Received XX Month, XXXX; revised XX Month, XXXX; accepted XX Month, XXXX; Date of publication XX Month, XXXX; date of current version XX Month, XXXX.

Digital Object Identifier 10.1109/OJVT.2024.1234567

Deep Reinforcement Learning Assisted UAV Path Planning Relying on Cumulative Reward mode and Region Segmentation

Zhipeng Wang, Soon Xin Ng and Mohammed El-Hajjar

[1]School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, United Kingdom
Email:{z.wang@soton.ac.uk, sxn@ecs.soton.ac.uk, meh@ecs.soton.ac.uk }

ABSTRACT In recent years, unmanned aerial vehicles (UAVs) have been considered for many applications, such as disaster prevention and control, logistics and transportation, and wireless communication. Most UAVs need to be manually controlled using remote control, which can be challenging in many environments. Therefore, autonomous UAVs have attracted significant research interest, where most of the existing autonomous navigation algorithms suffer from long computation time and unsatisfactory performance. Hence, we propose a Deep Reinforcement Learning (DRL) UAV path planning algorithm based on cumulative reward and region segmentation. Our proposed region segmentation aims to reduce the probability of DRL agents falling into local optimal trap, while our proposed cumulative reward model takes into account the distance from the node to the destination and the density of obstacles near the node, which solves the problem of sparse training data faced by the DRL algorithms in the path planning task. The proposed region segmentation algorithm and cumulative reward model have been tested in different DRL techniques, where we show that the cumulative reward model can improve the training efficiency of deep neural networks by 30.8% and the region segmentation algorithm enables deep Q-network agent to avoid 99% of local optimal traps and assists deep deterministic policy gradient agent to avoid 92% of local optimal traps.

INDEX TERMS UAV path planning, Deep Reinforcement Learning, Autonomous Navigation, Experience Replay, Cumulative Reward Model, Region Segmentation.

I. Introduction

In recent years, unmanned aerial vehicles (UAVs) have been widely used in many applications including weather forecast, disaster prevention and control [1], logistics and transportation, information collection, wireless communication as well as special environments operation [2]–[5]. UAVs can be roughly divided into two categories, namely fixed wing UAV [6], [7] and rotor UAV [8]. Fixed wing UAV has the characteristics of fast flight speed and large load capacity but poor flexibility. It is suitable for flying in an open environment and is often used in information collection, disaster early warning and geographic mapping [9]. The rotor UAV has the characteristics of small volume and strong flexibility, but its load capacity is generally small and its endurance time is shorter than that of the fixed wing UAVs [10]. This kind of UAV is suitable for performing tasks in small and medium-sized environments [11], such as urban logistics distribution, air base station (UAV assisted communication) [12], warehouse management and farm management [13], [14].

As mentioned above, UAVs have a very wide range of application scenarios. Here, as an example, we consider the use of UAVs to assist communication networks [15], [16], where UAVs can serve as aerial base stations [17], [18]. Zeng et al. presented in [19] the advantages, challenges, and potential of UAV assisting in wireless communication. The authors in [20] proposed a system for radio surveillance using the Twin Delayed Deep Deterministic Policy Gradient (TD3) model, and [21] combined the UAVs with the reconfigurable intelligent surface (RIS) for UAV Jamming Rejection and Path Planning, which shows the potential of combining the UAV and deep reinforcement learning framework. In this case, we consider using the UAVs to cruise at a fixed flight altitude, while continuously moving between optimal deployment positions. After the best deployment location is decided using techniques such as [22], [23], the dynamic deployment problem of the UAVs becomes a collision free path planning problem.

Many autonomous navigation schemes proposed for UAV autonomous navigation require a range of sensor arrays to collect a variety of information, and can require large computation capability [24]–[26]. On the other hand, as a resource limited platform, UAV has limited computational capabilities and energy supply [27]. Hence, there are two potential solutions for these challenges. The first is to optimize the algorithm of autonomous navigation and hence to reduce the computational power demand of autonomous navigation [28]. The second is to send the information to the cloud that has sufficient computing capabilities using wireless networks [29], [30]. *In this paper, we focus on the first scheme, that is to improve the UAV autonomous navigation algorithm, while also reducing its computational requirements.*

The UAV flight path in a three-dimensional open space may be obstructed by various objects. Therefore, obstacle avoidance is key in autonomous UAV navigation systems. There are many kinds of obstacles, such as buildings, trees, power poles and street lamps, which do not move in a short time. We consider this kind of obstacles as static obstacles. In addition to static obstacles, some dynamic obstacles may appear in the flight space of UAV, such as other non cooperative UAVs, i.e. UAVs that cannot communicate with each other, birds and some other flying objects such as balloons [31]. These dynamic obstacles pose a great challenge to the obstacle avoidance mechanism of UAVs [32], [33]. Furthermore, [34] proposes a resource allocation algorithm that can be applied to edge computing network of vehicles to effectively reduce the load of the communication server. This technique has the potential to be extended to the UAV systems. On the other hand, although DRL is considered the most reliable autonomous navigation technology for the UAVs at present, its training time consumption and computational power requirements are very large [35]. This makes the application of such algorithms in autonomous navigation UAV quite challenging.

A. Related Works

The path planning methods of the UAV has several categories, including sampling-based, heuristic algorithms and deep reinforcement learning based techniques. In the following sections, we present a review of these techniques.

1) Sampling-based Algorithms

The first category is the path planning algorithm based on sampling, which includes A-star [36], rapidly-exploring random tree (RRT) [37], cell decomposition [38], artificial potential field (APF) [39], [40] and probabilistic road-map (PRM) algorithms [41]. These sampling based methods require the knowledge of the working environment in advance or the ability to fully sample the environmental information and redefine the entire area as a node set [42]. Then the best path is obtained by the optimal combination of nodes in the node set. These algorithms have their own advantages and disadvantages. Taking RRT as an example, the path planning algorithm based on RRT can efficiently sample the space and quickly and effectively search for the path that meets the requirements in high-dimensional space [43], [44]. The algorithm has great advantages in time complexity and space complexity. However, the RRT has certain requirements for the search space. If the search space is too complex or narrow, such as a complicated maze, the efficiency of RRT will be greatly reduced. Even if the path planning is successfully completed, the search time will still be very long.

Similar to the RRT algorithm, the PRM algorithm is also based on sampling and search. The difference is that the PRM is based on a comprehensive query and the RRT is based on a single point query [45], [46]. Since the PRM has less stringent requirements on the environment than the RRT algorithm, the universality of the PRM algorithm is wider than the RRT algorithm [41], [47]. However, the computing resources required by these methods increase dramatically with the increase in the environment space. The above techniques all have the problem of falling into the local optimal trap, where the APF technology is unable to find the global optimal path when facing large obstacles [48].

2) Heuristic Algorithms

The second category of path planning is based on biologically inspired heuristic algorithms [49], with the most representative algorithms being the genetic algorithm [50]–[52], ant colony optimization algorithm [53]–[55] and other artificial intelligence algorithms [35]. The generation of such algorithms is inspired by biological behaviour and they all need to know the details of the environment. Due to strict modeling, such algorithms can often obtain accurate path planning solutions. However, the employment of this kind of algorithm is challenging for large-scale and complex problems [56]. Among various biological heuristic algorithms, artificial intelligence method can basically complete the path planning task in an acceptable time when the environmental scale increases [56], [57]. Therefore, algorithms based on appropriate artificial intelligence techniques have shown good potential in path planning in medium-sized complex environments [58], [59].

3) Deep Reinforcement Learning Algorithms

Deep Q-Network (DQN) algorithm framework, which is a kind of deep reinforcement learning (DRL) algorithm has been proposed in [60]. The framework uses the experience replay method showing good performance in the process of learning how to play video games. Therefore, we consider applying the DQN algorithm framework of ”experience replay” [60] in the area of UAV path planning. Although DQN has better performance and more robust in large-scale and complex environments compared to other path planning algorithms, the DQN algorithm experiences high time complexity and divergence of neural networks. Moreover, similar to other path planning techniques, the DQN algorithm encounters the problem of local optimal traps in path planning tasks. *In order to improve the efficiency and robustness of the DQN path planning algorithm, we consider modeling the distribution of obstacles in the environment as a cumulative prior reward to pre-train UAVs.*

Although DRL algorithm is currently considered the most suitable technology for autonomous navigation of unmanned aerial vehicles, DRL still faces two main challenges, one is the low training efficiency of deep neural networks [61], [62], and the other is the risk of falling into local optimal traps [63]. In other fields of deep learning, there is also the problem of falling into local optimal traps, which some researchers refer to as the overfitting problem of deep neural networks [64], [65]. The above problems are mainly caused by the sparse training data or the probability distribution of the training data. The main solution to this problem is to reduce the sparsity of the training data [66], [67] and break the unique statistical characteristics of the training data [68]. However, for DRL algorithms, the training data is obtained through exploration, and some studies have proposed methods to solve special probability distributions, such as ”experience replay” mentioned earlier. For autonomous navigation of unmanned aerial vehicles, experience replay will make the reward values sparse. This is because the most crucial experience in the path planning problem is the transition upon reaching the destination, which is often given very high reward values. However, such experience only appears once in each training episode. After using experience replay, the key experience mentioned above may not even exist in the experience pool. Even if it exists, the probability of it being selected for learning is very low.

Against this backdrop we propose a method that can solve the above problems, which can effectively improve the training efficiency of neural networks for various DRL algorithms for autonomous navigation of unmanned aerial vehicles while minimizing the possibility of falling into local optimal traps.

B. Contributions

Our contributions can be listed as follows:

- We propose a framework of DRL for UAV path planning that is applicable to most current autonomous navigation algorithms for DRL UAV agents, such as DQN, deep deterministic policy gradient (DDPG), Actor-Critic(AC) and stochastic value gradient (SVG). More explicitly, this framework can be applied in both the Markov decision process (MDP) and the partially observable Markov decision process (POMDP) environments.
- We propose a cumulative reward model based on distance and obstacle density in order to reduce the training overhead. More specifically, compared with the traditional reward model, the DRL algorithm based on the cumulative reward model has a higher training efficiency.
- We propose a region segmentation algorithm that allows the DRL UAV agents to avoid falling into local optimal traps. The region segmentation algorithm can avoid falling into local optimal traps 99% of the time using DQN, while also avoiding falling into local optimal traps 92% of the time using DDPG.
- The region segmentation algorithm we propose applies soft decision reward value evaluation, which can mitigate issues related to slow convergence or network divergence in the POMDP environment.

The contributions of our paper are compared to the literature shown in Table 1. The rest of the paper is organized as follows. In Section II we introduce the system model. In Section III, we introduce the cumulative reward model based on distance, obstacle density and region segmentation algorithm. In Section IV, we demonstrate the feasibility of our proposed method through simulation results and compare it with the traditional reward model. In Section V, we offer our conclusions and propose potential future directions.

II. System Model and Problem Formulation

In our proposed system, when the deployment point of the UAV is given, the UAV will plan a collision free path to move from its current position to the deployment point. Taking the environment shown in Fig. 1 as an example, if the yellow dot represents the deployment location of the drone, the drone will plan a collision free path to fly to the deployment location. In this section, we formulate our problem and then introduce the background knowledge related to the DRL techniques used in our system.

A. Problem Formulation

We aim to solve the path planning problem of UAV, where the path should be as short as possible without collision. In this paper, we assume that the UAV cruises at a fixed flight altitude, where we only need to consider obstacles at the flying altitude of the UAV [35]. In order to facilitate the modeling of the workspace, we take a cross-section of the 3D workspace as shown in Fig. 1 at the flight height of the UAV to get the 2D simulation environment shown in Fig. 2. In Fig. 2 the red dot represents the starting position of the UAV, the black blocks represent the obstacles, the white blocks represents the non-obstacle node, and the blue block represents the destination.

In this grid simulation environment, the path planning result of UAV can be expressed as an ordered path vector $\boldsymbol{P} = [P_1, P_2, \ldots, P_{last}]$, where P_i represents the i-th node

TABLE 1: Novelty comparison with the state-of-the-art literature.

	our paper	[43]	[57]	[45]	[46]	[49]	[50]	[51]	[52]	[55]	[35]	[58]	[69]
Reinforcement Learning	✓		✓								✓	✓	✓
Deep neural network	✓		✓	✓		✓		✓				✓	✓
Path planning efficiency analysis	✓	✓		✓			✓	✓	✓		✓	✓	✓
Local optimal trap avoidance	✓				✓	✓	✓		✓	✓			

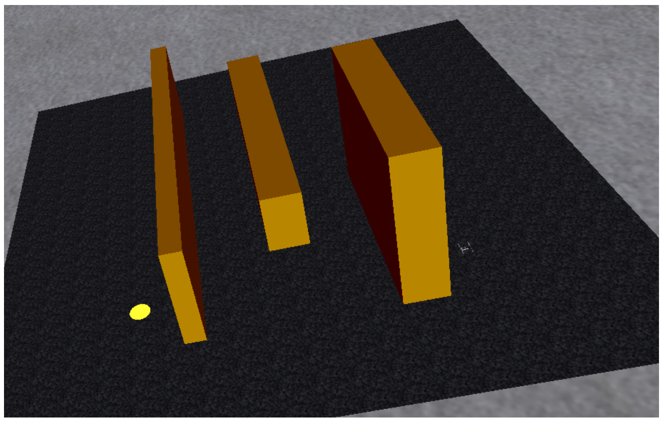


FIGURE 1: 3D view of UAV working environment in Simulink VRSink.

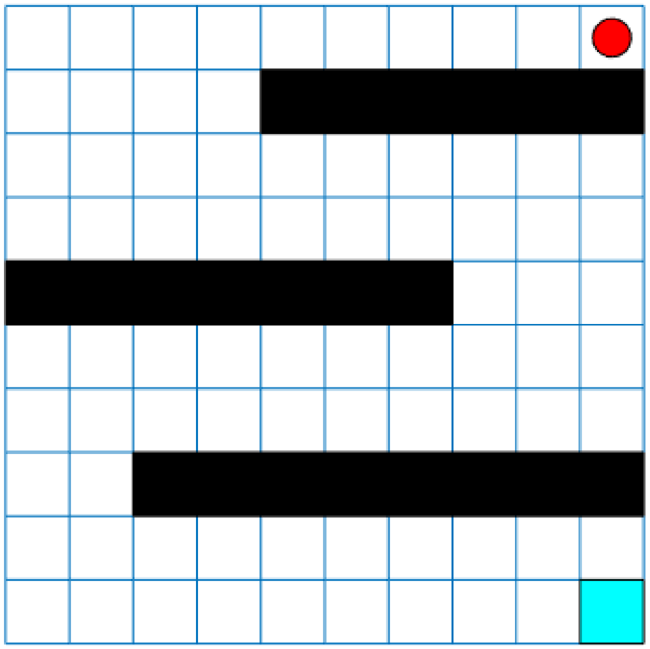


FIGURE 2: UAV working environment 2D model

in the path, P_{last} represents the last node in the path. The ordered vector $\boldsymbol{P}$ satisfies the condition $P_1 = S_{start}, P_{last} = S_{end}, \forall \hat{p} \subseteq \boldsymbol{P}$ satisfies $\hat{p} \in \boldsymbol{S}, \hat{p} \notin \boldsymbol{Obs}$, where $\boldsymbol{S}$ is the set of all possible states, S_{start} is the start node, S_{end} is the destination node, and $\boldsymbol{Obs}$ is the set of all obstacle nodes.

B. System Models

Our proposed UAV DRL autonomous navigation system considers two typical scenarios: discrete MDP environment and continuous POMDP environment, where the system model in both cases is different. In a discrete MDP environment, our UAV autonomous navigation system uses a positioning system to grasp the position of the UAV and the obstacle distribution of the environment, then produce a discrete workspace based on the environment information. Then, a UAV agent can be trained to complete real-time path planning by generating transaction experience through exploring the environment and training the neural network with the collected transaction experiences. During this process, the flight control and positioning system is always considered accurate and reliable.

We consider the path planning problem of UAV as a MDP, which includes several main parts: state set S, action set $\mathbf{A} = a_t = \{$Forward, backward, left, right, left front, right front, left back, right back$\}$, transfer function $T(s^{'}|s,a)$, and reward function $R(s^{'}|s,a)$. The main advantage of Q-learning is that it uses the time differential (TD) method (a combination of Monte Carlo and dynamic programming) for off-line learning, and Bellman equation can be used to solve the optimal strategy of the Markov process [70], [71].

The solution of the Bellman equation is based on the optimal cumulative expectation $V^*(s)$, which can be expressed as follows [72]:

$$V^*(s) = \max_{\pi} E\left[\sum_{t=0}^{t=n} \gamma^t R(S_{t+1}, A_t, S_t|\pi, s)\right], \quad (1)$$

where γ is the discount factor, t is current time index, and π is the action policy. The Q value $Q(s,a)$ is a function of the action state value, expressed as the expected accumulation of the action reward value as follows:

$$Q(s,a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots |a,s)], \quad (2)$$

where r_{t+n} is the reward value of next n steps of current time index t. The update of Q value is based on the Bellman's equation:

$$Q(s_{t+1},a_{t+1}) = (1-\alpha)Q(s_t,a_t) + \alpha[R(s_t,a_t) + \gamma \max_a (Q(s_{t+1},a))], \tag{3}$$

where α is the learning rate of the agent. If the agent executes action a_t in state s_t at time t, it will get immediate reward $R(s_t,a_t)$ and delayed reward $\max_a(Q(s_{t+1},a))$. It is not difficult to see that Q-learning considers not only the current reward expectation, but also the maximum reward that may be obtained in the future when making decisions according to the Q value [73].

However, there is an obvious limitation of using tables to store Q values. That is, when the working area expands or the motion freedom of the agent increases, the table size will grow exponentially. In order to overcome this problem, DQN has been proposed, which uses deep neural network to map the relationship between input information and Q value [74]. This method has many advantages, such as making Q-learning work in a high-dimensional and complex environment without worrying about the size of the Q table. On the other hand, it can adapt to various input signals, which improves the flexibility of the system [75].

Furthermore, we need a loss function to guide the adjustment of neural network weight. In DQN, the loss function can be expressed as the expectation of the square of the difference between the observed value (real value) and the network predicted value:

$$L_i(\theta_i) = ((R_{t+1} + \gamma \max_{a'} Q(S_{t+1},a'|\theta^-) - Q(S_t,A_t|\theta))^2, \tag{4}$$

where θ is the parameter set of the agent's network and it represents the agent's policy of action. DQN uses a target network, which has parameters from several steps earlier. A target network is used to estimate Q-values at the step $t+1$. Furthermore, $(R_{t+1} + \gamma \max_{a'} Q(S_{t+1},a'|\theta^-))$ is the observed Q value and $Q(S_t,A_t|\theta)$ is the predicted Q value of the network [76].

With the loss function, we only need to use the gradient descent method to continuously update the network weight to achieve the purpose of training. The process of network update can be expressed as:

$$\theta_{i+1} = \theta_i - \nabla_{\theta_i} L_i(\theta_i). \tag{5}$$

When the working environment of the UAVs is considered as a POMDP environment, the UAV agent cannot directly obtain the state s_t but can obtain environmental information by observing the feedback $O(o_t|s_{t+1},a_t)$. In this situation, the state transition cannot satisfy the Markov property. Hence, the UAV agent needs to traverse all observation history $O(o_t,o_{t-1},\ldots,o_1,o_0)$ to estimate state s_t. However, the DDPG and other DRL algorithms based on the AC framework can address this. In POMDP, both the state space and action space are continuous spaces, so the relationship between the action space $a(v_t,\vartheta_t)$ and the state space $S_{t+1}(x_{t+1},y_{t+1})$ of the UAV is given by:

$$\begin{cases} x_{t+1} = x_t + v_{t+1} \times \cos(\vartheta_{t+1}) \times \Delta t \\ y_{t+1} = y_t + v_{t+1} \times \sin(\vartheta_{t+1}) \times \Delta t \end{cases}, \tag{6}$$

where v_t is the flight velocity of the UAV in the horizontal direction at time t, ϑ_t is the horizontal direction angle of the UAV at time t.

In the DDPG algorithm, the action strategy of the UAV agent is represented by a neural network θ^μ, and the value estimation network is represented by θ^Q. Then, to update the action strategy network θ^μ, back propagation through time (BPTT) can be used as follows [69]:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_\tau \left[\sum_t \gamma^{t-1} \frac{\partial \theta^Q(h_t,a)}{\partial a}\bigg|_{a=\theta^\mu(h_t)} \frac{\partial \theta^\mu(h_t)}{\partial \theta} \right], \tag{7}$$

where $\theta^Q(h_t,a)$ is the observation history of the value network at time t, and $\theta^\mu(h_t)$ is the observation history of the action strategy network at time t. However, there are further challenges as reported in [77]. For example, the correlation between the training data or the fast update speed of the network may lead to the divergence of the neural network [78]. In this regard, Volodymyr Mnih and his colleagues optimized the DQN in [60].

The correlation between training data can lead to the divergence of the network, especially when using data with time series characteristics for training. Mobile path is a kind of data with continuous time characteristics. Therefore, in order to break the correlation between training data, experience replay is introduced [60]. This method divides the transition experience sequence of UAVs into some small fragments and puts them in the experience pool. In each training, a certain amount of experience fragments are randomly selected from the experience pool for training. This can effectively avoid the network divergence caused by the use of correlated data. The algorithm framework of Double Deep Q-network (DDQN) with experience replay is shown in Fig. 3. Under this framework, the agent selects actions based on random strategies or the current target Q-network and executes selected actions in the simulator, which is simulated and generates transitions. The transitions generated by the simulator will be stored as experience in the experience pool. When the network needs to be updated, the system will randomly extract experience from the experience pool to update the main Q-network, and after a certain delay, copy the weight of the main network to the target Q-network.

Unfortunately, experience replay brings inefficient learning to DDQN and DDPG UAV agents in path planning tasks, which can be attributed to the inappropriate design of the reward model. In the traditional reward model used in [79], the reward value of the UAV, when encountering obstacles, is a negative reward value R_{crash}, while the reward value when reaching the destination is a positive reward value R_{reach}. Then, any other legal reward value is set to 0. There is no problem with this reward model itself, but it adds challenges in the experience replay [80]. The

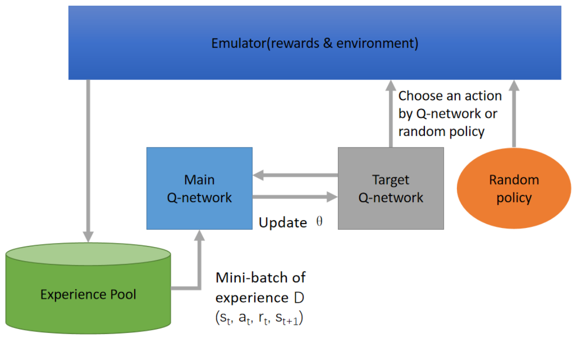

Emulator(rewards & environment)
Choose an action by Q-network or random policy
Main Q-network
Target Q-network
Random policy
Update θ
Experience Pool
Mini-batch of experience D
(s_t, a_t, r_t, s_t+1)

FIGURE 3: The algorithm framework of DDQN with experience replay.

experience $(s_t, a_t, r(s,a), s_{t+1})$ obtained by the UAV when exploring in the simulator or real environment is stored in the experience pool. At each training, a batch of experience will be randomly selected from the experience pool for learning [60].

It can be observed that only when the ”key experience” is learned can DRL agents be greatly improved, where a ”key experience” is obtained when a UAV would reach the destination in the next state s_{t+1}. However, after the introduction of experience replay as defined in [60], these key experiences only account for a small proportion in the experience pool. Moreover, these experiences may not be learned by agents, because of the limited size of the experience pool. More specifically, when new experiences are saved, the oldest experience will be deleted if the experience pool is full. Therefore, key experiences may be deleted without being learned by the UAV. More specifically, since the experience transitions in the experience pool is randomly selected, an experience transition may be discarded before it is learned by the UAV. *In order to mitigate this issue and to improve the training efficiency, we propose a cumulative reward model in the next section. This reward model trains UAVs using prior information such as obstacle density information in the environment and the distance from the current position to the endpoint, for improving the neural networks convergence speed.*

III. Proposed method

To address the problem of sparse key experiences in the experience pool caused by traditional reward models, we propose a cumulative reward model, which can make positive rewards more evenly distributed in the experience pool to solve the problem of neural network divergence and of the slow training caused by sparse rewards. In this section, a detailed introduction of our proposed cumulative reward model and the region segmentation is presented. Then, the framework of our proposed DQN and DDPG algorithms based on the cumulative reward model and the region segmentation will be discussed in detail.

A. Cumulative reward model based on distance and obstacle density

We propose to reduce the sparsity of reward model and computational resources required for training without changing the DRL algorithm framework with experience replay, while providing training efficiency to achieve better real-time performance. The training efficiency here refers to the number of episodes required for DRL UAV agents to converge during cold start. Cold start is the process in which UAVs freely explore in an unfamiliar environment in real environments or simulators until they can complete path planning in that environment. Correspondingly, hot start refers to the scenario when the UAVs already have a neural network adapted to the environment, which can directly perform real-time path planning based on the output of the neural network.

Our proposed method is based on the idea that the reward model can make all exploration experience beneficial to the network convergence, where even exploration experiences that do not reach the destination can still guide the agent to the destination. Inspired by the artificial potential field method [39], we propose a cumulative reward model, which provides cumulative reward values for the agent’s exploration experience based on the distance between the agent and the target point as well as the obstacle density in the adjacent space of the current node.

Specifically, for all non-obstacle areas, the reward value is calculated according to the Euclidean geometric distance between the current position and the end point as well as the obstacle density. In this way, the training is effective even when the UAV does not reach the end point or when the experience of reaching the end point has not been learned. This reward value is a cumulative value, which represents the cumulative value of all rewards for the UAV from the starting point to the current position. The corresponding cumulative reward function in a 2D environment can be computed as:

$$Reward(x,y,l) = \frac{C}{\sqrt{(x-x_d)^2+(y-y_d)^2} \times D(x,y,l)}, \tag{8}$$

where (x,y) is the destination coordinates, l is the size of the safety range, C is a reward constant, and D is the spatial obstacle density in the adjacent space of the current node. More explicitly, the value of C is half or less of the reward value for arriving at the destination, while D can be expressed as:

$$D(x,y,l) = \frac{1+Obs(x,y,l)}{S(x,y,l)}, \tag{9}$$

where $S(x,y,l)$ is the number of points in the adjacent points in size l square of the current point (x,y) for MDP. When for POMDP, $S(x,y,l)$ is the area with size l of current coordinate (x,y). The value of $Obs(x,y,l)$ for MDP is the number of obstacle nodes in the adjacent nodes in size l square of the current point (x,y), and that for POMDP is the area of obstacles in the area with size l of the current coordinate (x,y). The application of the cumulative reward model has

TABLE 2: The time frequency of the cumulative reward model in environments of different sizes.

Environment size	MDP Environment	POMDP Environment
T(n=10,m=10)	7.93×10^2	7.93×10^4
T(n=20,m=50)	7.993×10^3	7.993×10^6
T(n=50,m=100)	3.999×10^4	1.999×10^8
T(n=100,m=100)	7.999×10^4	7.999×10^8

low computational complexity in a 2D MDP environment with rectangular edges of length m and n, since its algorithm time frequency $T(n, m) = 8 \times m \times n - 7$, where $T(\cdot)$ represents the number of repetitions of key code in the algorithm. Hence, $O(n, m) = m \times n$ in a 2D MDP environment with rectangular edges of m and n, where $O(\cdot)$ is the time complexity of the algorithm. In the POMDP environment, due to the need for the UAV to update environmental information through exploration, if the distribution of obstacles in the environment remains unchanged, it often only requires a few updates to get all the information. Considering the worst-case scenario, in a rectangular POMDP environment with side lengths of m and n, the algorithm has a time complexity of $O(n, m) = n^2 \times m^2$. The time complexity of the cumulative reward model is shown in Table 2 for different sizes of the MDP and the POMDP environments. In both the MDP and POMDP environments, the aggressive reward strategy not only accelerates the convergence speed of the neural networks, but it also increases the likelihood of neural networks falling into local optimal traps. Hence, to reduce the risk of DRL UAV agents falling into local optimal traps, we propose a region segmentation algorithm to re-estimate the reward value generated by the cumulative reward model, as discussed in the following section.

B. Region segmentation

In the scenario where there are large obstacles in the environment as shown in Fig. 2, we found that the agent may be blocked by the longest obstacle and repeat heading to the same area. In other words, the agent is led to some local optimal trap. Furthermore, the agent will also be blocked by large-size obstacles such as large buildings. In order to overcome these challenges, we propose the region segmentation algorithm to help the agent ignore the action with higher reward value but actually blocked.

Firstly, the whole environment is divided into several small areas of $N \times N$ according to the set size, where the border information is shared between the adjacent areas. Then, each small area border is divided into λ parts, where eight ordered borders can be obtained as shown in Fig. 4. A clustering algorithm is used to detect the connectivity between the ordered borders, where we store the connectivity result (1 or 0) in the connectivity table $C(i, j, k)$. This connectivity detection algorithm is shown in Algorithm 1.

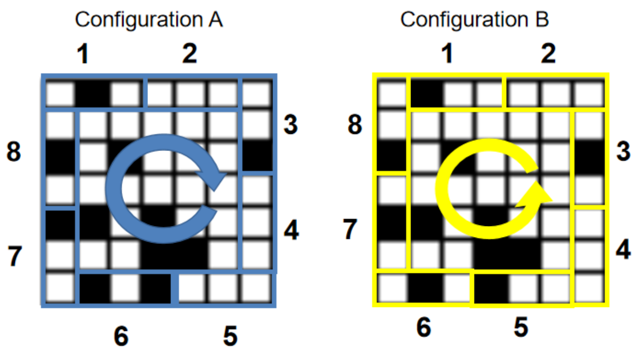


FIGURE 4: Two different configurations of small region border setting.

Algorithm 1 Connectivity detection algorithm based on clustering

```
1: Initialize divided small regions.
2: Initialize the value of $\alpha$ and $\beta$ ($\alpha \neq \beta$).
3: for each $j \in [1, 4 \times \lambda]$ do
4:     Copy small region $i$.
5:     Set the value of all non-obstacle nodes on border $j$ to $\alpha$;
6:     Store all non-obstacle nodes on border $j$ into waiting list $L$;
7:     while $L \neq \emptyset$ do
8:         Set the non-obstacle node value in the adjacent nodes of the first node in the waiting list L to $\beta$;
9:         Store the node whose value has changed into the waiting list.
10:    end while
11:    for each $k \in [1, 4 \times \lambda]$ do
12:        if there are nodes on the ordered border $k$, the value is $\beta$ then
13:            $C(i, j, k) = 1$;
14:            $C(i, k, j) = 1$;
15:        end if
16:    end for
17: end for
```

Based on Algorithm 1, we have connectivity information between cell block borders. When calculating the reward value, we take the connectivity of the corresponding two borders as the Boolean value (0 or 1) multiplied by the reward value calculated by the cumulative reward function:

$$R_{(i,j,k)}(x, y) = R(x, y) \times C(i, j, k), \qquad (10)$$

where $R(x, y)$ is the reward value obtained by the cumulative reward function and $C(i, j, k)$ is the logical value in the connectivity table that is related to region i, from border j to border k. Furthermore, (10) is called the hard decision of the reward value evaluation. Note that this reward value evaluation method can cause fluctuations in the neural net-

TABLE 3: The time frequency of the region segmentation.

Parameters	MDP Environment	POMDP Environment
$T(i=16, n=5, \lambda=2, N=20, M=20)$	2.56×10^4	1.024×10^7
$T(i=100, n=5, \lambda=2, N=50, M=50)$	1.6×10^5	4×10^8
$T(i=50, n=10, \lambda=2, N=50, M=100)$	3.2×10^5	1.6×10^9
$T(i=25, n=20, \lambda=2, N=100, M=100)$	6.4×10^5	6.4×10^9

work in the POMDP environment, because the reward value of actions reaching the blocked areas are set to 0. In training, learning those transitions experience may cause a significant increase or decrease of the learning curve. Therefore, we propose a soft decision reward value represented as:

$$R_{(i,j,k)}(x,y) = \begin{cases} R(x,y) & ,C(i,j,k)=1 \\ \omega \times R(x,y) \times \rho^{R(x,y)} & ,C(i,j,k)=0 \end{cases} \tag{11}$$

where ω and ρ are constants that satisfy $\omega, \rho \in (0,1)$.

In this way, the UAV can avoid moving into a local optimal trap. The region segmentation method can solve not only the local optimal trap problem faced by the cumulative reward model, but also the problem of agents falling into the local optimal trap in the artificial potential field method. The time frequency and the time complexity of the region segmentation in MDP environment are given by $T(i,n,\lambda) = 16 \times i \times n^2 \times \lambda^2$ and $O(i,n,\lambda) = i \times n^2 \times \lambda^2$, where i is the number of segmented small regions, n is the length of the sides of these square small areas, and λ is the number of partitions of the small regions' boundaries. Normally, the values of i and n do not exceed 100, and the value of λ is within 10. In the scenario presented in this paper, $\lambda = 2$. In the worst-case scenario of POMDP environment, the time complexity of this algorithm is $O(i,n,\lambda,N,M) = i \times n^2 \times \lambda^2 \times N \times M$, where N and M is the side length of the whole working environment. The time complexity of the region segmentation algorithm in MDP and POMDP environments for different is shown in Table 3.

In the process of using the region segmentation method, we must ensure the uniqueness of the connectivity table. In other words, regardless of the size of the segmented region, the border must be small enough. As shown in Fig. 4 the clockwise Configuration A or anticlockwise Configuration B rule is used to determine which border the node at the corner of a small area belongs to. Specifically, if Configuration A is used, the nodes in the corners of the small region need to be divided into their clockwise boundaries. For example, we can observe from Fig. 4 in Configuration A, the node in the upper right corner is assigned into the clockwise boundary (boundary 3), while in Configuration B the node in the upper right corner is assigned to the counter clockwise boundary (boundary 2). It is worth noting that two adjacent small regions need to use different corner node attribution rules, as shown in Fig. 5, where we can see that adjacent small regions with different configurations can overlap perfectly.

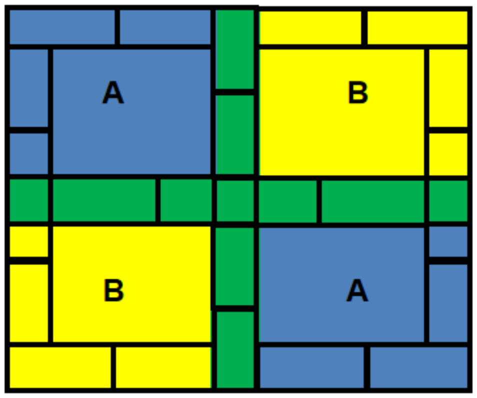


FIGURE 5: The overlapping scheme of adjacent small regions (blue region use configuration A, yellow region use configuration B, green shows the overlapping borders).

C. Deep reinforcement learning model combining cumulative reward model and region segmentation algorithm

Our proposed reward model and region segmentation algorithm can be applied to various deep learning algorithms and are suitable for MDP and POMDP environments. The following will take DQN algorithm for path planning in MDP environment and DDPG for path planning in POMDP environment as examples.

The pseudo code of the proposed DQN path planning algorithm based on the above cumulative reward model and region segmentation for UAVs is shown in Algorithm 2.

The difference between our proposed DQN algorithm 2, which combines the cumulative reward model and region segmentation, and the traditional DQN algorithm is elaborated in the following. Initially, it is necessary to initialize the experience pool and neural network weights. Then, region segmentation is preformed and the connectivity detection algorithm 1 is used to obtain a connectivity table. A collision free path will appear as all 1 in the connectivity table. Afterwards, the reward value is evaluated based on the cumulative reward model formula. Then, using random exploration and storing the exploration experience in the experience pool, and a batch of experience from the experience pool is sampled for learning. When using these experiences for training, if it is found that the action in the current experience corresponds to a value of 0 in the connectivity table, it indicates the potential collisions or traps. In this case, the reward value in

Algorithm 2 DQN path planning algorithm for UAVs based on the cumulative reward model and region segmentation in MDP environment

```
    Initialize experience replay memory $\mathcal{M}$ to capacity $\eta$.
 2: Initialize the network weight randomly to $\theta$.
    Run algorithm 1 to obtain connectivity information of small regions.
 4: **for** episode = 1 to M, **do**
      Initialize sequence $s_1 = x_1$ and preprocessed sequence $\phi_1 = \phi(s_1)$
 6:   **for** t = 1 to T, **do**
        According to the value of $\epsilon$, choose the random exploration strategy or the current network output $A(\theta)$ to determine the current action.
 8:     Execute the action and calculate the reward value $R_t = Reward(x, y)$ of the action according to the cumulative reward model based on (8), (9) and (10).
        Save the transition set $(s_t, a_t, R_t, s_{t1})$ into $\mathcal{M}$.
10:     Sample random mini-batch $I$ from $\mathcal{M}$ and do gradient descent to update network weight $\theta$.
      **end for**
12: **end for**
```

the current experience is treated as 0 to prevent the network from converging to local optimal traps due to this experience. As can be seen from Algorithm 2, the current reward value R_t is positive and will increase when the agent moves closer to the destination. Hence, the reward of each exploration experience of the algorithm will give a positive feedback to the agent. Even if the experience of the agent arriving at the destination is not learned, the reward value will guide the agent to explore in the direction closer to the destination and with lower obstacles density.

The pseudo code of the proposed DDPG path planning algorithm based on the above cumulative reward model and region segmentation for UAVs is shown in Algorithm 3. In our proposed algorithm, the initialization is the same as the traditional DDPG algorithm. Firstly, the experience pool is initialized and the actor network, critic network, and their corresponding target network weights are initialized. Then, by adding random noise to the actor network output, the environment is randomly explored, and the exploration experience is stored in the experience pool. Due to the inability to directly obtain current state information in the POMDP environment, agents can only obtain observation information. Therefore, Monte Carlo methods can be used to assist neural networks in traversing the observation history. Then, based on the current traversal of observation history, the action reward value for state transitions obtained through inference will be calculated using the cumulative reward model. At this point, if the environmental information is updated, a connectivity detection algorithm needs to be run to update the connectivity table. Otherwise, the reward value is estimated based on the current connectivity table and the state transition experience is learned to adjust the weights of the critic network. The adjustment of the actor network weights needs to be updated through back propagation through time (BPTT). By repeating the above process, the UAV agent's neural network will gradually converge and avoid most potential local optima.

Algorithm 3 DDPG path planning algorithm for UAVs based on the cumulative reward model and region segmentation in POMDP environment

```
    Initialize experience replay memory $\mathcal{M}$ to capacity $\eta$.
    Randomly initialize the actor network $\mu(s|\theta^{\mu})$ with weight $\theta^{\mu}$.
 3: Randomly initialize the critic network $Q(s, a|\theta^{Q})$ with weight $\theta^{Q}$.
    Copy networks $\mu$ and $Q$ to get target networks $\mu^{'}$ and $Q^{'}$ .
    **for** episode = 1 to M, **do**
 6:   Initialize a random process $\mathcal{N}$ for action exploration.
      Receive an initial observation $o_0 = h_0$.
      **for** t = 1 to T, **do**
 9:     Select action $a_t = \mu(s_t|\theta^{\mu}) + \mathcal{N}_t$ and execute $a_t$ to get observe $o_t$.
        Run algorithm 1 to obtain connectivity information of small regions if the environment information has been updated.
        Calculate reward $R_t$ according to (8), (9) and (11).
12:     Save the transition set $(o_t, h_t, a_t, R_t)$ into $\mathcal{M}$.
        Sample random mini-batch $I$ from $\mathcal{M}$
        Set $y_i^t = R_i^t + \gamma Q^{'}(h_i^{(t+1)}, \mu'((h_i^{t+1}|\theta^{\mu'})|\theta^{Q'})$.
15:     Update critic network by minimizing the loss: $L = \frac{1}{NT}\sum_i \sum_t \left( y_i^t - Q\left(h_i^t, a_i^t\right) \frac{\partial Q\left(h_i^t, a_i^t\right)}{\partial \theta} \right)$
        Update actor network using BPTT: $\nabla_{\theta^u} J = \frac{1}{NT}\sum_i \sum_t \frac{\partial Q\left(h_i^t, \theta^{\mu}\left(h_i^t|\theta^{Q'}\right)\right)}{\partial \theta} \frac{\partial \theta^{\mu}\left(h_i^t|\theta^{\mu'}\right)}{\partial \theta}$
        Update target networks:
        $\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$
        $\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$
18:   **end for**
    **end for**
```

IV. Simulation results

In this section, computer simulations are used to validate, analyze and discuss the algorithms proposed in this paper. The simulation experiments are divided into two parts. The first part is to use DDQN, DDPG and fast recurrent stochastic value gradient (FRSVG) [69] algorithms combined with cumulative reward models and region segmentation algorithms for training UAV agents to conduct path planning task in high building density urban areas. Then, the training efficiency of DRL UAV agents using traditional reward models and cumulative reward models will be compared. The second

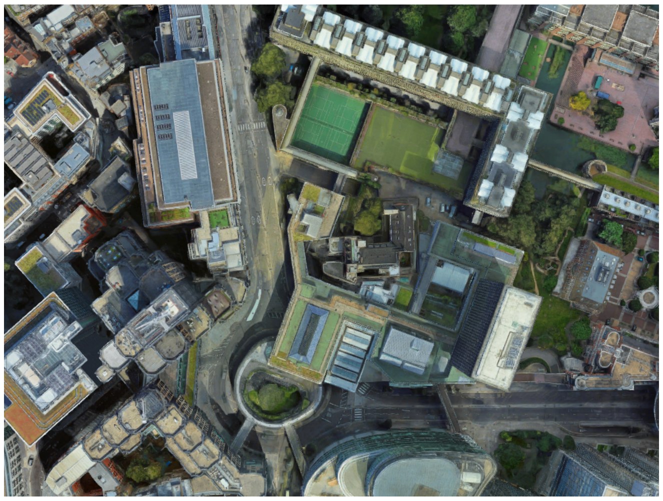


FIGURE 6: Satellite image of the real experimental environment.

part of the experiments involves conducting path planning for various UAV agents trained by DDQN, DDPG and FRSVG in four environments that are prone to falling into the local optimal trap, with and without the use of region segmentation algorithms. The probability of each DRL UAV agents falling into the local optimal trap is evaluated and compared.

A. Training efficiency of DRL agents using the cumulative reward model

We consider the scenario of UAV deployment in a real urban environment as shown in the satellite image of Fig. 6. In this scenario, the UAV performs the path planning task at a fixed flight altitude in the central area of the city with buildings of different heights. We chose a block near the Museum of London and used a shadow index algorithm [81] to estimate the height of buildings in the environment through satellite images. In this environment, the fixed flight altitude of the UAV is set at 20 meters above the ground, which is about 6 floors high. In this environment, the UAV will start from the area with dense buildings in the top left corner and plan a collision free path to the target point in the bottom right corner of the map.

After being processed and modeled by the shadow index algorithm of [81], the working environment in the simulator is shown in Fig. 7. The MDP environment for DQN is a grid world generated based on the binary map, while the POMDP environment for DDPG and FRSVG is a continuous numerical space based on the binary map. After the UAV is trained by DQN, DDPG and FRSVG algorithm based on the cumulative reward model, it can quickly plan a collision free flight path. The path planning results in the simulator are also shown in Fig. 7, where black nodes are obstacles and white nodes are free area, while the blue nodes constituted the path that the UAV chose. The flight trajectory of the UAV in the real world environment is shown in Fig. 8.

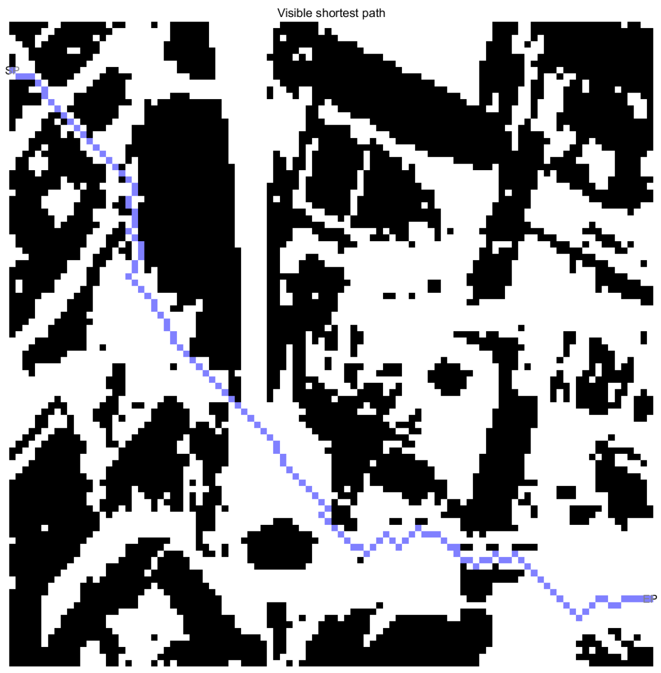


FIGURE 7: Path planning results of UAV in simulator environment.

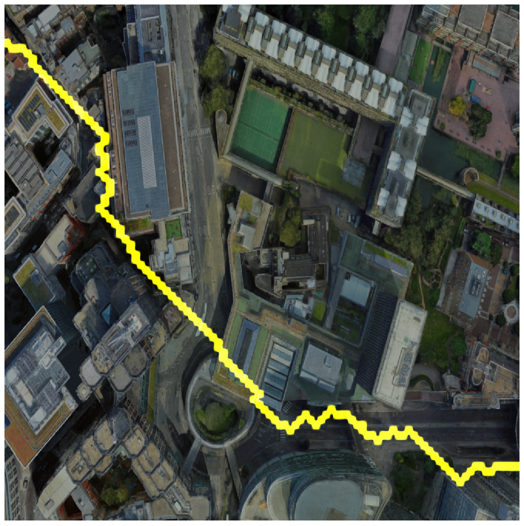


FIGURE 8: The flight trajectory of the UAV in the real world environment.

TABLE 4: Hyper-parameters of Deep Q-network.

Parameter	Value	Definition
Mini-batch	$2 \times N$	The sampling size of each training, N is the size of $N \times N$ environment.
γ	0.95	Discount factor
η	$N \times 100000$	Capacity of experience pool, N is the size of $N \times N$ environment.
ϵ	0.9:0.01:0.1	Possibility to select random actions, The initial value is 0.9, and the minimum value is 0.1. The decline step of each episode is 0.01.

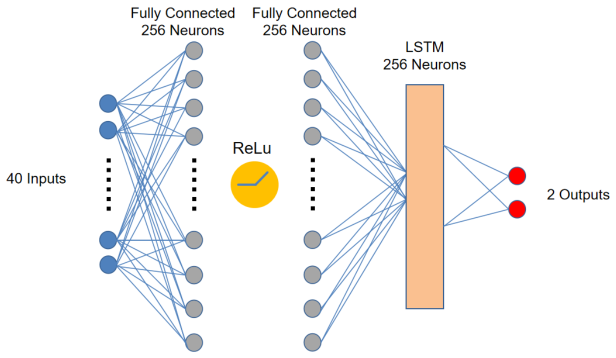


FIGURE 9: Actor neural network structure of DDPG in POMDP environment.

TABLE 5: Hyper-parameters of the DDPG model.

Parameter	Value	Definition
Mini-batch	$2 \times N$	The sampling size of each training, N is the size of $N \times N$ environment.
γ	0.95	Discount factor
η	$N \times 100000$	Capacity of experience pool, N is the size of $N \times N$ environment.
τ	0.001	Learning rate of actor and critic networks
σ_{Init}	1.0	Initial exploration variance
σ_{Min}	0.05	Final exploration variance
M	10000	Maximum number of training episodes
K	10	Steps delay of target networks update

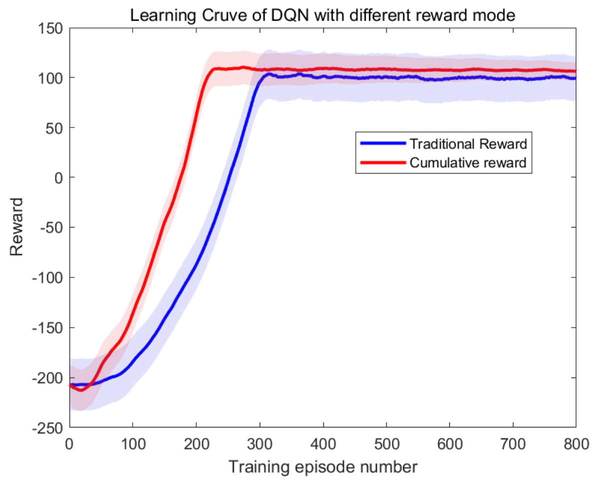


FIGURE 10: Learning curve of DQN agent in path planning training.

The hyper-parameters of DQN in the experiment are shown in Table 4. We change the frequency of ϵ value reduction to one reduction per episode rather than one reduction after a certain number of agent exploration steps. Such changes are conducive to the exploration of agents in large scale environments.

In this experiment, the network for DQN has four inputs and one output, two fully connected hidden layers, while each layer contains 128 neurons. The activation function uses rectified linear unit (ReLu). On the other hand, the neural network for DDPG and FRSVG has 40 inputs which are virtual sensor data, including ranging sensor data from 36 horizontal directions, as well as flight speed and 3D coordinates, and it has two outputs corresponding to velocity and horizontal direction angle, with two fully connected hidden layers, where each layer contains 256 neurons, where the LSTM network with 256 neurons is utilized. The actor neural network structure of DDPG and FRSVG is shown in Fig. 9. The hyper-parameters of the neural network training for DDPG and FRSVG are shown in Table 5.

Fig. 10 shows the average learning curve of 100 DQN agents, where we can see that the DQN agent using the cumulative reward model converges faster than the DQN agent using the traditional reward model, and the final stable reward value is slightly higher than the traditional reward model, this means that the cumulative reward model outperforms traditional reward models in terms of training efficiency and path planning quality. The shaded area represents the standard deviation of the learning curve, while the wider the shaded area, the greater the difference between different agents in the current episode. Fig. 11 shows the average learning curve of 60 DDPG agents, where it is shown that the DDPG agent using the cumulative reward model converges faster than the DDPG agent using the traditional reward model.

Fig. 12 shows the average learning curve of 60 FRSVG agents, where in this simulation, we compared our proposed cumulative reward model and the special reward model

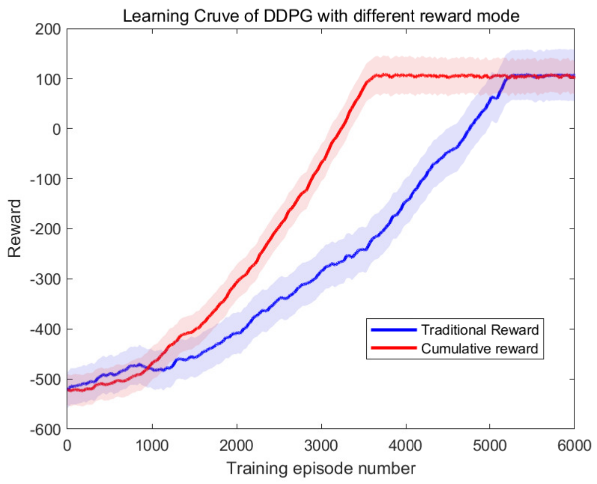


FIGURE 11: Learning curve of DDPG agent in path planning training.

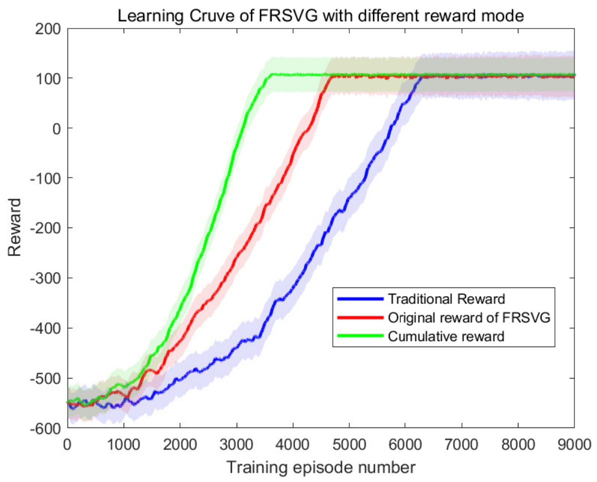


FIGURE 12: Learning curve of FRSVG agent in path planning training.

proposed with FRSVG in [69]. As can be seen in Fig. 12 the training efficiency of the cumulative reward model is higher than that of both the traditional reward model and the proposed reward model in [69].

B. Path planning results with region segmentation

We found that the DRL UAV agent based on the cumulative reward model could fall into a local optimal trap when encountering large closed obstacles. In order to verify that our proposed region segmentation method can solve the above problems, we placed three large enclosed obstacles with the size of 3×15 in a 20×20 environment as seen in Fig. 13. The area size used for the region segmentation is 6×6, and the entire 20×20 environment is divided into 16 small areas. Each border is divided into two parts, that is, each small region is divided into eight borders. In

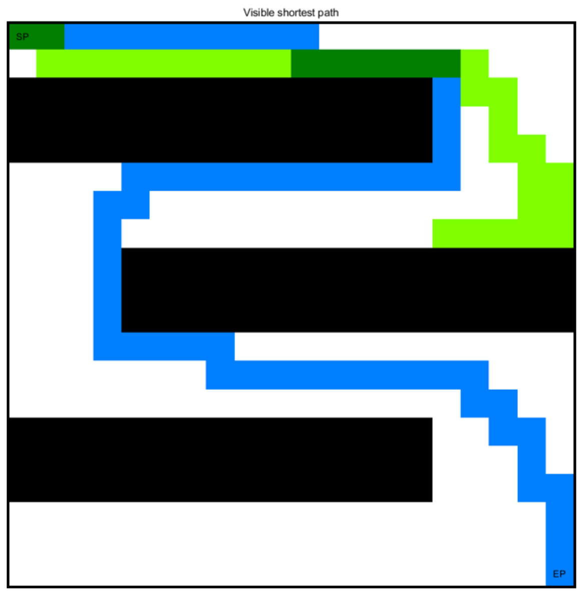


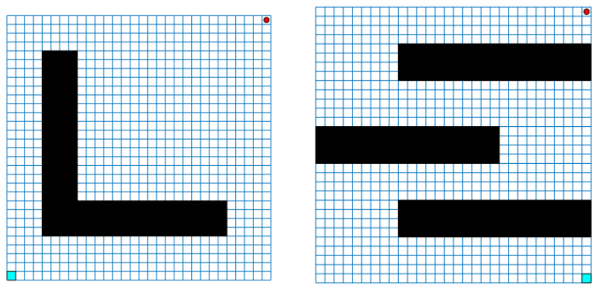
FIGURE 13: Path planning results of cumulative reward DQN with and without region segmentation method in a same 20×20 environment which has large closed obstacles.

this environment, the DQN is trained using the cumulative reward model without the region segmentation method and the cumulative reward model with the region segmentation method. The path planning results obtained after the same number of training episodes are shown in Fig. 13.

In Fig. 13, the "SP" is the start node, "EP" is the destination node, black nodes are obstacles, white nodes are free area, blue path represents the path using the region segmentation method, and the yellow path represents the path trained without the region segmentation method, while the green nodes represent the overlapping path. The results clearly show that the DQN agent trained without the region segmentation method falls into the local optimal trap when facing large closed obstacles. As seen in Fig. 13, the proposed region segmentation method is capable to mitigating the local optimal trap.

In order to quantify the performance of region segmentation methods in solving local optimal traps in UAV autonomous navigation problems, we considered four environments where UAV agents are prone to falling into local optimal traps. Among them, Environment 1 and Environment 2 only have large-sized obstacles, while Environment 3 and 4 add randomly distributed small-sized obstacles. Environments 1, 2, 3 and 4 are shown in Fig. 14a, Fig. 14b, Fig. 14c and Fig. 14d, respectively.

The red dots in the figure represents the starting points of the UAV, the light blue nodes represent the endpoint positions, and the black nodes represent the obstacle nodes. We trained 100 agents using DQN with and without Region Segmentation (RS), DDPG with and without RS, FRSVG with and without RS in each environment. The number of


(a) Environment 1 (b) Environment 2

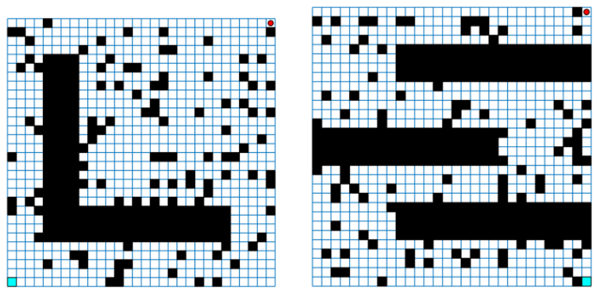

(c) Environment 3 (d) Environment 4

FIGURE 14: Four environments with large obstacles that the UAV agents are prone to falling into local optimal traps.

TABLE 6: Performance comparison of DQN, DDPG and FRSVG with or without Region Segmentation method (trapped rate with 100 agents)

Agent type	Env1	Env2	Env3	Env4
DQN	16%	21%	24%	26%
DQN-RS	0%	0%	0%	0%
DDPG	39%	37%	42%	38%
DDPG-RS	6%	8%	6%	8%
DDPG-RS-SD	4%	3%	4%	4%
FRSVG	11%	16%	14%	15%
FRSVG-RS	2%	2%	4%	3%
FRSVG-RS-SD	0%	0%	1%	0%

times the agents converged to the local optimal trap were recorded in percentage and presented in Table 6.

In the experimental results shown in Table 4, the RS represents the use of region segmentation algorithm, while SD represents the use of soft decision. From the experimental results, we can see that the region segmentation algorithm can avoid almost 99% of the local optimal traps in DQN, and does not require the use of soft decisions to correct the reward value. DDPG UAV agents are the easiest to fall into the local optimal trap, with the highest trapped rate reaching 42% when no region segmentation algorithm is used. However, after using the soft decision region segmentation algorithm, the trapped rate is only 6%. It is worth mentioning that the recently proposed FRSVG algorithm [69] can also benefit from the region segmentation algorithm. After combining with the soft decision aided region segmentation algorithm, the trapped rate is reduced from 16% to 0%, which means that the region segmentation algorithm can avoid 99% of local optimal traps. Given that the statistical sample is 100 agents per algorithm and per environment, 0% does not mean complete avoidance of the local optimal trap. When the number of agents reaches a very large value, there may be situations where they fall into the local optimal trap. Although the application of cumulative reward model and region segmentation method has improved the training efficiency and reliability of DRL based UAV autonomous navigation systems, it is still challenging to implement this system on the embedded UAV platforms to have the independent autonomous navigation UAV. Due to the fact that the UAV are resource limited platform, the energy it can carry is limited, and high-performance computing units consume a huge amount of energy. The energy that the UAV platform can carry is not enough to support long-term neural network training to update the neural network while moving. Therefore, our proposed algorithm can be applied to small and medium-sized 3D environments with known or unknown environmental information, while considering fixed flight altitude. The UAV using our proposed algorithm can perform autonomous navigation for logistics delivery, environmental information collection, and can support wireless communication systems.

V. Conclusions

In this paper, we proposed efficient path planning solution for the UAV using DRL, which is combined with the cumulative reward model and the region segmentation to improve the training efficiency and robustness of the DQN, DDPG and FRSVG frameworks. The reward model provides rewards for agents based on the distance between the current location and the destination, as well as depending on the density of obstacles in the adjacent area of the current location. This model can effectively solve the network divergence problem caused by the traditional reward model in the DRL algorithms with experience replay. We also introduced a region segmentation method to improve the robustness of DRL UAV agents training, when facing large enclosed obstacles. It can also reduce the probability of DRL agents converging to local optimal traps. Our simulations prove that the training efficiency of DRL UAV agents using our proposed cumulative reward model are 30.8% higher than that of the DRL UAV agents with traditional reward model in average. Moreover, the region segmentation method can help DQN and FRSVG agents to avoid 99% local optimal traps and help DDPG agents to avoid 92% local optimal traps. For future research, we will focus on extending the cumulative reward model and region segmentation algorithm to complex 3D environments.

REFERENCES

[1] Geng Sun, Long He, Zemin Sun, Qingqing Wu, Shuang Liang, Jiahui Li, Dusit Niyato, and Victor C. M. Leung. Joint Task Offloading and Resource Allocation in Aerial-Terrestrial UAV Networks with Edge and Fog Computing for Post-Disaster Rescue. *IEEE Transactions on Mobile Computing*, pages 1–18, 2024.

[2] Jeongeun Kim, Seungwon Kim, Chanyoung Ju, and Hyoung Il Son. Unmanned Aerial Vehicles in Agriculture: A Review of Perspective of Platform, Control, and Applications. *IEEE Access*, 7:105100–105115, 2019.

[3] Dyutimoy Nirupam Das, Rohan Sewani, Junwei Wang, and Manoj Kumar Tiwari. Synchronized Truck and Drone Routing in Package Delivery Logistics. *IEEE Transactions on Intelligent Transportation Systems*, 22(9):5772–5782, 2021.

[4] Suttinee Sawadsitang, Dusit Niyato, Puay Siew Tan, Ping Wang, and Sarana Nutanong. Shipper Cooperation in Stochastic Drone Delivery: A Dynamic Bayesian Game Approach. *IEEE Transactions on Vehicular Technology*, 70(8):7437–7452, 2021.

[5] Zhangfeng Ma, Bo Ai, Ruisi He, Gongpu Wang, Yong Niu, Mi Yang, Junhong Wang, Yujian Li, and Zhangdui Zhong. Impact of UAV Rotation on MIMO Channel Characterization for Air-to-Ground Communication Systems. *IEEE Transactions on Vehicular Technology*, 69(11):12418–12431, 2020.

[6] Pakorn Poksawat, Liuping Wang, and Abdulghani Mohamed. Gain Scheduled Attitude Control of Fixed-Wing UAV With Automatic Controller Tuning. *IEEE Transactions on Control Systems Technology*, 26(4):1192–1203, 2018.

[7] Boyang Zhang, Xiuxia Sun, Shuguang Liu, Maolong Lv, and Xiongfeng Deng. Event-Triggered Adaptive Fault-Tolerant Synchronization Tracking Control for Multiple 6-DOF Fixed-Wing UAVs. *IEEE Transactions on Vehicular Technology*, 71(1):148–161, 2022.

[8] Caiwu Ding and Lu Lu. A Tilting-Rotor Unmanned Aerial Vehicle for Enhanced Aerial Locomotion and Manipulation Capabilities: Design, Control, and Applications. *IEEE/ASME Transactions on Mechatronics*, 26(4):2237–2248, 2021.

[9] Wu Yi, Chen Liming, Kong Lingyu, Zhang Jie, and Wang Miao. Research on application mode of large fixed-wing UAV system on overhead transmission line. In *2017 IEEE International Conference on Unmanned Systems (ICUS)*, pages 88–91, 2017.

[10] Prithvi Krishna Chittoor, Bharatiraja Chokkalingam, and Lucian Mihet-Popa. A Review on UAV Wireless Charging: Fundamentals, Applications, Charging Techniques and Standards. *IEEE Access*, 9:69235–69266, 2021.

[11] Chuanzheng Li, Chuang Xue, and Yue Bai. Experimental investigation on aerodynamics of nonplanar rotor pairs in a multi-rotor UAV. In *2019 14th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 911–915, 2019.

[12] Xukai Zhong, Yiming Huo, Xiaodai Dong, and Zhonghua Liang. Deep Q-Network Based Dynamic Movement Strategy in a UAV-Assisted Network. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*, pages 1–6, 2020.

[13] Siva Sai, Akshat Garg, Kartik Jhawar, Vinay Chamola, and Biplab Sikdar. A Comprehensive Survey on Artificial Intelligence for Unmanned Aerial Vehicles. *IEEE Open Journal of Vehicular Technology*, 4:713–738, 2023.

[14] Seyed Saeed Khodaparast, Xiao Lu, Ping Wang, and Uyen Trang Nguyen. Deep Reinforcement Learning Based Energy Efficient Multi-UAV Data Collection for IoT Networks. *IEEE Open Journal of Vehicular Technology*, 2:249–260, 2021.

[15] Mingze Zhang, Yifeng Xiong, Soon Xin Ng, and Mohammed El-Hajjar. Content-Aware Transmission in UAV-Assisted Multicast Communication. *IEEE Transactions on Wireless Communications*, 22(11):7144–7157, 2023.

[16] Mingze Zhang, Yifeng Xiong, Soon Xin Ng, and Mohammed El-Hajjar. Deployment of Energy-Efficient Aerial Communication Platforms With Low-Complexity Detection. *IEEE Transactions on Vehicular Technology*, 72(9):12016–12030, 2023.

[17] Mingze Zhang, Mohammed EI-Hajjar, and Soon Xin Ng. Intelligent Caching in UAV-Aided Networks. *IEEE Transactions on Vehicular Technology*, 71(1):739–752, 2022.

[18] Hasini Viranga Abeywickrama, Ying He, Eryk Dutkiewicz, Beeshanga Abewardana Jayawickrama, and Markus Mueck. A Reinforcement Learning Approach for Fair User Coverage Using UAV Mounted Base Stations Under Energy Constraints. *IEEE Open Journal of Vehicular Technology*, 1:67–81, 2020.

[19] Yongs Zeng, Qingqing Wu, and Rui Zhang. Accessing From the Sky: A Tutorial on UAV Communications for 5G and Beyond. *Proceedings of the IEEE*, 107(12):2327–2375, 2019.

[20] Xin Yuan, Shuyan Hu, Wei Ni, Xin Wang, and Abbas Jamalipour. Deep Reinforcement Learning-Driven Reconfigurable Intelligent Surface-Assisted Radio Surveillance With a Fixed-Wing UAV. *IEEE Transactions on Information Forensics and Security*, 18:4546–4560, 2023.

[21] Shuyan Hu, Xin Yuan, Wei Ni, Xin Wang, and Abbas Jamalipour. RIS-Assisted Jamming Rejection and Path Planning for UAV-Borne IoT Platform: A New Deep Reinforcement Learning Framework. *IEEE Internet of Things Journal*, 10(22):20162–20173, 2023.

[22] Xiao Zhang and Lingjie Duan. Fast Deployment of UAV Networks for Optimal Wireless Coverage. *IEEE Transactions on Mobile Computing*, 18(3):588–601, 2019.

[23] Omar Bouhamed, Hakim Ghazzai, Hichem Besbes, and Yehia Massoud. A Generic Spatiotemporal Scheduling for Autonomous UAVs: A Reinforcement Learning-Based Approach. *IEEE Open Journal of Vehicular Technology*, 1:93–106, 2020.

[24] Abhishek Vashist, Sharan Vidash Vidya Shanmugham, Amlan Ganguly, and Sai Manoj P D. DQN Based Exit Selection in Multi-Exit Deep Neural Networks for Applications Targeting Situation Awareness. In *2022 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6, 2022.

[25] Yue Yang, Xiaoxiong Liu, Weiguo Zhang, Xuhang Liu, and Yicong Guo. Multilayer Low-Cost Sensor Local-Global Filtering Fusion Integrated Navigation of Small UAV. *IEEE Sensors Journal*, 22(18):17550–17564, 2022.

[26] Jono Vanhie-Van Gerwen, Kurt Geebelen, Jia Wan, Wout Joseph, Jeroen Hoebeke, and Eli De Poorter. Indoor Drone Positioning: Accuracy and Cost Trade-Off for Sensor Fusion. *IEEE Transactions on Vehicular Technology*, 71(1):961–974, 2022.

[27] Antonio Loquercio, Elia Kaufmann, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Learning high-speed flight in the wild. *Science Robotics*, 6(59):eabg5810, 2021.

[28] Philipp Foehn, Elia Kaufmann, Angel Romero, Robert Penicka, Sihao Sun, Leonard Bauersfeld, Thomas Laengle, Giovanni Cioffi, Yunlong Song, Antonio Loquercio, and Davide Scaramuzza. Agilicious: Open-source and open-hardware agile quadrotor for vision-based flight. *Science Robotics*, 7(67):eabl6259, 2022.

[29] Sixing Yin, Lihua Li, and F. Richard Yu. Resource Allocation and Basestation Placement in Downlink Cellular Networks Assisted by Multiple Wireless Powered UAVs. *IEEE Transactions on Vehicular Technology*, 69(2):2171–2184, 2020.

[30] Cheng Zhan and Yong Zeng. Energy Minimization for Cellular-Connected UAV: From Optimization to Deep Reinforcement Learning. *IEEE Transactions on Wireless Communications*, 21(7):5541–5555, 2022.

[31] Ang Gao, Qi Wang, Wei Liang, and Zhiguo Ding. Game Combined Multi-Agent Reinforcement Learning Approach for UAV Assisted Offloading. *IEEE Transactions on Vehicular Technology*, 70(12):12888–12901, 2021.

[32] Hongda Chen, Kuochu Chang, and Craig S. Agate. UAV Path Planning with Tangent-plus-Lyapunov Vector Field Guidance and Obstacle Avoidance. *IEEE Transactions on Aerospace and Electronic Systems*, 49(2):840–856, 2013.

[33] Chao Yin, Zhenyu Xiao, Xianbin Cao, Xing Xi, Peng Yang, and Dapeng Wu. Offline and Online Search: UAV Multiobjective Path Planning Under Dynamic Urban Environment. *IEEE Internet of Things Journal*, 5(2):546–558, 2018.

[34] Zemin Sun, Geng Sun, Yanheng Liu, Jian Wang, and Dongpu Cao. BARGAIN-MATCH: A Game Theoretical Approach for Resource Allocation and Task Offloading in Vehicular Edge Computing Networks. *IEEE Transactions on Mobile Computing*, 23(2):1655–1673, 2024.

[35] Yu-Jia Chen, Deng-Kai Chang, and Cheng Zhang. Autonomous Tracking Using a Swarm of UAVs: A Constrained Multi-Agent Reinforcement Learning Approach. *IEEE Transactions on Vehicular Technology*, 69(11):13702–13717, 2020.

[36] Gang Tang, Congqiang Tang, Christophe Claramunt, Xiong Hu, and Peipei Zhou. Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. *IEEE Access*, 9:59196–59210, 2021.

[37] Cong Zhao, Yifan Zhu, Yuchuan Du, Feixiong Liao, and Ching-Yao Chan. A Novel Direct Trajectory Planning Approach Based on Generative Adversarial Networks and Rapidly-Exploring Random Tree. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):17910–17921, 2022.

[38] Dae Hwan Kim, Giang Hoang, Min-Ji Bae, Jin Wook Kim, Suk Min Yoon, Tae-Kyeong Yeo, Hong Sup, and Sang-Bong Kim. Path tracking control coverage of a mining robot based on exhaustive path planning with exact cell decomposition. In *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, pages 730–735, 2014.

[39] Zhenhua Pan, Chengxi Zhang, Yuanqing Xia, Hao Xiong, and Xiaodong Shao. An Improved Artificial Potential Field Method for Path Planning and Formation Control of the Multi-UAV Systems. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 69(3):1129–1133, 2022.

[40] Jiayi Sun, Jun Tang, and Songyang Lao. Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm. *IEEE Access*, 5:18382–18390, 2017.

[41] Pritam Ojha and Atul Thakur. Real-Time Obstacle Avoidance Algorithm for Dynamic Environment on Probabilistic Road Map. In *2021 International Symposium of Asian Control Association on Intelligent Robotics and Industrial Automation (IRIA)*, pages 57–62, 2021.

[42] Haeyeon Lee, H. Kamaya, and K. Abe. Labeling Q-learning embedded with knowledge update in partially observable mdp environments. In *Second IEEE International Conference on Computational Cybernetics, 2004. ICCC 2004.*, pages 329–333, 2004.

[43] Guanchong Niu, Lan Wu, Yunfan Gao, and Man-On Pun. Unmanned Aerial Vehicle (UAV)-Assisted Path Planning for Unmanned Ground Vehicles (UGVs) via Disciplined Convex-Concave Programming. *IEEE Transactions on Vehicular Technology*, 71(7):6996–7007, 2022.

[44] Jingcheng Zhang, Yuqiang An, Jianing Cao, Shibo Ouyang, and Lei Wang. UAV Trajectory Planning for Complex Open Storage Environments Based on an Improved RRT Algorithm. *IEEE Access*, 11:23189–23204, 2023.

[45] Jie Qi, Hui Yang, and Haixin Sun. MOD-RRT*: A Sampling-Based Algorithm for Robot Path Planning in Dynamic Environment. *IEEE Transactions on Industrial Electronics*, 68(8):7244–7251, 2021.

[46] Jiankun Wang, Wenzheng Chi, Chenming Li, Chaoqun Wang, and Max Q.-H. Meng. Neural RRT*: Learning-Based Optimal Path Planning. *IEEE Transactions on Automation Science and Engineering*, 17(4):1748–1758, 2020.

[47] Antouan Anguelov, Roumen Trifonov, and Ognian Nakov. Colony Intelligence for Autonomous Wheeled Robot Path Planning. In *2020 28th National Conference with International Participation (TELECOM)*, pages 137–140, 2020.

[48] Praveen Kumar Selvam, Gunasekaran Raja, Vasantharaj Rajagopal, Kapal Dev, and Sebastian Knorr. Collision-free Path Planning for UAVs using Efficient Artificial Potential Field Algorithm. In *2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring)*, pages 1–5, 2021.

[49] Daqi Zhu, Xiang Cao, Bing Sun, and Chaomin Luo. Biologically Inspired Self-Organizing Map Applied to Task Assignment and Path Planning of an AUV System. *IEEE Transactions on Cognitive and Developmental Systems*, 10(2):304–313, 2018.

[50] Jianqiang Li, Genqiang Deng, Chengwen Luo, Qiuzhen Lin, Qiao Yan, and Zhong Ming. A Hybrid Path Planning Method in Unmanned Air/Ground Vehicle (UAV/UGV) Cooperative Systems. *IEEE Transactions on Vehicular Technology*, 65(12):9585–9596, 2016.

[51] Jiabao Wen, Jiachen Yang, and Tianying Wang. Path Planning for Autonomous Underwater Vehicles Under the Influence of Ocean Currents Based on a Fusion Heuristic Algorithm. *IEEE Transactions on Vehicular Technology*, 70(9):8529–8544, 2021.

[52] Yuwen Pan, Yuanwang Yang, and Wenzao Li. A Deep Learning Trained by Genetic Algorithm to Improve the Efficiency of Path Planning for Data Collection With Multi-UAV. *IEEE Access*, 9:7994–8005, 2021.

[53] Giovanni Iacovelli and Luigi Alfredo Grieco. Drone Swarm as Mobile Relaying System: A Hybrid Optimization Approach. *IEEE Transactions on Vehicular Technology*, 70(11):12272–12277, 2021.

[54] Lanyong Zhang and Ruixuan Zhang. Research on UAV cloud control system based on ant colony algorithm. *Journal of Systems Engineering and Electronics*, 33(4):805–811, 2022.

[55] Wu Husheng, Li Hao, and Xiao Renbin. A blockchain bee colony double inhibition labor division algorithm for spatio-temporal coupling task with application to UAV swarm task allocation. *Journal of Systems Engineering and Electronics*, 32(5):1180–1199, 2021.

[56] Ronglei Xie, Zhijun Meng, Lifeng Wang, Haochen Li, Kaipeng Wang, and Zhe Wu. Unmanned Aerial Vehicle Path Planning Algorithm Based on Deep Reinforcement Learning in Large-Scale and Dynamic Environments. *IEEE Access*, 9:24884–24900, 2021.

[57] Yuntao Xue and Weisheng Chen. A UAV Navigation Approach Based on Deep Reinforcement Learning in Large Cluttered 3D Environments. *IEEE Transactions on Vehicular Technology*, 72(3):3001–3014, 2023.

[58] Shuai Li, Fan Wu, Siyu Luo, Zhengrong Fan, Jienan Chen, and Shengli Fu. Dynamic Online Trajectory Planning for a UAV-Enabled Data Collection System. *IEEE Transactions on Vehicular Technology*, 71(12):13332–13343, 2022.

[59] Yuwen Qian, Kexin Sheng, Chuan Ma, Jun Li, Ming Ding, and Mahbub Hassan. Path Planning for the Dynamic UAV-Aided Wireless Systems Using Monte Carlo Tree Search. *IEEE Transactions on Vehicular Technology*, 71(6):6716–6721, 2022.

[60] David Silver Volodymyr Mnih, Koray Kavukcuoglu. Human-level control through deep reinforcement learning. *Nature*, page 529–533, 2015.

[61] Shimin Gong, Meng Wang, Bo Gu, Wenjie Zhang, Dinh Thai Hoang, and Dusit Niyato. Bayesian Optimization Enhanced Deep Reinforcement Learning for Trajectory Planning and Network Formation in Multi-UAV Networks. *IEEE Transactions on Vehicular Technology*, 72(8):10933–10948, 2023.

[62] Minah Seo, Luiz Felipe Vecchietti, Sangkeum Lee, and Dongsoo Har. Rewards Prediction-Based Credit Assignment for Reinforcement Learning With Sparse Binary Rewards. *IEEE Access*, 7:118776–118791, 2019.

[63] Fuchen Kong, Qi Wang, Shang Gao, and Hualong Yu. B-APFDQN: A UAV Path Planning Algorithm Based on Deep Q-Network and Artificial Potential Field. *IEEE Access*, 11:44051–44064, 2023.

[64] Zeju Li, Konstantinos Kamnitsas, and Ben Glocker. Analyzing Overfitting Under Class Imbalance in Neural Networks for Image Segmentation. *IEEE Transactions on Medical Imaging*, 40(3):1065–1077, 2021.

[65] Liaqat Ali, Atiqur Rahman, Aurangzeb Khan, Mingyi Zhou, Ashir Javeed, and Javed Ali Khan. An Automated Diagnostic System for Heart Disease Prediction Based on χ^2 Statistical Model and Optimally Configured Deep Neural Network. *IEEE Access*, 7:34938–34945, 2019.

[66] Lu Yi and Man-Wai Mak. Improving Speech Emotion Recognition With Adversarial Data Augmentation Network. *IEEE Transactions on Neural Networks and Learning Systems*, 33(1):172–184, 2022.

[67] Matvey Gerasyov and Ilya Makarov. Dealing With Sparse Rewards Using Graph Neural Networks. *IEEE Access*, 11:89180–89187, 2023.

[68] Jongwan Kim and Hoon Kim. Length of Pseudorandom Binary Sequence Required to Train Artificial Neural Network Without Overfitting. *IEEE Access*, 9:125358–125365, 2021.

[69] Yuntao Xue and Weisheng Chen. A UAV Navigation Approach Based on Deep Reinforcement Learning in Large Cluttered 3D Environments. *IEEE Transactions on Vehicular Technology*, 72(3):3001–3014, 2023.

[70] Amit Konar, Indrani Goswami Chakraborty, Sapam Jitu Singh, Lakhmi C. Jain, and Atulya K. Nagar. A Deterministic Improved Q-Learning for Path Planning of a Mobile Robot. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(5):1141–1153, 2013.

[71] Dongcheng Li, Wangping Yin, W. Eric Wong, Mingyong Jian, and Matthew Chau. Quality-Oriented Hybrid Path Planning Based on A* and Q-Learning for Unmanned Aerial Vehicle. *IEEE Access*, 10:7664–7674, 2022.

[72] Kyriakos G. Vamvoudakis and Nick-Marios T. Kokolakis. 2020.

[73] Meng Zhao, Hui Lu, Siyi Yang, and Fengjuan Guo. The Experience-Memory Q-Learning Algorithm for Robot Path Planning in Unknown Environment. *IEEE Access*, 8:47824–47844, 2020.

[74] Nan Zheng and Pinaki Mazumder. *Fundamentals and Learning of Artificial Neural Networks*, pages 11–60. 2020.

[75] Muhammad Affan, Junaid Jawaid, Syed Umaid Ahmed, Ali Isfand yar Manek, and Riaz Uddin. Solving Combinatorial Problems through Off-Policy Reinforcement Learning Methods. In *2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE)*, pages 1–5, 2020.

[76] Chunxue Wu, Bobo Ju, Yan Wu, Xiao Lin, Naixue Xiong, Guangquan Xu, Hongyan Li, and Xuefeng Liang. UAV Autonomous Target Search Based on Deep Reinforcement Learning in Complex Disaster Scene. *IEEE Access*, 7:117227–117245, 2019.

[77] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. 2018.

[78] Dongbin Zhao, Derong Liu, F. L. Lewis, Jose C. Principe, and Stefano Squartini. Special Issue on Deep Reinforcement Learning and Adaptive Dynamic Programming. *IEEE Transactions on Neural Networks and Learning Systems*, 29(6):2038–2041, 2018.

[79] Liangheng Lv, Sunjie Zhang, Derui Ding, and Yongxiong Wang. Path Planning via an Improved DQN-Based Learning Policy. *IEEE Access*, 7:67319–67330, 2019.

[80] Xinyuan Zhou, Peng Wu, Haifeng Zhang, Weihong Guo, and Yuanchang Liu. Learn to Navigate: Cooperative Path Planning for Unmanned Surface Vehicles Using Deep Reinforcement Learning. *IEEE Access*, 7:165262–165278, 2019.

[81] Nada Kadhim and Monjur Mourshed. A Shadow-Overlapping Algorithm for Estimating Building Heights From VHR Satellite Images. *IEEE Geoscience and Remote Sensing Letters*, 15(1):8–12, 2018.