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.2025.1234567

3D Spatial Information Compression based Deep Reinforcement Learning for UAV Path Planning in Unknown Environments

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 }

Mohammed El-Hajjar would like to acknowledge the support of the Future Telecoms Research Hub, Platform for Driving Ultimate Connectivity (TITAN), sponsored by the Department of Science Innovation and Technology (DSIT) and the Engineering and Physical Sciences Research Council (EPSRC) under Grants EP/X04047X/1, EP/Y037243/1 and EP/X04047X/2.

ABSTRACT In the past decade, unmanned aerial vehicles (UAVs) technology has developed rapidly, while the flexibility and low cost of UAVs make them attractive in many applications. Path planning for UAVs is crucial in most applications, where the path planning for UAVs in unknown, while complex 3D environments has also become an urgent challenge to mitigate. In this paper, we consider the unknown 3D environment as a partially observable Markov decision process (POMDP) problem and we derive the Bellman equation without the introduction of belief state (BS) distribution. More explicitly, we use an independent emulator to model the environmental observation history, and obtain an approximate BS distribution of the state through Monte Carlo simulation in the emulator, which eliminates the need for BS calculation to improve training efficiency and path planning performance. Additionally, we propose a three-dimensional spatial information compression (3DSIC) algorithm to continuous POMDP environment that can compress 3D environmental information into 2D, greatly reducing the search space of the path planning algorithms. The simulation results show that our proposed 3D spatial information compression based deep deterministic policy gradient (3DSIC-DDPG) algorithm can improve the training efficiency by 95.9% compared to the traditional DDPG algorithm in unknown 3D environments. Additionally, the efficiency of combining 3DSIC with fast recurrent stochastic value gradient (FRSVG) algorithm, which can be considered as the most advanced state-of-the-art planning algorithm for the UAV, is 95% higher than that of FRSVG without 3DSIC algorithm in unknown environments.

INDEX TERMS 3D path planning, Deep Reinforcement Learning, 3D Spatial Information Compression, Unmanned Aerial Vehicles, Unknown Environment, Partially Observable Markov Decision Process.

I. Introduction

THE unmanned aerial vehicles (UAVs) technology has developed rapidly, making it essential in many application scenarios such as logistics transportation [1], weather forecasting [2], disaster prevention and control [3], wireless communication [4]–[7] and agriculture [8]. Rotor UAVs have high flexibility, which is reflected in their motion flexibility, and in their ability to configure corresponding components and modules according to the needs of different application scenarios [9]. However, there are still many challenges to the implementation of UAVs. For example, most UAVs fields rely on remote control signals for motion management, which not only limits the flexibility of UAVs but also requires human supervision. In addition, as a limited resource platform, UAVs are in short supply of energy, while efficient and high-performance path planning can improve the working endurance of UAVs. Therefore, path planning and autonomous navigation of UAVs have been actively researched in recent years [10].

A. Background and Motivation

In the past decade, many UAV path planning techniques have emerged. One major category is sampling based static path planning algorithms, such as A-star [11], [12], Dijkstra [13], rapidly-exploring random tree (RRT) [14], [15], probabilistic road-map (PRM) [16], and artificial potential field (APF) [17] algorithms. These sampling based global static path planning algorithms require complete environmental information knowledge and perform well in small-sized 2D environments, but their training efficiency and path planning performance are unsatisfactory when the environment size increases or expands to 3D environments. More importantly, these algorithms are unable to track the dynamic changes in the environment and cannot complete path planning when environmental information is unknown [18]–[20]. The algorithm proposed in [21] can perform path planning in 3D space, and the probability of finding a path is 20% higher than traditional PRM algorithms. *However, although the path planning speed of sampling based techniques has been improved a lot, these techniques cannot meet the real time and reliability requirement of path planning tasks for UAVs in 3D environments, due to the extremely large search space.*

For the path planning problem of UAVs in 3D environments, some biologically inspired optimization algorithms perform better than sampling based algorithms. For example, [22] proposed a 3D path planning algorithm for UAVs based on ant colony optimization (ACO) and it was shown to be effective in disaster prevention and control scenarios. Furthermore, a genetic algorithm (GA) based UAVs 3D path planning algorithm was proposed in [23], which considers both 3D UAVs paths and multi UAV conflict problems, which showed high robustness in discrete 3D space.

Deep reinforcement learning (DRL) is considered one of the promising algorithms for solving UAV path planning problems [36]. Among all the DRL algorithms, Deep Q-Network (DQN), actor-critic (AC), and deep deterministic policy gradient (DDPG) are three commonly used algorithms for solving UAV path planning problems, especially when path planning tasks are considered as Markov decision process (MDP). An improved DQN algorithm was proposed in [37], which introduced an empirical value evaluation network to effectively eliminate path planning drift and improve the accuracy of path planning. The introduction of experience replay [38] has improved the network convergence stability of DRL algorithms such as DQN, DDPG, and AC, while introducing the problem of key experience sparsity when using DRL algorithms for path planning tasks [39]–[41]. A cumulative reward model was proposed in [31] to reduce the network divergence caused by sparse rewards, and also introduced a region segmentation algorithm to further reduce the possibility of intelligent agents trained by multiple DRL algorithms falling into local optima.

There are still challenges in applying the DRL algorithm to the UAV path planning in 3D environments. For example, there are the challenges of low reliability in complex environments, low training efficiency in large environments, and inability to face rapidly changing dynamic environments [10].When the size of the 3D environment increases, the search space of the UAVs will exponentially increase, which undoubtedly poses a challenge to the early random exploration of DRL algorithms. In addition, UAVs have a larger action space and more decision branches in 3D environments compared to 2D environments, which leads to problems such as low training efficiency. Furthermore, the low training efficiency makes it difficult for DRL based UAV agents to face rapidly changing dynamic obstacles [10], [28], [34]. Researchers have made great efforts in the path planning problem of UAVs in 3D environments. For example, [24] proposed an improved sparse A-star algorithm, which has been proven to be capable of 3D path planning for UAVs. A 3D tangent (3D-TG) method based on obstacle geometry information was proposed in [25], which can complete UAV path planning in a 3D urban environment. In [29], a high efficient state decomposition deep deterministic strategy gradient algorithm has been proposed, which has improved convergence rate, navigation performance, and generalization ability compared to traditional DDPG algorithms. In [34], a 3D spatial information compression (3DSIC) algorithm was proposed and combined with DDPG, which highly improved the path planning efficiency by compressing the 3D environment into a 2D environment. *However, the original 3DSIC algorithm need to acquire environmental information for spatial information compression. This cannot be easily obtained when facing unknown 3D environments.*

As mentioned above, another challenge is the uncertainty of the environment. The path planning of UAVs in unknown environments is considered a partially observable Markov decision process (POMDP), and RL is considered one of the best solutions [26], [35]. However, there are not many studies considering using DRL methods to solve POMDP problems in unknown 3D environments. Nguyen [33] proposed a random finite set track-before-detect (TBD) multi-object filter to solve the POMDP problem in UAV path planning. Ragi [26] proposed a POMDP framework to solve the navigation problem of UAVs in 2D unknown wind affected environments. Although [35] proposes a fast recurrent stochastic value gradient (FRSVG) algorithm that can achieve path planning in unknown 3D environments through the value gradient update with recurrent networks, it does not process spatial information to reduce the redundancy of spatial information. Therefore, the training efficiency of the FRSVG algorithm in the POMDP environment can be further improved by combining it with the 3DSIC algorithm [34] extended to the PODMP environment. Additionally, unlike DDPG which updates the network through the policy gradient, FRSVG can be a typical case of the value gradient algorithm in 3D unknown continuous environment.

At the time of writing, research on the path planning problem of UAVs in 3D unknown environments is still very

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

	our paper	[24]	[25]	[26]	[27]	[28]	[29]	[30]	[31]	[32]	[33]	[34]	[35]
Partially Observable Markov Decision Process	✓			✓						✓	✓		✓
Deep Reinforcement Learning	✓			✓		✓	✓		✓	✓		✓	✓
Information Compression	✓		✓									✓	
3D environment	✓	✓	✓		✓	✓	✓	✓		✓		✓	✓
Continuous Workspace	✓		✓		✓	✓	✓	✓	✓		✓	✓	✓

limited and mainly focus on improving algorithms rather than information processing.

In this work, we propose a 3D spatial information compression (3DSIC) method that utilizes the topological features of spatial information for compression and simplification, reducing information redundancy while retaining key information of 3D POMDP environment. Additionally, we introduced an emulator to record and update the environmental information collected by the UAV through the sensor system, while making the system satisfy the POMDP attributes and performing the 3DSIC through the emulator. More specifically, the Monte Carlo simulation in the emulator is conducted to obtain the probability of the correspondence between observations and different states, for solving the uncertainty problem of states and observations in the POMDP environment. Then, the latest environmental information in the emulator is used to compress the spatial environment with the 3DSIC algorithm, and obtain the compressed 2D observed environment. This simplifies the navigation problem in the unknown 3D space into a known environment of 2D path planning problem.

B. Contributions

The contributions of our paper are compared to the literature in Table 1. Our contributions can be summarized as follows:

- We extend the 3DSIC proposed in [34] to accommodate the features of POMDP, enabling it to be applied in completely unknown 3D environments, and achieving higher online training efficiency compare with DRL algorithms without 3DSIC.
- We propose the 3DSIC-DDPG algorithm in POMDP environment, which combines 3DSIC and DDPG algorithm. In addition, we combine the FRSVG algorithm with the 3DSIC algorithm under POMDP. During training, UAV agents consider both compressed 3D spatial information and obstacle distribution information at the current flight altitude, enabling efficient path planning training in a cluttered 3D space.
- We use an emulator established based on the ranging sensor data to replace the belief function. The probability of action choice in the history can be produced by Monte Carlo simulation of the emulator. This eliminates the permanent impact of bad decisions in the belief function on training and improves the success rate of path planning.
- The simulation results show that the training efficiency of 3DSIC-DDPG is 95.9% higher than that of traditional DDPG in the 3D POMDP environment, and the training efficiency of 3DSIC-FRSVG is 96.5% higher than that of traditional FRSVG in the 3D POMDP environment.

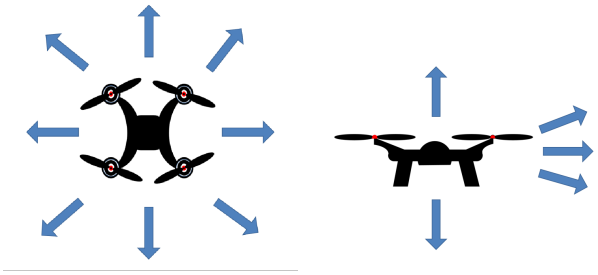
The rest of the paper is organized as follows. In Section II we introduce the system model. In Section III, we introduce the 3DSIC algorithm, and the 3DSIC-DDPG algorithm in partially observable environment. In Section IV, we demonstrate the feasibility of our proposed method through simulation results and compare with state-of-the-art. In Section V, we offer our conclusions and propose potential future directions.

II. System Model and Problem Formulation

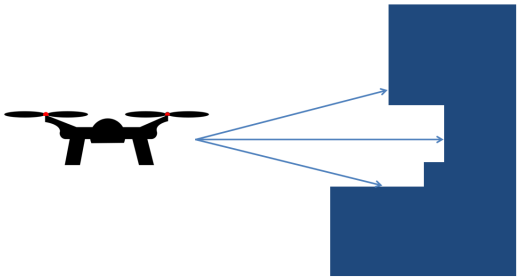
A. System model

In our proposed system, when the destination is given to the UAV, it can plan a collision-free path from the current location to the destination in a completely unknown environment without any additional instruction. In this paper, we consider the UAV path planning problem as POMDP continuous 3D environments. A discrete 3D environment is a 3D mesh based on division of a real 3D working environment, dividing space into many fixed sized cube nodes. The motion of UAVs is seen as a state-transition process within these cube nodes. Each cube node has its 3D coordinates and sorting number. Then, we transfer the discrete 3D environment distribution to a continuous 3D space as we explain in Section III.

The continuous 3D space directly uses the 3D workspace as the emulator model, where the UAV directly uses the current 3D coordinates to represent the state $S_t(x_t, y_t, z_t)$ and complete interaction with the entire state space, where x_t, y_t and z_t are the 3D coordinates of the UAV at time t. The motion and flight control in continuous 3D space are very complex, which are not the focus of this study. In this contribution, the horizontal flight speed, the horizontal heading angle, and vertical flight speed are used to represent the action space $a(V_t^h, V_t^v, \vartheta_t)$, where V_t^h is the horizontal flight speed and its value is limited to (0, 2m/s), V_t^v is the vertical flight speed and these speeds have 3 option values (-1m/s, 0, 1m/s), while ϑ_t is the horizontal heading angle



(a) Schematic diagram of distance measuring sensor distribution



(b) Schematic diagram of distance measuring sensor situation

FIGURE 1: Schematic diagram of the distribution of distance measuring sensors on the UAV.

satisfying $\vartheta_t \in (0, 2\pi)$ [35]. Hence, the state update of the UAVs in a continuous 3D environment can be represented by the following formula:

$$\begin{cases} x_{t+1} = x_t + V_{t+1}^h \times \cos(\vartheta_{t+1}) \times \Delta t \\ y_{t+1} = y_t + V_{t+1}^h \times \sin(\vartheta_{t+1}) \times \Delta t \\ z_{t+1} = z_t + V_{t+1}^v \times \Delta t \end{cases}. \tag{1}$$

In a 3D unknown environment, the UAV only has the coordinate information of itself and the target point, without the distribution information of obstacles in the surrounding environment. UAVs observe the surrounding environment information through a group of measuring sensors. The sensor group includes 26 ranging sensors, one sensor directly above and one sensor directly below the UAV. Eight ranging sensors are set at every $\pi/4$ rad angle in the horizontal direction of the UAV, and 16 ranging sensors are set at each horizontal position with $\pi/12$ rad elevation and depression angles. The sensor layout diagrams are shown in Fig. 1a.

The observation set O_t include the obstacle coordinates calculated from the distance sensor data. As shown in Fig. 1b, the coordinates of obstacles in the horizontal direction can be directly calculated using the current drone coordinates and ranging sensor values. The sensor values with elevation and depression angles need to be calculated using trigonometric functions to determine the relative height and distance. The calculation process can be represented as follows:

$$\Delta h = L \times \sin(\pi/12), \tag{2}$$

$$\Delta d = L \times \cos(\pi/12), \tag{3}$$

where Δh is the relative height, Δd is the relative distance and L is the ranging sensor value.

B. Energy Consumption Model

The energy consumption model of the UAV is important to determine the reward model and energy cost penalty. Yan et al. derived a simplified energy consumption model for UAVs in [42], which can be used to calculate both the vertical and horizontal motion energy consumption, where the energy consumption of UAVs in the vertical motion can be expressed as:

$$P_{vertical}(t) = P_0 \left(1 + \frac{3\|\mathbf{V}_i + \mathbf{a}t\|^2}{U_{tip}^2}\right) \tag{4}$$

$$+ \; P_1 \left(\sqrt{1 + \frac{\|\mathbf{V}_i + \mathbf{a}t\|^4}{4v_0^4}} - \frac{\|\mathbf{V}_i + \mathbf{a}t\|^2}{2v_0^2}\right)^{\frac{1}{2}}, \tag{5}$$

where $\mathbf{V}_i$ is the initial velocity and a is the acceleration or deceleration, P_0 and P_1 are functions of the speed related to the physical properties of the UAV and the flight environment, U_{tip} represents the tip speed of the rotor blade and the mean rotor induced velocity is denoted as v_0. In the horizontal direction, the power consumption at a speed of $\mathbf{v}(t)$ is

$$P_{horizontal}(t) = \|\mathbf{F}\|\|\mathbf{v}(t)\|, \tag{6}$$

where $\mathbf{F}$ is the pulling force, $\mathbf{v}(t) = \|\mathbf{V}_i + \mathbf{a}t\|$ is the instantaneous velocity at time t . When flying at a constant speed of V, $\|\mathbf{F}\| = d_0 \rho s A V^2/2$ equals to the fuselage drag $D = \rho S V^2/2$, where S is the rotor solidity, A is the rotor disc area, and ρ is the air density.

C. Problem Formulation

In this 3D simulation environment, the path planning result of the 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 coordinate in the path, P_{last} represents the last node coordinate 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 location, S_{end} is the destination coordinate, and $\boldsymbol{Obs}$ is the set of all obstacles.

Explicitly, we consider the path planning problem of the UAV as a POMDP, which includes several main parts: state set $\boldsymbol{S}$, action set $\mathbf{A} = a_t$, transfer function $T(s'|s, a)$, observing space $\boldsymbol{Z}$, observation probability $\boldsymbol{O}$ and reward function $R(s'|s, a)$. Note that $\boldsymbol{O}$ is the probability of observing certain environmental information after performing a specific action in a specific state. Although POMDP introduces uncertainty, the Bellman equation can be used to solve the optimal strategy of the Markov process [43], [44]. Therefore, the

DDPG algorithm is also applicable for solving continuous POMDP problems.

There are four neural networks in the DDPG algorithm, which are actor network, critic network, target actor network, and target critic network [29]. The training process of the DDPG agent is to continuously update these four networks, making the weights of the target action network approach the direction of optimal policy. As mentioned earlier, the update of the target critic network is achieved through the Bellman equation. The solution of the Bellman equation is based on the optimal cumulative expectation $V^*(s)$, which can be expressed as follows [45]:

$$V^*(s) = \max_{\theta^\mu} E\left[\sum_{t=0}^{t=n} \gamma^t R(S_{t+1}, A_t, S_t|\theta^\mu, s)\right], \quad (7)$$

where γ is the discount factor, t is the current time index, θ^μ is the actor network weight. 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:

$$Q(s_t, a_t) = r(s,a) + \gamma Q(s_{t+1}, \theta^\mu(s_{t+1})), \quad (8)$$

where $r(s,a)$ is the reward value of action a at state s, $\theta^\mu(s_{t+1})$ is the action output of the target actor network, which can be considered as a_{t+1} as well. Then we need to define a loss function to determine the direction of the critic network update. The Q-value output by the critic network is expected to be close to the actual value obtained by the system, so we usually use the time difference mean square error of the two as the loss function, which can be expressed as:

$$L(\theta^Q) = \frac{1}{N}\sum_i \left(y_i - Q(s_i, a_i|\theta^Q)\right)^2, \quad (9)$$

where N is the mini-batch size, i is the index for the number of experience, θ^Q is the weight of the critic network, and y_i is the practical value of the experience.

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 [46].

Additionally, DDPG also requires agents to conduct large-scale random explorations in the early stages of training, and update the neural network through the interaction experience generated by these random explorations and the environment. In the DDPG algorithm, the random exploration of UAV agents is achieved by adding noise to the actions output by the actor network:

$$a_t = \mu(s_t|\theta^\mu) + \mathcal{N}, \quad (10)$$

where $\mathcal{N}$ is the noise that follows a Gaussian distribution with a mean of 0 and variance of 0.2 to 0.05, and θ^μ is the network weight for actor network μ.

The update of the actor network is achieved by maximizing the Q value of the critic network. Therefore, its objective function can be expressed as:

$$J(\theta^\mu) = \frac{1}{N}\sum_i Q(s_i, \mu(s_i|\theta^\mu)|\theta^Q), \quad (11)$$

where θ^μ is the actor network weight. Then, we can use gradient ascent to update the actor network as:

$$\nabla_{\theta^\mu} J \approx \frac{1}{N}\sum_i \nabla_a Q(s,a|\theta^Q)\nabla\theta^\mu \mu(s|\theta^\mu)\big|a = \mu(s). \quad (12)$$

The update of the target network is achieved by replicating the actor network and critic network through soft updates after a certain period of time. Its specific expression is as follows:

$$\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'} \quad (13)$$

$$\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}, \quad (14)$$

where τ is the soft update coefficient, which satisfies $\tau \in (0,1)$.

When solving POMDP problems the agent can only infer the current state based on the observation information z_0 to z_t of all historical actions a_0 to a_t, because it cannot obtain accurate state information. This history is described as $h_t = z_0, z_1 \dots z_t | a_0, a_1 \dots a_t$. From this, we can define the state function $V^*(h)$ and value function $Q^*(h,a)$ in POMDP. It is not difficult to find that h_t needs to traverse all histories in order to obtain the state. As time t increases, the size of h_t will become very large, and the effectiveness of the information will significantly decrease. To solve this problem, a probability distribution $b(s)$ of the state space is usually defined, which can be defined more specifically as [47] :

$$b_t(s) = Pr(s_t = s|h_t, b_0), \quad (15)$$

where b_0 is the distribution of the initial state. (15) represents the probability of being in state s after passing through history h_t under the initial distribution b_0. Then, we can derive the state value and Q-value function expressions in PODMP problem as follows:

$$V^*(b) = \max_\pi \left[R(b_t, a_t) + \gamma \sum Pr(z_t|b_t, a_t)V^*(b_{t-1}(a,z))\right], \quad (16)$$

$$Q^*(b_t, a_t) = R(b_t, a_t) + \gamma \sum Pr(z_t|b_t, a_t)V^*(b_{t-1}(a,z)), \quad (17)$$

where $b_{t-1}(a,z)$ is the belief transition from observation z when executing action a to BS b_{t-1}.

III. Proposed Information Compression and Path Planning Techniques

In this section, our proposed 3DSIC algorithm will be introduced in detail. Furthermore, the 3DSIC-DDPG algorithm will be introduced as a scheme for applying the 3DSIC algorithm in continuous POMDP 3D space.

A. 3D Spatial Information Compression in POMDP environment

In [34], we proposed a 3DSIC algorithm, which can be combined with DDPG algorithm to achieve UAV path planning in a 3D environment. The proposed techniques in [34] showed higher training efficiency than the traditional DDPG. However, the original 3DSIC algorithm proposed in [34] does not consider the POMDP environment, this means that the original 3DSIC algorithm cannot handle the uncertainty between observations and states in POMDP environments, and relies on known global 3D spatial information which can not be obtained in a 3D PODMP environment. Thus, in this paper, we extend the 3DSIC to a POMDP environment by introducing an emulator to store and update the observed 3D spatial environment. Meanwhile, the emulator can also adapt to the uncertainty of the POMDP environment through Monte Carlo simulation in the emulator.

In the POMDP environment, the UAV initially has no knowledge of any environmental information and only perceives the environment through distance sensors. In this case, we consider first establishing a discrete emulator for the surrounding environment of the UAV. The establishment and update of an emulator require the collection of environmental information. In this study, we consider using only ranging sensors to collect environmental information for establishing and maintaining the emulator. In the scenario shown in Fig. 2, the heading angle of the drone's movement is θ_t^1, and the reading of a horizontal ranging sensor is l_t^n, with an azimuth angle of θ_t^2. The update of the obstacle coordinates in the horizontal direction can be represented as follows:

$$\begin{cases} x_{ob} = x + l_t^n \times \cos\left(\theta_t^1 + \theta_t^2\right) \\ y_{ob} = y + l_t^n \times \sin\left(\theta_t^1 + \theta_t^2\right) \\ z_{ob} = z \end{cases}, \tag{18}$$

where x_{ob}, y_{ob} and z_{ob} are the 3D coordinates of the detected obstacle, x, y and z are the current coordinate of the UAV. If the distance sensor has an elevation angle of θ_t^3, updating the obstacle coordinates is slightly more complex and can be expressed as:

$$\begin{cases} x_{ob} = x + l_t^n \times \cos\left(\theta_t^3\right) \times \cos\left(\theta_t^1 + \theta_t^2\right) \\ y_{ob} = y + l_t^n \times \cos\left(\theta_t^3\right) \times \sin\left(\theta_t^1 + \theta_t^2\right) \\ z_{ob} = z + l_t^n \times \sin\left(\theta_t^3\right) \end{cases}. \tag{19}$$

It is worth noting that our simulation system did not take into account the pitch angle during the UAV flight, but it is necessary to consider the pitch angle during the UAV flight when conducting experiments on real-world UAVs. In our simulation system, the body of drone is assumed to maintain horizontal during flight, which means that the body of the UAV is never tilted, and therefore the range sensor will not have any additional tilt angle. However, in the real world, rotary wing drones have a certain inclination angle during flight, which can cause the range sensor and the fuselage to tilt together. Therefore, when updating the emulator, it is necessary to consider the impact of its body tilt on the range sensor. When deploying this model on real-world UAVs, sensor fusion can be used to correct the offset caused by the pitch angle for the ranging sensors based on the flight attitude and pitch angle calculated by the gyroscope and accelerometer [48], [49].

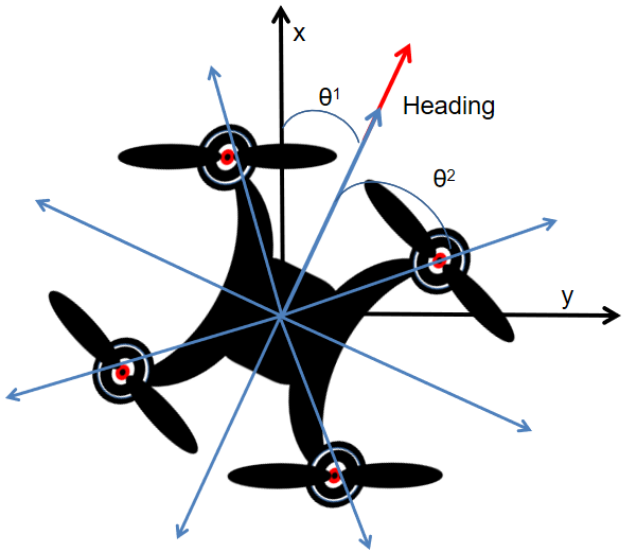


FIGURE 2: Schematic diagram of UAV ranging sensors with heading angle.

For the meshed 3D space in the emulator, the spatial information is first set to binary, where the non-obstacle node is set to 0, and the obstacle node is set to 1. However, in the POMDP environment we are considering, the action space and state space of the drone are both continuous. Therefore, when the collected environmental information shows that there are obstacles within the cube represented by a discrete node, this node is considered an obstacle node in the emulator. If no obstacles are detected within the cube represented by a discrete node, the node will be updated as a free node. Then the 3D space in the emulator is cut in the direction perpendicular to the Z axis to obtain multiple 2D planes with the same size as shown in Fig. 3. It is worth noting that during initialization, the entire environment is unknown and all nodes will be initialized as obstacle nodes. When the UAV first obtains observation data, it starts updating environmental information. Therefore, when compressing information, unexplored unknown areas will also be treated as obstacles.

Each of the resultant two dimensional planes represents the distribution of obstacles at its height. The spatial information of these planes is two-dimensional, where the $m \times n$ two-dimensional plane space information can be expressed as:

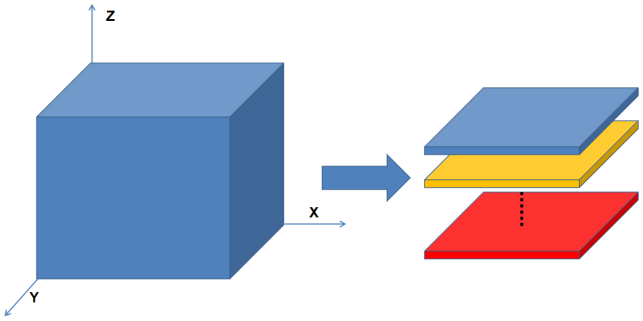


FIGURE 3: Schematic diagram of 3D space cutting.

$$\mathbf{S}_i = \begin{pmatrix} S_{i,11} & \cdots & S_{i,1n} \\ S_{i,21} & \cdots & S_{i,2n} \\ \cdot & \cdot & \cdot \\ S_{i,m1} & \cdots & S_{i,mn} \end{pmatrix}, i \in (1, 2, \ldots, h), \tag{20}$$

where i is the number of the plane, $S_{i,11}$ presents the cube node of plane i, the 2D coordinate of a plane $X = 1$ and $Y = 1$ is free space or obstacle. If the value of $S_{i,11}$ equals to 1 then this node is obstacle and if the value of $S_{i,11}$ equals to 0 that means node $S_{i,11}$ is free space. Then, we can reduce the dimension of 2D information of each plane, and convert these 2D information into row vectors as follows:

$$\mathbf{S}_i = \begin{pmatrix} S_{i,11} & \cdots & S_{i,1n} \\ S_{i,21} & \cdots & S_{i,2n} \\ \cdot & \cdot & \cdot \\ S_{i,m1} & \cdots & S_{i,mn} \end{pmatrix} \Rightarrow \left(S_{i,1} \cdots S_{i,n} \cdots S_{i,mn}\right). \tag{21}$$

Taking the 3D space of $m \times n \times h$ as an example, the spatial information of each 2D plane after dimension reduction can be combined to obtain the 3D spatial information $\mathbf{S}$, which can be expressed as:

$$\mathbf{S} = \begin{pmatrix} S_{1,1} & \cdots & S_{1,n} & \cdots & S_{1,mn} \\ S_{2,1} & \cdots & S_{2,n} & \cdots & S_{2,mn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ S_{h,1} & \cdots & S_{h,n} & \cdots & S_{h,mn} \end{pmatrix}. \tag{22}$$

Here, the reduced dimension 3D information has not been compressed. If we conduct information compression now, the compressed information will contain the obstacle distribution of all flight height. When planning the path of the UAV, we only want to consider the obstacle distribution information of the layers adjacent to the current flight altitude layer of the UAV. Therefore, we need to introduce a compression field to limit the range of the 3DSIC method.

Since the spatial information is discrete, the compressed field also needs to be discrete. For example, if we only consider the current layer of the UAV flight altitude and the upper and lower 2 layers adjacent to the current layer, the compression field will have 5 elements. In the Section IV, we will study the effect of the number of layers to compress. The compression field can be expressed as a row vector β, whose dimension is determined by the number of layers to be compressed. For example, with 5 layers compression, β can be represented as:

$$\beta = \begin{pmatrix} \rho_1 & \rho_2 & \rho_3 & \rho_4 & \rho_5 \end{pmatrix}, 0 \leq \rho_i \leq 1, \sum_{i=1}^{5} \rho_i = 1, \tag{23}$$

where ρ_i is the compression coefficient of the i-th layer from bottom to top in the compression field.

If we do the dot product operation for the compression factor β and the spatial information of the corresponding layer after dimension reduction S, we can get:

$$\begin{aligned} \mathbf{S}_c &= \beta \cdot \mathbf{S} \\ &= \begin{pmatrix} \rho_1, \rho_2, \rho_3, \rho_4, \rho_5 \end{pmatrix} \cdot \begin{pmatrix} S_{11} & \cdots & S_{1n} & \cdots & S_{1mn} \\ S_{21} & \cdots & S_{2n} & \cdots & S_{2mn} \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ S_{51} & \cdots & S_{5n} & \cdots & S_{5mn} \end{pmatrix} \\ &= \begin{pmatrix} S_c(1) & S_c(2) & \ldots & S_c(n) & \ldots & S_c(mn) \end{pmatrix}, \end{aligned} \tag{24}$$

where $\mathbf{S}_c(i)$ is the compressed spatial information element. For the example of compressing 5 layers, it can be expressed as:

$$\begin{aligned} S_c(i) = \rho_1 \times S_{1i} + \rho_2 \times S_{2i} + \rho_3 \times S_{3i} + \\ \rho_4 \times S_{4i} + \rho_5 \times S_{5i}, i \in (1, 2, \ldots, mn). \end{aligned} \tag{25}$$

Here, if the compressed row vector is restored to two-dimensional plane information, the two-dimensional plane information is 3D spatial information weighted by the compression factor weight of the current layer and the adjacent upper and lower four layers of spatial information. Its physical meaning can be explained as follow: if the value of a node is not 1, it means that there must be a layer in the 3D space adjacent to the current layer that can pass through the node without collision. If a node value is 1, it means that the current layer and adjacent layers cannot pass through the node.

The original 3D environment information may lose spatial information after being compressed into 2D environment using the 3DSIC. The information entropy of the information loss changes according to the size of the compressed environment and is affected by the probability of obstacle distribution in the environment. The loss of information entropy after applying the 3DSIC is shown in Fig. 4, where we consider the actual size of the compressed environment to be 500 times of the square of the environment size value shown in the figure. For example, if the size of the environment is 3×3, the actual environment size compressed is $3^2 \times 500$, since the resolution is 0.1 for the x and y axis, and the number of layers to compress is 5. We can see that the loss of information increases with the size of the environment, and at the same time, the information entropy caused by compression is the highest when the

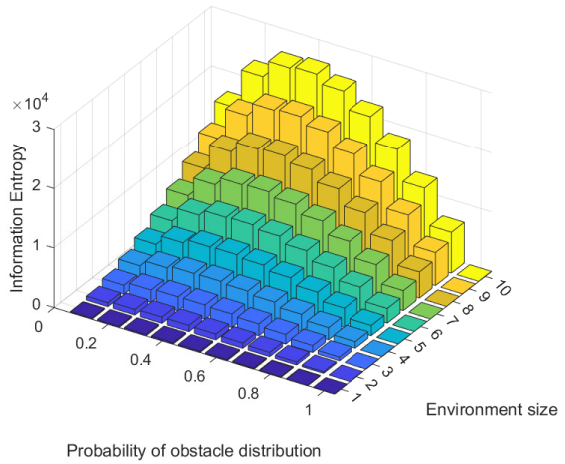


FIGURE 4: Information entropy loss after 3DSIC.

probability of obstacle distribution is 0.5 as shown in Fig. 4. Considering that obstacles in urban environments are mainly made up of buildings, due to gravity, the vertical changes of obstacles are relatively small, so the information loss caused by compression is not significant.

B. 3DSIC-DDPG in Continuous POMDP Environment

The output of the 3DSIC algorithm is a 2D normalized map. This map is discrete, so we can directly combine 3DSIC and DDPG algorithms to use the compressed normalized discrete map as the weight coefficients for the DDPG reward model and penalty value model. Therefore, the final reward value for each state transition can be expressed as:

$$R(s_t, a_t, s_{t+1}) = (1 - C(s_{t+1})) \times r(s_t, a_t) + C(s_{t+1}) \times (p_e(a_t) + p_{col}(s_{t+1})), \quad (26)$$

where $R(s_t, a_t, s_{t+1})$ is the final reward value of transition (s_t, a_t, s_{t+1}, R), $C(s_{t+1})$ is the value at position s_{t+1} in compressed normalized discrete map, $p_e(a_t)$ is the motion energy cost punishment of action a_t, and $p_{col}(s_{t+1})$ is the collision punishment of position s_{t+1} in the original 3D map.

In a 3D continuous environment, we can also divide the 3D environment into many equally sized cube nodes, and then use these nodes to compress 3D spatial information. However, since the workspace and action space are both continuous, the coordinates of the UAV may not be integers. Therefore, for each node, there is a corresponding range $C_{range}(x)$ and $C_{range}(y)$, and when the current coordinates of the UAV are in the range of node (i, j), that is when $x_t \in C_{range}(x_i)$ and $y_t \in C_{range}(y_j)$, the UAV is considered to be in node cube (i, j). Similarly, obstacles in a continuous environment can also be stored in a discrete emulator. When any detected obstacle exists inside a discrete cube, it is considered an obstacle cube. The state-action history of the drone is obtained by sampling on a continuous timeline, and the state information of each sampling point includes the current 3D coordinates, heading angle, horizontal velocity, and vertical velocity of the drone. This enables discrete emulator to store the history of continuous state-actions of drones in continuous 3D space and the distribution of obstacles in the searched environment. It is worth noting that we need to set the duration Δt of each action to precisely allow the UAV to switch altitude to the integer z index of discrete 3D environment, which can be presented as $V^h_{t+1} \times \Delta t = \Delta z_{index}$.

When the environment is considered as a 3D PODMP environment, obtaining the compressed state is still difficult, although the compressed map is also probability distribution driven. The conventional solution is to use BS $b_t(s)$ to represent the probability of being in state s in the current observation history h_t. After using the 3DSIC, there are not many possible combinations of s and h_t due to the compression of reward space and interaction space to 2D. Hence, we propose an independent emulator that updates the environmental information in the emulator based on all observation histories, where the emulator is an agent modeled environment based on observation information. After compression, we use Monte Carlo simulation of the current action a_t and observation history information h_t in the simulator to represent the distribution of BS for state s, thereby avoiding directly calculating BS $b_t(s|h_t, b_0, a_t)$. More specifically, it avoids using a huge table to store historical observation results. It can be observed that when the number of Monte Carlo simulations is large enough, the probability distribution and belief distribution obtained from the simulation are roughly equal. Therefore, we can directly use the probability distribution obtained by Monte Carlo simulation of the emulator as the observation probability o_t, and use the corresponding historical h_t, action a_t, and reward value R_t as state transition experiences into the experience pool, which is (o_t, h_t, a_t, R_t). The framework of the 3DSIC-DDPG algorithm in POMDP environment is shown in Fig. 5. Unlike traditional DDPG, the observation data and experience obtained by the UAV agents interacting with the environment will not be directly stored in the experience pool or used for training. The observation data are first used to build the emulator, which does not need to include all environmental information, but only need to model the environment based on historical observations. Then, when new experience arises, Monte Carlo simulation is first performed through the emulator to obtain the probability of certain actions occurring. With the emulator, we can use Eq. (17) to update the Q values without using table or linked lists to save history.

By conducting Monte Carlo simulations in the emulator, we can easily obtain the mapping distribution of observation history, states and actions. The belief state obtains a mapping of uncertainty between observations and states by introducing the belief function $b(\cdot)$ into the uncertain properties of the POMDP, while the emulator obtains an approximation of the uncertainty mapping between states and observations through Monte Carlo simulation and introduce it into the

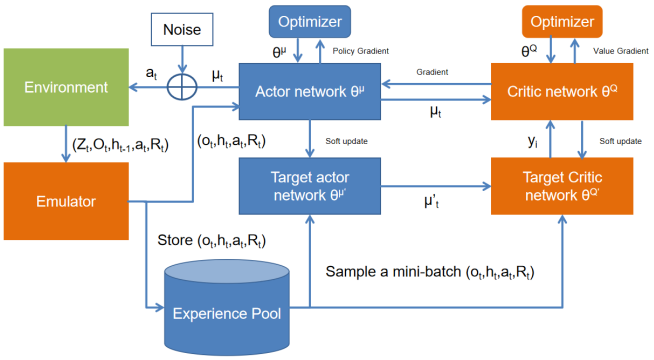


FIGURE 5: Framework Diagram of 3DSIC-DDPG in POMDP environment.

loss function in the POMDP environment. It is not difficult to obtain its representation in POMDP by replacing the state s in the update equation of the critic network:

$$L = \frac{1}{NT}\sum_i\sum_t\left(y_i^t - Q\left(h_i^t, a_i^t\right)\frac{\partial Q\left(h_i^t, a_i^t\right)}{\partial\theta}\right), \quad (27)$$

where t is the time index and h_i^t is the history from the emulator of experience i. Then, the actor network can be updated with the state s replaced by history h as well. It can be expressed as:

$$\nabla_{\theta^u} J = \frac{1}{NT}\sum_i\sum_t \frac{\partial Q\left(h_i^t, \theta^\mu\left(h_i^t|\theta^{Q'}\right)\right)}{\partial\theta}\frac{\partial\theta^\mu\left(h_i^t|\theta^{\mu'}\right)}{\partial\theta}. \quad (28)$$

The pseudo-code of 3DSIC-DDPG is shown in Algorithm 1. It is worth noting that in a continuous 3D POMDP environment, each $C(\cdot)$ records compressed value of each node and a specific range constraint for each node, so the node value $C(h_t)$ can be determined by the x and y coordinates of h_t.

Algorithm 1 DDPG algorithm based on 3D spatial information compression in continuous 3D POMDP environment

Initialize experience replay memory $\mathcal{M}$ to capacity η.
Randomly initialize the actor network $\mu(s|\theta^\mu)$ with weight θ^μ.
Randomly initialize the critic network $Q(s,a|\theta^Q)$ with weight θ^Q.
Copy networks μ and Q to get target networks μ' and Q'.
for episode = 1 to M, **do**
  Initialize a random process $\mathcal{N}$ for action exploration.
  Initialize compressed normalized map by execute 3DSIC algorithm with β and $h_1(z)$.
  for t = 1 to T, **do**
    Select action $a_t = \mu(h_t|\theta^\mu) + \mathcal{N}_t$ and execute a_t to get observation o_{t+1}.
    if $h_{t+1}(z) \neq h_t(z)$ **then**
      Update compressed normalized map by execute 3DSIC algorithm with β and $h_{t+1}(z)$.
    end if
    Calculate reward $R_t = (1 - C(i,j)) \times r(h_t, a_t) + C(i,j) \times (p_e(a_t) + p_{col}(h_{t+1})), x_{t+1} \in C_{range}(x_i)$ and $y_{t+1} \in C_{range}(y_j)$
    Save the transition set (o_t, h_t, a_t, R_t) into $\mathcal{M}$.
    Sample random mini-batch I from $\mathcal{M}$
    Set $y_i^t = R_i^t + \gamma Q'(h_i^{(t+1)}, \mu'((h_i^{t+1}|\theta^{\mu'})|\theta^{Q'})$.
    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 after K steps:
    $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
    $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
  end for
end for

IV. Simulation Results

In this section, computer simulations are used to validate, and analyze the algorithms proposed in this paper. The simulation environment is considered as a 3D POMDP environment, and we will test the impact of combining our proposed 3DSIC and policy gradient algorithm DDPG on training efficiency and path planning performance in POMDP environment. In addition, we will also combine 3DSIC with FRSVG [35] to test the impact of 3DSIC algorithm on the value gradient algorithm. Our benchmark schemes for comparison are the traditional DDPG algorithm and the traditional FRSVG algorithm.

A. Training efficiency of DDPG agent in complex 3D POMDP environment

We consider using complex urban environments as simulation environments, which have complex 3D obstacle layouts and are POMDP environments for UAVs that rely on sensors to perceive environmental information. Hence, we chose a block near the Museum of London and used a shadow index algorithm [50] to estimate the height of buildings in the environment through satellite images. The satellite image is shown in Fig. 6. Then, a 100-cube 3D environment is modeled based on the result of buildings' distribution and height.

In order to compare the training efficiency and path planning quality of DDPG algorithm using 3DSIC and DDPG algorithm not using 3DSIC, the reward value curve of each agent during the training process is recorded and used for comparison. To avoid the influence of special circumstances, we use 10 different random number seeds to train 10 3DSIC-DDPG agents and 10 DDPG agents. In addition, for fairness, each group of 3DSIC-DDPG and DDPG agents will use the same random number generator seed to ensure similar early random exploration experiences. The quality of path

FIGURE 6: Satellite image of the real experimental environment.

planning will be compared with the total number of actions in the path based on the results of path planning tests after training. A lower number of actions means that the total length of the path is shorter. The hyper-parameters of the neural network training for 3DSIC-DDPG and DDPG are shown in Table 2. In the reward model for all DRL agents, after considering the energy consumption during horizontal and vertical movements of the UAV, as well as the computational cost of updating the 2D map using 3DSIC after altitude changes, the average comprehensive cost per action during horizontal movement is one-fourth of that during vertical movement. Hence, the penalty value for all horizontal movements is -0.5, and the penalty value for vertical movements is -2. This setting reduces the frequency of UAVs changing their flight altitudes. Furthermore, [31] found that the above traditional reward model can cause data sparsity in the experience pool and proposes a cumulative reward model that can effectively solve the problem of sparse rewards caused by traditional reward models. The cumulative reward model in the 2D environment can be represented as:

$$Reward(x, y, l) = \frac{C}{\sqrt{(x - x_d)^2 + (y - y_d)^2} \times D(x, y, l)}, \tag{29}$$

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 could be half of or less than the reward value for arriving at the destination, while D can be expressed as:

$$D(x, y, l) = \frac{1 + Obs(x, y, l)}{S(x, y, l)}, \tag{30}$$

where $S(x, y, l)$ is the measure of area in the adjacent area in size l square of the current coordinate (x, y). The 2D form of the cumulative reward model can be directly applied to compressed 2D environments by setting the value

TABLE 2: Hyper-parameters of the 3DSIC-DDPG and DDPG model.

Parameter	Value	Definition
I	$m + n$	The sampling size of each training, m and n are the size of $m \times n$ environment.
γ	0.95	Discount factor
η	$m \times n$ $\times 50000$	Capacity of experience pool.
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	100000	Maximum number of training episodes
K	10	Steps delay of target networks update
β	(0.1, 0.2, 0.4, 0.2, 0.1)	compress factor

of $Obs(x, y, l)$ to the sum of the node values within the range l of (x, y) nodes in the compressed map, because the compressed value also contains information from all compressed layers. If we consider applying the cumulative reward model to the original 3D environment, we need to extend it to 3D, which can be expressed as:

$$R(x, y, z, l) = \frac{C}{\sqrt{(x-x_d)^2+(y-y_d)^2+(z-z_d)^2} \times D(x,y,z,l)}, \tag{31}$$

where z is the current z coordinate and z_d is the z coordinate of the destination, while the evaluation of $D(x, y, z, l)$ uses the volume to replace the area measurement.

Furthermore, the value calculation of the reward model can take into account both the 3D environment information before 3DSIC and the compressed 2D environment after 3DSIC. Hence, there are 2D and 3D reward evaluation methods for traditional reward and cumulative reward model.

In this experiment, the neural network for 3DSIC-DDPG and DDPG has 32 inputs which are 6 3D coordinates elements and 26 ranging sensor data, and it has 3 outputs corresponding to vertical velocity, horizontal velocity and horizontal direction angle, with two fully connected hidden layers, where each layer contains 256 neurons, then connected with a long-short term memory (LSTM) layer with size of 256 between fully connected layers and output layer. The schematic diagram of the actor network structure is shown in Fig. 7.

Fig. 8 shows the average learning curve of 10 3DSIC-DDPG agents and 10 DDPG agents. The 3DSIC-DDPG agents converge on average at 18626 episodes, while the traditional DDPG agents converge at 39460 episodes. The training efficiency of 3DSIC-DDPG agents is 2.118 times that of traditional DDPG agents. However, during path plan-

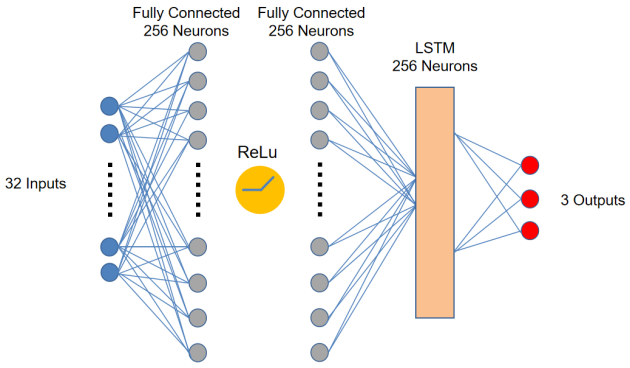


FIGURE 7: Actor neural network structure of 3DSIC-DDPG and DDPG

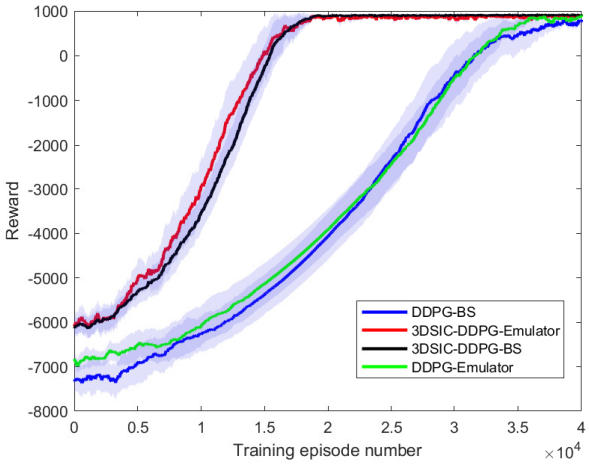


FIGURE 8: Learning curve of 3DSIC-DDPG and DDPG in 3D POMDP Environment

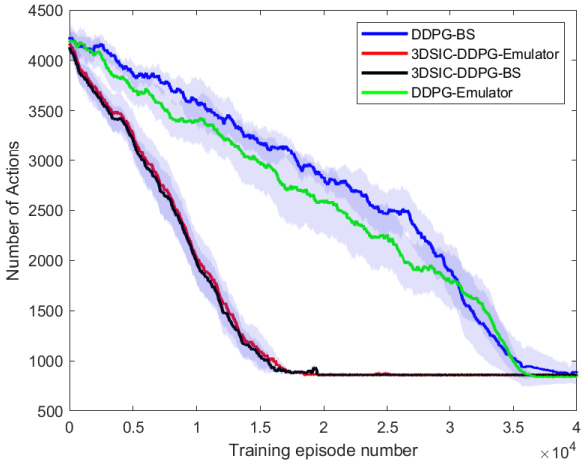


FIGURE 9: Actions curve of 3DSIC-DDPG and DDPG in 3D POMDP Environment

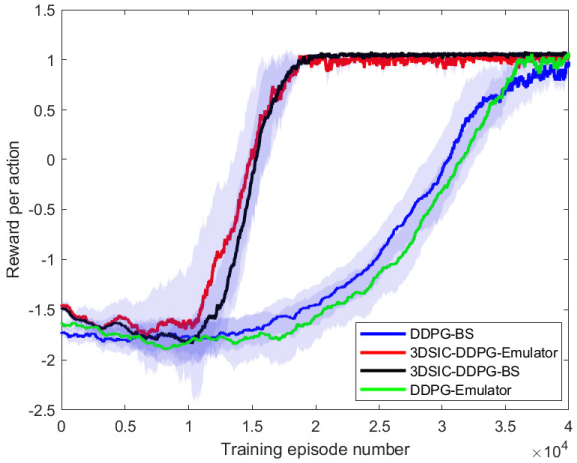


FIGURE 10: Reward per action curve of 3DSIC-DDPG and DDPG in 3D POMDP Environment

ning testing, we found that not all actor networks obtained from the convergence points of 3DSIC-DDPG agents passed the path planning test, indicating that the quality of path planning was lower than expected. Meanwhile, we also noticed that all actor networks trained for a period of time after using converged episodes can pass path planning tests.

From our analysis, we find that the convergence of the critic network is faster than that of the action network, which is described in [51] as an overestimation of the action network. In other words, as long as the actor is given more time, the actor network can converge, or the Twin-delayed deep deterministic policy gradient (TD3) structure can be used to introduce four networks to eliminate overestimation of the actor network [51]. This can greatly increase the complexity of the system. Hence, we consider simultaneously detecting the total number of actions at the end of each episode and drawing action curves to track the convergence of the actor network. In addition, we also used rewards per action (RpA) as a tracking measure for the convergence of the entire system.

The experimental results are shown in Fig. 9 and Fig. 10, which indicate that the actual convergence position of the actor network of 3DSIC-DDPG-Emulator is at 19851 episodes, while the action network of DDPG-BS converges at 38904 episodes. The convergence of the entire system is consistent with that of the actor network, and compared to traditional DDPG, the 3DSIC-DDPG algorithm has a 95.9% improvement in training efficiency. Additionally, the agents using the emulator converge earlier than those using the BS. In theory, using an emulator can reduce the impact of learned experiences on environmental changes, while BS cannot adapt well to environmental changes.

In addition, to verify the actual performance improvement caused by the increase in training efficiency, we designed a

TABLE 3: Probability of collision free path planning for different DDPG algorithm in dynamic environment.

Type of agent	number of collision paths	Speed of dynamic obstacles	Collision-free rate
3DSIC-DDPG	102	0.1 m/s	89.8%
3DSIC-DDPG	109	0.2 m/s	89.1%
3DSIC-DDPG	146	0.5 m/s	85.4%
DDPG	127	0.1 m/s	87.3%
DDPG	174	0.2 m/s	82.6%
DDPG	218	0.5 m/s	78.2%

validation experiment by implementing trained 10 3DSIC-DDPG models and 10 traditional DDPG models in an environment with dynamic obstacles at different rates of change. The dynamic obstacle is a small cube with a size of $1m \times 1m \times 1m$, with a distribution density of 0.5% and movement speeds of $0.1m/s$, $0.2m/s$ and $0.5m/s$. Please note that in this simulation, dynamic obstacles are added in the path planning stage rather than in the training stage. Additionally, note that the 0.5% distribution density is used as an example and any other distribution density value would work. We conducted 1000 simulations for each agent and fixed the pseudo-random generator seeds as 1 to 1000 with the same random number generator type. The simulation results are recorded in Table 3. Based on the simulation results, it can be seen that the collision-free probabilities of 3DSIC-DDPG and the DDPG agents are both close to 90%, when there are obstacles with a motion speed of $0.1m/s$ in the environment. Although the collision-free probability of DDPG agents is only 78.2% when the obstacle movement speed increases to $0.5m/s$, 3DSIC-DDPG can still maintain a collision-free probability of 85.4%.

B. Training efficiency of value gradient agent in complex 3D PODMP environment

In addition to policy gradient updates, there are also value gradient methods for updating deep reinforcement learning. For example, [35] proposed a fast recurrent stochastic value gradient (FRSVG) algorithm. In order to comprehensively test the effectiveness of 3DSIC, we use a combination of 3DSIC and FRSVG for training and compare it with traditional FRSVG to evaluate training efficiency and path planning quality.

Fig. 11 shows the average learning curve of 10 3DSIC-FRSVG agents and 10 FRSVG agents. Similarly, the FRSVG algorithm also experiences overestimation of actors, so we use the actions curve and reward per action curve to represent the convergence process of the system, as shown in Fig. 12 and Fig. 13, respectively. The simulation results show that the average convergence position of 3DSIC-FRSVG agents is 19541 episodes, while the average convergence position of traditional FRSVG agents is 38417 episodes. Compared

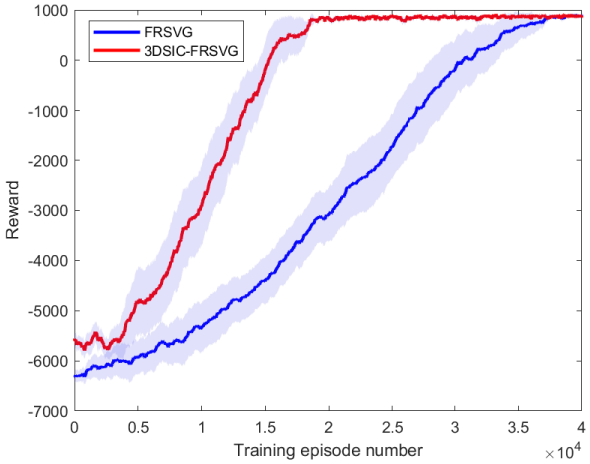


FIGURE 11: Learning curve of 3DSIC-FRSVG and FRSVG in 3D POMDP Environment

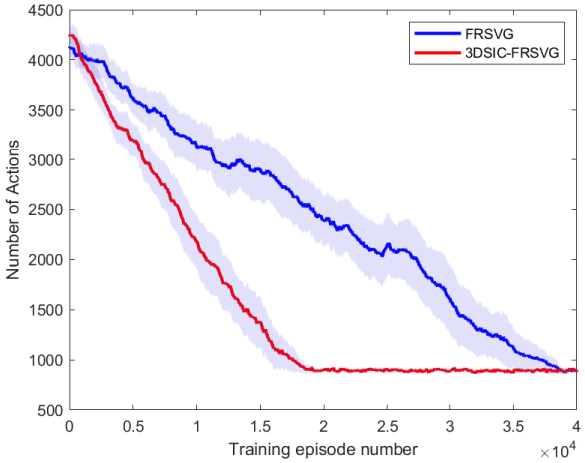


FIGURE 12: Actions curve of 3DSIC-FRSVG and FRSVG in 3D POMDP Environment

to FRSVG, the training efficiency of 3DSIC-FRSVG has increased by 96.5%. It is worth noting that the 3DSIC-FRSVG algorithm may encounter overfitting problems. When this problem occurs, the path planning test can pass, but all agents have very similar learning and action curves with very small variances. This may result in the trained neural network lacking universality in path planning problems. The displayed simulation results are the training results of 3DSIC-FRSVG agents with dropout and data enhancement utilized.

To compare the impact of combining 3DSIC with different reward models on training efficiency, we simulated four different embedding methods of reward models in a complex forest environment of 20 cubes. The simulation results are shown in Fig. 14, where ”2D traditional reward” denote the evaluated reward value in the compressed 2D environment,

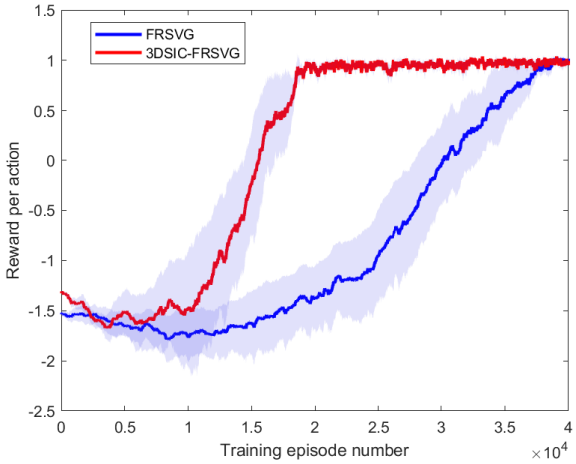


FIGURE 13: Reward per action curve of 3DSIC-FRSVG and FRSVG in 3D POMDP Environment

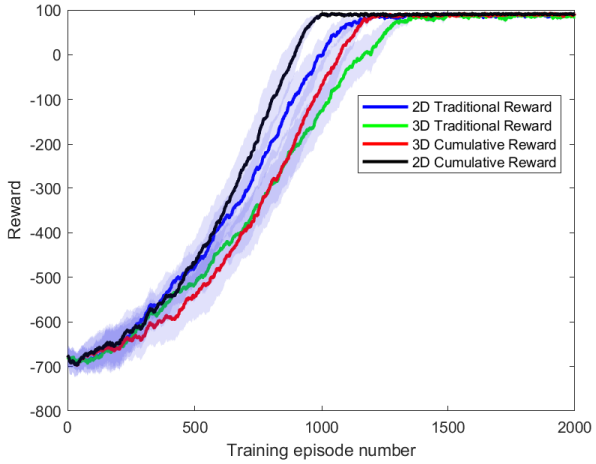


FIGURE 14: Learning curve of traditional reward model and cumulative reward model in 3D POMDP environment

and "3D traditional reward" means the evaluated reward value in the original 3D environment before 3DSIC processed for the traditional reward scheme. Similarly, the "2D cumulative reward" and "3D cumulative reward" denote that of the cumulative reward scheme. Note that for every reward model we trained 10 agents to obtain the average learning curve. It can be observed that the combined cumulative reward model after compression shows the highest training efficiency. The 2D traditional reward scheme and 3D cumulative reward scheme converge very closely in terms of the training episode number. On the other hand, the 2D traditional reward curve increases faster than that of the 3D cumulative reward curve at the early training stage. This indicates that the 3D cumulative reward model may have higher efficiency in larger environments compared to the 2D traditional reward model. Moreover, these simulation results also indicate that the compressed environment with 3DSIC still has information redundancy. More specifically, evaluating reward values in the original 3D environment takes into account more environmental information than evaluating reward values in the 2D environment after 3DSIC. In this case, similar convergence episode number of the 2D traditional reward scheme and the 3D cumulative reward scheme illustrate that in some situation, considering less spatial information can achieve similar training efficiency. This means that the training efficiency improvement brought by the 3DSIC algorithm may be even greater with well designed reward evaluation method.

C. Path planning performance of 3DSIC-DDPG and 3DSIC-FRSVG

In theory, the path planned by combining 3DSIC algorithm and DRL algorithm is difficult to be globally optimal. However, in the POMDP environment, it is hard for the DRL algorithms to achieve global optimal path planning results without using the 3DSIC algorithm. From the action curve simulation results shown in Fig. 9 and Fig. 12, it can be seen that there is no significant difference in the number of actions of the converged path between the 3DSIC-DRL algorithms and the DRL algorithms without 3DSIC. In addition, we plotted a set of path planning results for 3DSIC-DDPG, DDPG, 3DSIC-FRSVG, and FRSVG algorithms in a 3D simulation environment, as shown in Fig. 15 and Fig. 16. Among them, the red paths are the paths planning result without using the 3DSIC algorithm, while the green paths are planned by 3DSIC-DDPG or 3DSIC-FRSVG. It can be observed that the path planning quality of DDPG and 3DSIC-DDPG is similar, and the path planned after information compression is not worse than using DDPG directly, while the total number of actions is sometimes less than traditional DDPG. The path planning quality of 3DSIC-FRSVG is slightly inferior to traditional FRSVG, as evidenced by the fact that the average number of actions taken by 10 3DSIC-FRSVG agents (885.9 actions) is 19.7 more than that of FRSVG (866.2 actions). However, from Fig. 16, we can see that the path planned by traditional FRSVG is shorter but more risky, since sometimes it is very close to obstacles. On the other hand, 3DSIC-FRSVG agents will choose to travel in a safer location.

Additionally, besides path length and training efficiency, the reliability of path planning is crucial. To compare the collision free path planning probabilities of DDPG with and without 3DSIC in POMDP environment, we conducted 5 rounds of training on 3DSIC-DDPG with emulator, 3DSIC-DDPG with belief function, DDPG with emulator, and DDPG with belief function. Then, we added random exploration noise to each trained agent and conducted 100 path planning tests, where we also recorded the number and probability of collision free path planning. The simulation results are shown in Table 4. If the Agent type name

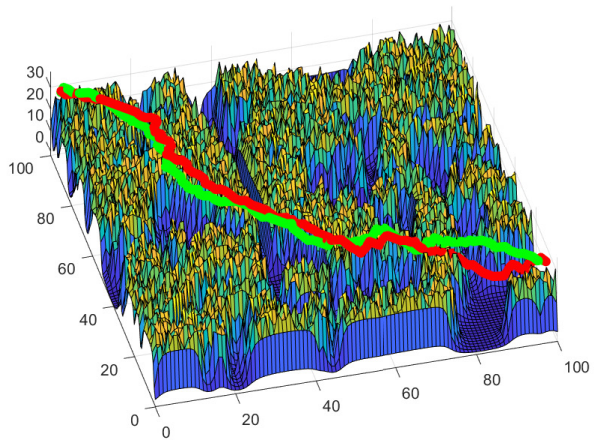


FIGURE 15: Path planning result of 3DSIC-DDPG and DDPG in 3D POMDP Environment, where the red path is planned by DDPG agent, while the green path is planned by 3DSIC-DDPG agent.

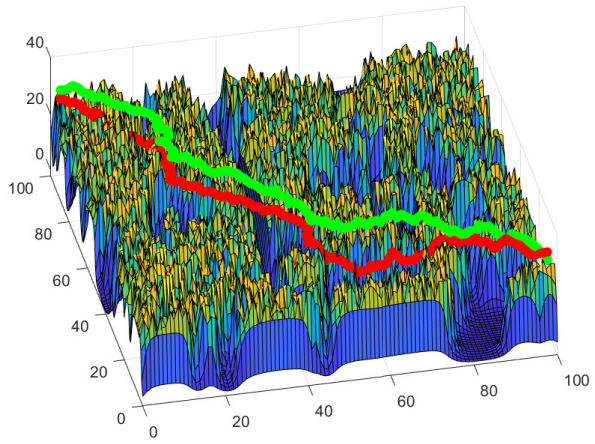


FIGURE 16: Path planning result of 3DSIC-FRSVG and FRSVG in 3D POMDP Environment, where the red path is planned by FRSVG agent, while the green path is planned by 3DSIC-FRSVG agent.

contains (e), it indicates the mapping between observations and history states is obtained through an emulator. While the agent type with (b) indicates that the mapping between observation and the history state is obtained through the BS. It can be seen from the simulation results that using 3DSIC to compress spatial information does indeed have an impact on the reliability of path planning. However, when exploring noise with very small variance, the impact can be almost ignored. In addition, when the noise variance is small, the impact of using an emulator or BS mapping observation on the reliability of path planning cannot be determined. Additionally, when the variance of random exploration noise

TABLE 4: Probability of collision free path planning for different DDPG algorithm with emulator and belief state.

Type of agent	number of collision paths	Variance of exploration noise	Collision free rate
3DSIC-DDPG(e)	44	0.05	91.2%
3DSIC-DDPG(b)	31	0.05	93.8%
3DSIC-DDPG(e)	82	0.2	83.6%
3DSIC-DDPG(b)	127	0.2	74.6%
DDPG(e)	26	0.05	94.8%
DDPG(b)	29	0.05	94.2%
DDPG(e)	118	0.2	76.4%
DDPG(b)	152	0.2	69.6%

increases, the probability of collision free path planning using agents with emulators is significantly higher than that using agents with BS. This is also consistent with the analysis that bad experiences in the BS can have a permanent impact on the system. In addition, simulation results indicate that using 3DSIC can provide agents with resistance to random exploration noise. However, this feature has two aspects: on one hand it reduces the sensitivity of the trained agent to noise, while on the other hand its ability to track environmental changes will be weaker.

V. Conclusions

In this paper, we extended the 3DSIC algorithm to a 3D unknown environment. Additionally, we derived the Bellman equation representation in the POMDP case and constructed an independent emulator to update the environmental information. We first obtained an approximate solution of the belief state distribution through Monte Carlo simulation. Then, we combined the 3DSIC and the DDPG algoritm for a POMDP environment. Our proposed 3DSIC algorithm can efficiently plan paths in unknown 3D environments, where the simulation results show that our proposed 3DSIC-DDPG algorithm has a 95.9% improvement in training efficiency compared to the traditional DDPG algorithm in the 3D POMDP environment. Compared with the FRSVG algorithm, using the 3DSIC algorithm could improve the training efficiency by 96.5% in the 3D PODMP environment, while maintaining the quality of the path planning.

REFERENCES

[1] 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.

[2] Jiapeng Yin, Peter Hoogeboom, Christine Unal, Herman Russchenberg, Fred van der Zwan, and Erik Oudejans. UAV-Aided Weather Radar Calibration. *IEEE Transactions on Geoscience and Remote Sensing*, 57(12):10362–10375, 2019.

[3] Shams ur Rahman, Geon-Hwan Kim, You-Ze Cho, and Ajmal Khan. Positioning of UAVs for throughput maximization in software-defined disaster area UAV communication networks. *Journal of Communications and Networks*, 20(5):452–463, 2018.

[4] Ming Yan, Rui Xiong, Yan Wang, and Chunguo Li. Edge Computing Task Offloading Optimization for a UAV-Assisted Internet of Vehicles via Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, 73(4):5647–5658, 2024.

[5] 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.

[6] 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.

[7] Ming Yan, Chien Aun Chan, André F. Gygax, Chunguo Li, Ampalavanapillai Nirmalathas, and I Chih-Lin. Efficient Generation of Optimal UAV Trajectories With Uncertain Obstacle Avoidance in MEC Networks. *IEEE Internet of Things Journal*, 11(23):38380–38392, 2024.

[8] 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.

[9] 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.

[10] Harrison Kurunathan, Hailong Huang, Kai Li, Wei Ni, and Ekram Hossain. Machine Learning-Aided Operations and Communications of Unmanned Aerial Vehicles: A Contemporary Survey. *IEEE Communications Surveys and Tutorials*, 26(1):496–533, 2024.

[11] 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.

[12] Haoxin Liu and Yonghui Zhang. ASL-DWA: An Improved A-Star Algorithm for Indoor Cleaning Robots. *IEEE Access*, 10:99498–99515, 2022.

[13] Nelapati Lava Prasad and Barathram Ramkumar. 3-D Deployment and Trajectory Planning for Relay Based UAV Assisted Cooperative Communication for Emergency Scenarios Using Dijkstra's Algorithm. *IEEE Transactions on Vehicular Technology*, 72(4):5049–5063, 2023.

[14] 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.

[15] Reza Mashayekhi, Mohd Yamani Idna Idris, Mohammad Hossein Anisi, Ismail Ahmedy, and Ihsan Ali. Informed RRT*-Connect: An Asymptotically Optimal Single-Query Path Planning Method. *IEEE Access*, 8:19842–19852, 2020.

[16] 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.

[17] 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.

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

[19] 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.

[20] 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.

[21] Matej Novosad, Robert Penicka, and Vojtech Vonasek. CTop-PRM: Clustering Topological PRM for Planning Multiple Distinct Paths in 3D Environments. *IEEE Robotics and Automation Letters*, 8(11):7336–7343, 2023.

[22] Yuting Wan, Yanfei Zhong, Ailong Ma, and Liangpei Zhang. An Accurate UAV 3-D Path Planning Method for Disaster Emergency Response Based on an Improved Multiobjective Swarm Intelligence Algorithm. *IEEE Transactions on Cybernetics*, 53(4):2658–2671, 2023.

[23] Yu Wu, Kin Huat Low, Bizhao Pang, and Qingyu Tan. Swarm-Based 4D Path Planning For Drone Operations in Urban Environments. *IEEE Transactions on Vehicular Technology*, 70(8):7464–7479, 2021.

[24] Zhe Zhang, Jian Wu, Jiyang Dai, and Cheng He. A Novel Real-Time Penetration Path Planning Algorithm for Stealth UAV in 3D Complex Dynamic Environment. *IEEE Access*, 8:122757–122771, 2020.

[25] Huan Liu, Guohua Wu, Ling Zhou, Witold Pedrycz, and Ponnuthurai Nagaratnam Suganthan. Tangent-Based Path Planning for UAV in a 3-D Low Altitude Urban Environment. *IEEE Transactions on Intelligent Transportation Systems*, 24(11):12062–12077, 2023.

[26] Shankarachary Ragi and Edwin K. P. Chong. UAV Path Planning in a Dynamic Environment via Partially Observable Markov Decision Process. *IEEE Transactions on Aerospace and Electronic Systems*, 49(4):2397–2412, 2013.

[27] Fatemeh Rekabi-Bana, Junyan Hu, Tomáš Krajník, and Farshad Arvin. Unified Robust Path Planning and Optimal Trajectory Generation for Efficient 3D Area Coverage of Quadrotor UAVs. *IEEE Transactions on Intelligent Transportation Systems*, 25(3):2492–2507, 2024.

[28] Hao Xie, Dingcheng Yang, Lin Xiao, and Jiangbin Lyu. Connectivity-Aware 3D UAV Path Design With Deep Reinforcement Learning. *IEEE Transactions on Vehicular Technology*, 70(12):13022–13034, 2021.

[29] Lijuan Zhang, Jiabin Peng, Weiguo Yi, Hang Lin, Lei Lei, and Xiaoqin Song. A State-Decomposition DDPG Algorithm for UAV Autonomous Navigation in 3-D Complex Environments. *IEEE Internet of Things Journal*, 11(6):10778–10790, 2024.

[30] Hu Teng, Ishtiaq Ahmad, Alamgir Msm, and Kyunghi Chang. 3D Optimal Surveillance Trajectory Planning for Multiple UAVs by Using Particle Swarm Optimization With Surveillance Area Priority. *IEEE Access*, 8:86316–86327, 2020.

[31] Zhipeng Wang, Soon Xin Ng, and Mohammed EI-Hajjar. Deep reinforcement learning assisted uav path planning relying on cumulative reward mode and region segmentation. *IEEE Open Journal of Vehicular Technology*, 5:737–751, 2024.

[32] Chao Wang, Jian Wang, Yuan Shen, and Xudong Zhang. Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach. *IEEE Transactions on Vehicular Technology*, 68(3):2124–2136, 2019.

[33] Hoa Van Nguyen, Hamid Rezatofighi, Ba-Ngu Vo, and Damith C. Ranasinghe. Online UAV Path Planning for Joint Detection and Tracking of Multiple Radio-Tagged Objects. *IEEE Transactions on Signal Processing*, 67(20):5365–5379, 2019.

[34] Zhipeng Wang, Soon Xin Ng, and Mohammed El-Hajjar. A 3d spatial information compression based deep reinforcement learning technique for uav path planning in cluttered environments. *IEEE Open Journal of Vehicular Technology*, 6:647–661, 2025.

[35] 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.

[36] 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.

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

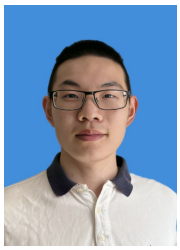
[38] David Silver Volodymyr Mnih, Koray Kavukcuoglu. Human-level control through deep reinforcement learning. *Nature*, page 529–533, 2015.

[39] 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.

[40] 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.

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

[42] Hua Yan, Yunfei Chen, and Shuang-Hua Yang. New Energy Consumption Model for Rotary-Wing UAV Propulsion. *IEEE Wireless Communications Letters*, 10(9):2009–2012, 2021.
[43] 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.
[44] 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.
[45] Kyriakos G. Vamvoudakis and Nick-Marios T. Kokolakis. 2020.
[46] Jiehong Wu, Ya'nan Sun, Danyang Li, Junling Shi, Xianwei Li, Lijun Gao, Lei Yu, Guangjie Han, and Jinsong Wu. An Adaptive Conversion Speed Q-Learning Algorithm for Search and Rescue UAV Path Planning in Unknown Environments. *IEEE Transactions on Vehicular Technology*, 72(12):15391–15404, 2023.
[47] Michael C. Fowler, T. Charles Clancy, and Ryan K. Williams. Intelligent Knowledge Distribution: Constrained-Action POMDPs for Resource-Aware Multiagent Communication. *IEEE Transactions on Cybernetics*, 52(4):2004–2017, 2022.
[48] Grigore Stamatescu, Iulia Stamatescu, Dan Popescu, and Cristian Mateescu. Sensor fusion method for altitude estimation in mini-UAV applications. In *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, pages SSS–39–SSS–42, 2015.
[49] Reham Abdelfatah, Ahmed Moawad, Nancy Alshaer, and Tawfik Ismail. UAV Tracking System Using Integrated Sensor Fusion with RTK-GPS. In *2021 International Mobile, Intelligent, and Ubiquitous Computing Conference (MIUCC)*, pages 352–356, 2021.
[50] 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.
[51] Xuqiong Luo, Qiyuan Wang, Hongfang Gong, and Chao Tang. Uav path planning based on the average td3 algorithm with prioritized experience replay. *IEEE Access*, 12:38017–38029, 2024.



Zhipeng Wang received the B.Eng. degree in telecommunication engineering from Chengdu University of Information Technology, Chengdu, China, in 2019. He received the M.Sc. degree (First class) in electronic communication and computer engineering from the University of Nottingham, U.K., in 2020. He received the Ph.D. degree in electronic and electrical engineering form the University of Southampton, U.K. in 2025. His current research interests include deep reinforcement learning, high dimensional information compression, UAV autonomous navigation, 3D path planning, partially observable Markov decision process.

Dr Soon Xin Ng(S'99-M'03-SM'08) received the B.Eng. degree (First class) in electronic engineering and the Ph.D. degree in telecommunications from the University of Southampton, U.K., in 1999 and 2002, respectively. He is currently a Professor of Next Generation Communications at the University of Southampton. His research interests include adaptive coded modulation, coded modulation, channel coding, space-time coding, joint source and channel coding, iterative detection, OFDM, MIMO, cooperative communications, distributed coding, quantum communications, quantum error correction codes, joint wireless-and-optical-fibre communications, game theory, artificial intelligence and machine learning. He has published over 300 papers and co-authored two John Wiley/IEEE Press books in this field.

Mohammed El-Hajjar (M'02, SM'14) is a Professor of Signal Processing for Wireless Communications in the School of Electronics and Computer Science in the University of Southampton. He is the recipient of several academic awards and has published a Wiley-IEEE book and more than 100 IEEE journal and conference papers and in excess of 10 patents. Mohammed's research interests include the design of intelligent and energy-efficient transceivers, MIMOs, millimeter wave communications, non-terrestrial networks and machine learning for wireless communications. Mohammed's research is funded by the Engineering and Physical Sciences Research Council, the Royal Academy of Engineering and many industrial partners.