A UAV Navigation Approach Based on Deep Reinforcement Learning in Large Cluttered 3D Environments

Yuntao Xue and Weisheng Chen

***Abstract*—This paper is concerned with the problem of using deep reinforcement learning methods to enable safe navigation of UAVs in unknown large-scale environments containing a large number of obstacles. The ability of UAV to navigate autonomously is a precondition for performing disaster rescue and logistics deliveries. In this paper, the perception-constrained navigation of UAV is modeled as a partially observable Markov decision process (POMDP), and a fast recurrent stochastic valued gradient algorithm based on the actot-critic framework is designed for online solution. Compared with traditional SLAM-based and reactive obstacle avoidance-based approaches, the proposed navigation method can map the raw sensing information into navigation signals, and the learned policy deployed to the UAV can avoid the memory occupation and computational consumption required for real-time map building. Through experiments in a newly designed simulation environment, it is demonstrated that the proposed algorithm can enable the UAV to navigate safely in unknown cluttered large-scale environments and outperform the state-of-the-art DRL algorithm in terms of performance.**

***Index Terms*—Autonomous UAV navigation, deep reinforcement learning, partially observable Markov decision process, continuous control.**

I. Introduction

OVER the last decade, the domain of unmanned aerial vehicles (UAVs) has rapidly expanded to the applications in a variety of industries, including scientific research, infrastructure inspections, and aerial photography. Researchers are dedicated to improving UAVs autonomy and designing an intelligent system that allow them to accomplish autonomous navigation without human intervention for downstream tasks. The objectives of this work is to provide a method that allows UAVs to travel autonomously from an arbitrary starting point to a target point in an unknown large-scale complicated environment, such as densely forested areas or densely populated cities. The technology is essential for a multitude of applications, including cargo transportation, disaster relief, and aerial surveillance, all of which contribute to the creation of safer and smarter cities [1].

Manuscript received 9 May 2022; revised 14 July 2022; accepted 30 October 2022. Date of publication 2 November 2022; date of current version 14 March 2023. This work was supported by the National Natural Science Foundation of China under Grant 62073254. The review of this article was coordinated by Dr. Benedetta Picano. *(Corresponding author: Weisheng Chen.)*

The authors are with the School of Aerospace Science and Technology, Xidian University, Xi'an 710071, China (e-mail: ytxue@stu.xidian.edu.cn; wshchen@126.com).

Digital Object Identifier 10.1109/TVT.2022.3218855

For UAV navigation system solutions, a variety of methods have been developed, including non-learning based methods and learning based methods. The most classical non-learning method is referred to as simultaneously localization and mapping (SLAM), which entails creating a map of the environment and estimating the pose of the UAV using SLAM, followed by path planning navigation [2], [3], [4], [5], [6], [7], [8]. Li [2] utilized graph SLAM [7] and A-star path planning, completing the direct navigation method of the micro UAV in the industrial environment without any specialization. Another class of non-learning based method is sensing and avoidance. The intuition is that, UAV avoided collisions by turning the course of the vehicle and navigates using path planning [9], [10], [11], [12], [13], [14], [15], [16]. Cho [11], for instance, used an optical flow module to avoid wall-like frontal obstacles and trajectory tracking module generates control commands for navigation in 3-D textured environments. Liu [9] proposed a planning method, which imposed a cost map to avoid collisions on the top of the probability occupancy representation of the gradually observed environment, and then finded the optimal trajectory in the cost map. The third traditional approach employs a filtering technique for localization to retracing preset path points for navigation [17], [18], [19], [20], [21], [22], [23]. Methods in this category are generally committed to improving the accuracy of UAV positioning by reducing time measurement errors using various position techniques such as Kalman filter [17] or Particle filter [18]. The accurate localization of the UAV reduces the error between the actual trajectory and the predefined path. Non-learning based methods all have one thing in common, that is, they need accurate external path planning, which will be challenged when the environment is complex and unknown. Several machine learning methods have been proposed for this purpose, including reinforcement learning (RL) and imitation learning [24], [25], [26]. Pham [25], for example, proposed a method to train UAV to navigate to the target point using PID and Q learning algorithms without a mathematical model in an unknown environment. Loquercio [26] built a convolutional neural network named DroNet that a UAV can learn to fly in the city by imitating a manned vehicle, and has shown great generalization ability in a wide variety of scenarios.

When it comes to navigation behavior in large scale complicated unknown environments, UAVs encountered the following challenges: 1) The flight environment contains dense obstacles, such as thick forests and crowded cities. In this situation, sensing

and avoidance-based methods lose efficiency since this method is adapted to an environment where obstacles are sparse; 2) The area of the environment is large. In large-scale scenarios in the real world, SLAM-based methods will fail because the continuous mapping process will result in unsustainable processing costs; 3) The real-world environment is often unknown. The detection and navigation of an unknown environment will be complicated by interference from unknown obstacles throughout the UAV flight. This procedure necessitates numerous iterative calculations. While reinforcement learning is suitable for unknown situations, it requires improvement in large and cluttered situations; 4) Sensors have limited sensing capacity. The sensing range and field of vision of most radars and cameras are restricted, and GPS in dense cities would be ineffective. When the capabilities of the sensor are limited, UAVs are often caught in a dilemma or even destroyed catastrophically.

The aforementioned issues obstruct the practical deployment of UAV navigation algorithms. In order to solve these challenges, UAVs need to resort to the historical states they have observed, and then decide which action should be taken. The decision paradigm can conform to the partially-observed Markov Decision process (POMDP). The MDP model makes decisions based on the system's current actual state, although the accurate state of the system is often difficult to acquire. POMDP makes the assumption that the system's state information is only partially known and cannot be observed directly. The agent can only indirectly determine the MDP's potential state via observation of the system, and then take action to move to another potential state while obtaining rewards. Therefore, this work models the perceptually constrained UAV navigation problem as POMDP and solves it by redesigning the reward and state space using a DRL-based approach.

Modeling the navigation task as POMDP needs to meet the following requirements. Firstly, it should be model-free, because in the real environment, the agent cannot know the state transition contingency and the observation probability a priori. Secondly, The action space of a UAV should be continuous since its control signal is continuous. Finally, its observation signal and reward form should be tailored to provide the UAV with adequate information to achieve a certain goal. Therefore, this work designed an observation space, with a variety of sensors so that the UAV can obtain its own state information and interactive information about the environment. Additionally, a reward function based on UAV domain knowledge was developed to reward or penalize behaviors performed in a certain state.

By retaining the memory of previous observations and actions, the model-free POMDP problem is solved by RL, creating stochastic or deterministic control policy [27], [28], [29], [30], [31]. Among these methods [32], the policy gradient method obtains a parameterized stochastic policy for model-free POMDP with continuous space through gradient descent in the parameter space. A major disadvantage of these algorithms is that they cannot be implemented end-to-end. In addition, when a nonlinear function approximated such as a neural network is used to represent the action-value function, RL will become unstable or even diverge [33]. With the emergence of deep learning technology, various techniques for solving Markov decision processes (MDPs) [34] and POMDPs problems have begun to integrate reinforcement learning with deep neural networks. Hausknecht [35] used model-free RL algorithm with recurrent network to solve POMDPs with discrete action space. Lillicrap [36] introduced the DDPG framework, which combined a deterministic policy-based actor-critic framework [37] and a deep neural network to perform continuous control tasks. Heess [38] used a recurrent neural network with backpropagation and extended the deterministic policy gradient algorithm and the stochastic value gradient algorithm to obtain RDPG and RSVG(0) for solving problems in partially observable domains.

Many researchers have used the theoretical DRL algorithm to control UAV navigation, which has many advantages over the non-learning based navigation method. For example, RL can learn correct actions when UAVs have little prior knowledge of their environment. A model-based RL method for quadrotor autonomous navigation control was created by Imanberdiyev et al. [39]. The ROS-based experimental environment of RL is set to a discrete state, which is incompatible with real UAV navigation, due to the dimensional restriction of RL. The "dimension curse" of RL can be solved by combining with deep neural networks. A DQN-based obstacle avoidance technique was created by Singla et al. for UAVs with monocular cameras [40]. In order to maintain important information for obstacle avoidance in lengthy observation sequences, it employs a technique that integrates a recurrent neural network with temporal attention. However, its action space is discrete, and vision-based sensors are susceptible to illumination and texture changes in applications. Tong [41] divided the UAV navigation task into obstacle avoidance and objective achievement, and trained them independently for dynamic environments. However, its experimental setting was set in a 2D environment, which is not comparable to the real 3D environment. In response to the above problems, a rangefinder sensor was used that is more generalizable for navigation in a continuous space in a specially designed 3D environment containing complex obstacles.

Our method is motivated by Wang's proposed RDPG-based method for UAV navigation in continuous action spaces [42], which can run efficiently online and be applied to large-scale environments. But it is generally believed that in partially observable domains, stochastic policy outperform deterministic policy because of more possibilities to explore [38]. Therefore, the corresponding stochastic value policy version of DPG, namely SVG(0), is concerned in our work to solve the navigation of UAV in unknown environment. However, the sample efficiency of the RSVG(0) algorithm is low, and the parameters can only be updated after the end of the episode, which makes the algorithm unable to learn quickly. In addition, RSVG(0) updates the parameters based on the whole episode rather than the historical trajectory, which is not suitable for the learning framework of memory replay. Specifically, RSVG(0) is actually based on a highly correlated state sequence update policy, which affects its convergence speed. In this regard, we proposed an online update DRL algorithm, Fast Recurrent SVG(0), to solve the POMDPs problem and apply it to the continuous navigation control of UAV.

The main contributions of this paper are as follows:

1) The challenge of UAV navigation in unknown complex environments is addressed in this research through the proposal of an actor-critic based motion framework. Specifically, UAV navigation was modeled as POMDP with limited sensing capabilities, and online fast stochastic policy gradient were used to solve it.
2) For domains with partial observability, we derived the stochastic policy gradient algorithm. The success rate of UAV navigation can be increased by theoretically and experimentally proving that stochastic policy produce better exploration behavior than deterministic policy in unknown environments.
3) By incorporating knowledge from the navigation domain, the reward function and action space were redesigned to address the reward sparsity problem in RL. The UAV can obtain reward signals at each step, and then perform dual-channel control of steering and speed for obstacle avoidance. The effectiveness of the navigation framework is verified in a 3D simulation scenario with randomly generated obstacles.

The structure of the entire manuscript is shown below. Section II introduces the concepts of MDPs and POMDPs, as well as the DDPG and RSVG(0) algorithms for solving POMDPs in continuous space. Section III defines the navigation of UAV. The perception-constrained UAV navigation problem is modeled as a POMDP in Section IV. A new DRL algorithm proposed in Section V. Section VI demonstrates simulation results and discussion. Section VII summarizes the full paper and looks forward to future work.

II. Background

This section describes the concepts of MDP and POMDP, then introduces the solution algorithms DDPG and RSVG(0) for POMDP in continuous space.

A. MDPs and POMDPs

This work consider the Markov decision process in discrete time with continuous state and action. MDP uses an environmental state $\mathcal{S}$ and a series of action $\mathcal{A}$, and has an initial state distribution $\mathbf{s}^0 \sim p^0(\cdot)$, a transfer distribution $\mathbf{s}^{t+1} \sim p(\cdot \mid \mathbf{s}^t, \mathbf{a}^t)$, and a reward function $r^t = r(\mathbf{s}^t, \mathbf{a}^t, t)$. In each time step, the agent executes action a_t under the state s_t according to the policy $\mathbf{a} \sim p(\cdot \mid \mathbf{s}; \theta)$, and then enters the next state s_{t+1} and a reward r_t at the same time. What we consider is a time-invariant stochastic policy $\mathbf{a} \sim p(\cdot \mid \mathbf{s}; \theta)$. The goal of policy optimization is to find the policy parameter θ, It can maximize the expectation of the sum of future rewards. The state space and action space of the MDP can be discrete or continuous. Since the control signal of the UAV is continuous, this article focuses on the MDP and POMDP with continuous state space and action space. In order to facilitate the calculation, we will turn the summation into an integral in the following.

MDPs are usually solved according to RL, which obtains the optimal poilcy based on trial and error learning. The policy can be deterministic or stochastic, and since a stochastic policy increases the probability of being completely explored in an unknown environment, this paper focuses on the stochastic policy, expressed as: $a \sim \pi(\cdot \mid s) : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is in state $\mathcal{S}$ The probability of a set of actions taken according to the policy. If the action space is continuous, the stochastic policy can be expressed as a function $a \sim \pi(\cdot \mid s, \theta)$, where θ is the parameter of the function. In order to simplify the representation, the policy π be called in the following description is a function of θ. The size of the state space and the size of the action space of the Markov decision process are defined as $|S|$ and $|A|$, respectively. Solving the MDP through the value iteration algorithm requires scanning the entire state space to converge, so its computational complexity can be expressed as $\mathcal{O}(|S|^2|A|)$.

The task of reinforcement learning is to estimate the value functions of states and the action-value functions of state-action pairings. The value function is that the agent starts from the initial state and determines the expectation of the subsequent actions according to the policy π. For the stochastic policy, the value function is defined as:

$$V_\pi(s_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}) \mid s_t\right] \tag{1}$$

Among them, $\gamma \in [0, 1]$ is the discount factor, and the action-value function is defined as:

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}) \mid s_t, a_t\right] \tag{2}$$

For the optimal stochastic policy, we expect that the agent can maximize the expectation of future discount rewards according to its take action:

$$J = \mathbb{E}_\tau\left[\sum_{t=1}^{\infty} \gamma^{t-1} r(s_t, a_t)\right] \tag{3}$$

When the agent cannot directly observe the state s_t, instead it receives observations from a set of $\mathcal{O}$ which are conditioned on the underlying state $p(o_t \mid s_t)$,then MDP becomes POMDP. Discrete-time POMDP formally is a 7-tuple $(S, A, T, R, \Omega, O, \gamma)$, in each time step, the environment is in a certain state $s_t \in \mathcal{S}$, the agent takes action $a_t \in \mathcal{A}$ and according to the transition probability $\mathcal{T}(s_{t+1} \mid s_t, a_t)$ to reach the next state s_{t+1}, the agent receives the observation $o_t \in \mathcal{O}$ according to the conditional observation probability $O(o_t \mid s_{t+1}, a_t)$ and gets a reward $R(s_t, a_t)$ simultaneously. The observation sequence no longer satisfies the Markov property: $p(o_{t+1} \mid a_t, o_t, a_{t-1}, o_{t-1}, \ldots, o_0) \neq p(o_{t+1} \mid o_t, a_t)$. Therefore, the agent needs to access the entire historical trajectory $h_t = (o_t, a_{t-1}, o_{t-1}, \ldots, o_0)$ infer the current state s_t to make a decision.

The aim of RL in partially observable domains is that the agent learns an optimal policy $a_t \sim \pi(\cdot \mid h_t)$ which maps from the history to a distribution over action $P(\mathcal{A})$ witch maximizes the expected discounted reward.

B. Policy Gradient and Value Gradient

In order to overcome the problem of MDPs, the Deterministic Policy Gradient (DPG) algorithm was proposed to deal with

continuous action space. The essential premise of the DPG algorithm for the fully observed situation is that the policy can be updated using backpropagation for a deterministic policy μ^θ with parameters θ and access to the real action-value function associated with the existing policy Q:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{s\sim\rho^\mu}\left[\left.\frac{\partial Q^\mu(s,a)}{\partial a}\right|_{a=\mu^\theta(s)} \frac{\partial \mu^\theta(s)}{\partial \theta}\right] \tag{4}$$

where the expected value is related to the discounted state visitation distribution ρ^μ caused by the current policy μ^θ. In practice, an approximation (critic) Q^ω with parameters ω that is differentiable in a and can be learnt, for example, using Q-learning, replaces the precise action-value function Q^μ.

In order to ensure that the DPG algorithm can be applied to a large observation space, the researchers used a neural network to approximate the function. Lillicrap [36] combined with deep neural network to make necessary modifications to DPG (using experience replay and target network), and Silver [43] proposed the Deep DPG algorithm that robustly solved challenging problems across a variety of MDP domains with continuous action spaces.

The optical policy and associated action-value functions are based on the entire previous observation-action history function in the case of partial observability. Heess [38] introduced a recurrent neural network to enable the network to learn to save past information, which is essential for solving a part of the observable domain. Correspondingly, writing $\mu(h)$ and $Q(h,a)$ instead of $\mu(s)$ and $Q(s,a)$. It is also called Recurrent DPG, and its policy is updated as follows:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_\tau\left[\sum_t \gamma^{t-1}\left.\frac{\partial Q^\mu(h_t,a)}{\partial a}\right|_{a=\mu^\theta(h_t)} \frac{\partial \mu^\theta(h_t)}{\partial \theta}\right] \tag{5}$$

RDPG is an algorithm for learning deterministic policy, and examples where deterministic policy perform poorly in partially respectable situations are discussed in the literature [44]. Therefore, Heess also introduced a stochastic policy version of DPG: SVG(0), and combined with the Recurrent neural network to realize the use of past information to make it used for POMDP. SVG(0) updates the policy by backpropagating $\partial Q/\partial a$, specifically, using "re-parameterization" to represent the stochastic policy as a fixed, independent noise source and a parameterized deterministic function that transforms a draw from that noise source, ie $a = \pi^\theta(h,\nu)$ with $\nu \sim \beta(\cdot)$, where β is a certain kind of fixed distribution, such as Gaussian distribution, then the update of the stochastic policy is as follows:

$$\frac{\partial J(\theta)}{\partial \theta} = \mathbb{E}_{\tau,\nu}\left[\sum_t \gamma^{t-1}\left.\frac{\partial Q^{\pi^\theta}(h_t,a)}{\partial a}\right|_{a=\pi^\theta(h_t,\nu_t)} \frac{\partial \pi^\theta(h_t,\nu_t)}{\partial \theta}\right] \tag{6}$$

where the expectation now explicitly over entire trajectories $\tau = (s_1, o_1, a_1, s_2, o_2, a_2, \ldots)$ which are drawn from the trajectory distribution induced by the current policy and $h_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t)$ is the observation-action trajectory prefix at time step t. During the learning process, a RSVG(0) agent uses the current policy to interact with the environment. After that, the obtained state information is placed in the replay buffer, and then a mini-batch is sampled evenly in the replay buffer to update the actor and critic.

Although RSVG(0) has demonstrated its efficacy in continuous control tasks in a part of the observable domain, it still has to be improved. 1) RSVG(0) is an offline update algorithm. It only updates the parameters after an episode has finished, resulting in limited usage and exploration of the area. 2) RSVG(0) converges slowly. As shown in Section V, the parameters should be updated based on the historical trajectory rather than the entire episode. The stochastic gradient is shown in (5) to be closely related to the historical paths in an episode. As a result, while training with the stochastic gradient descent (SGD) approach, the sampled mini-batch is the set of history trajectory correlations, resulting in a slow convergence of RSVG(0). To address this issue, this paper proposed a new DRL-based method for solving the POMDP problem.

III. Problem Formulation

The problem of UAV navigation in unknown complicated situations is formulated in this section.

A. UAV Coordinate System

Assuming a UAV is launched from any location (starting point) in the three-dimensional world, and its mission is to navigate a complicated environment to achieve its destination (target point). In this process, the earth coordinate system can be used to describe its absolute position, that is, $\varphi = [x, y, z, \vartheta_x, \vartheta_y, \vartheta_z]$. The body coordinate system can characterize the UAV's linear and angular motion, namely $\mu = [a, b, c, \vartheta_a, \vartheta_b, \vartheta_c]$. The UAV can only obtain the information about the relative position of the target point and the current state of its sensors. The fixed world coordinate system and the body coordinate system together represent the speed, direction and height of the UAV, then UAV needs to rely on limited sensor information to obtain internal state descriptions and complete navigation tasks.

B. The Dynamics of UAV

There are some researches on the UAV navigation problem that assumes the flight in a fixed altitude, while this is not in line with the real application scenario. To accurately simulate UAV navigation, the flight height in this work is set to $[0m, 100\tilde{}m]$. Therefore, the vector describing the position, direction and movement of the UAV can be simplified as $\zeta = [x, y, z, \vartheta]$, and then the dynamics of the UAV can be expressed as:

$$\begin{cases} v_{t+1} = v_t + \rho_t \\ \phi_{t+1} = \phi_t + \varphi_t \\ x_{t+1} = x_t + v_{t+1} \times \cos(\phi_{t+1}) \\ y_{t+1} = y_t + v_{t+1} \times \sin(\phi_{t+1}) \end{cases} \tag{7}$$

where v_t and ϕ_t represent the speed of UAV and first perspective direction at time step t, and ρ_t and φ_t represent throttle and steering commands. The navigation task is to control a UAV to take off from a random starting point to reach the target point without colliding with obstacles. We regarded the overall

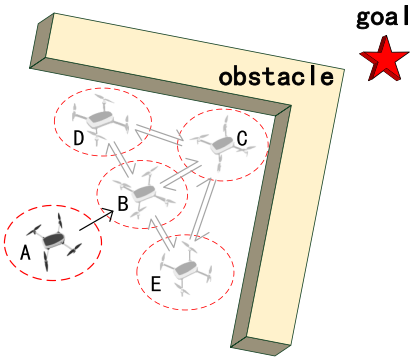


Fig. 1. A schematic diagram of the observability of UAV navigation in a large and cluttered environment. The red dotted coil outside the UAV indicates its perception range. The arrow indicates the direction where the drone may move. The arrow indicates the direction in which the UAV may move.

observation of the UAV as the system state, and the throttle and steering commands as agent's actions. According to the model-free property, the state transition probability is unknown and is determined automatically by the environment.

IV. DRL Model for UAV Navigation

In this section, practical examples were used to illustrate the advantages of modeling UAV navigation in an unknown and cluttered environment as POMDP, and POMDP modeling was given in details.

A. UAV Navigation as a POMDP

A UAV is assumed to have arrived at point A, as indicated in Fig. 1. It will move forward to point C to get closer to the target point. At this point, the UAV finds an obstacle in front of it and it should choose to move to the left, right or backwards. If it chooses to reach point D on the left, the UAV without memory storage forgets the environmental structure at point C and is instead encouraged to approach the target point, thus reaching point B or C. Similarly, if the UAV chooses to reach point E on the right side of point C, or directly chooses to go back to point B, it will eventually fall into the dilemma between point B and point C, looping and unable to escape. There are two reasons for this situation. One is due to the limited sensing capabilities of the UAV's on-board sensors, it can only perceive obstacles within a limited range. Another key aspect is that if UAVs cannot remember the environment they have experienced, they cannot build the environment gradually and make better decisions. POMDP's decision approach is more advantageous for the exploration of cluttered and unknown environments because MDP assumes that the environment state is fully observable, but its decision relies only on the state of the previous moment and cannot summarize the information of the entire historical trajectory to make a decision.

B. Description of Observation Space and Action Space

The UAV must be able to receive information from at least three sources in order to navigate, including information about its internal state, its interaction with the environment, and its relationship with the destination. First of all, the internal state of the UAV is its pose information, including position and orientation. Secondly, the UAV should perceive the obstacle information in the surrounding environment, which can be obtained from the image returned by the camera, the radar signal returned by the radar, or the distance information returned by the ranging sensor. In this work, we used seven ranging sensors to perceive UAV and environmental information, which was expressed as $[d_0, \ldots, d_6]$, as shown in Fig. 2(a). In order to adapt to three-dimensional navigation, 5 range sensors are also arranged on each vertical plane, expressed as $\psi = \{d_{i0}, d_{i1}, d_{i2}, d_{i3}, d_{i4}\}_{i=0\ldots6}^{7}$, as shown in Fig. 2(b). Finally, the relative location of the UAV with respect to the destination should be obtained. As shown in Fig. 2(c), this is represented by the vector $\xi = [d_7, \varphi_v, \varphi_h]$, which contains the distance information between the UAV and the destination, as well as the horizontal and vertical angle relationship between the UAV's first view direction and the goal point. This can be collected using equipment such as GPS. Combine the three kinds of information to get the description of the observation space, namely $o = [\psi, d_7, \varphi_v, \varphi_h]$, where $\varphi_v, \varphi_h \in [-\pi, \pi]$, $d_{i0} \sim d_{i6} \in [0, 10]$ and $d_7 \in [0, +\infty]$.

The action space of the UAV is obtained by its control signal. Considering the flight status of the UAV in a three-dimensional environment, the vector ρ was adopted as the steering signal with a range of $-\frac{6}{\pi}$ to $\frac{6}{\pi}$, σ as the throttle signal with a range of $\in [0, 2]$, and regard them as the action of the UAV.

C. Reward Design

In order for the UAV to gradually learn to make the most rational decisions based on sensed environmental information, the proper reward function must be designed to help it recognize the quality of the action. For UAV navigation in a large-scale unknown and complicated environment, a typical reward is a sparse reward, which is obtained only when the UAV reaches the ultimate target point. Although the reward design is simple, because the initial policy is random, the UAV has a low probability of successfully completing the navigation task in a large-scale complicated environment, resulting in very slow algorithm convergence. In order to speed up training, non-sparse rewards were proposed in RL. Recently, many methods using internal rewards have been proposed to solve agent exploration problems, including curiosity-driven, count-based and state-based methods. These methods [45] all shaped the formation of rewards so that the agent has internal incentives to complete exploration or navigation tasks, and ensured the invariance of the policy. However, the formation of the rewards of these methods depended on the change of the potential function of the state. It is difficult to realize in practice which hinders the integration of the rewards obtained from domain knowledge [46].

In our work, a non-sparse reward was designed for UAV navigation to achieve corresponding rewards and punishments in every action, and finally navigated to the target point in an optimal way. This non-sparse reward combined the domain knowledge of UAV navigation to get a satisfactory policy, which consists of four parts: transfer reward, obstacle penalty, speed

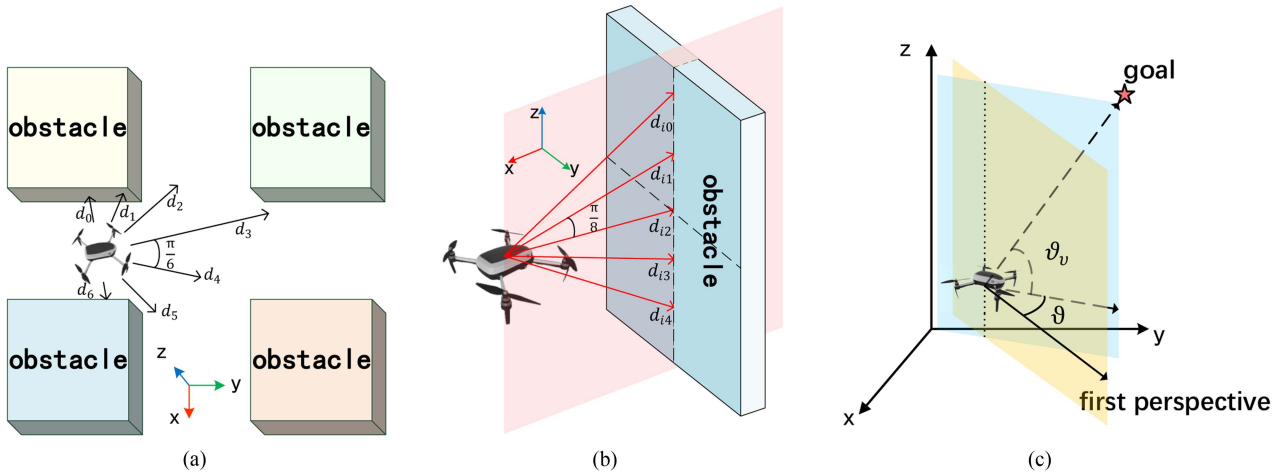


Fig. 2. Schematic diagram of UAV perception in a 3D environment. d_i in (a) and (b) represents the distance information to the obstacle perceived by the onboard rangefinder. The angle in (c) represents the relationship of the UAV to the goal.

penalty, safe space reward and step penalty. The transfer reward design is:

$$r_{trans} = \epsilon(d_s - d_{s-1}) \tag{8}$$

where ϵ is a normal number, and $d_s - d_{s-1}$ represents the shortened distance between the current position of the UAV the destination after one step transfer. By utilizing reward transfer, The UAV was incentivised effectively to move closer to the destination while restricting its movement away from it. Additionally, the most crucial component of navigating in a complicated environment is ensuring the safety of UAV. If the UAV collides with an obstacle, the result is catastrophic. To prevent the UAV from approaching the obstacle too closely, the obstacle penalty was designed as follows:

$$r_{obs} = -\lambda e^{-\sigma d_{\min}} \tag{9}$$

where λ and σ are two constants, and $d_{\min}$ is the limit distance that UAV can approach the obstacle in a safe flying condition. However, a single distance limit cannot fully guarantee the safety of UAV flight. Additionally, the UAV's speed approaching the obstacle should be regulated. The cost function that constrains the speed is also an exponential function, which needs to be punished when approaching or exceeding the maximum speed. As a result, to prohibit the UAV from approaching obstacles and speeding excessively close to them, the speed penalty was designed in the same manner as the equation (8):

$$r_{velocity} = -\alpha e^{\beta v_{\max}} \tag{10}$$

where α and β are two normal numbers, and $v_{\max}$ is the maximal speed of UAV specified by us. To guide the UAV to move in a safe area, it gets a safe area reward r_{safe} when its first view is towards safe space. Finally, in order to avoid the UAV's behavior that is detrimental to quickly reaching the destination, a constant number of steps is imposed on the UAV after each transfer. In short, the final non-sparse reward can be expressed as:

$$r_{final} = r_{trans} + r_{bar} + r_{velocity} + r_{safe} + r_{step} \tag{11}$$

The parameters in formula (10) must be fine-tuned in response to various scenarios in order for the final policy to be appropriate for navigation tasks. This relationship will be discussed in further detail in Section VI.

V. Deep Reinforcement Learning Algorithm to Addressing POMDPs

In this section, the proposed DRL algorithm is derived, followed by a presentation of the entire FRSVG(0) framework.

A. The FRSVG(0) Algorithm

To address the POMDP problem, this section developed a DRL algorithm that can run quickly online. Both the policy function and the corresponding action-value function in a partially observable environment are functions that are dependent on previous historical observations h_t. In order to make decisions based on historical data, recurrent neural networks have learned about the past. h_t is made up of the information o_t observed at time step t and the earlier historical information h_{t-1}. The value function and action-value function of the latent state s_t under observation h_t are defined as:

$$V_\pi^{h_t}(s_t) = \mathbb{E}\left[\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k}\middle|\, s_t, h_t, \pi\right] \tag{12}$$

$$Q_\pi^{h_t}(s_t, a_t) = \mathbb{E}\left[\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k}\middle|\, s_t, h_t, a_t, \pi\right] \tag{13}$$

The reward obtained by taking action when observing h_t with the latent state s_t is defined as:

$$R_\pi^{h_t}(s_t, a_t) = \mathbb{E}\left[r_{t+1}\middle|\, s_t, h_t, a_t, \pi\right] \tag{14}$$

As mentioned in Section II, the data of the latent state s_t of h_t obtained from the framework of experience replay adopted by the RSVG(0) algorithm is highly correlated. In order to reduce data correlation, the algorithm update is performed using the historical trajectory h_t instead of the entire trajectory τ. The

distribution relationship between the historical trajectory h_t and its latent state s_t is defined as $s_t \sim p(s_t|h_t)$, then the value function and action-value function of the historical trajectory h_t are:

$$V_\pi(h_t) = \mathbb{E}_{s_t|h_t}\left[V_\pi^{h_t}(s_t)\right] \tag{15}$$

$$Q_\pi(h_t, a_t) = \mathbb{E}_{s_t|h_t}\left[Q_\pi^{h_t}(s_t, a_t)\right] \tag{16}$$

The reward obtained from the action a_t taken in the historical trajectory h_t is stated as follows:

$$R_\pi(h_t, a_t) = \mathbb{E}_{s_t|h_t}\left[R_\pi^{h_t}(s_t, a_t)\right] \tag{17}$$

Bringing the relationship between the value function and the Q function into (16) can get:

$$\begin{aligned} &Q_\pi(h_t, a_t) \\ &= \mathbb{E}_{s_t\sim p(s_t|h_t)}\left[Q_\pi^{h_t}(s_t, a_t)\right] \\ &= \mathbb{E}_{s_t\sim p(s_t|h_t)}\left\{\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}\left[r_{t+1}\right]\right\} + \gamma\mathbb{E}_{s_t\sim p(s_t|h_t)} \\ &\quad \left\{\mathbb{E}_{o_{t+1},s_{t+1}\sim p(s_{t+1}|s_t,a_t)p(o_{t+1}|s_{t+1})}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\} \\ &= R_\pi(h_t, a_t) + \gamma\mathbb{E}_{h_{t+1}\sim p(h_{t+1}|h_t,a_t)} \\ &\quad \left\{\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|h_{t+1})}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\} \\ &= R_\pi(h_t, a_t) + \gamma\sum_{h_{t+1}} p(h_{t+1} \mid h_t, a_t) V_\pi(h_{t+1}) \end{aligned} \tag{18}$$

Based on the above definition, we calculate the policy gradient of the desired objective function of the optimal stochastic policy in (3) according to [34] as:

$$\begin{aligned} \nabla J(\theta) &= \nabla V(h_0) \\ &= \sum_h \sum_{t=0}^{\infty} \gamma^t Pr(h_0 \to h, t, \pi) \sum_a \frac{\partial\pi^\theta(h_t,\nu_t)}{\partial\theta} Q_\pi(h_t, a_t) \\ &= \sum_h d_\pi(h) \sum_a \frac{\partial\pi^\theta(h_t, \nu_t)}{\partial\theta} Q_\pi(h_t, a_t) \\ &= \mathbb{E}_{h\sim d_\pi(h)}\mathbb{E}_{a\sim\pi^\theta(h_t,\nu_t)}\left[\frac{\partial\log\left(\pi^\theta(h_t,\nu_t)\right)}{\partial\theta} Q_\pi(h_t, a_t)\right] \end{aligned} \tag{19}$$

where $Pr(h_0 \to h, t, \pi)$ represents the probability of transferring to the historical trajectory h after t time steps from the initial historical trajectory h_0 under the action of the policy π. Calculate the gradient of the actor according to the parameter θ, the stochastic gradient policy theorem of POMDP can be obtained.

Theorem: (POMDP's stochastic policy gradient theory) In the continuous action space, consider the challenge of learning a stochastic policy for POMDP. The stochastic policy is updated as follows:

$$\nabla J(\theta) = \mathbb{E}_{h\sim d_\pi(h)}\mathbb{E}_{a\sim\pi(a|h,\nu)}\left[\nabla_\theta \log(\pi(a|h, \nu))Q_\pi(h, a)\right] \tag{20}$$

where $d_\pi(h)$ is the expected un-normalized history trajectory distribution.

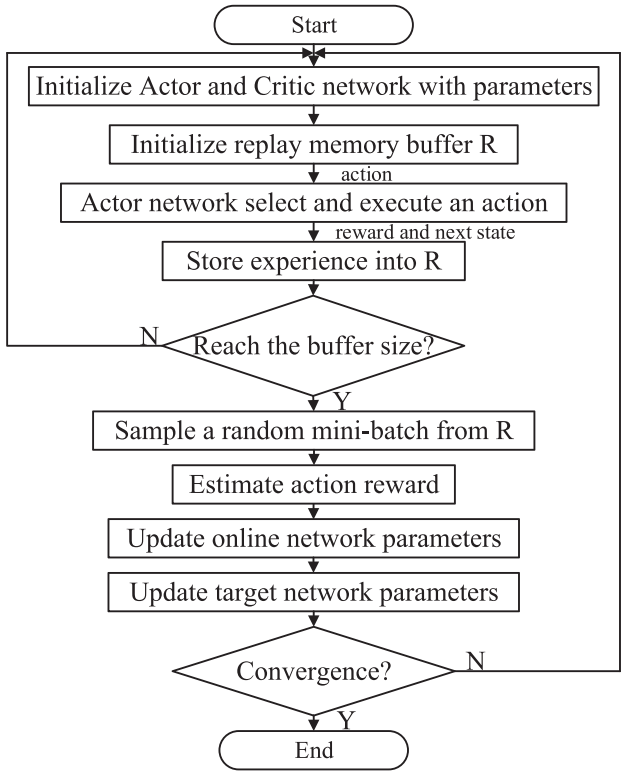


Fig. 3. The flow chart of FRSVG(0) algorithm.

Remark: Rather than the entire episode, the stochastic gradient in (20) is tied to the historical trajectory. The agent now can undertake parameter optimization online every time it interacts with the environment, rather than waiting for the end of the episode, enabling the agent to learn to explore the unknown world faster and better.

The stochastic gradient in (6) is equal to the discount of the gradient in (20) for the entire episode, indicating that the actor-critic parameters in RSVG(0) are actually optimized by a sequence of highly associated historical trajectories. Specifically, assuming that the potential state of h_t is s_t, RSVG(0) actually updates its parameters based on the highly correlated state sequence $(s_0, s_1, s_2 \ldots)$. In the deep reinforcement learning algorithm based on the AC framework, both actor and critic networks are updated by function approximators, which can lead to bias in the state distribution in (6), but the on-policy based stochastic policy gradient algorithm in (20) can be more stable and converge faster. The conclusion will be verified in the following experimental discussion.

The gradient in (20) involves the expectation of the unknown historical trajectory distribution $d_\pi(h)$, so we must estimate it by sampling the state and action space. Although deterministic policies have certain advantages in speed, stochastic policies perform better in unknown environments.A DRL algorithm of Fast Recurrent Stochastic Value Gradients(0) is proposed to improve the inefficiency of data utilization and thus speed up the algorithm convergence, and the FRSVG(0) algorithm with stochastic policy outperforms the algorithm with deterministic

Algorithm 1: FRSVG(0).

Initialize critic network $Q^{\omega}(a_t, h_t)$ and actor $\pi^{\theta}(h_t)$ with parameters ω and θ.
Initialize target critic networks $Q^{\omega'}(a_t, h_t)$ and target actor network $\pi^{\theta'}(h_t)$ with weights $\omega' \leftarrow \omega, \theta' \leftarrow \theta$.
Initialize replay buffer R.
for episodes = 1, M **do**
  Initialize a stochastic process N for action exploration
  Receive an initial observation o_0 (or history trajectory h_0)
  for t = 1, T **do**
    Execute action $a_t = \pi^{\theta}(h_t, \nu)$ with $\nu \sim \beta$, obtain reward r_t and observation o_t
    Store transition (h_{t-1}, a_t, o_t, r_t) into R
    Update history trajectory $h_t = [h_{t-1}, o_t, a_t]$
    Sample a random minibatch of N transition $\{(h_{i-1}, o_i, a_i, r_i)\}_{i=1}^{L}$ from R
    Set $y_t^i = r_t^i + \gamma Q^{\omega'}(h_{t+1}^i, \pi^{\theta'}(h_{t+1}^i, \nu))$ with $\nu \sim \beta$
    Compute critic update (using BPTT)

$$\Delta\omega = \frac{1}{NT}\sum_i \sum_t \left(y_t^i - Q^{\omega}\left(h_t^i, a_t^i\right) \frac{\partial Q^{\omega}\left(h_t^i, a_t^i\right)}{\partial \omega} \right)$$

    Compute actor update (using BPTT)

$$\Delta\theta = \frac{1}{NT}\sum_i \sum_t \frac{\partial Q^{\omega}\left(h_t^i, \pi^{\theta}\left(h_t^i, \nu\right)\right)}{\partial a} \frac{\partial \pi^{\theta}\left(h_t^i, \nu\right)}{\partial \theta}$$

    with $\nu \sim \beta$
    Update the target networks:

$$\omega' \leftarrow \tau\omega + (1-\tau)\omega'$$
$$\theta' \leftarrow \tau\theta + (1-\tau)\theta'$$

  end for
end for

policy in POMDP environment. Fig. 3 is the flow chart of the FRSVG(0) algorithm.

B. The FRSVG(0) Framework

Algorithm 1 shows the proposed algorithm. The FRSVG(0) algorithm framework primarily consists of the recurrent actor network, the recurrent critic network, and an experience replay mechanism. The algorithm takes the observation information of the initial state o_0 as input, and outputs a stochastic action policy $\pi^{\theta}(h_t, \nu)$ with $\nu \sim \beta$ selects the action. During the learning process, the agent outputs an action a_t according to the observation information o_t in the current state s_t at time step t, and evaluates the action according to the obtained reward r_t. Then store the historical observation information, current observation information, action information and reward information $\{h_{t-1}, a_t, o_t, r_t\}$ into the experience pool. At the same time, the neural network randomly samples the historical trajectory from the experience pool to replace the expectation in (20), and updates the network parameters through the stochastic gradient descent method. In practice, experience replay can improve the efficiency of the algorithm. The action policy that maximizes the reward is output to reach the target point by randomly extracting data from the memory buffer for training. Moreover, the FRSVG(0) algorithm divides the network structure into an online network and a target network. The actor network in the online network outputs the stochastic action policy π with parameter ω, and the critic network calculates the Q value evaluation action with parameter θ. The target network maintains the same structure as the online network, and the parameters are ω' and θ' respectively. ω' and θ' delay the update of tracking algorithms ω and θ by soft update, the use of the target network greatly improves the stability of the algorithm. The stochasticity policy can encourage the agent to fully explore the state space and action space [38]. In comparison to RSVG(0), FRSVG(0) can learn policy online using historical trajectories, rather than updating the parameters at the conclusion of each episode. Fig. 4 shows the framework of the FRSVG(0) algorithm.

VI. Simulation Results

In this section, a simulation environment is used to verify and discuss the proposed algorithm.

A. Experimental Settings

As shown in Fig. 5, four different types of large-scale cluttered scenarios were constructed for algorithm validation. Each scenario spans more than 1.0 square kilometers, and the terrain is randomly created with a variety of complex obstacles. Throughout the implementation phase, the agent will random choose an environment in which to advance an episode based on a stochastic policy that has been destroyed by exploration noise. After each episode, the starting and target points of UAV are randomly generated in the scene.

Compared with RSVG(0), the actor and critic in (20) use two LSTMs for approximation to summarize the information obtained from experience. The network structure is shown in Fig. 6. The input of the actor network is the information observed by all sensors $o = [\psi, \xi]$, that is, the distance information perceived by the rangefinder $\psi = \{d_{i0}, d_{i1}, d_{i2}, d_{i3}, d_{i4}\}_{i=0\ldots6}^{7}$ and relative position information of UAV and target point $\xi = [d_7, \varphi_v, \varphi_h]$. The input information has a total of 35 dimensions. After passing through the fully connected layers of size 500 and 400, the LSTM network of size 400 is connected, and the output is the UAV's normalized to [−1,1] steering signal ρ and accelerator ignal σ. The critic network shares the same structure to speed up training, except that it also takes action as input. The speed range of the UAV is limited to [0 m/s, 3m/s], and the flying altitude range is limited to [0 m, 100 m]. The safe space reward was set to $r_{safe} = 0.2$ and the step penalty was set to $r_{step} = -0.5$, respectively. The maximum simulation time for each episode is set to 300 s. In addition, Adam optimizer [47] is used to learn network parameters, and the learning rates for actor and critic are set to 10^{-4} and 10^{-3} respectively. The parameters of the critic network are regularized with weight decay of 10^{-2}, the

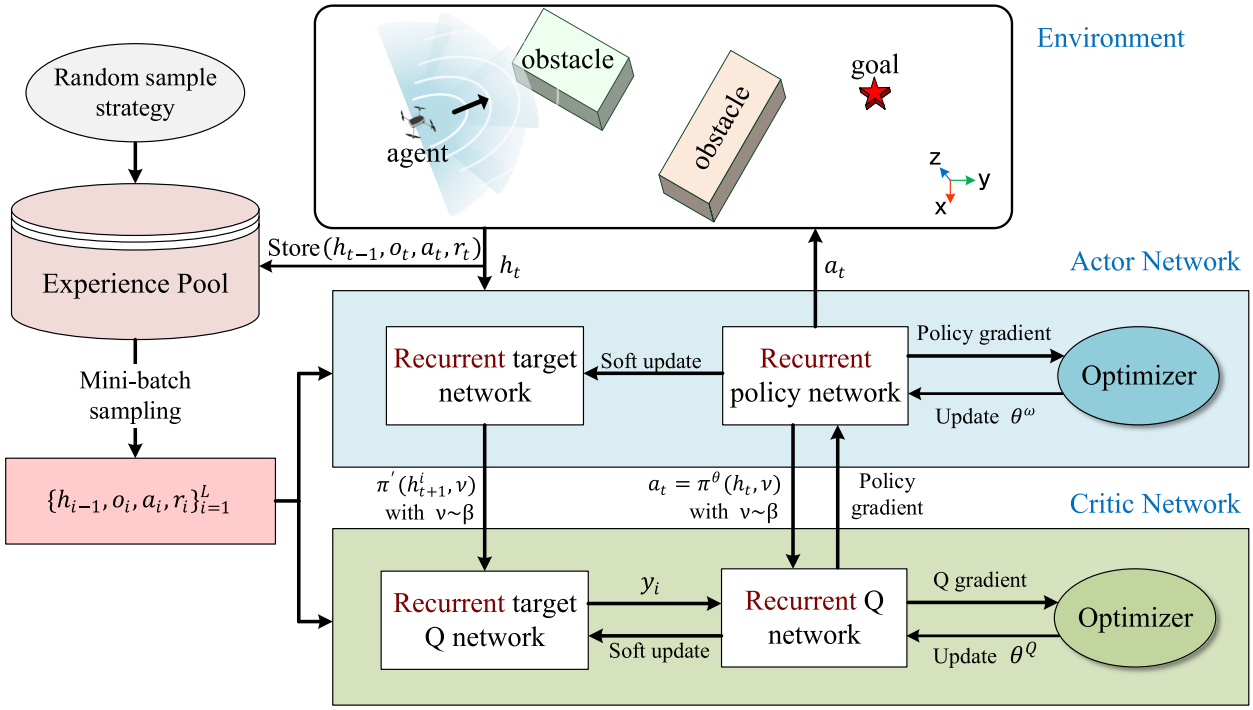


Fig. 4. The framework of FRSVG(0) algorithm for autonomous navigation of UAV.

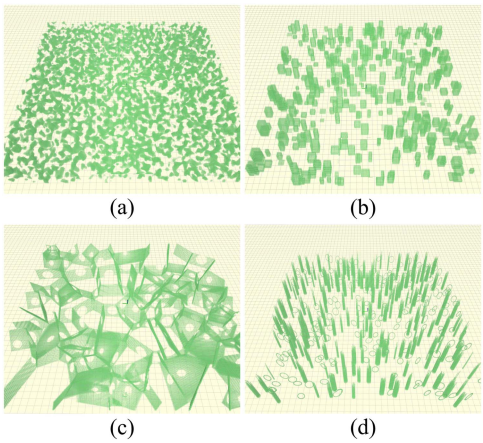


Fig. 5. Demonstration diagram of a simulated large-scale cluttered unknown environment, and call them ENV-I, II, III, IV, respectively. Obstacles (or buildings, plants, etc.) vary in size, shape, and interaction space in various types of environments. At the beginning of each episode, the agent will randomly select a map among the four scenes, and the obstacles in the environment will also be rearranged.

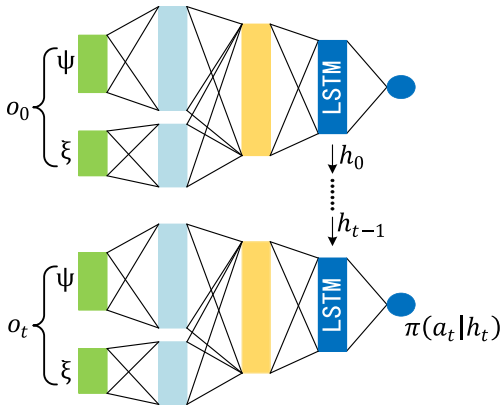


Fig. 6. The structure of the actor network.

discounted factor is $\gamma = 0.99$ and the soft target update rate is $\varepsilon = 0.001$.

B. Navigation Behavior of FRSVG(0)

The trained FRSVG(0) agaent was employed to accomplish numerous navigation tasks in a simulated environment to test the efficiency of the proposed algorithm. It can be seen from Fig. 7 that by summarizing the information from the historical trajectory and issuing control commands, the agent can reach the destination from the point of departure smoothly. Its successful navigation demonstrates that DRL is an effective solution for resolving UAV navigation challenges in an unknown and cluttered environment on a large scale.

To demonstrate that the FRSVG(0) agent with memory outperforms the DDPG agent without memory in navigation tasks, the starting and target points are randomly selected in ENV-IV, which has the highest obstacle coverage and the most cluttered obstacle structure among all scenes, and then navigation experiments are conducted with agents trained by both algorithms. As illustrated in Fig. 8, agents trained on the two algorithms are capable of successfully navigating cluttered environments. However, since the FRSVG(0) agent makes decisions based on previous observations and action sequences, it can benefit from information about the structure of the environment in the memory store and thus does not fall into local traps. DDPG agents, on the other hand, can only make decisions based on

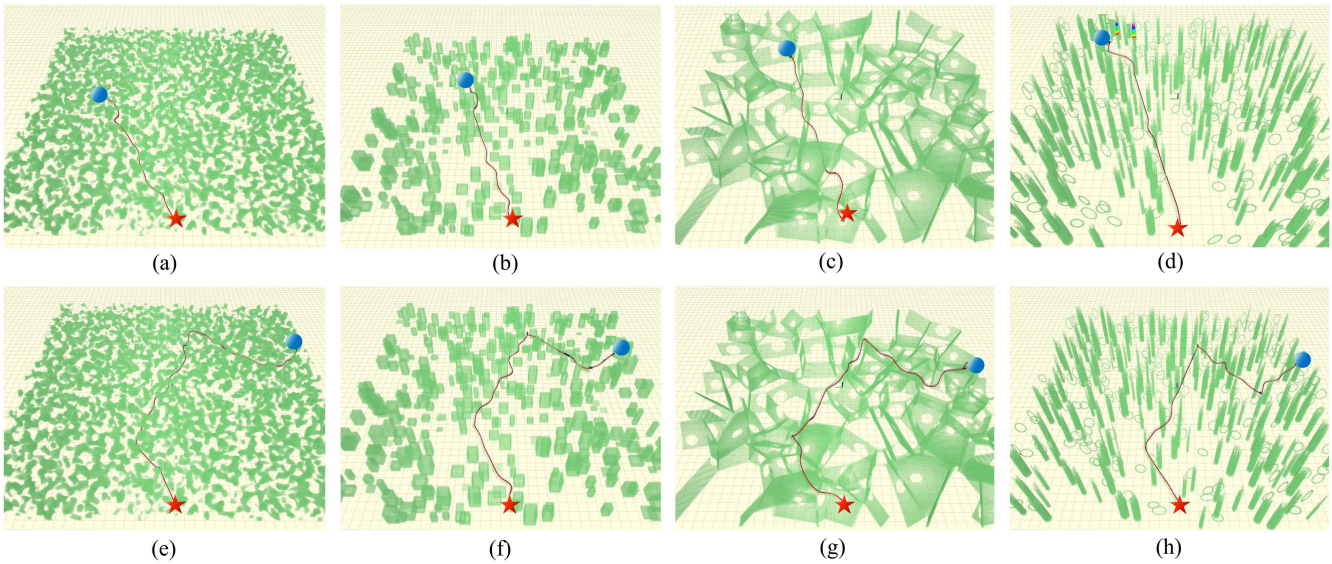


Fig. 7. Schematic diagram of the navigation trajectory of the FRSVG(0) agent trained in the simulation environment. The red circles and blue pentagrams represent the starting point and target point of an arbitrary choice, respectively. The curve connecting it represents the trajectory of the UAV.

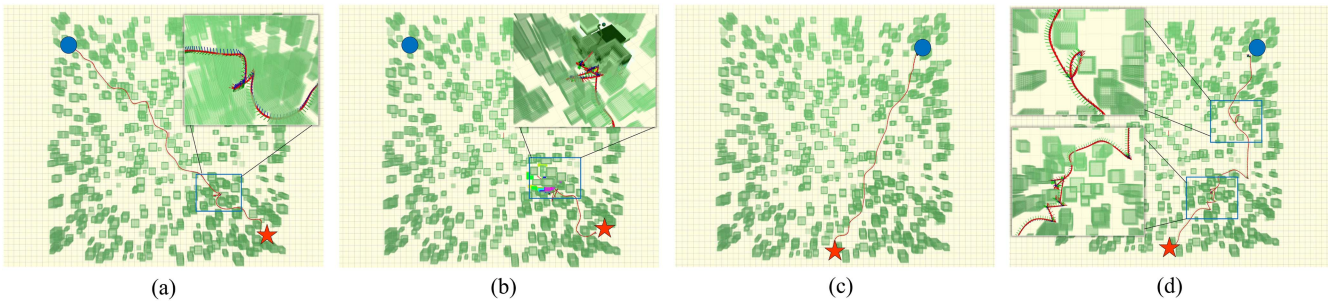


Fig. 8. The top view of the navigation trajectory of the FRSVG(0) agent and the DDPG agent in ENV-IV. (a) and (c) demonstrate the navigation trajectory of the FRSVG (0) agent, and (b) and (d) demonstrate the navigation trajectory of the DDPG agent.

present impressions, and are easily trapped because they can't remember or summarize their experiences.

Similar conclusions can be drawn from the final performance of the two algorithms. The convergence curves for DDPG and FRSVG (0) are shown in Fig. 9, which were obtained by utilizing the Monte Carlo approach to assess the objective function (3). In terms of implementation details, the average reward accumulated over 250 training rounds of agent was used as the reward. Larger rewards indicate better performance of UAV navigation. As can be observed, while the method based on FRSVG(0) is somewhat slower to convergence than the algorithm based on DDPG, when it does, the estimated normalized return of DDPG is significantly less than the estimated normalized return of FRSVG (0). This also demonstrates that stochastic policy is preferable than deterministic policy when it comes to exploring an unknown environment.

The success rate, lost rate, and collision rate of the DDPG agent were evaluated and compared with the proposed FRSVG(0) agent for the navigation tasks in four environments. For each agent conducting 500 navigation tasks in each environment, the success rate is the percentage of agents that

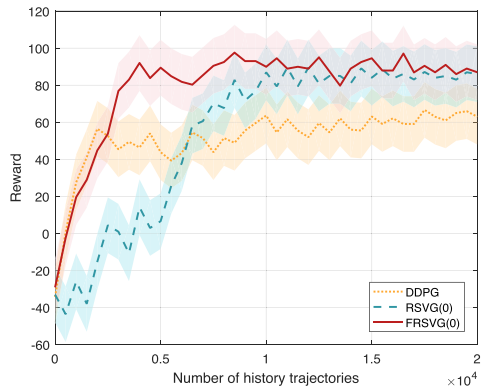


Fig. 9. The reward curves of DDPG, RSVG(0), and FRSVG(0) change as the number of training episodes increases. The average reward using 250 experiments in the operation is used as a normalized Reward.

successfully reach the target point without colliding, the trapped rate is the percentage of agents that are trapped in a local trap and cannot reach the target point within a predetermined time, and the collision rate is the percentage of agents that

TABLE I
STATISTICS OF THE SUCCESS RATE, LOST RATE AND COLLISION RATE OF DDPG, RSVG(0) AND FRSVG(0) IN DIFFERENT SCENARIOS

Results / Environments	Success rate			Collision rate			Trapped rate		
	DDPG	RSVG(0)	FRSVG(0)	DDPG	RSVG(0)	FRSVG(0)	DDPG	RSVG(0)	FRSVG(0)
ENV-I	51.50%	81.00%	**98.25%**	**0.25%**	1.00%	2.25%	47.50%	21.25%	**1.00%**
ENV-II	48.25%	78.75%	**98.50%**	**0.50%**	1.25%	3.50%	59.25%	16.75%	**0.25%**
ENV-III	37.00%	65.25%	**96.75%**	**1.25%**	1.75%	1.75%	52.75%	13.75%	**1.75%**
ENV-IV	34.75%	59.25%	**97.00%**	**0.75%**	13.50%	**0.75%**	61.50%	22.50%	**0.50%**
Average	42.88%	71.06%	**97.63%**	**0.69%**	4.38%	2.06%	55.25%	18.56%	**0.88%**

fail the task due to colliding with an obstacle. The results are shown in Table I. As can be seen, the success rate of DDPG did not exceed 58% in any of the four environments, and the success rate after averaging was only 46.73%, implying that the agent using DDPG has less than a half chance of successfully navigating, which is unacceptable in practical applications. In comparison, the success rate of FRSVG(0) agent exceeded 95% in all environments, with the average success rate of 98.1%, and the UAV navigation performance was significantly improved. Although both algorithms have lower collision rates, the DDPG agent is significantly more likely to get lost in ENV-IV, the most complicated environment, than the FRSVG(0) agent is. This demonstrates that the FRSVG(0) algorithm utilizing recurrent networks can remember past experiences to make decisions, thus reducing the probability of being trapped in a local environment. Naturally, the stochastic policy of FRSVG(0) increases computational effort when compared to the DDPG algorithm, which uses a feed-forward neural network as a function approximator, and this work approximates the actor network and the critic network using two recurrent neural network representations containing LSTMs. The greater complexity of the recurrent neural network results in a more secure navigation behavior than the feedforward neural network, which is critical for practical UAV applications.

C. Performance of FRSVG(0)

The following experiments were performed out with the identical hyperparameters set to confirm that the FRSVG(0) algorithm trained in two stages is more effective than RSVG(0) in accomplishing navigation tasks. As seen in Fig. 9, FRSVG(0) requires just 3000 episodes to attain convergence, whereas RSVG(0) requires 10000 rounds. Because interaction with the real world is fraught with danger in real-world applications, FRSVG(0) converges more quickly and with fewer episodes than RSVG (0). It may be suggested that it is more efficient for learning from small samples. This improvement is thought to be the result of two advancements: Firstly, FRSVG(0) is permitted for online learning. Unlike RSVG(0), which updates the parameters after the episode is complete, the FRSVG(0) agent updates the parameters whenever it interacts with the environment, allowing for the utilization of the most previous experience. Secondly, FRSVG(0) data consumption methods are more efficient. The potential state of the historical trajectory is

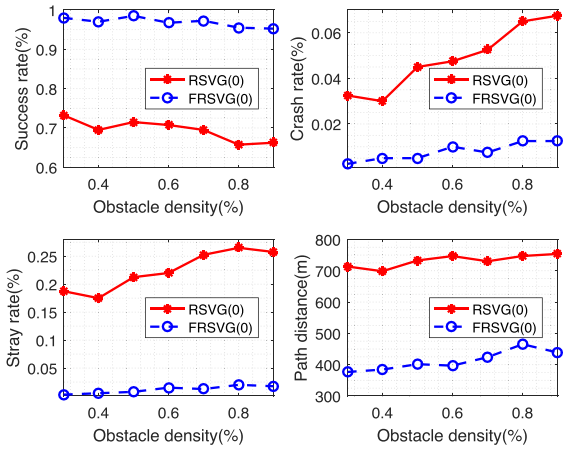


Fig. 10. The success rate, the trappped rate, the collision rate and average trajectory lengths of the FRSVG(0) agent and the RSVG(0) agent after 250 experiments in the environment of different obstacle densities.

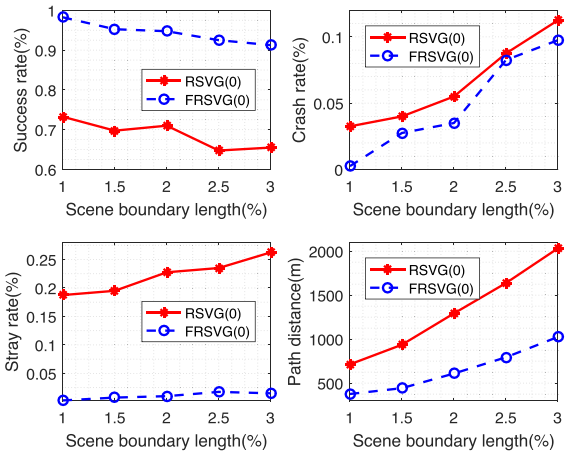
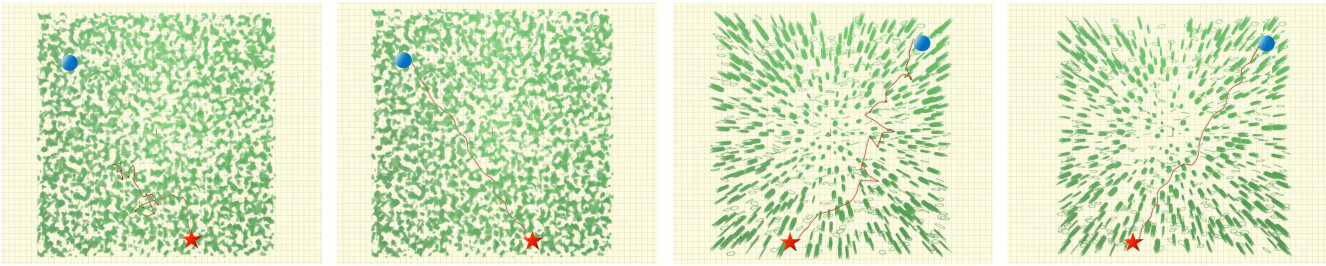


Fig. 11. The navigation success rate, trapped rate, collision rate and average flight length of the FRSVG(0) agent in a cluttered unknown environment at different scales. By setting the simulation environment parameters to expand the environment scale in Fig. 5, the generalization ability of environment verification algorithms of different sizes can be obtained.

heavily associated within an episode, which reduces the data utilization efficiency of RSVG(0). In comparison, FRSVG(0) optimizes parameters using independent historical trajectories sampled from the replay memory, which is more efficient.

While the normalized returns of FRSVG(0) and RSVG(0) convergence approaches the same level, the policies learnt by



(a) (b) (c) (d)

Fig. 12. Schematic diagram of the influence of rewards on navigation behavior. We have selected two sets of representative navigation behaviors in Scene I and Scene II. (a) and (c) show the navigation trajectory of the FRSVG(0) agent in $r_{step} = 0.0$. (b) and (d) show the navigation trajectory of the FRSVG(0) agent at $r_{step} = -0.6$.

TABLE II
FRSVG(0) TRAINING STATISTICS DESIGNED WITH DIFFERENT REWARD PARAMETERS

r_{vel};r_{step}	Success rate	Trapped rate	Collision rate	Flight length
-3.0;0.0	14.75%	72.50%	3.25%	1274.63m
-0.5;-0.6	95.25%	2.25%	1.00%	425.78m

the two are not identical. Table II shows that the RSVG(0) algorithm has a less than 89% chance of successfully completing the navigation task in all scenarios, with an average success rate of only 81.25%, over 17% lower than the proposed FRSVG(0) agent. Additionally, the trapped rate is considerably larger in RSVG(0) agents than in FRSVG(0) agents. As a result, despite the fact that both algorithms are meant to address the POMDP problem, our FRSVG(0) algorithm is more efficient.

The improvement of FRSVG(0) over RSVG(0) is also reflected in its powerful generalization ability. Apart from varying the geometry of obstacles, the simulated environment in which we operate could also alter the size and coverage of obstacles. By increasing the volume of obstacles, the UAV's flight space becomes increasingly complicated, which makes the navigation algorithm more challenging. Navigation experiments were done in settings with varied obstacle densities to fairly evaluate the generalization ability of the FRSVG(0) method and the RSVG(0) algorithm, then the results obtained are given in Fig. 10 and Fig. 11. Fig. 10(a) illustrates that the FRSVG(0) agent's navigation success rate does not drop when the density of obstacles in the scenario increases, whereas the RSVG(0) agent's navigation success rate reduces dramatically. Additionally, in comparison to our FRSVG(0), the trapped rate and collision rate of RSVG(0) both rise as obstacle density increases. Under various obstacle densities, the FRSVG(0) agent has a high success rate, a low trapped rate, and a low collision rate. This demonstrates that our algorithm is capable of not just remembering the structure of a complicated environment, but also of dealing with it. Fig. 10(d) illustrates how the navigation trajectory lengths of FRSVG(0) and RSVG(0) change when the obstacle density rises in the ENV-IV environment. As can be observed, the FRSVG(0) agent's flight distance is less than that of the RSVG(0) agent in the same obstacle density environment, indicating that our FRSVG(0) method is capable of handling it. It performs better in local traps and is capable of escaping the trap and flying to the target. Furthermore, when the obstacle density rises, the navigation task gets more complex, and the trajectory to the target point becomes more tortuous.

Furthermore, a flying experiment was preformed in a larger environmental area, and the results are shown in Fig. 11. As the area of the environment becomes larger, the success rate of both FRSVG(0) and RSVG(0) algorithms decreases, but the trapped rate, collision rate, and flight route distance all increase. Nonetheless, our FRSVG(0) agent achieves a success rate of more than 92% in the biggest region of the environment, which is 14% higher than the success rate of the smallest RSVG(0) agent. Furthermore, the FRSVG(0) agent's trajectory distance is less than that of the RSVG(0) agent, which becomes more apparent in more complicated environments. Despite the fact that the agents have never been trained in large-scale scenarios, the proposed FRSVG(0) algorithm gives a high generalization capacity.

D. The Impact of Reward

The navigation behavior of FRSVG(0) is influenced not only by the design of the observation state, but also by the reward parameters. The research presented in this study focuses into the effects of reward on UAV activities by changing safe space reward, step penalty, and speed penalty. By resetting the step penalty to $r_{step} = 0.0$, and the speed penalty to $r_{vel} = -3.0$, and retrain the FRSVG(0) agent with the other parameters unchanged. The performance of the agent acquired by training with the original settings after altering the two reward parameters is shown in Table II. As shown in the table, when the step penalty is set to 0 and the maximum speed penalty is set to -3.0, the agent's action selection becomes more conservative and prefers to hover safely. As a result, the success rate was reduced by 80%, and the trapped rate and average flight path increased from 2.25% and 425.78 m to 72.50% and 1274.63 m respectively. At this point, the agent's flying behavior became inefficient, and the agent was urged to fly to a safe area, ignoring the goal. Flying is shown in Fig. 12. When an agent is punished for taking a small number of steps, the trained agent is easily transferred to a free area and is unable to reach the goal. Second, when the speed penalty is reduced to 0, the agent is more likely to fly at a high rate of speed

near the obstacle, increasing the collision rate. Thus, although non-sparse rewards may be used in conjunction with domain knowledge to accelerate learning and enhance performance, the hyperparameter design portion needs careful planning and many trials to fine-tune.

VII. Conclusion and Future Work

This research focuses on the development of a deep reinforcement learning algorithm for safe UAV navigation in large scale unknown complicated situations. This problem was modeled as a POMDP, and it is suggested to solve it using a DRL algorithm based on the stochasticity value function. The use of the DRL algorithm eliminated the requirement for map rebuilding and path planning, enabling the UAV to travel from any location to the destination. FRSVG(0) is more efficient than RSVG(0), as shown by theoretical derivation and simulation data. Additionally, it may be expanded to larger, more complex, and distributed applications. Nonetheless, the sensor used in this article was a rangefinder. In actuality, though, a camera or radar may be a better option. There is much work that can be done.

References

[1] H. Shakhatreh et al., "Unmanned aerial vehicles (UAVs): A survey on civil applications and key research challenges," *IEEE Access*, vol. 7, pp. 48572–48634, 2019.

[2] H. Li and A. V. Savkin, "Wireless sensor network based navigation of micro flying robots in the industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3524–3533, Aug. 2018.

[3] S. Hening, C. A. Ippolito, K. S. Krishnakumar, V. Stepanyan, and M. Teodorescu, "3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments," in *Proc. AIAA Inf. Syst.-AIAA Infotech, Aerosp.*, 2017, Art. no. 0448.

[4] G. A. Kumar, A. K. Patil, R. Patil, S. S. Park, and Y. H. Chai, "A LiDAR and IMU integrated indoor navigation system for UAVs and its application in real-time pipeline classification," *Sensors*, vol. 17, no. 6, 2017, Art. no. 1268.

[5] H. Zhou, D. Zou, L. Pei, R. Ying, P. Liu, and W. Yu, "StructSLAM: Visual SLAM with building structure lines," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1364–1375, Apr. 2015.

[6] H. Qin et al., "Autonomous exploration and mapping system using heterogeneous UAVs and UGVs in GPS-denied environments," *IEEE Trans. Veh. Technol.*, vol. 68, no. 2, pp. 1339–1350, Feb. 2019.

[7] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *Int. J. Robot. Res.*, vol. 25, no. 5-6, pp. 403–429, 2006.

[8] R. Gomez-Ojeda, F.-A. Moreno, D. Zuniga-Noël, D. Scaramuzza, and J. Gonzalez-Jimenez, "PL-SLAM: A stereo slam system through the combination of points and line segments," *IEEE Trans. Robot.*, vol. 35, no. 3, pp. 734–746, Jun. 2019.

[9] S. Liu, M. Watterson, S. Tang, and V. Kumar, "High speed navigation for quadrotors with limited onboard sensing," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1484–1491.

[10] W. Dong, G.-Y. Gu, X. Zhu, and H. Ding, "High-performance trajectory tracking control of a quadrotor with disturbance observer," *Sensors Actuators A: Phys.*, vol. 211, pp. 67–77, 2014.

[11] G. Cho, J. Kim, and H. Oh, "Vision-based obstacle avoidance strategies for MAVs using optical flows in 3-D textured environments," *Sensors*, vol. 19, no. 11, 2019, Art. no. 2523.

[12] W. G. Aguilar, L. Álvarez, S. Grijalva, and I. Rojas, "Monocular vision-based dynamic moving obstacles detection and avoidance," in *Proc. Int. Conf. Intell. Robot. Appl.*, Springer, 2019, pp. 386–398.

[13] X.-Z. Peng, H.-Y. Lin, and J.-M. Dai, "Path planning and obstacle avoidance for vision guided quadrotor UAV navigation," in *Proc. 12th IEEE Int. Conf. Control Automat.*, 2016, pp. 984–989.

[14] K. R. Sapkota et al., "Vision-based unmanned aerial vehicle detection and tracking for sense and avoid systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 1556–1561.

[15] I. Mahjri, A. Dhraief, A. Belghith, and A. S. AlMogren, "SLIDE: A straight line conflict detection and alerting algorithm for multiple unmanned aerial vehicles," *IEEE Trans. Mobile Comput.*, vol. 17, no. 5, pp. 1190–1203, May 2018.

[16] C. Luo, S. I. McClean, G. Parr, L. Teacy, and R. De Nardi, "UAV position estimation and collision avoidance using the extended kalman filter," *IEEE Trans. Veh. Technol.*, vol. 62, no. 6, pp. 2749–2762, Jul. 2013.

[17] S. T. Goh, O. Abdelkhalik, and S. A. R. Zekavat, "A weighted measurement fusion Kalman filter implementation for UAV navigation," *Aerosp. Sci. Technol.*, vol. 28, no. 1, pp. 315–323, 2013.

[18] M. Mallick and A. Marrs, "Comparison of the KF and particle filter based out-of-sequence measurement filtering algorithms," in *Proc. 6th Int. Conf. Inf. Fusion*, 2003, pp. 422–430.

[19] P. Furgale and T. D. Barfoot, "Visual teach and repeat for long-range rover autonomy," *J. Field Robot.*, vol. 27, no. 5, pp. 534–560, 2010.

[20] A. Miller and B. Miller, "Stochastic control of light UAV at landing with the aid of bearing-only observations," in *Proc. 8th Int. Conf. Mach. Vis., Int. Soc. Opt. Photon.*, 2015, vol. 9875, Art. no. 987529.

[21] S. Karpenko, I. Konovalenko, A. Miller, B. Miller, and D. Nikolaev, "Visual navigation of the UAVs on the basis of 3D natural landmarks," in *Proc. 8th Int. Conf. Mach. Vis., Int. Soc. Opt. Photon.*, 2015, vol. 9875, Art. no. 98751I.

[22] Y. Zhang, X. Yuan, Y. Fang, and S. Chen, "UAV low altitude photogrammetry for power line inspection," *ISPRS Int. J. GEO- Inf.*, vol. 6, no. 1, 2017, Art. no. 14, doi: 10.3390/ijgi6010014.

[23] F. Expert and F. Ruffier, "Flying over uneven moving terrain based on optic-flow cues without any need for reference frames or accelerometers," *Bioinspiration Biomimetics*, vol. 10, no. 2, 2015, Art. no. 026003.

[24] A. Faust et al., "PRM-RL: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 5113–5120.

[25] H. X. Pham, H. M. La, D. Feil-Seifer, and L. V. Nguyen, "Autonomous UAV navigation using reinforcement learning," 2018, *arXiv:1801.05086*. [Online]. Available: http://arxiv.org/abs/1801.05086

[26] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1088–1095, Apr. 2018.

[27] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1-2, pp. 99–134, 1998.

[28] M. T. Spaan, "Partially observable Markov decision processes," in *Reinforcement Learning*. Berlin, Germany: Springer, 2012, pp. 387–414.

[29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3, pp. 229–256, 1992.

[30] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Solving deep memory POMDPs with recurrent policy gradients," in *Proc. Int. Conf. Artif. Neural Netw.*, 2007, pp. 697–706.

[31] F. Jurčíček, B. Thomson, and S. Young, "Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs," *ACM Trans. Speech Lang. Process.*, vol. 7, no. 3, pp. 1–26, 2011.

[32] D. Wierstra and J. Schmidhuber, "Policy gradient critics," in *Proc. Eur. Conf. Mach. Learn.*, Springer, 2007, pp. 466–477.

[33] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

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

[35] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," in *Proc. AAAI Fall Symp. Ser.*, Sep. 2015, pp. 29–37.

[36] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*. [Online]. Available: https://arxiv.org/abs/1509.02971

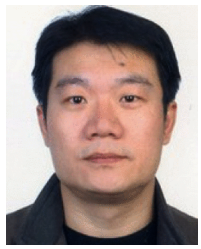
[37] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[38] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015, *arXiv:1512.04455*. [Online]. Available: https://arxiv.org/abs/1512.04455

[39] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *Proc. 14th Int. Conf. Control, Automat., Robot. Vis.*, 2016, pp. 1–6.

[40] A. Singla, S. Padakandla, and S. Bhatnagar, "Memory-based deep reinforcement learning for obstacle avoidance in UAV with limited environment knowledge," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 107–118, Jan. 2021.
[41] G. Tong, N. Jiang, L. Biyue, Z. Xi, W. Ya, and D. Wenbo, "UAV navigation in high dynamic environments: A deep reinforcement learning approach," *Chin. J. Aeronaut.*, vol. 34, no. 2, pp. 479–489, 2021.
[42] C. Wang, J. Wang, Y. Shen, and X. Zhang, "Autonomous navigation of UAVs in large-scale complex environments: A deep reinforcement learning approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2124–2136, Mar. 2019.
[43] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[44] S. P. Singh, T. Jaakkola, and M. I. Jordan, "Learning without state-estimation in partially observable Markovian decision processes," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1994, pp. 284–292.
[45] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, vol. 99, 1999, pp. 278–287.
[46] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing arbitrary reward functions as potential-based advice," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2652–2658.
[47] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–41.

Yuntao Xue received the B.Sc. degree from the Taiyuan University of Technology, Taiyuan, China, in 2017. He is currently working toward the Ph.D. degree with the School of Aerospace Science and Technology, Xidian University, Xi'an, China. His research interests include autonomous UAV navigation and deep reinforcement learning.



Weisheng Chen received the B.S. degree in mathematics and applied mathematics from the Department of Mathematics, Qufu Normal University, Qufu, China, in 2000, the M.Sc. degree in operational research and cybernetics and Ph.D. degree in applied mathematics from the Department of Applied Mathematics, Xidian University, Xi'an, China, in 2004 and 2007, respectively. From 2008 to 2009, he was a Visiting Scholar with the Automation School, Southeast University, Nanjing, China. He is currently a Professor with the School of Aerospace Science and Technology, Xidian University. His research interests include multiagent systems, adaptive control, event-triggered control, distributed optimization, learning, and control.