Article

Multiple Unmanned Aerial Vehicle Autonomous Path Planning Algorithm Based on Whale-Inspired Deep Q-Network

Wenshan Wang, Guoyin Zhang, Qingan Da, Dan Lu, Yingnan Zhao *, Sizhao Li and Dapeng Lang

College of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China; wangwenshan@hrbeu.edu.cn (W.W.); zhangguoyin@hrbeu.edu.cn (G.Z.); da_qing_an@hrbeu.edu.cn (Q.D.); ludan@hrbeu.edu.cn (D.L.); sizhao.li@hrbeu.edu.cn (S.L.); langdapeng@hrbeu.edu.cn (D.L.)
* Correspondence: zhaoyingnan@hrbeu.edu.cn

Abstract: In emergency rescue missions, rescue teams can use UAVs and efficient path planning strategies to provide flexible rescue services for trapped people, which can improve rescue efficiency and reduce personnel risks. However, since the task environment of UAVs is usually complex, uncertain, and communication-limited, traditional path planning methods may not be able to meet practical needs. In this paper, we introduce a whale optimization algorithm into a deep Q-network and propose a path planning algorithm based on a whale-inspired deep Q-network, which enables UAVs to search for targets faster and safer in uncertain and complex environments. In particular, we first transform the UAV path planning problem into a Markov decision process. Then, we design a comprehensive reward function considering the three factors of path length, obstacle avoidance, and energy consumption. Next, we use the main framework of the deep Q-network to approximate the Q-value function by training a deep neural network. During the training phase, the whale optimization algorithm is introduced for path exploration to generate a richer action decision experience. Finally, experiments show that the proposed algorithm can enable the UAV to autonomously plan a collision-free feasible path in an uncertain environment. And compared with classic reinforcement learning algorithms, the proposed algorithm has a better performance in learning efficiency, path planning success rate, and path length.

Keywords: deep reinforcement learning; deep Q-network; whale optimization algorithm; multi-UAV; path planning

check for updates

Citation: Wang, W.; Zhang, G.; Da, Q.; Lu, D.; Zhao, Y.; Li, S.; Lang, D. Multiple Unmanned Aerial Vehicle Autonomous Path Planning Algorithm Based on Whale-Inspired Deep Q-Network . *Drones* **2023**, *7*, 572. https://doi.org/10.3390/drones7090572

Academic Editors: Kaiyu Qin, Haitao Nie, Yikang Yang, Xue Li, Jinliang Shao, Lei Shi and Mengji Shi

Received: 28 July 2023
Revised: 30 August 2023
Accepted: 6 September 2023
Published: 8 September 2023

1. Introduction

Due to the continuous development of multi-sensor data fusion technology and autonomous flight control technology, Unmanned Aerial Vehicles (UAVs) have become an important tool in the field of emergency rescue. Particularly, in the face of natural disasters and emergencies, UAVs are widely used in search and rescue operations [1], for example, the 9.0-magnitude earthquake in Japan in 2011 [2], the 8.1-magnitude earthquake in Nepal in 2015 [3], and Hurricane Harvey in Texas in 2017 [4], which caused massive damage and casualties. UAVs were used to assist rescue teams to provide necessary rescue services for trapped people in danger. However, due to the lack of effective path planning, some UAVs collided during the flight, resulting in mission delays and resource waste, and even secondary disasters, which posed additional risks to both rescue workers and trapped people. Therefore, effective and collision-free path planning is critical for the successful rescue mission of UAVs. A better path planning strategy can avoid collisions between UAVs or even with buildings or other obstacles, improve rescue efficiency, and ensure the smooth progress of rescue operations [5].

The complexity of the rescue environment is the primary challenge of the path planning problem [6]. In general, obstacles (e.g., buildings, bridges, trees, gravel) in the disaster area are relatively messy, and the geographical environment is complex. Planning a collision-free flight path for UAVs has practical significance in assisting rescue missions. At present, some studies [7–10] (e.g., the A star algorithm, Dijkstra's algorithm, and the Bellman–Ford algorithm) have been used in collision-free path planning algorithms. These algorithms evaluate the quality of the path with certain rules and heuristic functions on the existing map and perform path planning by searching and optimizing to find the best path. However, some disasters may change the inherent topography of the affected area and damage the structure of buildings and infrastructure, which will cause unexpected changes in the task area.

For this reason, it is more practical to study the path planning problem in an uncertain task environment. In a disaster environment, the communications and navigation infrastructure may not be able to function properly, requiring UAVs to perform tasks autonomously without a GPS signal or effective communication with the ground command centers [11], which will cause traditional path planning algorithms to fail to obtain the latest maps and environmental information. In addition, unforeseen changes may occur in disaster environments, such as road damage, obstacles, and personnel evacuations. These changes will put forward higher requirements for path planning and navigation algorithms that can adapt to environmental changes in real time and quickly re-plan paths to meet these challenges. Traditional algorithms are usually designed based on specific maps and road network structures. When the map changes, the algorithm may not be able to adapt to the new environment. In order to meet these challenges, it is necessary to adopt more flexible and real-time path planning algorithms, such as the path planning algorithm based on reinforcement learning. The path planning algorithm based on reinforcement learning has the advantages of adaptability, real-time planning, and a learning ability. Researchers try to introduce the reinforcement learning (RL) model into the path planning algorithm [12–14] and learn flexible and feasible strategies through the continuous trial and error of the agent in unknown environments.

However, training RL models requires a large amount of training data, training time, and computing resources [15]. In addition, as the complexity of the task environment increases, RL-based path planning algorithms face challenges when dealing with complex environmental constraints (such as avoiding dangerous areas or other agents, energy constraints, and time constraints). It is necessary to design appropriate state representation and action selection strategies. The heuristic algorithm can provide prior knowledge for the reinforcement learning model by considering and dealing with these complex constraints using certain rules and heuristic functions. For example, the genetic algorithm [16] uses operations such as selection, crossover, and mutation; the particle swarm algorithm [17] updates the velocity and position of particles; and the bee colony algorithm [18] uses a local search and information exchange. These algorithms then combine fitness functions and constraints to generate UAV path solutions that adapt to the environment. These heuristic algorithms have advantages in terms of computational efficiency, data requirements, and solution speed, and they can effectively reduce path redundancy and unnecessary trial and error.

To this end, we propose a multi-UAV path planning algorithm based on a whale-inspired deep Q-network (WDQN). Specifically, we use the deep Q-network (DQN) as the framework and design a comprehensive reward function to balance the three factors of path length, obstacle avoidance, and energy consumption. Then, a deep neural network is used to approximate the Q-value function, and the comprehensive reward function is used to evaluate and reward the planned path in real time during training. In addition, the WoA is introduced when training the DQN, and the WoA balances exploration and exploitation to help the DQN model explore a wider range of actions and states, enabling it to discover new and possibly better strategies. Moreover, the search strategy is dynamically adjusted according to the fitness of the solution to help the DQN model to focus the search on promising areas of the state–action space, thereby improving the convergence speed and learning efficiency. Simulation experiments show that the proposed algorithm can enable UAVs to explore unknown and complex environments and plan collision-free feasible paths.

Finally, compared with the classic RL algorithm, the WDQN has a better performance in learning efficiency, path planning success rate, and path length.

The main contributions of this paper are as follows:

- We combine a WoA and DQN to propose a path planning algorithm for UAV-assisted disaster rescue. The introduced WoA significantly improves the learning efficiency of the DQN, and the designed comprehensive reward function effectively improves the sparse reward problem of the RL algorithm in complex environments.
- We train the model with increasing scene complexity to improve the robustness and generalization of the algorithm. During the training phase, we gradually increase the number of obstacles in the task scene, so that the UAV can gradually adapt to the complex environment and learn a better pathfinding strategy.
- Compared with the global path planning algorithm, this algorithm is suitable for disaster areas with limited communication and has good scalability; that is, it can apply the experience learned in the training process to unknown environments. In addition, the comparative experiments with the classical RL algorithms show that the WDQN has a better training efficiency and path planning ability.

The rest of the paper is organized as follows. In Section 2, we introduce research work related to this paper. The construction of a disaster scenario and the formulation of the rescue problem are described in Section 3. In Section 4, the WDQN-based multi-UAV path planning algorithm proposed in this paper is described in detail. In Section 5, the relevant parameters of the algorithm are introduced, and experiments are carried out to validate the proposed algorithm. The full text is summarized in Section 5.

2. Related Works

Multi-UAV reliable communication and path planning in emergency rescue scenarios are two current research hotspots [19–21]. On the one hand, the development of UAV communication security is constantly advancing in the direction of encrypted communication technology [22], anti-jamming technology [23], and anti-attack technology [24]. On the other hand, the development of path planning for UAVs is shifting from methods based on known maps to exploring unknown environments. In this paper, we assume that the communication environment of multi-UAVs is reliable and focus on the collision-free path planning algorithm in emergency scenarios. The following reviews the existing path planning algorithms from two aspects, namely global path planning based on a fully known environment and local path planning based on an uncertain environment [25].

2.1. Global Path Planning

Global path planning based on a completely known environment is a static planning algorithm. Global path planning is to calculate the optimal path from the starting point to the end point through an algorithm based on the known environmental map. Common global path planning algorithms include Dijkstra's algorithm, the A* algorithm, the sampling-based Rapidly Exploring Random Tree Star (RRT*) algorithm, etc.

Dijkstra's algorithm is widely used in solving the shortest path problem in weighted graphs. Dijkstra's algorithm is an efficient choice, especially when the target is uncertain. However, its time complexity and space complexity are relatively high, which is not suitable for complex environments. In order to avoid the sequential bottleneck problem that may occur in Dijkstra's algorithm, Wang et al. [26] first used the extended hierarchical graph to model the dynamic grid environment, then used the matrix alignment method to perform parallel exploration in the simulated environment, and finally traced the safe path according to the navigator matrix. This method shows a good performance in a large-scale dynamic grid environment. The A* algorithm is currently one of the most widely used graph traversal and path exploration techniques, which is an extension of the Dijkstra algorithm. The traditional A* algorithm has the advantages of simple principles and a fast calculation speed. But as the map grows larger, its computational load will increase exponentially, resulting in a high memory usage, long computing time, and low exploration efficiency [27]. To this end, Guilherme et al. [28] proposed an improved A* algorithm for UAVs for target search and rescue, which introduced a truncation mechanism to prevent UAVs from falling into the local optimum. The RTT* algorithm has great advantages in search efficiency and search quality, and it has been successfully applied in the field of unmanned driving and UAV navigation. Since the classic RTT* algorithm adopts a fixed step size expansion strategy, setting the step size too large or too small will directly affect the search results. To this end, Wang et al. [29] designed a dynamic step size strategy. When the underground intelligent vehicle is far away from obstacles, a larger step size is used to speed up the convergence speed; when it is close to obstacles, a smaller step size is used to ensure the safety of the path. Hu et al. [30] first used the bidirectional extending random tree search strategy to plan the static path and then pruned the random tree according to the change in environmental information, so as to reduce the calculation amount of the RRT* algorithm in uniform random sampling. Compared with the classic RRT algorithm, the improved algorithms converge faster and have higher accuracy in path planning. But it is easy for them to fall into local optimum, and the energy of the system is relatively large. Therefore, intelligent algorithms, such as PSO and the WOA, are often used to improve efficiency and quality. Dong et al. [31] used the Adaptive Whale Algorithm (AWOA) to optimize the deployment of UAVs, used the variable length population strategy to find the optimal number of stopping points, and introduced nonlinear parameters and partial mutation rules to balance exploration and exploitation, so as to minimize the energy consumption of UAVs. Yu et al. [32] improved the PSO algorithm through the simulated annealing algorithm, and proved by experiments that the algorithm can plan high-quality paths for UAVs.

The performance of the global path planning algorithm is highly dependent on the predicted environment information. When the task environment of UAVs undergoes unpredictable changes, the paths planned by these methods will no longer be reliable.

2.2. Local Path Planning

Local path planning based on uncertain environments belongs to the dynamic programming algorithm. When the environmental information is completely unknown or partially known, this type of algorithm uses the local information obtained by UAVs to plan a collision-free path. Classical local path planning usually introduces technologies such as the dynamic window method (DWA), the artificial potential field method, and reinforcement learning. These algorithms have a good robustness to environmental errors and noise.

Traditional DWA algorithms prefer to bypass the periphery of the dense obstacle area, which increases the total distance. Additionally, the objective cost function fails when a "C"-shaped obstacle is encountered, resulting in no path [33] being found. Tan et al. [34] introduced the concept of the obstacle search angle to the traditional DWA. When the static obstacle is not within a certain angle range of the agent's forward direction, the impact on the agent's subsequent navigation is not considered, which improves the adaptability of obstacle avoidance in different scenarios. The artificial potential field method simulates the repulsive field generated by obstacles and the gravitational field generated by the end point, so that the robot can avoid obstacles in the task environment and move forward to the end point. This method is relatively simple in theory and operation, but there are also problems, such as unreachable goals and the way it is easy to fall into the local optimum [35]. To this end, Sun et al. [36] introduced the gravity barrier function in the repulsion function and further restricted the flight boundary and flight speed of the UAV, so as to avoid falling into the local optimum and improve the safety and stability of the track route. However, the ability to deal with unknown task environments using methods based on the artificial potential field method is limited.

In recent years, methods based on reinforcement learning have been widely used. The Q-learning algorithm is one of the commonly used reinforcement learning methods, in which UAVs can learn the optimal action strategy by interacting with the environment. For example, Souto et al. [37] used Q-learning as a reinforcement learning technique to control the UAV to move to the target, aiming to reduce its energy consumption in the disaster area rescue process. However, the Q-learning algorithm faces challenges when dealing with continuous state and action spaces. To solve this problem, deep reinforcement learning (DRL) is introduced into local path planning tasks. DRL not only has the powerful perception and representation ability of a deep neural network when processing high-dimensional decision-making information but also inherits the learning mechanism of reinforcement learning in the interaction with the environment, which can realize real-time path planning for agents [38]. Among them, the DQN is one of the most classic methods. Zhang et al. [39] transformed the trajectory optimization problem into a constrained Markov decision process with a UAV as an agent and proposed a trajectory planning method based on the deep Q-network. Although the DQN is effective in obstacle avoidance and path planning, the algorithm has the problems of large randomness in action selection and slow convergence during training. To this end, Huang et al. [40] proposed the Dueling DQN algorithm based on the tree sampling mechanism. In addition, Wang et al. [41] proposed a DDQN based on a greedy strategy to solve the problem of mountain rescue in complex environments.

These algorithms enable UAVs to perform tasks in uncertain environments. Due to the lack of a grasp on global information, the planned path may not be optimal or even feasible. In addition, the path planning problem in an unknown environment needs to consider multiple possible states and transition probabilities, which will lead to a high computational complexity of the algorithm.

3. System Model

In order to simulate the real flight environment, a disaster search and rescue environment model is established first, which is the basis of UAV path planning. In this disaster search and rescue environment, the operating environment of UAVs is considered a two-dimensional nonlinear horizontal plane. The scientific issues considered in this paper are described as follows. In a complex and unknown disaster environment, multiple UAVs autonomously avoid unknown environmental obstacles during high-speed movement, search, approach, and find trapped people in the shortest possible time, aiming to provide reliable action routes and more time for rescue teams. Each UAV is considered a particle moving in two dimensions, independently making new decisions at all times to find its target. As shown in Figure 1, we model the entire auxiliary rescue mission from three aspects: environment information processing, value function acquisition, and action selection.

3.1. Environment Information Processing

The environmental information processing module converts the current environmental information of the UAV into a state representation. In the search process, many objects can be regarded as obstacles, such as trees, houses, mountains, etc. For the convenience of discussion, Figure 1a is a simplified environment model of a multi-UAV auxiliary rescue mission. We first divide the entire rectangular task area $\Re = H_x \times H_y$ into M cells and denote the position of the i-th cell as $\left(h_x^i, h_y^i\right)$, where $1 < i < M$. Let N denote the number of UAVs participating in the disaster search and rescue mission, which are distributed in the entire mission area. We use $\left(p_{uxj}^t, p_{uyj}^t\right) \in \Re$ to represent the position of UAV_j at time t, where $1 < j < N$. In the case of a fixed scanning angle, the flight altitude of the UAV is directly proportional to the coverage radius of the scanning area. Therefore, for each cell, the UAV's flight altitude can be adjusted based on a determined coverage radius to achieve accurate coverage of each cell. In other words, the UAV_j flight altitude al_j can be calculated based on the fixed scanning angle θ and coverage radius cr_j using the formula $al_j = cr_j \Big/ \tan\left(\frac{\theta}{2}\right)$. O is the set of obstacles in the environment, which are randomly distributed in the task area. Let $\left(p_{oxk}, p_{oyk}\right) \in \Re$ denote the position of obstacle k. Considering that the people on the ground will be trapped in the ruins when the house collapses, the movement of the trapped people is often limited by the space environment, and it is assumed that the position of the trapped people does not change with time. The coordinates of the trapped person $j(1 < j < N)$ can be denoted by $\left(p_{txj}, p_{tyj}\right)$.

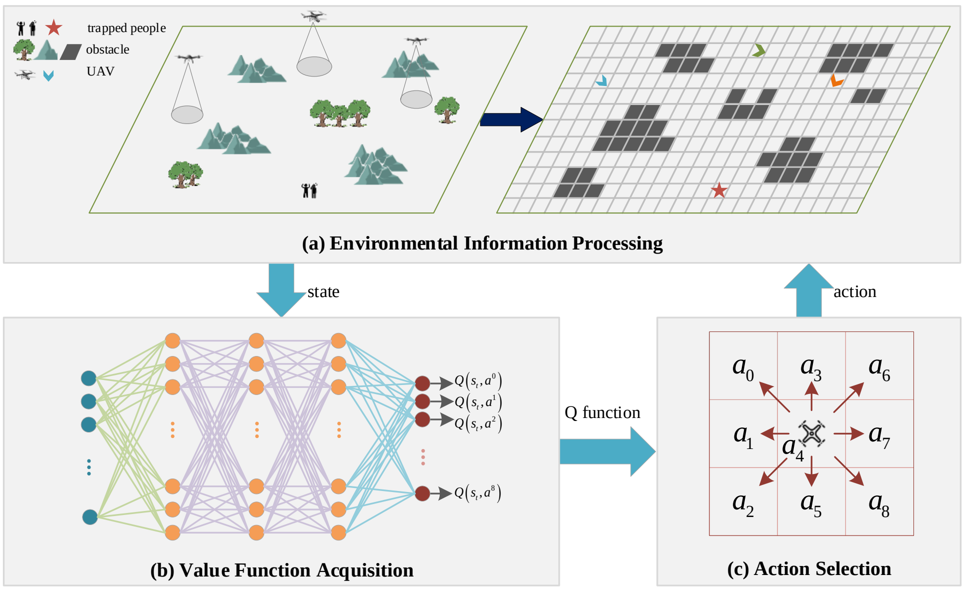


Figure 1. The schematic diagram of multi-UAV collaborative path planning based on reinforcement learning.

3.2. Value Function Acquisition

The purpose of the value function acquisition module is to train a path planning policy network for UAVs through learning algorithms. In this module, we build a neural network to represent the Q function, which consists of an input layer, multiple hidden layers, and an output layer. The input layer receives the current state representation as input, which represents the currently known information about the environment and the state of each UAV. The hidden layer maps the input to a higher-level feature representation through the calculation of multiple layers of neurons and the processing of activation functions. The output layer gives the Q-value of each possible action and selects the optimal action to guide the UAV's behavioral decision making. The details will be introduced in Section 4.

3.3. Action Selection

The UAV has a fixed field of view (FOV) and can randomly select any surrounding cells to move during flight. As shown in Figure 1c, the movement of the UAV mainly includes the following nine actions: east, south, west, north, southeast, southwest, northeast, northwest, and hovering. The action selection module first selects the optimal action according to the action value function and then transmits it to each UAV in the environment in the form of coordinates.

4. Method

In order to reduce the frequency of blind trial and error for UAVs in the search process, we design a whale-inspired deep Q-network. A WOA is used to accumulate prior knowledge about action selection. In this section, we first briefly introduce the basic principles of the WOA and DQN then introduce the proposed algorithm in detail.

4.1. Action Decision Based on WoA

The WoA is an optimization algorithm inspired by the foraging behavior of whales in nature. Each UAV is considered a whale, and there are three hunting models for each whale, which are the encircling model, the searching model, and the bubble-net attacking model.

4.1.1. Encircling Model

At this time, the whale chooses to swim towards the optimal individual, and the position update formula of the whale is as follows.

$$X_i(t+1) = X_{best}(t) - A \cdot \|C \cdot X_{best}(t) - X_i(t)\|, \tag{1}$$

$$\begin{cases} A = 2a \cdot \beta_1 - a \\ C = 2\beta_2 \\ a = 2 - t/T \end{cases} \tag{2}$$

where t represents the current number of iterations, X_{best}. represents the whale currently in the optimal position, β_1 and β_2 represent random variables in the range of $(0,1)$, respectively, and T represents the maximum number of iterations. The initial value of a is set to 2; it will decrease to 0 as the number of iterations increases.

4.1.2. Searching Model

At this time, the whale randomly selects an individual to approach, and the position update formula of the whale is as follows.

$$X_i(t+1) = X_{rand}(t) - A \cdot \|C \cdot X_{rand}(t) - X_i(t)\|, \tag{3}$$

where X_{rand} is the position of the whale randomly selected in the current group.

4.1.3. Bubble-Net Attacking Model

Whales also need to constantly adjust their position when using bubble nets to drive away prey. In this case, the position update formula of the whale is as follows.

$$X_i(t+1) = \|X_{best}(t) - X_i\| \cdot e^{bl} \cdot \cos(2\pi l) + X_{best}(t) \tag{4}$$

In the formula, b is a constant, and the default value is 1. l is a random number uniformly distributed in the range $(0,1)$.

4.2. Path Exploration Based on DQN

RL is a machine learning method. The agent obtains feedback and rewards from the environment by constantly trying different actions and gradually forms an optimal strategy. The interaction process between the agent and the environment can be represented by the quaternion (S, A, R, P, γ) of the Markov decision process (MDP), where S represents the state space of the agent, A represents the action space of the agent, $R = S \times A$ represents the immediate reward obtained by the agent after taking an action in the current state, $P : S \times A \times S \to [0,1]$ represents the transition probability of the agent from the current state to the next state after taking an action, and γ is a discount factor.

The core idea of the DQN is to use a deep neural network (DNN) to approximate the Q-value function used to evaluate the value of taking a specific action in a given state, thereby maximizing the discounted cumulative reward R in the reinforcement learning environment.

$$R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i) \tag{5}$$

where $\gamma \in [0,1]$ represents the discount factor and r represents the instant reward obtained at time t.

The DQN uses the basic idea of a Q-learning algorithm, which is to guide the agent to take the best action in a given state by learning a value function $Q(s,a)$. Unlike traditional Q-learning, the DQN uses a DNN to approximate the optimal value function. Specifically, a Q-network takes a state s as input and outputs a corresponding Q-value a to each action.

By optimizing the parameters of the Q-network, it can accurately predict the cumulative reward of performing each action in a given state; that is,

$$Q(s,a;\theta) \approx Q(s,a) \tag{6}$$

And the Q-function is shown in Equation (7).

$$Q_\pi(s,a) = \mathrm{E}_\pi[R_t | S_t = s, A_t = a] \tag{7}$$

where θ represents the weight of the Q-network. The optimal Q-function is shown in Equation (8).

$$Q^*_\pi(s,a) = \max_\pi Q_\pi(s,a) = \max_\pi \mathrm{E}_\pi[R_t | S_t = s, A_t = a] \tag{8}$$

The function Q^* returns the maximum reward of all strategies, which can be expressed by the Bellman equation as follows.

$$Q^*_\pi(s,a) = \mathrm{E}[y_t], \tag{9}$$

$$y_t = r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta'\right), \tag{10}$$

where s_{t+1} and a_{t+1} represent the subsequent state and action in the next time step, respectively. θ' represents the weight of the target Q-network, and y_t represents the target value.

In the DQN, the parameters θ of the Q-network are updated by optimizing the loss function, that is, minimizing the mean square error between the target Q-value and the current Q-value. The formula of the loss function is as follows.

$$Loss(\theta) = \mathrm{E}\left[\left(r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta'\right) - Q(s,a;\theta)\right)^2\right] \tag{11}$$

We choose the gradient descent method to update the parameters of the Q-network to gradually approximate the true Q-value function. The update formula of parameter θ is

$$\theta = \theta - \alpha \cdot \nabla Loss(\theta), \tag{12}$$

$$\nabla Loss(\theta) = \nabla\left(r_t + \gamma \max_{a_{t+1}} Q\left(s_{t+1}, a_{t+1}; \theta'\right) - Q(s,a;\theta)\right)^2 \tag{13}$$

where a represents the learning rate, which is used to control the step size of the parameter update. ∇ represents the gradient operation, and $\nabla Loss(\theta)$ represents the gradient of the loss function with respect to the network parameters. The network parameters are updated iteratively, so that the Q-function gradually approaches the true Q-value function. Finally, we can obtain the appropriate action through the well-trained network.

4.3. The Autonomous Path Planning Strategy Based on WDQN

For the path planning task, we proposes a whale-inspired DQN algorithm. The core idea of the WDQN is to use the WOA to provide the DQN with the decision-making experience required for training, improve the interaction efficiency between agents and the environment, and accelerate the convergence speed of the DQN algorithm. The framework of the WDQN algorithm consists of four parts, as shown in Figure 2. First of all, the input of the proposed algorithm is a two-dimensional vector composed of the real-time position of each UAV and the environment information. Second, we design a comprehensive reward function to generate dynamic rewards in real time based on environment information, which enables UAVs to have a good control performance. Then, each UAV is regarded as a whale, and the corresponding fitness function value is calculated according to the reward obtained during training. The action decision output by the WOA is added to the replay buffer of the Q-network for training, so that the model can obtain more valuable experience in the early stage of training. Finally, we increase the complexity of the environment step by step to improve the robustness and generalization of the algorithm.

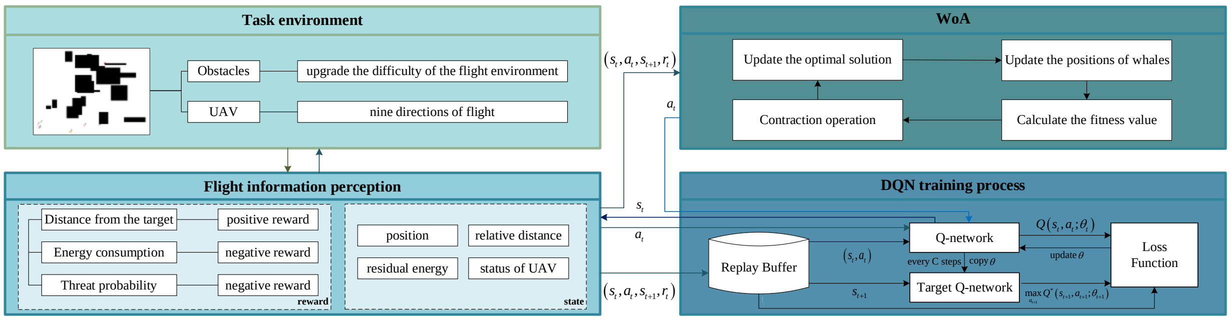


Figure 2. The framework of WDQN algorithm.

4.3.1. Reward Function

The setting of the reward function is crucial to the performance of reinforcement learning algorithms. Choosing an appropriate reward function can effectively promote the convergence of the algorithm, while an inappropriate one may make it difficult for the algorithm to converge [42]. In traditional reinforcement learning algorithms, a learner is rewarded only after completing a task, and there is no reward for the previous series of behaviors. It has been pointed out that this reward mechanism can lead to the sparse reward problem when faced with complex environments [43]. When the set of environmental states is large, the learner encounters a series of nonfeedback states before completing the task. Since effective rewards cannot be obtained in time, the algorithm will be difficult to converge. To solve this problem, we design a comprehensive reward function, as shown in Equation (14).

$$R = R_{target} + R_{avoid} + R_{cost} \tag{14}$$

The reward function is divided into three parts: the target reward function R_{target}, obstacle avoidance reward function R_{avoid}, and energy consumption reward function R_{cost}, which are used to evaluate the directionality, safety, and efficiency of the model, respectively:

(1) The target reward function R_{target} is used to guide the UAV to quickly reach the target position, which is calculated using the following formula.

$$R_{target} = \begin{cases} 1000, & if successful \\ \beta_1 \cdot (d_{origin}/d_{distance}) \cdot D, & else \end{cases} \tag{15}$$

$$d_{origin} = \sqrt{\left(p^0_{uxj} - p_{txj}\right)^2 + \left(p^0_{uyj} - p_{tyj}\right)^2}, \tag{16}$$

$$d^t_{distance} = \sqrt{\left(p^t_{uxj} - p_{txj}\right)^2 + \left(p^t_{uyj} - p_{tyj}\right)^2}, \tag{17}$$

where β_1 represents the weight of the target reward function, d_{origin} represents the initial Euclidean distance between the UAV and the target node, $d^t_{distance}$ represents the Euclidean distance between the UAV and the target node at time t, and $D = d^{t-1}_{distance} - d^t_{distance}$ represents the displacement of the UAV from time step $t-1$ to t.

(2) The obstacle avoidance reward function R_{avoid} is a negative reward used to measure the safe distance between the UAV and nearby obstacles. This function can guide the UAV away from obstacles and ensure the safety of the planned path, which is calculated using the following formula.

$$R_{avoid} = \begin{cases} -1000, & if collision \\ -\beta_2 \cdot \exp(d_{nearobs}), & else \end{cases}, \tag{18}$$

where β_2 represents the weight of the obstacle avoidance reward function and $d_{nearobs}$ represents the distance between the UAV and nearby obstacles. If there are no obstacles around the current position of the UAV, then $d_{nearobs} = 0$; otherwise, calculate the sum of the Euclidean distances to all surrounding obstacles.

(3) To make the UAV tend to plan shorter paths, the R_{cost} is designed as a reward function to measure the remaining energy, which is also a negative reward. It can be seen from the following formula that the faster the search, the higher the reward.

$$R_{cost} = -\beta_3 \cdot \exp\left(B_{cost}/B_{orgina}\right), \tag{19}$$

where β_3 represents the weight of the energy consumption reward function, B_{cost} represents the current energy consumption value, and B_{orgina} represents the initial energy value.

4.3.2. Complexity of Task Scene

In this paper, the reinforcement learning model is trained in the form of the complexity of the task scene. In order to simulate the disaster-affected scene, we assume that the task scene of UAVs contains N_{obs} obstacles (e.g., buildings trees, and vehicles), and use a quaternion (x, y, l, w) to identify the attributes of each obstacle. Among them, x and y represent the abscissa and ordinate of the obstacle center, respectively, and l and w represent the length and width of the obstacle, respectively. Obviously, as the number of obstacles increases, the complexity of the UAVs path planning increases. Therefore, we assume that the complexity L of the task scene is proportional to the number of obstacles N_{obs}, and the initial complexity of the training scene is $L_0 = 1$. Specifically, N_{obs} is a random integer in the interval $[L, 2L]$. In the initial stage of training, we use simple and easily avoidable obstacles to enable the UAV to quickly learn basic path planning techniques. As the training progresses, when the number N_{suc} of UAVs that find the target is greater than 80% of the total number of UAVs N, the complexity L of the scene will be upgraded, meaning that the terrain becomes gradually more complex and the passage becomes narrower. Otherwise, the difficulty remains the same. Namely,

$$L = \begin{cases} L+1, & if N_{suc} > 0.8 \cdot N \\ L, & else \end{cases} \tag{20}$$

4.3.3. WDQN for Multi-UAV Path Planning

In this paper, we use the Adam algorithm to adaptively adjust the learning rate and optimize the model parameters by calculating the first-order moment estimation and second-order moment estimation of the gradient, so as to speed up the convergence speed of the model and improve the robustness and stability of the algorithm. The update formula of the parameters is as follows.

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\overset{\wedge}{m_t}}{\sqrt{\overset{\wedge}{v_t}} + \sigma}, \tag{21}$$

where $\overset{\wedge}{m_t}$ represents the value of the first-order moment estimator m_t after bias correction, $\overset{\wedge}{v_t}$ represents the value of the second-order moment estimator v_t after bias correction, and $\sigma = 10e^{-8}$ is an offset used to prevent division by zero.

$$\overset{\wedge}{m_t} = \frac{m_t}{1 - \mu_1^t}, \tag{22}$$

$$\overset{\wedge}{v_t} = \frac{v_t}{1 - \mu_2^t}, \tag{23}$$

$$m_t = \mu_1 \cdot m_{t-1} + (1 - \mu_1) \cdot \nabla Loss(\theta), \tag{24}$$

$$v_t = \mu_2 \cdot v_{t-1} + (1-\mu_2) \cdot \nabla Loss(\theta), \tag{25}$$

where μ_1 and μ_2 represent the exponential decay rates of the first-order moment estimation and second-order moment estimation, respectively.

In order to guide the UAV to choose actions reasonably, we use the $\varepsilon-greedy$ strategy to balance exploration and utilization, thereby preventing the algorithm from falling into the local optimum. Decreasing ε gradually during the training process makes the Q-network perform more random exploration in the initial stage, which is beneficial to try more actions. As the training progresses, gradually decrease ε to make it more inclined to choose the optimal action for a better performance. The update formula of ε is as follows.

$$\varepsilon = \max\left(\frac{\varepsilon_{\min}-1}{T} \cdot t + 1, \varepsilon_{\min}\right), \tag{26}$$

where $\varepsilon_{\min}$ represents the minimum value of the exploration rate.

In summary, we give a detailed description of the WDQN-based multi-UAV autonomous path planning algorithm (see Algorithm 1).

Algorithm 1 Pseudocode of simulated WDQN

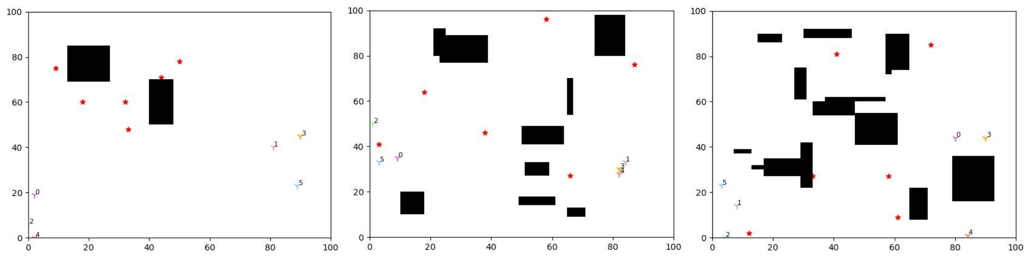
```
Input: Q-network weights $\theta$, maximum iterations $T$, learning rate $\alpha$, number of whales $W$,
    discount factor $\gamma$;
Output: The optimal policy $\pi$;
 1: Initialization the Q network, initialization the positions of UAV and the trapped people,
    initialization the level of obstacles $L$, initialization Q network $Q$ and target Q-network
    $Q'$, initialization greedy probability $\varepsilon$, set $t = 0$;
 2: for $t < T$ do
 3:     while $P_{uavs} != P_{obstacles}$ and $P_{uavs} != P_{people}$ do
 4:         update $\varepsilon$ according to Equation (26)
 5:         $eps = random(0,1)$
 6:         if $eps > 2\varepsilon$ then
 7:             $a_t = \arg\max_a Q(s_t, a_t; \theta)$
 8:         else if $\varepsilon < eps \leq 2\varepsilon$ then
 9:             $a_t$ selected according to WoA
10:         else
11:             $a_t$ selected a random action
12:         end if
13:         execute action $a_t$ and observe reward $r_t$ and new state $s_{t+1}$
14:         store the transition $(s_t, a_t, r_t, s_{t+1})$ into replay buffer
15:         set $s_t = s_{t+1}$
16:         if replay buffer is full then
17:             sample random minibatch of transition $(s_t, a_t, r_t, s_{t+1})$ from replay buffer
18:             set target value $y_t$ according to Equation (10)
19:             calculate the loss $Loss_t$ according to Equation (11)
20:             calculate gradient $\nabla Loss_t$ according to Equation (13)
21:             update the first moment estimator according to Equation (24)
22:             update the second moment estimator according to Equation (25)
23:             calculate bias-corrected first moment estimator according to Equation (22)
24:             calculate bias-corrected second moment estimator according to Equation (23)
25:             update Q-network weights according to Equation (21)
26:             every $C$ steps set $Q' = Q$
27:         end if
28:     end while
29:     calculate level of obstacles $L$ according to Equation (20)
30: end for
```

5. Simulation Results and Analysis

5.1. Simulation Settings

In order to better explore and learn the optimal path planning in different environments, it is necessary to train the WDQN model first. In the training phase, we simulated a 100 m $\times$ 100 m task area, in which we set up six randomly distributed UAVs and six randomly distributed trapped people. In addition, we gradually increased the complexity of the task scene to improve the training performance of the DQN. The initial complexity of the model training is 1, which means that the initial number of obstacles in the task scene is a random integer in the interval $[1, 2]$. The maximum complexity of the model training is 10, and in this case the number of obstacles in the task scene is a random integer in the interval $[10, 20]$.

A whole task environment consists of four parts: background, obstacles, UAVs, and targets. Figure 3 shows three examples of initial task environments with different complexities in a 100 m $\times$ 100 m task area at the test time. It can be seen that the red pentagrams represent targets, the Y-shaped marks of different colors represent UAVs with different numbers, and the black rectangles represent obstacles.



(a) Level 1 (b) Level 5 (c) Level 10

Figure 3. Examples of initial task environments with different complexities.

This paper uses the machine learning library PyTorch to build and train the WDQN. Table 1 describes the main parameters involved in the training phase of the model. At the beginning of each episode of the model training phase, the task environment is randomly initialized, including the positions of entities such as UAVs, targets, and obstacles. Each episode is terminated when the UAV hits an obstacle or runs out of battery or successfully finds the target.

Table 1. Summary of hyperparameters involved in training phase

Symbol	Description	Value
T	maximum episode	10,000
α	learning rate	0.001
γ	discount factor	0.9
$batsize$	batch size	128
$bufsize$	buffer size	100,000
$\varepsilon_{\min}$	the minimum of the exploration rate	0.1
μ_1	exponential decay rate of the first moment estimate	0.9
μ_2	exponential decay rate of the second moment estimate	0.999

The constants β_1, β_2, and β_3 in Equations (15), (18), and (19), respectively, represent the weights of the target reward function, the obstacle avoidance reward function, and the energy consumption reward function. The value range of these parameters is usually between $[0, 1]$, and $\beta_1 + \beta_2 + \beta_3 = 1$, indicating the importance of the corresponding reward function. When $\beta_1 = 0$, the target reward function does not work, and the path planning algorithm will not give priority to reaching the target position but may pay more attention to the optimization of obstacle avoidance and energy consumption. When $\beta_2 = 0$, the obstacle avoidance reward function does not work, and the algorithm is prone to collisions. When $\beta_3 = 0$, the energy consumption reward function will not work, and the algorithm will not give priority to the energy consumption index, which will easily increase the path cost. Therefore, appropriate values of β_1, β_2, and β_3 should be selected to meet the requirements of the task.

The performance of the WDQN algorithm with different reward function weight combinations is shown in Table 2. It shows the data obtained by taking the average value of 50 experiments in the scene of 100×100, with an obstacle complexity level of 7 and the number of targets set to six. Since this paper is applied in the emergency rescue scenario, it is most important to find a path to the trapped people; that is, the value of β_1 should not be too small. When the value of β_2 is too small, the ability of the algorithm to avoid obstacles will be reduced. And when the value of β_3 is too small, the path cost of the algorithm will increase. Therefore, comprehensively considering the three indicators of success rate, collision rate, and path length, we set $\beta_1 = 0.7$, $\beta_2 = 0.2$, and $\beta_3 = 0.1$. In this case, the algorithm achieves the best results.

Table 2. The performance of the WDQN algorithm with different reward function weight combinations.

β_1	β_2	β_3	Success Rate	Collision Rate	Average Path Length
0.5	0.4	0.1	85.1%	3.3%	512.93
0.5	0.3	0.2	83.8%	6.7%	447.21
0.5	0.2	0.3	73.3%	19%	367.33
0.5	0.1	0.4	61.7%	33.7%	301.79
0.6	0.3	0.1	90%	8.7%	331.71
0.6	0.2	0.2	73.3%	16.7%	298.62
0.6	0.1	0.3	73%	23%	256.65
0.7	0.2	0.1	96.8%	3.67%	262.75
0.7	0.1	0.2	93%	14%	260.85

The learning rate is a very important hyperparameter in the deep learning algorithm, which has an important impact on the training performance of the reinforcement learning model. In order to select appropriate parameters, we conducted seven independent experiments to compare the training performance of the WDQN under different initial learning rate settings, as shown in Figure 4. In terms of changes in the cumulative reward curve, when the learning rate is 0.01, the performance of the model is the worst. When the learning rate is 0.001, the training performance is the best. When the learning rate is lower than 0.001, the performance of the model gradually decreases, and the convergence of the cumulative reward function becomes worse. Therefore, we set the learning rate α to 0.001 in subsequent experiments.

5.2. Effectiveness of WDQN Algorithm

In this part, we tested the effectiveness and generalization of the proposed algorithm in three aspects: task scenes with different complexities, task targets with different numbers, and task areas with different sizes.

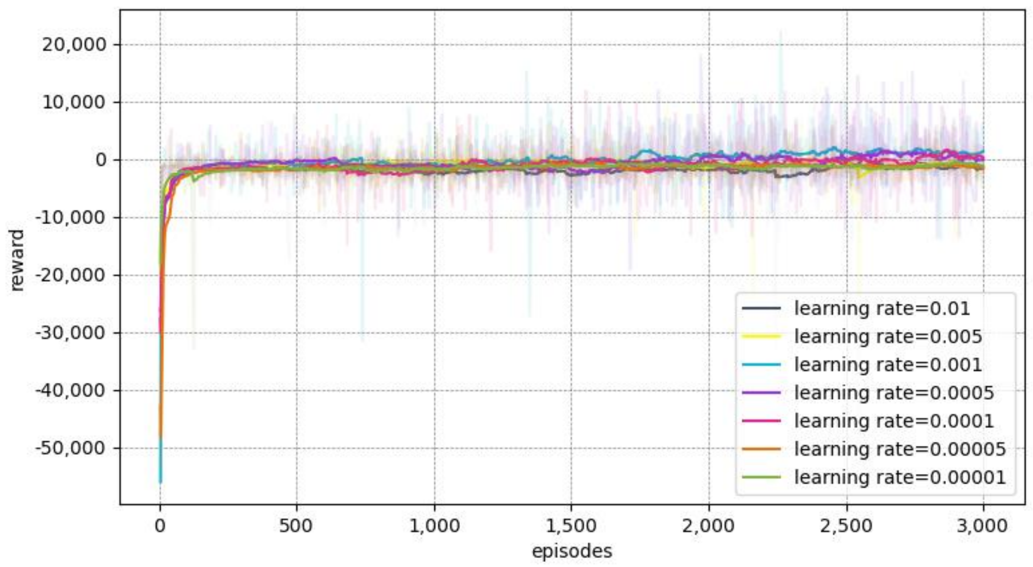
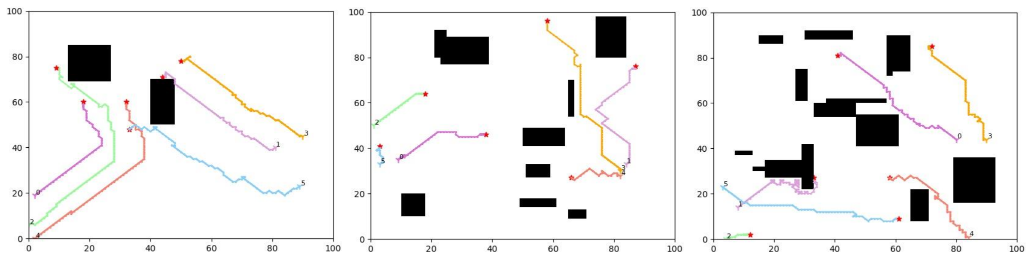


Figure 4. The rewards with different rates.

5.2.1. Experiments with Different Task Scene Complexities

Figure 5 shows the effectiveness tests of the proposed algorithm under three different task scene complexity settings, which are the test results of the corresponding task environments in Figure 3. The size of the test scene is fixed at 100 m $\times$ 100 m. The number of UAVs and targets is six, and they are randomly distributed in the test scene. The black rectangle is used to simulate obstacles. As the complexity of the task scene increases, the proportion of obstacles occupying the entire task area gradually increases. It can be seen from Figure 5 that although the difficulty of obstacle avoidance gradually increases the proposed algorithm can still enable UAVs to successfully avoid obstacles and find these targets. In the figure, the red pentagrams represent targets, the black rectangles represent obstacles, the Y-shaped marks of different colors represent UAVs with different numbers, and the line segments represent the paths taken by the UAVs. Since each UAV can only obtain the local information it is close to during the pathfinding process, it cannot predict the distribution of obstacles in the task environment. The final path of the UAV is usually not the shortest path to the target. This is normal and understandable in local path planning algorithms.



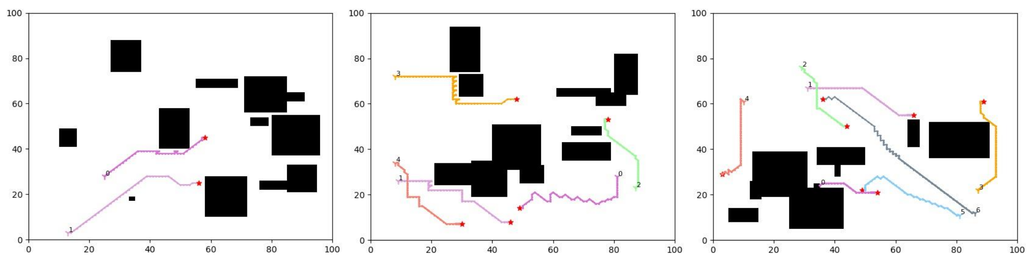
(a) Level 1 (b) Level 5 (c) Level 10

Figure 5. Path planning in three task scene of different complexities.

In order to test the execution time of path planning in task scenes of different complexities, we conducted 30 experiments with different complexity settings and recorded the execution times. The average execution time in level 1 is 225.21 s, the average execution time in level 5 is 126.52 s, and the average execution time in level 10 is 173.33 s. It can be seen that although fewer obstacles can reduce the limitation of the path planning algorithm it may increase the computational complexity and execution time of the algorithm. This is because the algorithm needs to consider more path selections and calculations to find the optimal path. And as the complexity of the task scenario increases, the execution time is normally distributed.

5.2.2. Experiments with Different Numbers of Task Targets

Figure 6 shows the test results of the proposed algorithm under three settings with different numbers of task targets. The size of the test environment is still 100 m × 100 m, and the test scene complexity is fixed at 10. The red pentagrams represent targets, the black rectangles represent obstacles, the Y-shaped marks of different colors represent UAVs with different numbers, and the line segments represent the paths taken by the UAVs. The number of UAVs and targets is gradually increasing, and they are randomly distributed in the mission area. It can be seen that the proposed algorithm has good scalability in the number of task targets. When the number of task targets in the test scene is more or less than that of the training scene, the proposed algorithm can still plan a safe and feasible path for each UAV.



(a) two task targets (b) five task targets (c) seven task targets

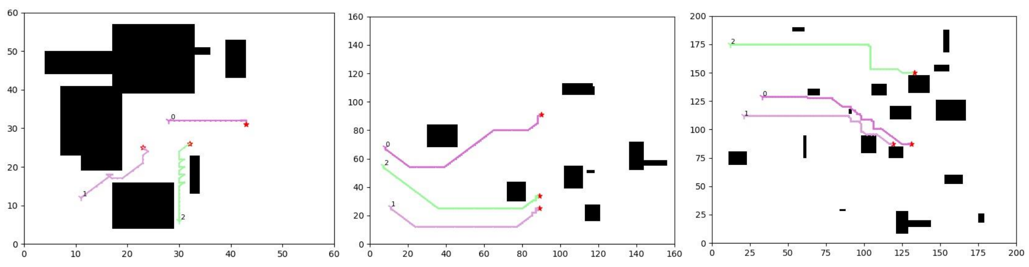
Figure 6. Path planning in different numbers of task targets.

In order to test the execution time of path planning with different numbers of task targets, we conducted 30 experiments with different numbers of task targets and recorded the execution time. The average execution time with two task targets is 15.95 s, the average execution time with five task targets is 71.39 s, and the average execution time with seven task targets is 193.65 s. It can be seen that as the number of task targets increases the execution time also increases.

5.2.3. Experiments with Different Sizes of Task Area

Figure 7 shows the test results of the proposed algorithm in three different sizes of task area. Since it is difficult to accommodate too many obstacles in a small-sized task area, we fixed the test scene complexity to 9, set the number of UAVs and targets to three, and randomly distributed them in the task area. The red pentagrams represent targets, the black rectangles represent obstacles, the Y-shaped marks of different colors represent UAVs with different numbers, and the line segments represent the paths taken by the UAVs. The sizes of the three test scenes are set to 60 m × 60 m, 160 m × 160 m, and 200 m × 200 m, respectively. It can be seen that the proposed algorithm has good scalability in the size of the task area. When the test scene is larger or smaller than the training scene, the proposed algorithm can still plan a safe and feasible path for each UAV.

In order to test the execution time of path planning under different task area sizes, we conducted 30 experiments for different task area sizes and recorded the execution time. The average execution time in the 60 × 60 area is 27.26 s, the average execution time in the 160 × 160 area is 70.73 s, and the average execution time in the 200 × 200 area is 239.31 s. It can be seen that as the size of the task area increases the execution time also increases.



(a) 60m × 60m (b) 160m × 160m (c) 200m × 200m

Figure 7. Path planning under different task area sizes.

5.3. Comparison and Analysis

In order to further evaluate the proposed WDQN-based path planning algorithm, we conducted multiple comparative experiments involving mainstream reinforcement learning models including Q-learning, the DQN, the DDQN, and the Dueling DQN. During the experiments, we focused on four metrics: the reward and loss in the training phase, the success rate, and the average path length in the testing phase.

5.3.1. Reward

The cumulative reward is the feedback signal provided to the agent in the task environment, which is used to guide the agent to learn and adjust its behavior. Rewards can be positive, negative, or zero, representing encouraging, punishing, or neutral evaluations of the agent's actions. In our experiments, the initial reward is set to a sufficiently small value. To measure the cumulative reward, we train these reinforcement learning models under the same experimental settings. The figure below shows the change curves in the cumulative reward obtained by each algorithm during the training phase.

First of all, observing the convergence of each curve in Figure 8, it can be found that there are unstable fluctuations in the curves of Q-learning and the DQN, and the convergence of the DDQN, Dueling DQN, and WDQN is relatively good, which shows that the proposed algorithm is easy to train. Then, observing the level of the reward value, it can be found that when the training tends to be stable (i.e., after 8000 episodes), the average reward obtained by UAVs in Q-learning and the DQN from the task environment is relatively low, while the average reward obtained by UAVs in the Dueling DQN and WDQN is relatively high, indicating that actions taken by UAVs can achieve better scores in path planning tasks. Finally, observing the training reward of the first 2000 episodes, it can be seen that the proposed WDQN can obtain a higher cumulative reward faster. This is because the introduction of the WOA enriches the action decisions in the replay buffer of the DQN and greatly speeds up the learning efficiency of the model. However, the learning progress of several other methods, especially Q-learning and the DQN, is relatively slow, and it is difficult to complete the path planning task in a short period of time. From the analysis of the above three points, it can be seen that the proposed model is superior to other reinforcement learning models in terms of stability, action decision quality, and learning efficiency.

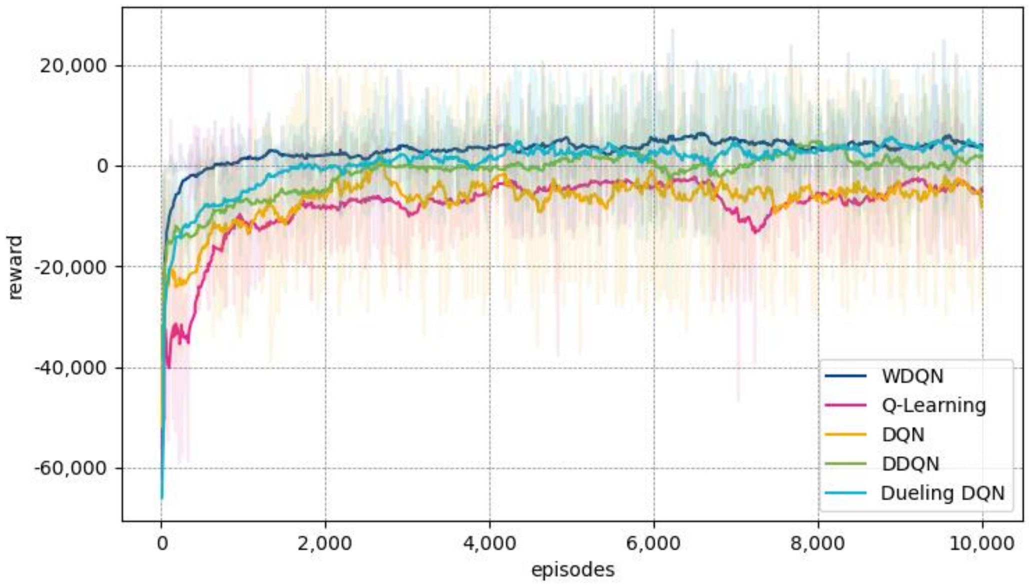


Figure 8. The reward curves of WDQN, Q-learning, DQN, DDQN, and Dueling DQN algorithm.

5.3.2. Loss

In reinforcement learning, the loss is a metric used to measure the discrepancy between the expected future reward and the actual observed reward. The goal of the UAV is to find the optimal path planning strategy by minimizing the loss function, so that the maximum reward can be obtained in the long-term accumulation process. In our experiments, the initial loss is set to a sufficiently large value. To measure the loss, we train these reinforcement learning models under the same experimental settings. The figure below shows the change curve of the loss obtained by each algorithm during the training phase.

First of all, observing the convergence of each curve in Figure 9, it can be found that the loss functions of Q-learning and the DQN have not achieved a satisfactory convergence, and the convergence efficiency of the DDQN, Dueling DQN, and WDQN is relatively high. Then, observing the value of the loss, it can be found that when the training tends to be stable (that is, after 8000 episodes) the loss of UAVs in the Q-learning and DQN algorithms is relatively high, indicating that there is a certain gap between the obtained path planning strategy and the expected planning effect. The loss of UAVs in the WDQN is the lowest, which shows that the proposed algorithm can show a better performance in path planning tasks. Finally, observing the changes in the loss of the first 2000 episodes, it can be seen that the proposed algorithm greatly reduces the loss of UAVs in a short period of time, which is due to the auxiliary effect of the WOA on the DQN model. The loss of other methods, especially Q-learning and the DQN, is maintained at a high level in the early stage, indicating that their early exploration and learning capabilities are relatively poor, and more episodes of training are needed to adapt to the rules of the path planning task. From the analysis of the above three points, it can be seen that the proposed model has more advantages than other reinforcement learning models in terms of stability, path planning quality, and learning efficiency.

5.3.3. Success Rate

The success rate of a path planning algorithm refers to the probability of planning a feasible path in an unknown test environment. To compare the success rate of these five well-trained reinforcement learning models, we conduct 1000 independent tests under three different scene complexity settings. Each test starts with changing the task environment, which means reinitializing the positions of objects, such as UAVs, targets, and obstacles within the 100 m $\times$ 100 m mission area. In this part, the success rate can be obtained by calculating the ratio between the number of feasible paths and the number of tests.

The figure below shows the success rate of five well-trained models for path planning under different scene complexity settings.

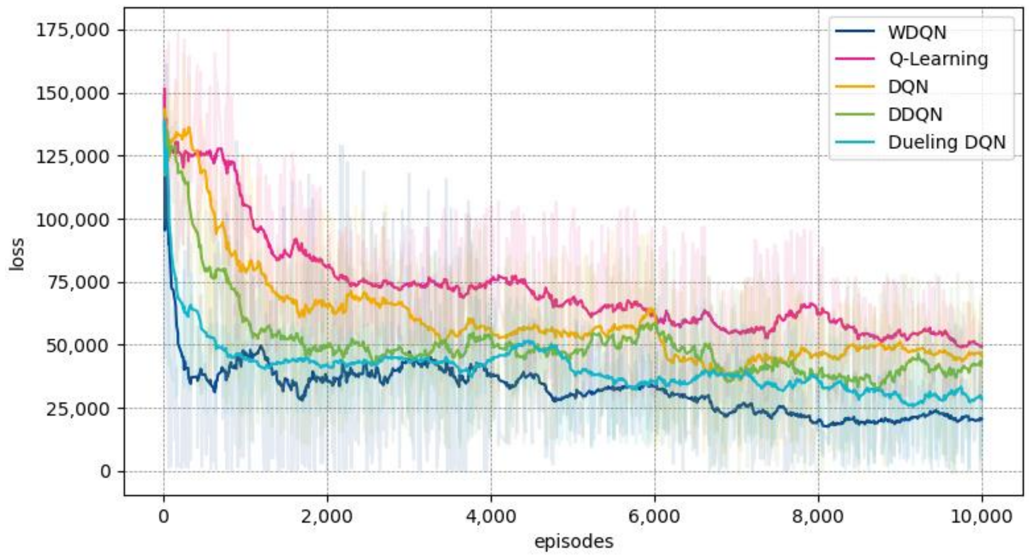


Figure 9. Comparison of loss during episodes.

First of all, observing the test results under the level 1 scene complexity setting, it can be seen from Figure 10 that the success rate of the path planning algorithm based on the WDQN is as high as 99.5%. Among the remaining four comparison methods, the success rates of Q-learning and the DQN are relatively good. This is because the network structure of these two models is simple, and the demand for training data is not large, so they are suitable for dealing with path planning problems in simple task scenes.

Then, observing the test results under the level 5 scene complexity setting, it can be seen that the relatively simple learning method of the Q-learning algorithm limits its performance in complex scenes, and its success rate is the lowest. Due to the more effective exploration and learning in the training phase, the WDQN and Dueling DQN show higher performances. Between the two algorithms, the success rate of the WDQN exceeds 95%. Finally, under the level 10 scene complexity setting, the performances of all five reinforcement learning models in the path planning task decrease. Among them, the success rate of the Dueling DQN is close to 70%, and the success rate of the DDQN, the DQN, and Q-learning is below 50%. Under this setting, the performance of the WDQN proposed in this paper is least affected by scene complexity, and its success rate still remains above 90%. Overall, our algorithm has good robustness under different scene settings and can perform path planning tasks with a high success rate.

5.3.4. Average Path Length

The average path length of a path planning algorithm refers to the average length of feasible paths planned in an unknown test environment. A short average path length means less distance, less time, and less energy consumption for the UAV to fly. To compare the average path lengths of these five well-trained reinforcement learning models, we conduct 1000 independent tests under three different scene complexity settings, and each test starts with changing the task environment. In order to ensure the fairness of the experiments, we set the number of UAVs and targets as one, the initial position of the UAV is fixed as [0,0], and the position of the trapped target is fixed as [95,95]. In addition, each algorithm has obstacles with the same properties when changing the task scene; that is, the number, location, and size of obstacles remain consistent. In this part, the average path length can be obtained by calculating the ratio between the total length of feasible paths and the number of feasible paths. Figure 11 shows the average path length of five reinforcement learning models under different scene complexity settings.

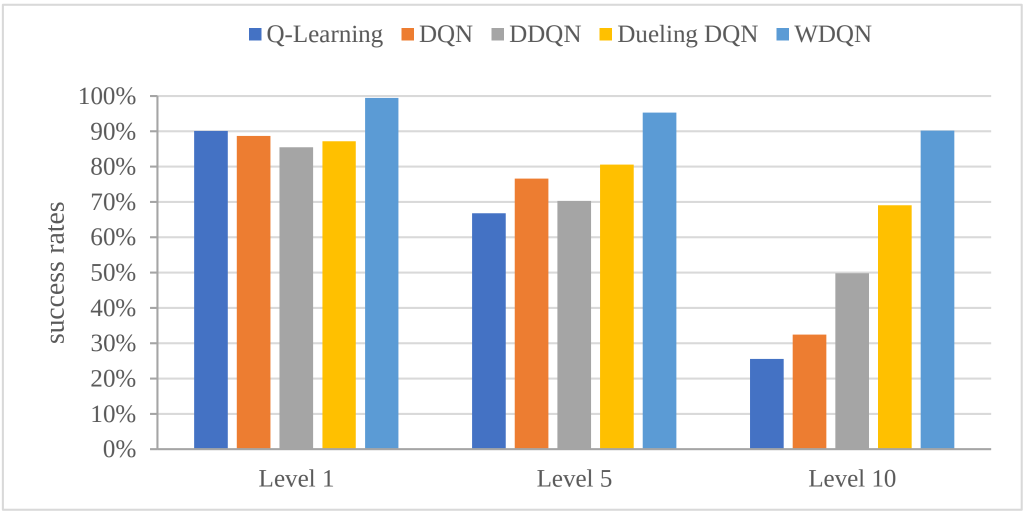
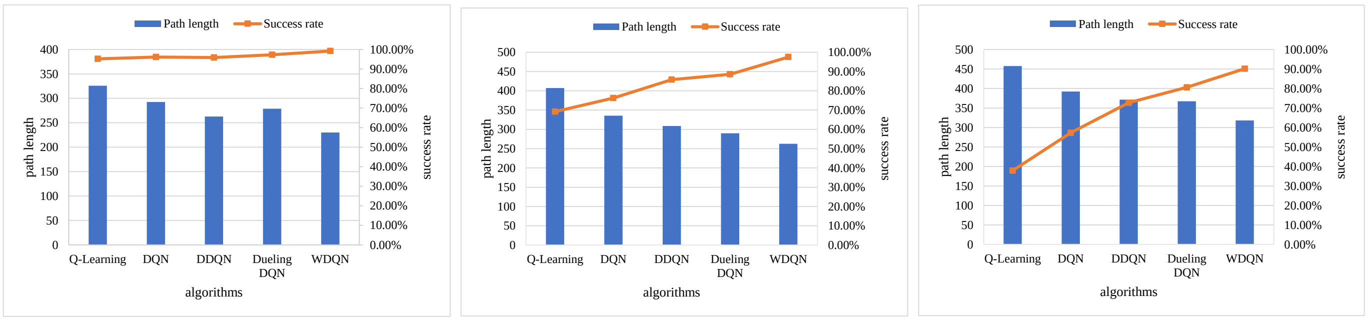


Figure 10. Comparison of success rates of different algorithms across varying difficulty levels.

According to the changes in the three sub-images, it can be seen that, in order to avoid obstacles, the average path length of these algorithms will increase as the scene complexity increases. In the tests of a single-UAV single-target search, the effective paths planned by Q-learning and the DQN are relatively long, and the success rate is low. It is difficult for these two algorithms to plan a satisfactory rescue route in a complex environment. Both the DDQN and Dueling DQN show similar performance in terms of success rate and effective path length. The WDQN algorithm proposed in this paper can plan the shortest path with a high success rate. Under these three levels of scene complexity settings, the average paths are 230.19 m, 262.755 m, and 318.153 m, respectively.



(a) Level 1 (b) Level 5 (c) Level 10

Figure 11. Comparison of average path lengths and success rates of different algorithms across varying difficulty levels.

6. Conclusions

In this paper, we study the path planning problem of multiple UAVs in emergency rescue scenarios based on the RL model and propose the WDQN-based path planning algorithm. The WDQN overcomes the limitations of traditional path planning algorithms in uncertain task scenes and improves the learning efficiency of RL models, enabling the UAV to autonomously plan a feasible collision-free path in an uncertain scene. Through continuous trial and error and studying feedback from the environment, the model gradually adapts and explores feasible collision-free paths in different scenes. In addition, we train the Q-network by gradually increasing the difficulty of obstacle avoidance, which enhances the robustness of the algorithm. In order to test the effectiveness of the proposed algorithm, we conduct experiments under different difficulty levels of obstacle avoidance, different numbers of task targets, and different scales of task areas. The results show that the proposed algorithm can successfully find targets and plan safe and feasible paths in all task scenes. Moreover, compared with several other RL algorithms, the proposed algorithm shows a better performance in terms of training efficiency, task completion rate, and average path length.

In three-dimensional space, the UAV needs to consider movement in both the horizontal and vertical directions, as well as different angles of rotation. This increases the number of actions in the UAV's action set to 27, which in turn increases the complexity of the action set and leads to a larger search space in the path planning algorithm. In addition, the jitter can lead to instability in the UAV's path, resulting in an increased energy consumption and flight risks, particularly in three-dimensional space. Consequently, in future work, we plan to propose a path planning strategy to address the complexity of a real-world three-dimensional environment, aiming to reduce the search space, accelerate the path planning process, and generate high-quality paths.

Furthermore, UAV path planning in large-scale mission environments is also a challenging problem. First of all, in large-scale task scenarios, UAVs need to plan paths in complex environments, avoid obstacles, and take into account factors such as task priorities and time constraints. This requires path planning algorithms to efficiently handle large-scale maps and tasks while ensuring the safety and efficiency of the path. In future work, we will consider dividing the large-scale region into multiple small task regions, decomposing the complex path planning problem into multiple relatively simple sub-problems. By decomposing the problem, we will reduce the computational complexity and the difficulty of path planning and design appropriate allocation algorithms and mechanisms to achieve intra-regional and inter-regional path planning and collaboration.

Author Contributions: Conceptualization, W.W.; methodology, W.W.; software, W.W.; validation, Q.D.; formal analysis, Q.D.; investigation, D.L. (Dan Lu) and D.L. (Dapeng Lang); resources, G.Z.; data curation, D.L. (Dan Lu) and D.L. (Dapeng Lang); writing—original draft preparation, W.W. and Q.D.; writing—review and editing, S.L. and Y.Z.; visualization, W.W.; supervision, S.L. and Y.Z.; project administration, G.Z.; funding acquisition, Y.Z. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the Basic Science Research Plan under Grant No. JCKY202160 4B002, and the China Scholarship Council under Grant No. 202006680073.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data available on request from the authors.

Acknowledgments: They authors thank the editor and anonymous reviewers for their helpful comments on improving the quality of this paper. They would also like to thank their laboratory team member for technical support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Zhang, Z.; Wu, J.; He, C. Search method of disaster inspection coordinated by multi-UAV. In Proceedings of the 2019 Chinese Control Conference (CCC), Guangzhou, China, 27–30 July 2019; pp. 2144–2148.
2. Hikichi, H.; Aida, J.; Kondo, K.; Tsuboya, T.; Kawachi, I. Residential relocation and obesity after a natural disaster: A natural experiment from the 2011 Japan Earthquake and Tsunami. *Sci. Rep.* **2019**, *9*, 374. [CrossRef] [PubMed]
3. Bhatta, S.; Dang, J. Seismic damage prediction of RC buildings using machine learning. *Earthq. Eng. Struct. Dyn.* **2023**, *52*, 3504–3527. [CrossRef]
4. Daud, S.M.S.M.; Yusof, M.Y.P.M.; Heo, C.C.; Khoo, L.S.; Singh, M.K.C.; Mahmood, M.S.; Nawawi, H. Applications of drone in disaster management: A scoping review. *Sci. Justice* **2022**, *62*, 30–42. [CrossRef]
5. Yamazaki, F.; Miyazaki, S.; Liu, W. 3D visualization of landslide affected area due to heavy rainfall in Japan from UAV flights and SfM. In Proceedings of the IGARSS 2018—2018 IEEE International Geoscience and Remote Sensing Symposium, Valencia, Spain, 22–27 July 2018; pp. 5685–5688.
6. Deng, X.; Li, R.; Zhao, L.; Wang, K.; Gui, X. Multi-obstacle path planning and optimization for mobile robot. *Expert Syst. Appl.* **2021**, *183*, 115445. [CrossRef]

7. Du, Y. Multi-UAV Search and Rescue with Enhanced a Algorithm Path Planning in 3D Environment. *Int. J. Aerosp. Eng.* **2023**, *2023*, 8614117. [CrossRef]
8. Dhulkefl, E.; Durdu, A.; Terzioğlu, H. Dijkstra algorithm using UAV path planning. *Konya J. Eng. Sci.* **2020**, *8*, 92–105. [CrossRef]
9. Jin, H.; Cui, W.; Fu, H. Improved RRT-connect algorithm for urban low-altitude UAV route planning. *J. Phys. Conf. Ser.* **2021**, *1948*, 012048. [CrossRef]
10. Ibrahim, M.S.; Rahman, S.; Hasan, M.S.; Ahmad, M.U.; Abrar, A. Flow-Based Path Planning for Multiple Homogenous UAVs for Outdoor Formation-Flying. In Proceedings of the 2022 7th International Conference on Mechanical Engineering and Robotics Research (ICMERR), Krakow, Poland, 9–11 December 2022; pp. 18–26.
11. Pairet Artau, È.; Hernández Vega, J.D.; Carreras Pérez, M.; Petillot, Y.R.; Lahijanian, M. Online Mapping and Motion Planning Under Uncertainty for Safe Navigation in Unknown Environments. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 3356–3378. [CrossRef]
12. Bayerlein, H.; Theile, M.; Caccamo, M.; Gesbert, D. Multi-UAV path planning for wireless data harvesting with deep reinforcement learning. *IEEE Open J. Commun. Soc.* **2021**, *2*, 1171–1187. [CrossRef]
13. Chen, Y.; Dong, Q.; Shang, X.; Wu, Z.; Wang, J. Multi-UAV autonomous path planning in reconnaissance missions considering incomplete information: A reinforcement learning method. *Drones* **2022**, *7*, 10. [CrossRef]
14. Liu, S.; Bai, Y. Uav intelligent coverage navigation based on drl in complex geometrical environments. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 1–12. [CrossRef]
15. Samir, M.; Sharafeddine, S.; Assi, C.M.; Nguyen, T.M.; Ghrayeb, A. UAV trajectory planning for data collection from time-constrained IoT devices. *IEEE Trans. Wirel. Commun.* **2019**, *19*, 34–46. [CrossRef]
16. Liu, H.; Ge, J.; Wang, Y.; Li, J.; Ding, K.; Zhang, Z.; Guo, Z.; Li, W.; Lan, J. Multi-UAV optimal mission assignment and path planning for disaster rescue using adaptive genetic algorithm and improved artificial bee colony method. *Actuators* **2021**, *11*, 4. [CrossRef]
17. Nayeem, G.M.; Fan, M.; Akhter, Y. A time-varying adaptive inertia weight based modified PSO algorithm for UAV path planning. In Proceedings of the 2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST), Dhaka, Bangladesh, 5–7 January 2021; pp. 573–576.
18. Kurdi, H.; Al-Megren, S.; Aloboud, E.; Alnuaim, A.A.; Alomair, H.; Alothman, R.; Muhayya, A.B.; Alharbi, N.; Alenzi, M.; Youcef-Toumi, K. Bee-inspired task allocation algorithm for multi-UAV search and rescue missions. *Int. J. Bio-Inspired Comput.* **2020**, *16*, 252–263. [CrossRef]
19. Khan, A.; Gupta, S.; Gupta, S.K. Emerging UAV technology for disaster detection, mitigation, response, and preparedness. *J. Field Robot.* **2022**, *39*, 905–955. [CrossRef]
20. Ullah, F.; Khan, S.I.; Munawar, H.S.; Qadir, Z.; Qayyum, S. Uav based spatiotemporal analysis of the 2019–2020 new south wales bushfires. *Sustainability* **2021**, *13*, 10207. [CrossRef]
21. Munawar, H.S.; Ullah, F.; Khan, S.I.; Qadir, Z.; Qayyum, S. UAV assisted spatiotemporal analysis and management of bushfires: A case study of the 2020 victorian bushfires. *Fire* **2021**, *4*, 40. [CrossRef]
22. Ko, Y.; Kim, J.; Duguma, D.G.; Astillo, P.V.; You, I.; Pau, G. Drone secure communication protocol for future sensitive applications in military zone. *Sensors* **2021**, *21*, 2057. [CrossRef]
23. Guo, S.; Guo, D.; Zhang, Q.; Wu, N.; Deng, J. Research progress of anti-jamming technology of unmanned aerial vehicle (UAV) data link. *IOP Conf. Ser. Mater. Sci. Eng.* **2020**, *816*, 012011.
24. Krichen, M.; Adoni, W.Y.H.; Mihoub, A.; Alzahrani, M.Y.; Nahhal, T. Security challenges for drone communications: Possible threats, attacks and countermeasures. In Proceedings of the 2022 2nd International Conference of Smart Systems and Emerging Technologies (SMARTTECH), Riyadh, Saudi Arabia, 9–11 May 2022; pp. 184–189.
25. Zhou, P.; Wu, D.; He, Y.; Pan, Y. Improved Path Planning Algorithm based on Fuzzy Control combining A* Artificial Potential Field Method. In Proceedings of the 2022 34th Chinese Control and Decision Conference (CCDC), Hefei, China, 15–17 August 2022; pp. 4640–4645.
26. Wang, J.; Li, Y.; Li, R.; Chen, H.; Chu, K. Trajectory planning for UAV navigation in dynamic environments with matrix alignment Dijkstra. *Soft Comput.* **2022**, *26*, 12599–12610. [CrossRef]
27. Li, J.; Liao, C.; Zhang, W.; Fu, H.; Fu, S. UAV Path Planning Model Based on R5DOS Model Improved A-Star Algorithm. *Appl. Sci.* **2022**, *12*, 11338. [CrossRef]
28. Farid, G.; Cocuzza, S.; Younas, T.; Razzaqi, A.A.; Wattoo, W.A.; Cannella, F.; Mo, H. Modified A-Star (A*) Approach to Plan the Motion of a Quadrotor UAV in Three-Dimensional Obstacle-Cluttered Environment. *Appl. Sci.* **2022**, *12*, 5791. [CrossRef]
29. Wang, H.; Li, G.; Hou, J.; Chen, L.; Hu, N. A path planning method for underground intelligent vehicles based on an improved RRT* algorithm. *Electronics* **2022**, *11*, 294. [CrossRef]
30. Hu, Z.; Qin, J.; Wang, Z.; He, J. Robot Path Planning Based on Multi-strategy Improved RRT* Algorithm. In Proceedings of the 2022 6th International Conference on Automation, Control and Robots (ICACR), Shanghai, China, 23–25 September 2022; pp. 17–24.
31. Dong, L.; Liu, Z.; Jiang, F.; Wang, K. Joint optimization of deployment and trajectory in UAV and IRS-assisted IoT data collection system. *IEEE Internet Things J.* **2022**, *9*, 21583–21593. [CrossRef]
32. Yu, Z.; Si, Z.; Li, X.; Wang, D.; Song, H. A novel hybrid particle swarm optimization algorithm for path planning of UAVs. *IEEE Internet Things J.* **2022**, *9*, 22547–22558. [CrossRef]

33. Yan, X.; Ding, R.; Luo, Q.; Ju, C.; Wu, D. A Dynamic Path Planning Algorithm Based on the Improved DWA Algorithm. In Proceedings of the 2022 Global Reliability and Prognostics and Health Management (PHM-Yantai), Yantai, China, 13–16 October 2022; pp. 1–7.
34. Tan, Z.; Wei, N.; Liu, Z. Local Path Planning for Unmanned Surface Vehicle based on the Improved DWA Algorithm. In Proceedings of the 2022 41st Chinese Control Conference (CCC), Hefei, China, 25–27 July 2022; pp. 3820–3825.
35. Li, Y.; Tian, B.; Yang, Y.; Li, C. Path planning of robot based on artificial potential field method. In Proceedings of the 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 4–6 March 2022; Volume 6, pp. 91–94.
36. Sun, Y.; Chen, W.; Lv, J. Uav Path Planning Based on Improved Artificial Potential Field Method. In Proceedings of the 2022 International Conference on Computer Network, Electronic and Automation (ICCNEA), Xi'an, China, 23–25 September 2022; pp. 95–100.
37. Souto, A.; Alfaia, R.; Cardoso, E.; Araújo, J.; Francês, C. UAV Path Planning Optimization Strategy: Considerations of Urban Morphology, Microclimate, and Energy Efficiency Using Q-Learning Algorithm. *Drones* **2023**, *7*, 123. [CrossRef]
38. Xu, X.; De Soto, B.G. Reinforcement learning with construction robots: A preliminary review of research areas, challenges and opportunities. In Proceedings of the ISARC: International Symposium on Automation and Robotics in Construction, Bogota, Colombia, 12–15 July 2022; IAARC Publications: Oulu, Finland, 2022; Volume 39, pp. 375–382.
39. Zhang, T.; Lei, J.; Liu, Y.; Feng, C.; Nallanathan, A. Trajectory optimization for UAV emergency communication with limited user equipment energy: A safe-DQN approach. *IEEE Trans. Green Commun. Netw.* **2021**, *5*, 1236–1247. [CrossRef]
40. Huang, Z.; Liu, S.; Zhang, G. The USV path planning of Dueling DQN algorithm based on tree sampling mechanism. In Proceedings of the 2022 IEEE Asia-Pacific Conference on Image Processing, Electronics and Computers (IPEC), Dalian, China, 14–16 April 2022; pp. 971–976.
41. Wang, Y.; Jiang, C.; Ren, T. UAV Path Planning Based on DDQN for Mountain Rescue. In *Proceedings of the International Conference on Intelligent Robotics and Applications*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 509–516.
42. Wang, B.; Li, S.; Gao, X.; Xie, T. UAV swarm confrontation using hierarchical multiagent reinforcement learning. *Int. J. Aerosp. Eng.* **2021**, *2021*, 1–12. [CrossRef]
43. Riedmiller, M.; Hafner, R.; Lampe, T.; Neunert, M.; Degrave, J.; Wiele, T.; Mnih, V.; Heess, N.; Springenberg, J.T. Learning by playing solving sparse reward tasks from scratch. In Proceedings of the International Conference on Machine Learning, Stockholm, Sweden, 10–15 July 2018; pp. 4344–4353.