligned} \quad (9)$$

where $f(x)$ represents the judgment function and 0 and 1 represent movable and completely immovable, respectively. $\Omega_{\text{radar}} = 1$ means UH is detected by radar; if $\Omega_{\text{radar}} = 0$, it means that it is not detected.

Modelling the battlefield environment is the first step in path planning. Through the above numerical analysis, we introduce the entire battlefield environment in detail, define the movement mode and behaviour constraints of UH, and clarify the passable and impassable areas in the environment. In order to successfully complete the raid mission, UH needs to reach the attack area safely, and in the process, it needs to avoid hitting mountains and being detected by radar. Since the radar threat area is dynamic, the best way to keep UH safe is to avoid crossing the radar coverage area. Therefore, to measure the quality of the planned path, indicators such as path length, path smoothness, whether there is a collision, and whether it crosses the radar coverage area should be integrated. The ideal planned path should have a short length and good smoothness, so as to effectively reduce the flight consumption of UH. Avoiding collisions and avoiding crossing the radar area are prerequisites for UH safety. It can be seen from Figure 3 that within the range of the flight height $h \in (0.2, 0.5)$, the probability of UH being detected by radar is not 100%, which makes it difficult for UH to accurately identify the radar coverage area and avoid crossing. To sum up, this path planning task is challenging.

4. Deep Reinforcement Learning Methods

It is the core content of path planning that requires a suitable algorithm model for path search. In this section, we introduce the reinforcement learning Q-learning algorithm and the deep reinforcement learning DQN algorithm, respectively, and explain the experience playback and target network mechanisms in the DQN algorithm. These algorithms are the key to completing the path search and are the basis of our proposed algorithm.

4.1. Reinforcement Learning. Four elements, state set S, action set A, state transition probability P, and reward set R, are included in the reinforcement learning model. Define the strategy $\pi : S \longrightarrow A$ which is the mapping from state to action. In the current state s, the learner will choose the action a according to the policy π. When action a is executed, the environment will transition to the next state s' with probability P and receive reward r from the environment. The purpose of reinforcement learning is to maximize the cumulative reward by adjusting the policy. Value functions can be used to judge the pros and cons of a strategy. Assuming that the initial state of the learner is $s_0 = s$, the state value function of the policy π is defined as

$$V^\pi = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_t = \pi(s_t), \quad (10)$$

where $\gamma \in (0, 1)$ is the decay factor, which is used to specify the decay degree of future rewards. Reinforcement learning can take the maximization function as the optimal strategy, which can be expressed as

$$\pi^* = \max_\pi V^\pi(s), \forall s \in S. \quad (11)$$

Due to the Markov property of reinforcement learning, the state action value function can be expressed as

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma V^\pi(s_{t+1}), \quad (12)$$

where $Q^\pi(s_t, a_t)$ represents the expected reward of choosing action a_t in state s_t. At this time, the optimal strategy should be reexpressed as

$$\pi^* = \max_\pi Q^\pi(s, a), \forall s \in S, \forall a \in A. \quad (13)$$

Q-learning is a relatively mature and widely used reinforcement learning algorithm. Q-learning is a reinforcement learning algorithm based on value function, and its update method can be expressed as

$$(s, a) = Q(s, a) + \alpha \left(r + \gamma \max_{a'} Q\left(s', a'\right) - Q(s, a) \right), \quad (14)$$

where $\alpha \in (0, 1]$ is the learning rate, which is used to control the proportion of future rewards in the learning process. For formula (14), if each state s and action a are visited infinitely, and the decay factor γ takes an appropriate value, then the Q

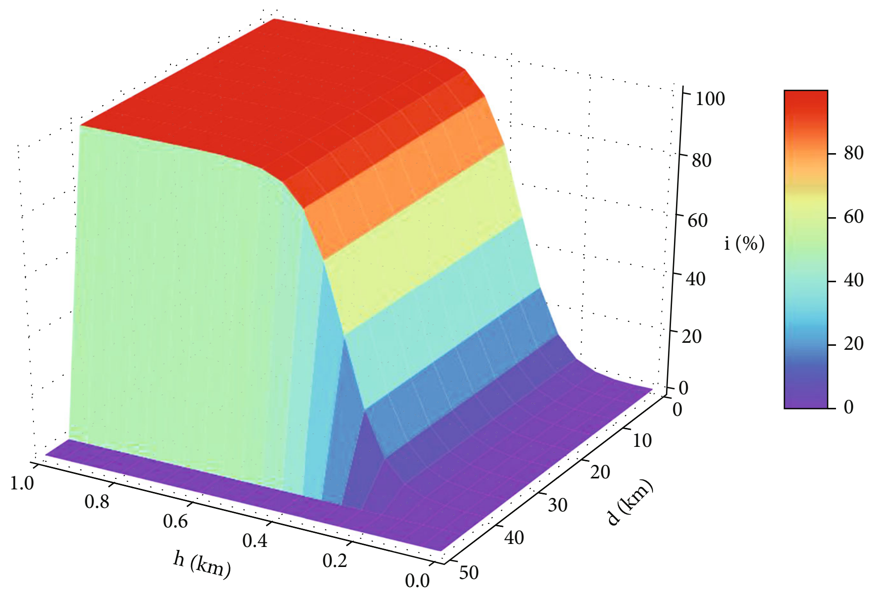


Figure 3: Radar detection probability model.

value will eventually converge to a fixed value. It is worth noting that the Q-learning algorithm needs to generate a Q table during operation to store the *Q* value corresponding to different actions in each state, so the algorithm needs to keep reading and writing the *Q* value to update the Q table.

4.2. Deep Reinforcement Learning. The combination of reinforcement learning and neural network has been studied in the early days, but simply combining the two has not achieved the desired effect [34]. The proposal of DQN algorithm provides a powerful boost for the development of deep reinforcement learning [35]. Experience replay and the proposal of target network mechanism are important reasons for the success of DQN algorithm. Since the correlation of samples in deep reinforcement learning is much larger than in simple reinforcement learning, the purpose of experience replay is to make the deep neural network converge to the same step size, so that the algorithm gradient descent moves in the same direction, thereby promoting the algorithm convergence. At the same time, the experience replay mechanism requires the algorithm to randomly sample training samples from the experience pool, which can improve data utilization. The experience replay mechanism effectively solves three problems: overcoming the correlation of empirical data, reducing the variance of parameter updates, and overcoming the nonstationary distribution problem [36].

The principle of the DQN algorithm is to combine reinforcement learning and deep neural networks, use the Q-learning algorithm to provide labelled samples for the neural network, and then use the gradient descent method of backpropagation to update the neural network parameters. The DQN algorithm uses a neural network to fit the update process of the Q-learning algorithm:

$$Q(s, a, \omega) \approx Q(s, a), \tag{15}$$

where ω represents the neural network parameters. The DQN algorithm takes the state s as the input and the Q value corresponding to different actions as the output, so that the Q value information is stored in the neural network node. Therefore, the DQN algorithm does not need to generate a Q table, and the update of the Q value is completed by updating the parameters of the neural network. The loss function $L(\omega)$ in the algorithm update process can be expressed as

$$L(\omega) = E\left[\left(r + \gamma \max_{a'} Q\left(s', a', \omega\right) - Q(s, a, \omega)\right)^2\right], \tag{16}$$

where $Q(s, a, \omega)$ is generated by the evaluate training network and $Q(s', a', \omega)$ is generated by the target value network. The parameters of the target value network are exactly the same as the training network. When the algorithm updates a certain number of steps, the parameters of the evaluate training network will be completely copied to the target network. The target value network can solve the problem of strong data dependence when a single network is updated, thus effectively promoting the convergence of the algorithm.

The algorithm update process uses the stochastic gradient descent algorithm to update the network parameter ω, and the update value $\Delta\omega$ can be expressed as

$$\frac{\delta L(\omega)}{\delta \omega} = \left[r + \left(\gamma \max_{a'} Q\left(s', a', \omega\right) - Q(s, a, \omega)\right)\right] \frac{\delta Q(s, a, \omega)}{\delta \omega}. \tag{17}$$

DQN algorithm code description can be seen in Algorithm 1:

5. Heuristic Deep Q-Network Algorithm

In this section, we detail the proposed process of the heuristic DQN algorithm and embed it in the UH path planning task. We first describe the state-action ensemble of the UH performing low-altitude raid task model and then design a heuristic synthetic reward function.

5.1. State and Action Sets. The partitioning of state and action sets is the first step in reinforcement learning algorithms. Since the UH is moving in the environment, its motion path is a time-dependent nonlinear function. Considering that the DQN algorithm requires the state to be discrete, the environment model needs to be discretized. The grid method can be used to discretize the system environment. First, the airspace environment is divided into 500 squares using 50×10 squares, and each square can correspond to a state s_i of the environment. At this time, the path of the UH is also discretized into a series of time-related location points. Combined with the movement speed of the UH, 10 s can be taken as a time step; that is, the UH will complete a state transition in each time step. Then, state s_i is represented in vector form:

$$s_i = [s_{i1}, s_{i2}, s_{i3}, s_{i4}], \tag{18}$$

where s_{i1} and s_{i2} can correspond to x and y in $X = (x, y)$, respectively, indicating the position information. s_{i3} indicates collision information, $s_{i3} = 1$ indicates a collision, and $s_{i3} = 0$ indicates no collision. s_{i4} means radar detection, $s_{i4} = 1$ means detected, and $s_{i4} = 0$ means not detected. Through the above analysis, the battlefield environment is divided into 500 states; that is, the state set S can be expressed as

$$s_i \in S, i \in [0,499]. \tag{19}$$

UH performs an action per time step. Since UH performs 8-dimensional motion, the action set A can be expressed as

$$a_i \in A, i \in [0, 7]. \tag{20}$$

The movement direction of a_i in formula (20) is consistent with that in Figure 2.

Path search is a key step in path planning, and the division of state-action sets is a prerequisite for path search. The state set can effectively pass the UH position and environmental constraints to the algorithm for path search. The action set defines the movement mode of the UH in the environment and further clarifies the action constraints.

5.2. Comprehensive Reward Function. The setting of the reward function is crucial for reinforcement learning algorithms. Selecting an appropriate reward function can effectively promote the algorithm convergence, while inappropriate reward function settings may lead to difficulty in algorithm convergence [37]. In traditional reinforcement learning algorithms, when a learner completes a task, there is a corresponding reward, while the previous series of behaviours are not rewarded. Some studies have pointed out that this kind of reward can lead to sparse reward problem in the face of complex environment [38]. When the set of environmental states is large, the learner encounters a series of nonfeedback states before completing the task. Since the effective reward cannot be obtained in time, the algorithm will be difficult to converge. In order to improve this problem, we design a heuristic reward function r_1, and its specific expression is

$$r_1 = \begin{cases} \dfrac{d_{\max} - d_{t+1}}{d_{\max} - d_{\min}}, & d_{t+1} < d_t, \\ -\dfrac{d_{t+1} - d_{\min}}{d_{\max} - d_{\min}}, & d_{t+1} \geq d_t, \end{cases} \tag{21}$$

where d_t and d_{t+1} are the distances between the UH and the radar target in the current state and the next state, respectively, and $d_{\max}$ and $d_{\min}$ represent the maximum and minimum distances, respectively. It can be seen from formula (21) that the value of the reward r_1 will be related to d_{t+1} in real time. When the UH takes a certain action a_t, its distance from the radar target will change. If UH is closer to the radar, a_t will get a positive reward; otherwise, it will be penalized (negative reward). The analysis shows that when the distance between the UH and the radar is large, the value of the negative reward is relatively large. At this time, the UH will quickly approach the radar under the constraint of negative reward. As the distance between the UH and the radar decreases, the value of the negative reward decreases, while the value of the positive reward increases. Then, the constraints of negative rewards will gradually weaken and the effects of positive rewards will increase. At this time, since UH takes suboptimal actions, it will not be severely punished, and it can continue to explore suboptimal actions while approaching the target, so as to seek the optimal path.

Considering the motion constraints, frequently changing the motion direction is unfavorable for UH, especially for large corners. Frequently changing the movement direction will increase the flight consumption of the UH and also affect the flight safety. In order to make the planning path smoother, the smooth path reward function r_2 is designed, and its specific expression is as follows:

$$r_2 = -\frac{\angle a_t a_{t+1}}{180^\circ}, \tag{22}$$

where $\angle a_t a_{t+1}$ represents the acute angle value of the angle between the motion directions of a_t and a_{t+1}. It can be seen from formula (22) that we punish the corner behaviour, and the value of the negative reward increases with the increase of the corner. This effectively constrains the corner behaviours and makes the planned path smoother.

It is worth noting that both r_1 and r_2 are rewards obtained when UH satisfies the passable condition C_m. When UH satisfies the impassable condition C_{um}, that is, the distance between UH and the obstacle is less than the

```
Algorithm: DQN algorithm
Initialization: initialize training network parameter $\omega$and target network parameter $\omega'$, $\omega = \omega'$.
Iterative process:
Repeat (for each episode)
        Initialization state $s$
          Repeat (for each step)
                Select action $a$ based on the $\varepsilon - greedy$ policy
                Perform action $a$ to get reward $r$ and next state $s'$
                Store transition $(s, a, r, s')$ in the experience memory
                Sample random mini batch$(s, a, r, s')$ from the experience memory
                $y_i = \begin{cases} r_j & \text{for} - terminal \\ r_j & +\gamma \max_{a'} Q(s', a', \omega) \text{ for non} - terminal \end{cases}$
                Loss function$L(\omega)$ is obtained
                Updating network parameters
                $s = s'$
          End Repeat ($s'$is the terminated state)
End Repeat (End of the training)
```

Algorithm 1: DQN algorithm.

safety radius R_{safe}, set the reward r_3 as

$$r_3 = -1000, \tag{23}$$

where the value of the negative reward is larger, because we want to strictly prohibit the occurrence of collision events. When UH completes the raid mission, set the reward r_4 to

$$r_4 = 1000. \tag{24}$$

We strongly approve of the behaviours of completing the task, so we can get a large reward. We summarize the above reward settings into a comprehensive reward function R_c:

$$R_c = \theta_1 r_1 + \theta_2 r_2 + \theta_3 r_3 + \theta_4 r_4, \tag{25}$$

where $\theta_1, \theta_2 \in (0,\infty)$ are reward coefficients. Under normal circumstances, the value of θ_3 is 0. When a collision event occurs, θ_3 is set to 1. At this time, the system will exit and start learning again. Similar to θ_3, the value of θ_4 is also 0 under normal circumstances. Only when the task is completed will θ_4 take the value of 1. At this time, the system will still exit and start learning again.

To sum up, the comprehensive reward function designed can generate dynamic rewards in real time in combination with environmental information, so that UH has good control performance and can make the planning path smoother. The heuristic comprehensive reward function can optimize the search according to the continuously estimated environmental cost information, which makes the process of reward accumulation smoother, thus effectively improving the sparse reward problem in complex environments. The positional relationship between UH and radar targets, motion constraints, and environmental constraints is effectively integrated by the reward function, which further improves the efficiency of path search.

5.3. Heuristic Deep Q-Network Algorithm Model.

5.3. Heuristic Deep Q-Network Algorithm Model. Figure 4 shows the algorithm model, which clearly shows the whole process of using H-DQN for path search. It can be seen from the figure that after the algorithm runs, the s containing the UH location information and the surrounding environment information will be used as the input of the neural network. After the Q values of the different actions corresponding to the state s are output by the neural network, the algorithm will select the action a corresponding to the largest Q value as the output according to the ε-greedy strategy. When UH performs action a, state s will change to state s'. At this time, the environment will evaluate the action a according to the comprehensive reward function and get the reward r. After this series of actions is completed, the complete quaternary information group (s, a, r, s') will be obtained. According to the experience replay mechanism, the quaternary information group will be stored in the experience pool. If the experience pool stores data of a certain size, it can start to random sample learning on the experience pool. In the process of learning, the current state s will be used as the input of Evaluate Net to obtain the actual value $Q(s, a\,;\omega)$, and the next state s' will be used as the input of Target Net to obtain the estimated value $Q(s', a'\,;\omega)$. Next, $Q(s, a\,;\omega)$, $Q(s', a'\,;\omega)$, and reward r are used as input to the loss function to get the mean squared error. Finally, the Evaluate Net is updated by the stochastic gradient descent method, thus completing the optimization of the action selection strategy. Through the above process, the proposed H-DQN algorithm can effectively use the environment model information to complete the search of the safe path. After a lot of training, the algorithm and the environment model fully interact, and the neural network parameters will tend to be stable. The trained network model can be loaded directly when it is used, so that the input state can be automatically recognized and the corresponding correct action can be obtained. We can get the complete safe flight path by outputting the

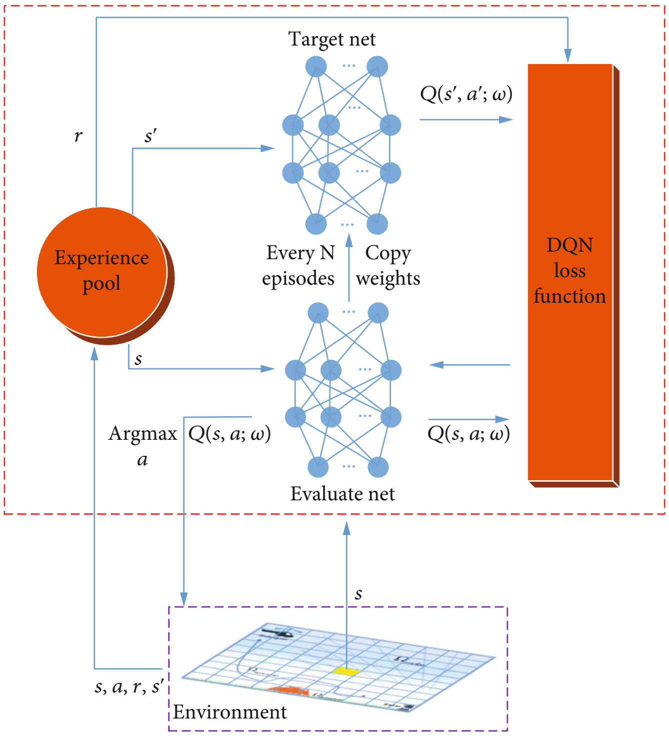


Figure 4: H-DQN algorithm model.

real-time positions of the UH after performing the corresponding actions in sequence. So far, we have completed the entire process of path planning.

The division of the state-action set is a key step in embedding the reinforcement learning algorithm model into the path planning problem, and designing an appropriate reward function is an important way to improve the performance of the algorithm. We describe these procedures in detail and frame the proposed algorithm so that it is convenient to extend these results to more general path planning problems. It is worth noting that the comprehensive reward function we designed has remarkable generality, not only applicable to the DQN algorithm and its various variants, but also can be combined with other intelligent algorithms that require reward constraints.

6. Simulation Experiment

In this section, the performance of the proposed H-DQN algorithm is evaluated through comparative experiments. To ensure the validity of the experiments, all experiments were carried out in the same environment. The construction of the experimental environment and the implementation of the algorithm are all done through Python code on the PyCharm platform. We are using Python-3.6.10 version, and the neural network is built with Tensorflow-2.6.0 version. All experiments were performed on the same computer with twelve Intel (R) Core (TM) i7-8700 CPU @ 3.20 GHz processor and one NVIDIA GeForce GT 430 GPU, running memory with 16 GB RAM.

6.1. Experimental Parameter Settings. The learning rate α is a parameter that has a great influence on the performance of reinforcement learning algorithms. In order to select appropriate parameters, a control experiment was carried out by selecting different learning rates. During the experiment, if the raid task is completed, UH will get 1 point; otherwise, it will not score. The score of the last 100 tasks performed by UH is used as the standard to measure the performance of the algorithm. In order to reduce the experimental error, the experiments under each parameter setting were carried out independently for 5 times, and the experimental results were averaged to obtain Figure 5.

Figure 5 shows the performance of the algorithm under different learning rate values. It can be seen from the figure that the performance of the algorithm is affected by the value of the learning rate. This is because the learning rate represents how well the learning effect is preserved with each algorithm update. The larger the value of the learning rate α, the more the learning effect is retained, and the faster the training speed will be. At this time, the algorithm is not stable enough and is prone to oscillation. The smaller the learning rate is, the less the learning effect is retained, and the slower the training speed will be. At this point, the algorithm will become relatively stable, but if the training speed is too slow, it will bring more time overhead, which is also unacceptable under certain circumstances. According to the results of this experiment, when the learning rate α is 0.005, the algorithm has the final performance, and the training time overhead at this time is also acceptable. In addition, after training, the algorithm scores can be stabilized within a certain range. It shows that the algorithm

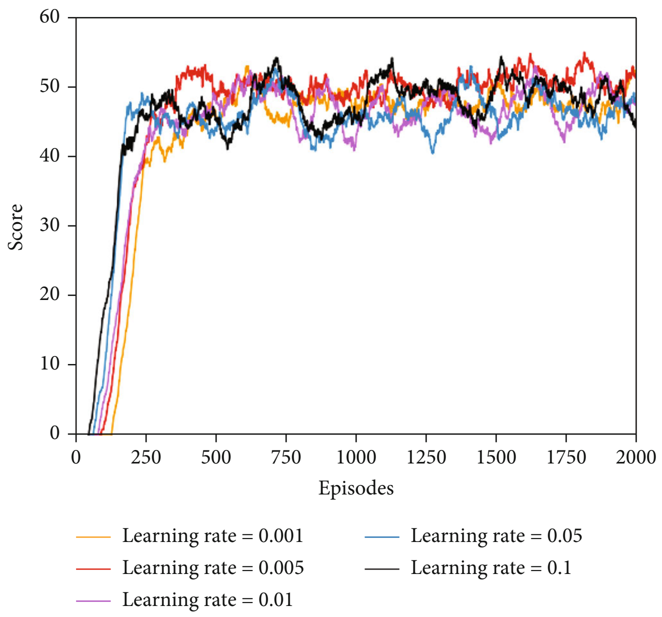


FIGURE 5: The score with different learning rates.

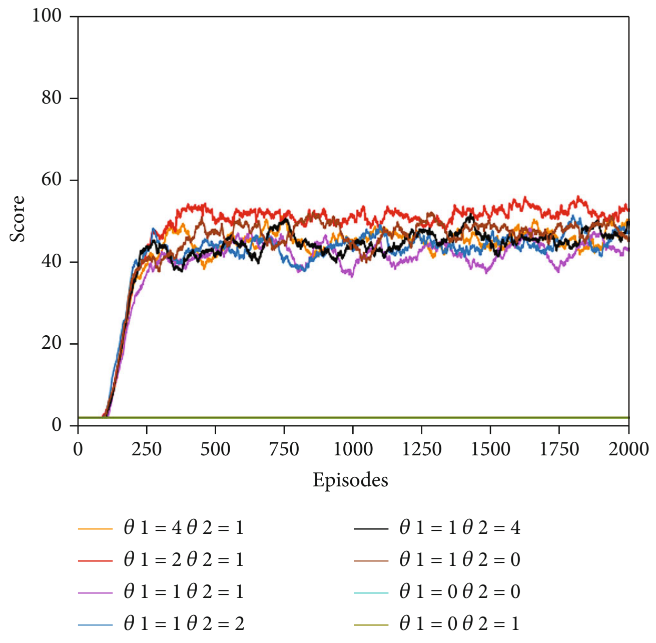


FIGURE 6: The score with different reward coefficients.

can converge smoothly under the background of this experiment.

In addition to the learning rate α, the decay factor γ is also an important parameter. The larger the decay factor γ is, the more the algorithm pays attention to past experience, and the smaller the value is, the more attention is paid to the current income. The value of γ can obtained based on past experience, and 0.9 is a common choice in the related literature. It is pointed out in the literature [39] that a suitable value of γ can be obtained by $\gamma = 0.1^{1/t}$. The t represents the number of steps to expect the algorithm to consider future rewards. A value of 0.9 means that the algorithm needs to consider the next 22 steps, which is appropriate in our experiments.

The influence of other parameters of the algorithm is as follows: if the exploration factor ε is too large, the algorithm tends to maximize the current profit and loses the motivation to explore, thus it may miss the bigger profit in the future. There are too few hidden layers and hidden layer neurons in a neural network to fit the data well and too many to learn effectively. If the batch size is too small, the sampling learning efficiency will be low, and if it is too large, the algorithm will easily converge to the local optimum. After many experiments, the algorithm parameters are finally set as follows: the learning rate α is 0.005, the decay factor γ is 0.9, and the exploration factor ε is 0.9. The input layer of the neural network and the state dimension are consistent with 4 neurons, the output layer and the action dimension are consistent with 8 neurons, and the hidden layer is two identical fully connected networks, each with 16 neurons. Neural network parameters ω are initialized with random values from a normal distribution with a mean of 0 and a standard deviation of 0.3. The parameter of Evaluate Net will be copied to Target Net every 200 episodes. The pool size is 3200 and the batch size is 32.

6.2. The Effect of Reward Coefficients. Since the comprehensive reward function designed this time is obtained by the weighted addition of each part of the reward function, the influence of the weight of each part of the reward function on the overall performance of the algorithm is a question worthy of discussion. In this section, we analyse the influence of each coefficient of the comprehensive reward function through multiple comparative experiments. Due to the particularity of the definition, θ_3 and θ_4 in equation (25) are not within the scope of our research and analysis, and we only talk about the effects of θ_1 and θ_2.The experiments

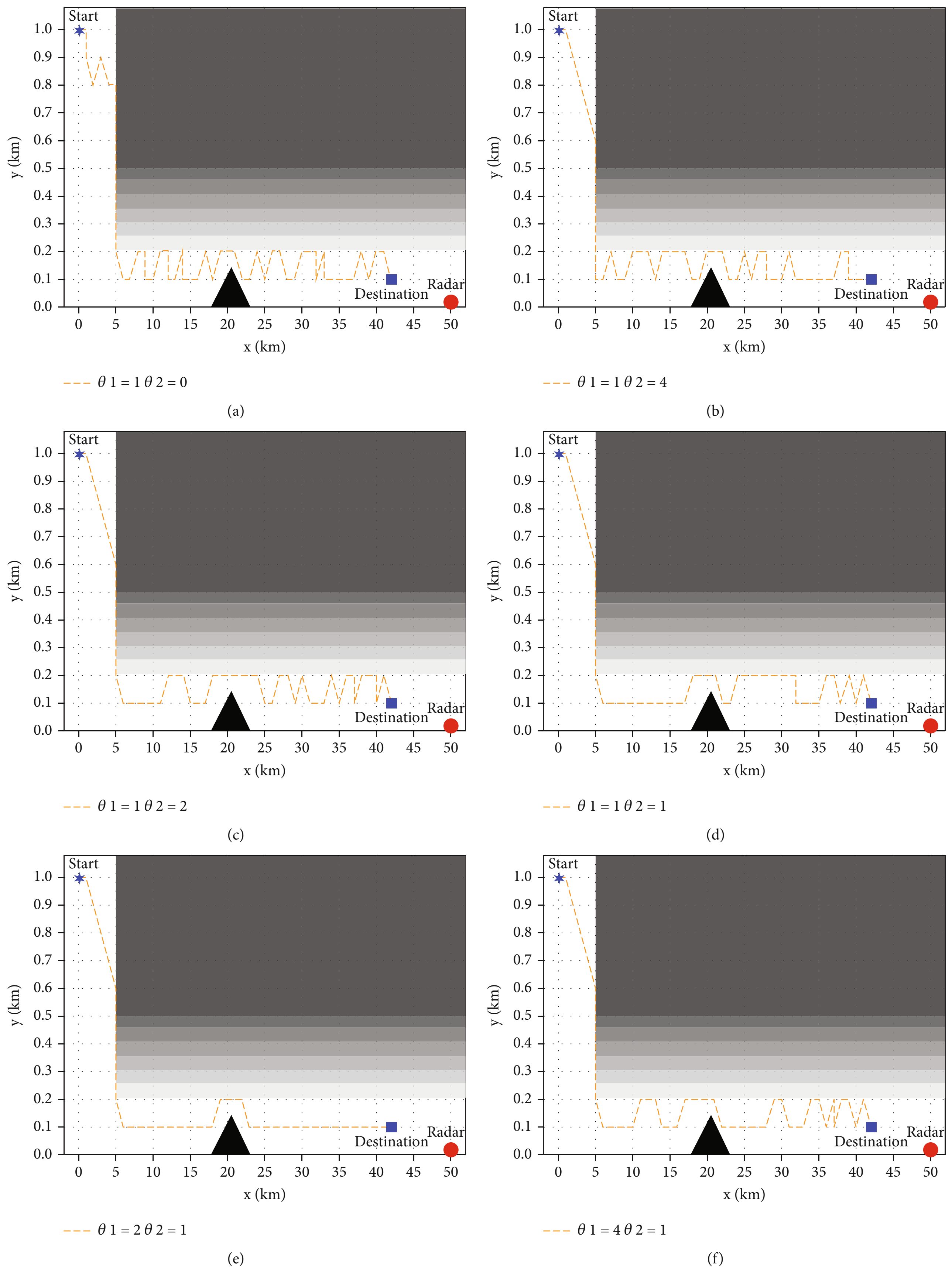
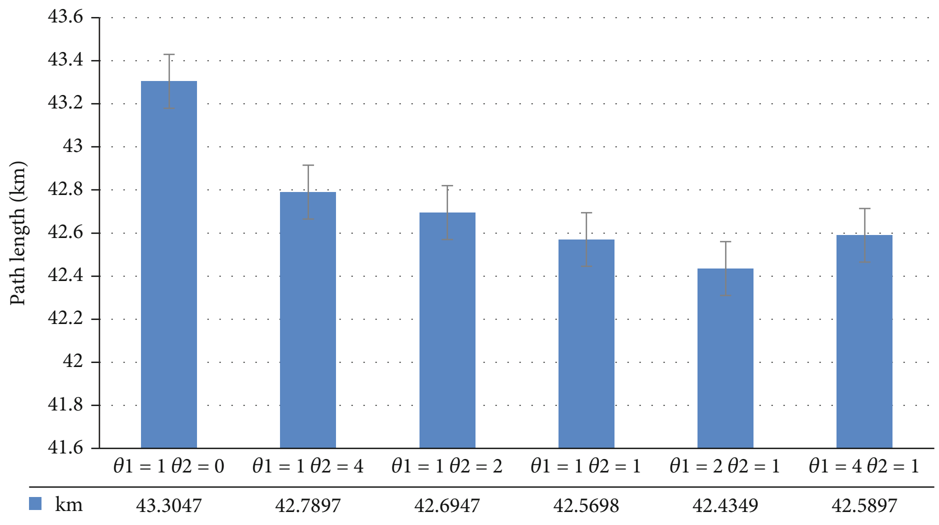
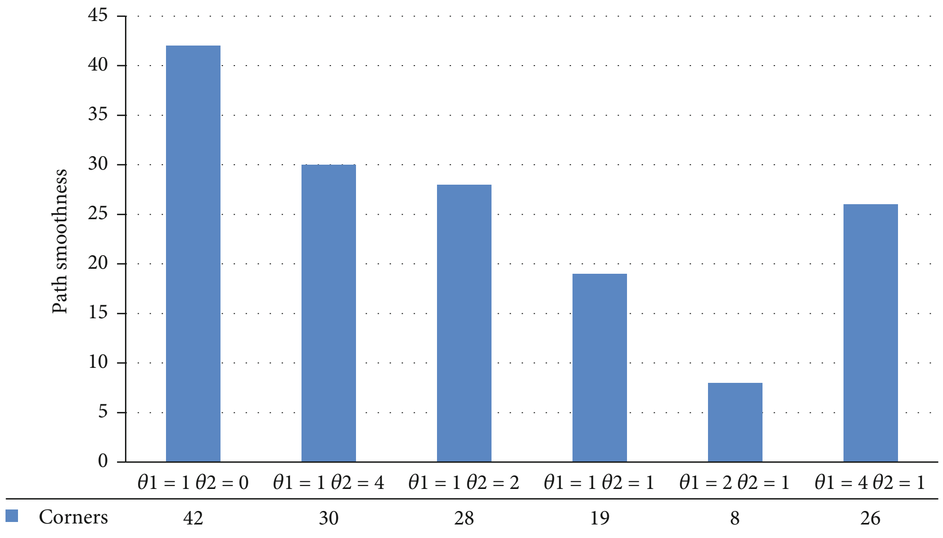


Figure 7: Path planning test for H-DQN in different reward coefficients. (a–f) The text results in the parameters θ_1 and θ_2 taking different values.



	θ1 = 1 θ2 = 0	θ1 = 1 θ2 = 4	θ1 = 1 θ2 = 2	θ1 = 1 θ2 = 1	θ1 = 2 θ2 = 1	θ1 = 4 θ2 = 1
🟦 km	43.3047	42.7897	42.6947	42.5698	42.4349	42.5897

Figure 8: Comparison of path length obtained for H-DQN in different reward coefficients. The interval on each bar denotes the standard deviation of the path length.



	θ1 = 1 θ2 = 0	θ1 = 1 θ2 = 4	θ1 = 1 θ2 = 2	θ1 = 1 θ2 = 1	θ1 = 2 θ2 = 1	θ1 = 4 θ2 = 1
🟦 Corners	42	30	28	19	8	26

Figure 9: Comparison of path smoothness obtained for H-DQN in different reward coefficients.

under each parameter setting were independently run 5 times, and the experimental results were averaged to obtain Figure 6.

Figure 6 shows the score of the algorithm when the parameters θ_1 and θ_2 take different values, respectively. We can see that, except for the two cases of $\theta_1 = 0, \theta_2 = 0$ and $\theta_1 = 0, \theta_2 = 1$, the algorithms under other parameter settings have converged after training. Since the algorithm cannot converge when the parameters $\theta_1 = 0, \theta_2 = 0$, it can be shown that the traditional sparse reward setting is difficult to adapt to the experimental background. At the same time, when the parameters $\theta_1 = 0, \theta_2 = 1$, the algorithm cannot converge, which means that without the guidance of the heuristic reward function, only the smooth reward cannot promote the algorithm to converge. Since the algorithm can successfully converge when $\theta_1 = 0, \theta_2 = 1$, it can be shown that the heuristic reward function effectively promotes the convergence of the algorithm. In the remaining five parameter settings, after training, the score of the algorithm can be stabilized within a certain range, which shows that the introduction of the smooth reward function does not affect the convergence ability of the algorithm. Using the trained algorithm for the path planning test results in Figure 7, the experiments under each parameter setting were independently run 5 times, and the experimental results were averaged to obtain Figures 8 and 9.

It can be seen from Figure 7 that when $\theta_1 = 1, \theta_2 = 0$, that is, only the heuristic reward function works alone, although the algorithm successfully completes the path planning task, the planned path oscillates to a large extent and the path is not smooth enough. Comparing Figure 7(a) with Figures 7(b)–7(f), it can be seen that the path planned by the algorithm after adding the smooth reward is significantly smoother. Figures 8 and 9 show the length and smoothness

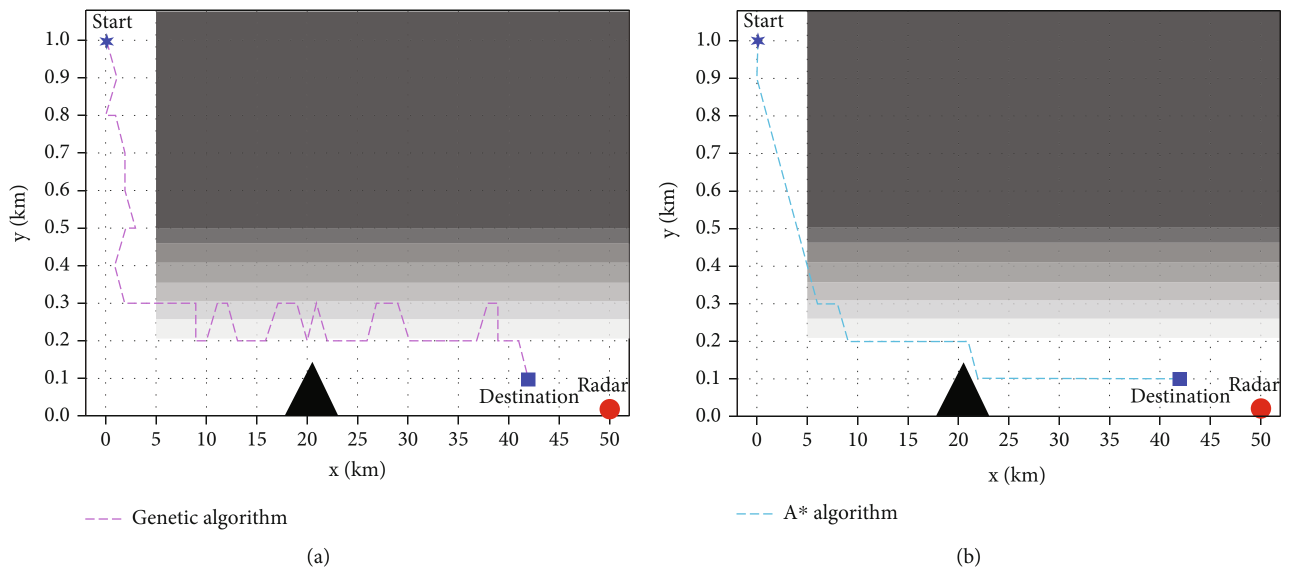


Figure 10: Path planning test for GA and A∗ algorithm.

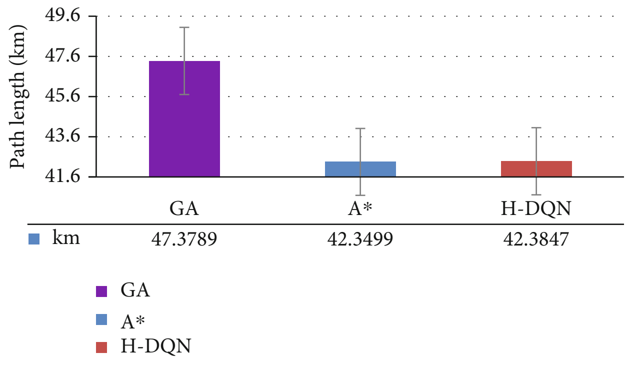


Figure 11: Comparison of path length obtained for GA, A∗, and H-DQN algorithms. The interval on each bar denotes the standard deviation of the path length.

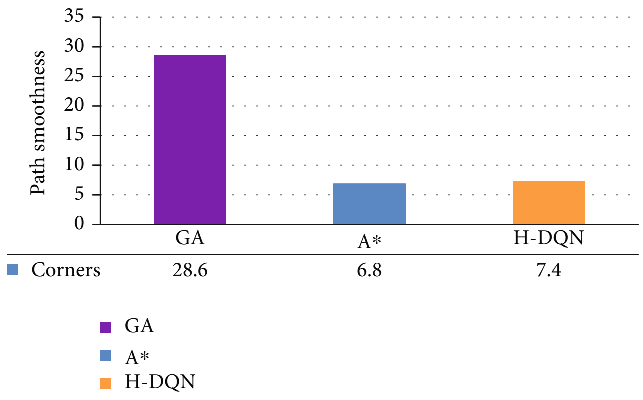


Figure 12: Comparison of path smoothness obtained for GA, A∗, and H-DQN algorithms.

of the paths planned by the algorithms with different values of θ_1 and θ_2, respectively. We can see that as the weight of θ_2 increases, the path planned by the algorithm becomes shorter and smoother, but when the weight of θ_2 is greater than a certain value, the performance of the algorithm begins to deteriorate.

It can be seen from the above experimental results that the heuristic reward can provide heuristic information for UH, guide UH to move closer to the goal, and effectively promote the algorithm convergence. The introduction of the smooth reward function can effectively promote the path planned by the algorithm to be smoother, but when the weight of the smooth reward function exceeds a certain value, its smoothing ability begins to gradually weaken. Combining Figures 7–9, we can see that in our experimental environment, when $\theta_1 = 2, \theta_2 = 1$, the algorithm can have the best performance.

6.3. Compare with Other Algorithms. In order to further prove the good performance of the H-DQN algorithm proposed in this paper, the more representative path planning algorithms A∗ algorithm and GA algorithm are selected for comparative experiments. During the experiment, the A∗ algorithm uses the Manhattan distance as the key value calculation method. The population size of the GA algorithm is 100, the terminating evolutionary generation is 200, and the crossover probability and mutation probability are both set to 0.8. The two algorithms have carried out 10 times independent experiments and selected the more representative path planning results to obtain Figure 10. Figures 11 and 12 are obtained after averaging the 10 times experimental results.

It can be seen from Figure 10 that although GA and A∗ algorithm have successfully completed the path planning task, the paths planned by the two algorithms all pass through the radar coverage area. Combining with Figure 7(e), it can be seen that the H-DQN algorithm does not cross the radar coverage area. On the one hand, because GA and A∗ algorithm is based on the real-time state of the environment model when searching for the path, there is a certain lag in the path planned in this way for the dynamic threat area that changes in real time, and it cannot fully reflect the real state of the environment model. On the other hand, since the radar coverage area in the environment model is in the form of probability distribution, in this case, the GA and A∗ algorithm will consider part of the area within the radar coverage area as safe when performing path search. Therefore, the path planned by the GA and A∗ algorithm has a high probability of passing through the radar coverage area. However, due to the constantly changing position of the safe area, the radar coverage area is inherently dangerous and should not be passed. In the process of path search, whenever UH is detected by radar, the H-DQN algorithm will get a negative reward. Then, after a lot of training, the radar coverage area will be explored by the H-DQN algorithm. Since the update strategy of the H-DQN algorithm is to maximize the cumulative reward, when the algorithm converges, the radar coverage area will be regarded as a forbidden area by the algorithm and will no longer be entered. In summary, the UH trained by H-DQN algorithm can accurately identify the radar coverage area and avoid passing, which is difficult for GA algorithm and the A∗ algorithm.

Figures 11 and 12 show the comparison of the average length and average smoothness of the paths planned by the GA, A∗, and H-DQN algorithms. We can see that among the three algorithms, the GA algorithm performs poorly, while the A∗ algorithm and the H-DQN algorithm perform relatively well. Further comparison shows that the difference in length and smoothness of the paths planned by the A∗ algorithm and H-DQN algorithm is small, and the performance of the A∗ algorithm is even better than that of the H-DQN algorithm. However, since the planned path should avoid crossing the radar coverage area, the H-DQN algorithm is a better choice.

7. Conclusions

In this paper, an H-DQN algorithm for path planning of UH in complex low airspace environments is proposed. Numerical modeling and analysis of UH's low airspace raid mission environment is carried out. On this basis, the corresponding state, behaviour space, and comprehensive reward function of the task model are given. In order to improve the sparse reward problem of traditional reinforcement learning algorithms, a heuristic reward function is designed to guide the algorithm to converge quickly. The introduction of a smooth reward function constrains the behaviour of UH and makes the planned path smoother. The simulation experiment analyses the influence of the weight of each part of the reward function on the performance of the algorithm and compares it with the traditional path planning algorithm. The experimental results show that the proposed H-DQN algorithm has better performance and faster convergence speed, which can help UH successfully complete the raid task. In the next step, we consider combining the comprehensive reward function with more intelligent algorithms to verify its effectiveness in different experimental backgrounds.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant No. 62071483 and Grant No. 61602505).

References

[1] Z. Fang, J. Wang, Y. Ren, Z. Han, H. V. Poor, and L. Hanzo, "Age of information in energy harvesting aided massive multiple access networks," *IEEE Journal on Selected Areas in Communications*, vol. 40, no. 5, pp. 1441–1456, 2022.

[2] T. Do-Duy, L. D. Nguyen, T. Q. Duong, S. R. Khosravirad, and H. Claussen, "Joint optimisation of real-time deployment and resource allocation for UAV-aided disaster emergency communications," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 11, pp. 3411–3424, 2021.

[3] Z. Qadir, F. Ullah, H. S. Munawar, and F. al-Turjman, "Addressing disasters in smart cities through UAVs path planning and 5G communications: a systematic review," *Computer Communications*, vol. 168, pp. 114–135, 2021.

[4] J. Chen, C. Du, Y. Zhang, P. Han, and W. Wei, "A clustering-based coverage path planning method for autonomous heterogeneous UAVs,," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[5] S. Zhang, Y. Zhou, Z. Li, and W. Pan, "Grey wolf optimizer for unmanned combat aerial vehicle path planning," *Advances in Engineering Software*, vol. 99, pp. 121–136, 2016.

[6] F. Ge, K. Li, Y. Han, W. Xu, and Y. Wang, "Path planning of UAV for oilfield inspections in a three-dimensional dynamic environment with moving obstacles based on an improved pigeon-inspired optimization algorithm," *Applied Intelligence*, vol. 50, no. 9, pp. 2800–2817, 2020.

[7] X. Zhang and H. Duan, "An improved constrained differential evolution algorithm for unmanned aerial vehicle global route planning," *Applied Soft Computing*, vol. 26, pp. 270–284, 2015.

[8] X. Yu, C. Li, and J. F. Zhou, "A constrained differential evolution algorithm to solve UAV path planning in disaster scenarios," *Knowledge-Based Systems*, vol. 204, p. 106209, 2020.

[9] J. Chen, Y. Zhang, L. Wu, T. You, and X. Ning, "An adaptive clustering-based algorithm for automatic path planning of heterogeneous UAVs," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2021.

[10] Z. Zhang, J. Wu, J. Dai, and C. He, "Rapid penetration path planning method for stealth UAV in complex environment with BB threats," *International Journal of Aerospace Engineering*, vol. 2020, Article ID 8896357, 15 pages, 2020.

[11] N. Wang, X. Jin, and M. J. Er, "A multilayer path planner for a USV under complex marine environments," *Ocean Engineering*, vol. 184, no. JUL.15, pp. 1–10, 2019.

[12] T. Phanthong, T. Maki, T. Ura, T. Sakamaki, and P. Aiyarak, "Application of A∗ algorithm for real-time path re-planning of an unmanned surface vehicle avoiding underwater obstacles," *Journal of Marine Science and Application*, vol. 13, no. 1, pp. 105–116, 2014.

[13] A. Ammar, H. Bennaceur, I. Châari, A. Koubâa, and M. Alajlan, "Relaxed Dijkstra and A∗ with linear complexity for robot path planning problems in large-scale grid environments," *Soft Computing*, vol. 20, no. 10, pp. 4149–4171, 2016.

[14] T. Whitaker, S. J. Cunningham, and C. Bobda, "Decentralised indoor smart camera mapping and hierarchical navigation for autonomous ground vehicles," *IET Computer Vision*, vol. 14, no. 7, pp. 462–470, 2020.

[15] H. Shorakaei, M. Vahdani, B. Imani, and A. Gholami, "Optimal cooperative path planning of unmanned aerial vehicles by a parallel genetic algorithm," *Robotica*, vol. 34, no. 4, pp. 823–836, 2016.

[16] Y. X. Wang, Y. Y. Tian, X. Li, and L. H. Li, "Self-adaptive dynamic window approach in dense obstacles," *Control and Decision*, vol. 34, no. 5, pp. 927–936, 2019.

[17] J. Chen, F. Ling, Y. Zhang, T. You, Y. Liu, and X. du, "Coverage path planning of heterogeneous unmanned aerial vehicles based on ant colony system," *Swarm and Evolutionary Computation*, vol. 69, article 101005, 2022.

[18] J. J. Shin and H. Bang, "UAV path planning under dynamic threats using an improved PSO algorithm," *International Journal of Aerospace Engineering*, vol. 2020, Article ID 8820284, 17 pages, 2020.

[19] Z. Fang, J. Wang, J. Du, X. Hou, Y. Ren, and Z. Han, "Stochastic optimization aided energy-efficient information collection in internet of underwater things networks," *IEEE Internet of Things Journal*, vol. 9, 2021.

[20] H. Sang, Y. You, X. Sun, Y. Zhou, and F. Liu, "The hybrid path planning algorithm based on improved A∗ and artificial potential field for unmanned surface vehicle formations," *Ocean Engineering*, vol. 223, no. 3–4, p. 108709, 2021.

[21] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[22] K. Tanakitkorn, P. A. Wilson, S. R. Turnock, and A. B. Phillips, "Grid-based GA path planning with improved cost function for an over-actuated hover-capable AUV," in *2014 IEEE/OES Autonomous Underwater Vehicles (AUV)*, pp. 1–8, Oxford, MS, USA, 2014.

[23] H. Kim, S. H. Kim, M. Jeon, J. H. Kim, S. Song, and K. J. Paik, "A study on path optimization method of an unmanned surface vehicle under environmental loads using genetic algorithm," *Ocean Engineering*, vol. 142, no. sep. 15, pp. 616–624, 2017.

[24] T. W. Zhang, G. H. Xu, X. S. Zhan, and T. Han, "A new hybrid algorithm for path planning of mobile robot," *The Journal of Supercomputing*, vol. 2, 2021.

[25] M. Sun, X. Xu, X. Qin, and P. Zhang, "AoI-energy-aware UAV-assisted data collection for IoT networks: a deep reinforcement learning method," *IEEE Internet of Things Journal*, vol. 8, no. 24, pp. 17275–17289, 2021.

[26] L. Chang, L. Shan, C. Jiang, and Y. Dai, "Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment," *Autonomous Robots*, vol. 45, no. 1, pp. 51–76, 2021.

[27] C. Chen, X. Q. Chen, F. Ma, X. J. Zeng, and J. Wang, "A knowledge-free path planning approach for smart ships based on reinforcement learning," *Ocean Engineering*, vol. 189, pp. 106299–106299, 2019.

[28] G. Han, A. Gong, H. Wang, M. Martinez-Garcia, and Y. Peng, "Multi-AUV collaborative data collection algorithm based on Q-learning in underwater acoustic sensor networks," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 9, pp. 9294–9305, 2021.

[29] M. Volodymyr, K. Koray, S. David et al. , "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] X. Wu, H. Chen, C. Chen et al., "The autonomous navigation and obstacle avoidance for USVs with ANOA deep reinforcement learning method," *Knowledge-Based Systems*, vol. 196, p. 105201, 2020.

[31] Y. Zhao, X. Qi, Y. Ma, Z. Li, and M. A. Sotelo, "Path following optimization for an underactuated USV using smoothly-convergent deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–13, 2020.

[32] J. Li, Y. Chen, X. N. Zhao, and J. Huang, "An improved DQN path planning algorithm," *The Journal of Supercomputing*, vol. 78, pp. 1–24, 2021.

[33] J. Wang, Z. Wu, S. Yan, M. Tan, and J. Yu, "Real-time path planning and following of a gliding robotic dolphin within a hierarchical framework," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 4, pp. 3243–3255, 2021.

[34] J. Tsitsiklis and B. Van Roy, *An Analysis of Temporal-Difference Learning with Function Approximation (Technical Report LIDS-P-2322)*, Laboratory for Information and Decision Systems, 1996.

[35] J. Chung, "Playing Atari with deep reinforcement learning," *Computer Ence*, vol. 21, pp. 351–362, 2013.

[36] J. Li, Y. Chen, X. N. Zhao, and J. Huang, "An improved DQN path planning algorithm," *The Journal of Supercomputing*, vol. 78, no. 1, pp. 616–639, 2022.

[37] B. Wang, S. Li, X. Gao, and T. Xie, "UAV swarm confrontation using hierarchical multiagent reinforcement learning," *International Journal of Aerospace Engineering*, vol. 2021, Article ID 3360116, 12 pages, 2021.

[38] M. Riedmiller, R. Hafner, T. Lampe et al., "Learning by playing solving sparse reward tasks from scratch," *International conference on machine learning*, 2018, pp. 4344–4353, Stockholm, Sweden, 2018.

[39] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, 2018.