Multi-Agent Deep Reinforcement Learning for UAVs Navigation in Unknown Complex Environment

Yuntao Xue and Weisheng Chen

***Abstract*—As unmanned aerial vehicles (UAVs) play an increasingly significant role in modern society, using reinforcement learning to build safe multi-UAV navigation algorithms has become a hot topic. One of the major issues is coordinating multi-UAV to navigate safely in complicated and unknown 3D environments. The current decentralized navigation systems suffer from the challenges of partially observable properties in unknown environments and unstable environments, resulting in poor learning effects. In this article, we proposed a new multi-agent recurrent deterministic policy gradient (MARDPG) algorithm based on the depth deterministic policy gradient algorithm for controlling the navigation action of multi-UAV. Specifically, each critic network learns centrally so that each UAV can estimate the policies of other UAVs, thereby resolving the problem of slow convergence caused by the unstable environment. Decentralized execution eliminates the need for communication resources between UAVs. Sharing parameters in critics network accelerates training. The added LSTM network enables UAVs to use historical exploration information to improve the prediction of action values without being trapped by local traps. Finally, thorough simulation results in a realistic simulation environment were provided to demonstrate the superiority in terms of convergence and efficacy over the state-of-the-art DRL approach.**

***Index Terms*—Multi-UAV navigation, autonomous systems, recurrent deterministic policy gradient, partially observable Markov decision process.**

I. Introduction

MULTI-UAV navigation is gaining popularity in the fields of robotics and artificial intelligence, and it has a wide range of applications, including search and rescue, wireless services, and crop inspections [1], [2], [3], [4], [5]. To perform successfully in the above applications, UAVs must have robust navigation capabilities, which implies that each UAV should be able to safely navigate from its starting place to the target point. However, in the actual world, the complicated workspace presents a significant difficulty for multi-UAV navigation. Due to their faster flying speeds and the peculiarities of aviation, UAVs, as opposed to conventional ground robots, have the potential to severely hurt pedestrians and property [6]. As a result, we concentrate on developing effective multi-UAV navigation algorithm in unknown complex environments.

With the advances in multi-UAV navigation research, various multi-UAV navigation system solutions have been developed, such as centralized-based approaches [7], [8], [9], [10]. The centralized-based approach assumes that a central server determines all agents' behaviors based on global knowledge about all agents, such as their state and observations. The centralized approach generates global control commands based on global observations, which can guarantee safety, integrity, and near-optimality. However, the high dependence on communication and the high consumption of computing resources affect the extension of this approach to more practical scenarios. In addition, this approach has low anti-interference performance. Once the central server is attacked, the entire system will be paralyzed, which is fatal and intolerable.

Traditional decentralized multi-UAV navigation systems [11] have been presented to overcome these issues, in which each agent makes an independent decision after considering the state inputs from its neighbors. The majority rules of such techniques are based on the velocity obstacles framework [11], [12], [13], which can avoid agent collisions and thus navigate safely in cluttered environments. However, it has a number of serious limitations that make implementation challenging. Firstly, it is assumed that each agent in the simulated environment needs to communicate velocity and attitude information with neighboring agents, which in practice requires the addition of communication bands and is susceptible to interference, such as the presence of complex electromagnetic signals and building occlusions in cities. Secondly, traditional agent-based velocity obstacles algorithms have many configurable parameters and necessitate complicated parameter fine-tuning designs for different situations, further limiting their generalization. Finally, the traditional decentralized approach is inferior than the centralized approach in terms of navigation speed and time performance.

There are new possibilities to develop autonomous navigation with recent advances in artificial intelligence and deep learning technology. The agent control process has previously been defined as MDP [14] and solved using reinforcement learning, which has been widely used in robot obstacle avoidance and navigation tasks. However, unprocessed high-dimensional sensing input information can lead to the inability of traditional RL algorithms to store a large number of value functions. To tackle this challenge, researchers combined deep learning (DL) and

Manuscript received 17 May 2023; revised 12 June 2023 and 12 July 2023; accepted 18 July 2023. Date of publication 24 July 2023; date of current version 23 February 2024. This work was supported by the National Natural Science Foundation of China under Grants 62073254 and 92271101. *(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).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIV.2023.3298292.

Digital Object Identifier 10.1109/TIV.2023.3298292

reinforcement learning (RL), resulting in deep reinforcement learning (DRL) [15], [16]. Compared with traditional decentralized navigation systems, deep reinforcement learning can process pixel-level image state input and map it into control commands for the agent. Although DRL algorithms have achieved good results in single UAV navigation situations, when applied to multi-UAV navigation, there are problems such as training failure to converge. The difficulty of multi-UAV navigation in complicated unknown circumstances will become more relevant in real-world tasks; for example, multi-UAV can significantly improve efficiency in cases like search and rescue in disaster areas and cargo delivery.

The navigation behaviors of different UAVs can affect each other in the situation of multi-UAV navigation. Each UAV cannot fully perceive the complete information of its environment, and it can only perceive local observations around itself, in this regard, the decision process fits the framework of decentralized partially observable Markov decision processes (Dec-POMDPs), then the multi-UAV navigation problem was solved by modeling it as Dec-POMDPs and designing an algorithm based on deep reinforcement learning in this work. In the partially observable case, the UAV does not have access to global information and is prone to local dilemmas in navigation, to give the UAV memory, we apply recurrent networks, which allow the past state to impact the decision of future action.

The main contributions of the approach proposed in this article are as follows.

1) We construct the navigation of multi-UAVs in an unknown and complex environment as a Dec-POMDPs problem, and propose a centralized training and decentralized execution MADRL algorithm to solve this problem. The centralized learning of each critic network enables each agent to consider the estimated policies of other agents when making decisions. Decentralized execution enables each agent to make decisions independently without communication.
2) We propose a MARDPG algorithm that integrates the recurrent neural network into MADRL to capture latent state information. Experiments in a simulation environment with limited UAV field of view prove that the extraction of historical information is helpful for multi-UAV navigation decisions.
3) To speed up training, we design more comprehensive state and reward information combined with domain knowledge combined with navigation. Additionally, the utilization of distance information collected by the rangefinder helped bridge the gap between the simulated environment and real-world applications. Experimentally verified, our strategy can be easily extended to other unseen complex environments.

The rest of this article is structured as follows. Related work is discussed in Section II. Section III presents the navigation model and problem formulation. Section IV defines the DRL model for UAV navigation. Section V shows the algorithm details. Section VI evaluates the algorithm performance in experiments. Section VII concludes this article. Notation is given in Table VI.

II. Related Work

Fan et al. [17] divided decentralized navigation methods into two types: sensor-level methods that directly map raw sensor information into control signals and agent-level methods that consider the observable states of other agents (information such as speed and position).

A. Sensor-Level Methods

The sensor-level decentralized navigation strategy is to directly map raw sensor information into collision-free control commands. Sensor-level navigation strategies are often modeled based on deep neural networks, trained using supervised learning on large datasets [18]. In [19], a convolutional neural network was trained to map raw pixels from a single camera to steering commands, validating the feasibility of an end-to-end approach for sensor-level navigation. Chen [20] trained a dual DRQN neural network model with LSTM units to deploy to a mobile robot for collision-free navigation. These works perform well on mobile robots with low moving speeds and small observation dimensions, but cannot be directly applied to UAV navigation with increased uncertainty due to high dimensions. Zhu [21] proposed a POMDP-based decentralized decision-making system for multi-UAV search targets in GPS-denied environments. However, it assumes that the map of the environment is known, making it difficult to extend the navigation strategy to unknown environments. Based on the high-dimensional observation of UAVs in the actual environment and more environmental noise characteristics, if the observable state information of the surrounding agents can be used, better navigation performance can be achieved. Therefore, in this article, the multi-UAV navigation problem was modeled as Dec-POMDPs, then the agent-level decentralized navigation method was used to solve it.

B. Agent-Level Methods

Agent-level decentralized navigation can be divided into two categories. One is the obstacle avoidance method based on velocity obstacles. Initially, the most classic ORCA-3D [11], [22], [23] provided sufficient conditions for multi-robot to avoid collisions when moving in a 3D dense and complex environment, and can calculate collision-free actions for all robots in a short time. By combining ORCA with deep reinforcement learning, the CADRL algorithm was proposed in [24], which developed real-time achievable interaction rules by learning a value function that implicitly encodes cooperative behavior. However, these methods are based on the assumption of perfect perception, which brings challenges to realistic sensor accuracy. Wang [6] proposed a distributed agent-level obstacle avoidance policy based on reinforcement learning with two-stage training, which can reach faster convergence rate. Another class of agent-level approaches split navigation into goal assignment and path planning problems. In [25], the MADDPG algorithm [26] was used to solve the target assignment and path planning problems of multi-UAV in a dynamic environment and achieve real-time performance. However, although these efforts have made significant

progress, they either ignore or partially consider, the problem of multi-UAV obstacle avoidance in environments with complex obstacles and the problem of UAVs making better decisions based on historical trajectories. In this article, an actor-critic model using a recurrent network structure is proposed to deal with the above-mentioned multi-UAV navigation collaboration problem in partially observable environments.

III. System Model and Problem Formulation

A. UAV Model

The UAV in this article is set up as a quadrotor and its model contains twelve states: absolute position in the world coordinate system (x, y and z), Euler angles ($\phi - roll$, $\theta - pitch$ and $\psi - yaw$) representing the rotation of the UAV, velocity in the body frame coordinate system (v_x, v_y and v_z) and angular velocity (p, q and r). The position, direction, and motion of the UAV are represented by the world coordinate system and the body coordinate system, which is called to as the UAV's internal state description.

For simplicity, we assume that the UAV is flying at a fixed speed in a 3D environment. We ignore the momentum of the UAV in flight and assume that the steering action and throttle action are immediately effective. Therefore, the dynamics of the UAV is given by

$$\begin{cases} \vartheta_{t+1} = \vartheta_t + \rho_t \\ \varphi_{t+1} = \varphi_t + \tau_t \\ x_{t+1} = x_t + v_t \times \cos\left(\vartheta_{t+1}\right) \cos\left(\varphi_{t+1}\right) \\ y_{t+1} = y_t + v_t \times \sin\left(\vartheta_{t+1}\right) \cos\left(\varphi_{t+1}\right) \\ z_{t+1} = z_t + v_t \times \sin\left(\varphi_{t+1}\right) \end{cases} \tag{1}$$

where ϑ_t and φ_t are the directional angles of the UAV in the horizontal and vertical planes, respectively, ρ_t and τ_t are the steering signals of the two angles, respectively, and v is the the constant velocity.

B. Description of Navigation Tasks

In this article, DRL is applied to control multi-UAV so as to accomplish navigation in complex environments. The environmental information is sensed by on-board sensors so that the UAV can be controlled accordingly. The basic DRL model is shown on the left side of Fig. 1. We control the UAV by adjusting its flight angle to avoid obstacles, ensuring a safe trajectory to the target location. After each maneuver, the UAV transitions to a new state and receives feedback from the environment, enabling it to update its strategy and optimize its behavior accordingly. To estimate the Q value and choose the action, two deep neural networks are used. Before each selection action, all state information from all parts must be integrated. In the case of multi-UAV, global information must be taken account.

The question is how to make a rational choice on the action selection of UAVs utilizing previous environmental information and the current state in order to achieve obstacle avoidance and complete navigation in complex environments. Each UAV is an agent. Therefore, the problem can be defined as a multi-agent problem. Different UAVs observe different things, and the actions of individual UAVs affect each other, and coordination among all individual UAVs is important. Therefore, navigation by coordinating multi-UAV and combining spatio-temporal relations is our ultimate goal.

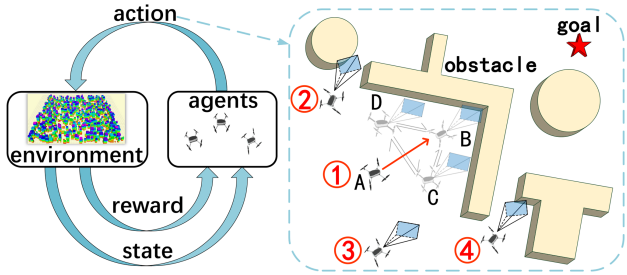


Fig. 1. Schematic diagram of multi-UAV navigation observability in complex environment. The left side shows the reinforcement learning interaction process, and the right side shows the navigation behavior of multi-UAV in the environment. The blue cone in front of the UAV shows its perception range, and the red arrow indicates that the drone is encouraged to move towards the target.

IV. Deep Reinforcement Learning Model

In this section, to address the problem of navigating multi-UAV, we first introduce by example why navigation of UAVs in complex environments is partially visible in the state. Then describe the details of Dec-POMDPs modeling.

A. UAV Navigation as a Dec-POMDPs

As shown in Fig. 1, if the target point of UAV 1 is on the other side of the obstacle, it will be encouraged to approach the target to reach point B. Due to the limited sensing distance of the on-board sensors (blue cone), the UAV can only detect obstacles ahead after reaching point C. For the sake of safety, the UAV will retreat to point A, or explore to the left and right sides to point C and point D. But when there is no obstacle within the sensing distance, the UAV will move towards point B again, and finally get trapped. If the UAV can remember the environment it has experienced previously, it will be able to make better decisions by gradually constructing the local environment structure based on its knowledge. During the centralized training stage, the UAV could learn the environment structure from the observations and actions of other UAVs, and if a neighboring UAV has traversed the same path, the UAV can learn the environment structure from the experience of other UAVs to determine a safe and efficient path. Each UAV can only observe its surrounding area and make decisions based on its historical trajectory, which can be described as Dec-POMDPs.

B. Observation and Agent Action

The UAV needs to receive the following three pieces of information in order to navigate to a target point in an environment with obstacles: its state information, the interaction with the environment, and the relationship with the target point. First, since the environment is unknown, the UAV cannot obtain its absolute position information, so we discard the position $[x, y, z]$ representation in the world coordinate system and use

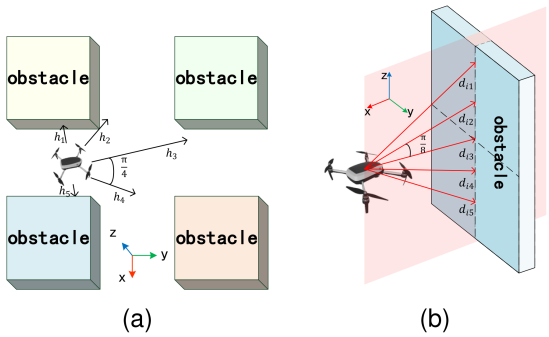


Fig. 2. (a) Schematic representation of the perception of the rangefinder carried by the UAV on the horizontal plane. (b) Schematic representation of the rangefinder perception carried by the UAV on the vertical plane. The horizontal and vertical planes contain a total of 25 perception data.

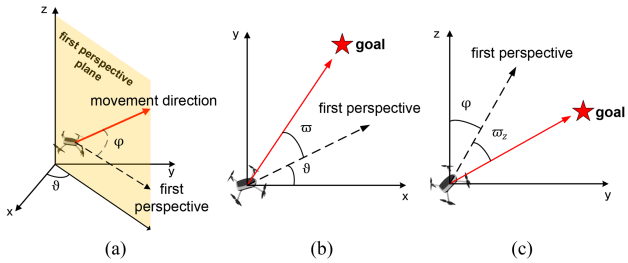


Fig. 3. (a) Represents the position of the UAV in the 3D world schematically. (b) and (c) Represent the relative positions of the UAV and the goal point in the x-y plane and the y-z plane, respectively.

the angle between the first perspective and the x-axis $[\vartheta]$, and the angle between the first perspective and the horizontal line $[\varphi]$ to describe its internal state, as shown in Fig. 3. A gyroscope can be used to measure these two angles in the real world. Second, a variety of sensory senses, such as visible cameras, radar, and ultrasonic rangefinders, can be used to determine the connection between the UAV and the surrounding obstacles. To bridge the gap from simulation to the real world, this article uses 25 ultrasonic rangefinders to sense obstacles, denoted as $\psi = \{d_{i0}, d_{i1}, d_{i2}, d_{i3}, d_{i4}\}_{i=1}^{5}$, as shown in Fig. 2. Finally, the UAV needs to detect its relative position and angular relationship with the target point. This information about the target position is input to the UAV at the beginning of the task, $\xi = [d_5, \varpi, \varpi_z]$. The above three observations together form the observation space of the UAV, namely $o = [\vartheta, \varphi, \psi, \xi]$, where $\vartheta, \varphi, \varpi, \varpi_z \in [-\pi, \pi]$, $\{d_{i0}, d_{i1}, d_{i2}, d_{i3}, d_{i4}\}_{i=1}^{5} \in [0, 10]$, and $d_5 \in [0, \infty]$.

Each UAV selects the action at each time step and shifts to a new state s_{t+1}. We need to control the steering angle and speed of the UAV. To prevent the UAV's control signals from changing too drastically in magnitude, we set the steering signals ρ and τ on the pinch angle ϑ and φ, respectively, to be limited to $[-\pi/6, \pi/6]$ at each time step.

C. Reward Design

An appropriate reward function must be chosen as feedback for the action in order for each agent to gradually learn to make the most rational decisions concerning environmental obstacles. Each decision is guided by appropriate rewards. Sparse rewards are a common method in navigation problems, in which the UAV reaches the goal point before receiving a reward, and this delayed feedback makes training in large and complicated environments inefficient. The approach takes a long time to converge, or may never converge. We combine domain knowledge of navigation with a generally acceptable policy to build a non-sparse reward for the agents to simulate real-world situations. The reward consists of four components: transfer reward, collision penalty, free space reward and step penalty. The transfer reward is denoted as:

$$r_{trans} = \alpha \left(d_s - d_{s-1} \right) \tag{2}$$

where α is a constant and $d_s - d_{s-1}$ denotes the shortened distance from the goal point after the UAV makes action at a time step. Transfer rewards can be used to encourage the UAV to get closer to the destination while preventing it from going away from the target point. Furthermore, the most crucial aspect of navigating complex environments for a UAV is to ensure its safety. The repercussions of colliding with an obstacle or a neighboring UAV would be catastrophic. Besides that, the UAV performs in a three-dimensional environment, to prevent it from flying just as high to avoid obstacle behavior, it will be penalized when it approaches a predefined altitude, which can essentially be considered the top obstacle. As a result, the two can be represented as a distance relationship with the UAV, and the collision penalty is described as follows:

$$r_{col} = -\lambda e^{-\sigma d_{min}} \tag{3}$$

where λ and σ are two constants and d_{min} is the shortest distance between the UAV and obstacles (obstacles include environmental static obstacles and other UAVs). The minimum distance d_{min} between the UAV and the obstacle and the minimum distance between the UAV and other UAVs are set to R and $2R$, respectively, where R is the radius of the spatial volume of the UAV. In addition, when the UAV's first perspective rangefinder detects no obstacle, it receives a free space reward r_{free}, which encourages it to explore toward a safe space. Finally, the UAV is penalized with a constant number of steps r_{step} for each movement to prevent it from making pointless and redundant moves. The total non-sparse reward can be expressed as:

$$r = \delta_1 r_{trans} + \delta_2 r_{col} + \delta_3 r_{free} + \delta_4 r_{step} \tag{4}$$

where δ_1, δ_2, δ_3 and δ_4 are scale factors. Different examples may lead to different policies. Fine-tuning of the parameters is required to obtain the desired policies, and details of the parameters are described in the experiments section.

V. Deep Reinforcement Learning Algorithm for UAVs Control

A. The MARDPG Algorithm

This work models the process of observing the environment and thus making decisions by UAVs with limited sensing capabilities as partially observable Markov decision processes [27]. In a navigation scenario with multi-UAV, this can be modeled as decentralized partially observable Markov decision processes (Dec-POMDPs) [28]. Dec-POMDPs are defined as 8-tuple

$(\mathcal{I}, \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{O}, \mathcal{R}, \gamma)$, where $\mathcal{I} = \{1, \ldots, N\}$ denotes N agents, $\mathcal{S}$ denotes the set of states of all agents, $\mathcal{S}$ describes the possible configurations of all UAVs, $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_N\}$ denotes the set of actions, $\Omega = \{\mathcal{O}_1, \ldots, \mathcal{O}_N\}$ denotes the set of partial observables by agents, and the transfer probability function is denoted as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \mapsto \mathcal{S}$, the observation probability function is denoted as $\mathcal{O} : S \mapsto \mathcal{O}$, $\mathcal{R}$ denotes the reward function, and $\gamma \in [0, 1)$ denotes the discount factor. At each time step t, the agent i receives a partial observation o_1^t and executes action a_t^i according to the stochastic policy $\pi^i(h_t^i; \theta^i)$ (or the deterministic policy $\mu^i(h_t^i; \theta^i)$), where θ^i is the policy parameter of agent and h_t^i is the observation history of agent i. The current state s_t of the Dec-POMDP takes the joint action set $\{a_t^1, \ldots, a_t^N\}$ according to the transfer probability function $\mathcal{P}(s_{t+1} \mid s_t, a_t^1, \ldots, a_t^N)$ to transfer to the next state s_{t+1} to obtain the joint reward $r_t = \mathcal{R}(s_t, a_t)$ and get a new joint observation $\Omega = \{o_{t+1}^1, \ldots, o_{t+1}^N | a_t^1, \ldots, a_t^N, s_{t+1}\}$. The goal of each agent i is to maximize the entire expected reward $\mathcal{R}_i = \sum_{t=0}^{T} \gamma^t r_i^t$ within time horizon T.

This work is concerned with learning using Recurrent Neural Network so that the UAV combined with the historical trajectory h_t^i can estimate its true state from local observations Ω_i. The recursiveness in the network structure assumes the function of memorizing some of the observations, making the system behavior minimally different from that in the fully observable case.

Q-learning [29] and DQN [16] are the most popularity model-free reinforcement learning algorithms that have been applied in multi-agent environments. Q-learning uses an action-value function $Q^\pi(s, a) = \mathbb{E}[R \mid s^t = s, a^t = a]$ as a policy π. This Q-function can be updated iteratively by:

$$Q^\pi(s, a) = \mathbb{E}_{s'} \left[r(s, a) + \gamma \mathbb{E}_{a' \sim \pi} \left[Q^\pi \left(s', a' \right) \right] \right] \tag{5}$$

DQN learns the target action-value function Q^* corresponding to the optimal policy, by minimizing the loss function:

$$\mathcal{L}(\theta) = \mathbb{E}_{s,a,r,s'} \left[\left(Q^*(s, a \mid \theta) - r + \gamma \max_{a'} \bar{Q}^* \left(s', a' \right) \right)^2 \right] \tag{6}$$

where $\bar{Q}$ is the objective value of the Q function for stabilizing the learning process. DQN also uses the experience replay buffer $\mathcal{D}$, containing (s, a, r, s'), for the purpose of stabilizing the learning.

Although DQN can be used directly for multi-agents scenarios by each agent learning the optimal policy independently, this is difficult to converge in practice. This is because, from the perspective of any one agent, the overall environment in the iteration is unstable as a result of including other agents as part of the environment, making it impossible to converge without satisfying Markov assumptions.

Policy gradient algorithm [30] is another popular reinforcement learning algorithm that maximizes the expected reward $J(\theta) = \mathbb{E}_{s \sim p^\pi, a \sim \pi_\theta}[R]$ by directly optimizing the policy parameter θ. According to the definition of the Q function, the gradient of the policy can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim p^\pi, a \sim \pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) Q^\pi(s, a) \right] \tag{7}$$

where p^π is the state distribution. The high variance problem of the policy gradient algorithm is more severe in a multi-agent environment. The reward of each agent depends on the actions of multiple agents, and not considering the actions of other agents will increase the variance of the gradient.

When navigating multi-UAV in a complex unknown environment, each UAV can only access local information about the environment, therefore learning through historical trajectories is necessary. Q-learning requires the same training and testing conditions, therefore this work is based on actor-critic framework [31], which takes a centralized training and decentralized execution approach. Each UAV can utilize the global information in the training phase to achieve the optimal policy. The policies learned by UAVs cannot use the shared information in the execution phase, i.e., there is no communication with other UAVs. The critic networks learn the approximation of $Q^\pi(s, a)$ through temporal difference. To reduce the problem of environmental instability and high variance in the multi-agent framework, the critic of each agent is trained using the observations and actions of all agents. Thus, the gradient for agent i can be written as

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim p^\pi, a_i \sim \pi_{\theta_i}} [\nabla_{\theta_i} \log \pi_{\theta_i}(a_i | o_i) Q_i^\pi (o_1, \ldots, o_N, a_1, \ldots, a_N)] \tag{8}$$

Extending the policy gradient to the deterministic policy $\mu_\theta : \mathcal{S} \mapsto \mathcal{A}$, the gradient of the objective function $J(\theta) = \mathbb{E}_{s \sim p^\mu, a \sim \mu_\theta}[R(s, a)]$ can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s \sim \rho^\mu} \left[\nabla_\theta \mu_\theta(a \mid s) \nabla_a Q^\mu(s, a) \big|_{a = \mu_\theta(s)} \right] \tag{9}$$

The gradient of the objective function of the deterministic policy for multi-agent can be written as

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s \sim \rho^\mu} [\nabla_{\theta_i} \mu_{\theta_i}(a_i | o_i) \nabla_{a_i} Q_i^\mu (o_1, \ldots, o_N, a_1, \ldots, a_N) |_{a = \mu_{\theta_i}(o_1, \ldots, o_N)})] \tag{10}$$

In this article, a recurrent multi-agent actor-critic model for partial observables is proposed. An LSTM network is added to the network structure and $h_{t,i}$ denotes the historical observation information of agent i at time step t. Each agent makes a decision for action based on the previous history trajectory: $a_{i,t} = \mu_{\theta_i}(h_{i,t})$. Then the gradient of the objective function of the deterministic policy based on previous history of the recurrent multi-agent can be written as:

$$\nabla_{\theta_i} J(\mu_i) = \mathbb{E}_{U(\mathcal{D})} \left[\nabla_{\theta_i} \mu_{\theta_i} (a_{i,t} \mid h_{i,t}) \nabla_{a_{i,t}} Q_i^\mu (h_{1,t}, \ldots, h_{N,t}, a_{1,t}, \ldots, a_{N,t} |_{a = \mu_{\theta_i}(h_{1,t}, \ldots, h_{N,t})}) \right] \tag{11}$$

Here the experience replay buffer $U(\mathcal{D})$ includes the tuple $(o_{1,t}, \ldots, o_{N,t}, a_{1,t}, \ldots, a_{N,t}, r_{1,t}, \ldots, r_{N,t})$ for store the experience of all agents. During the training update, experiences are sampled from the replay buffer and utilized to compute gradients, which are subsequently employed to update the parameters of the specific agent's actor network. It is important to note that although the experiences are originally collected by all agents, the training process focuses on modifying the parameters of the individual agent being trained. By considering

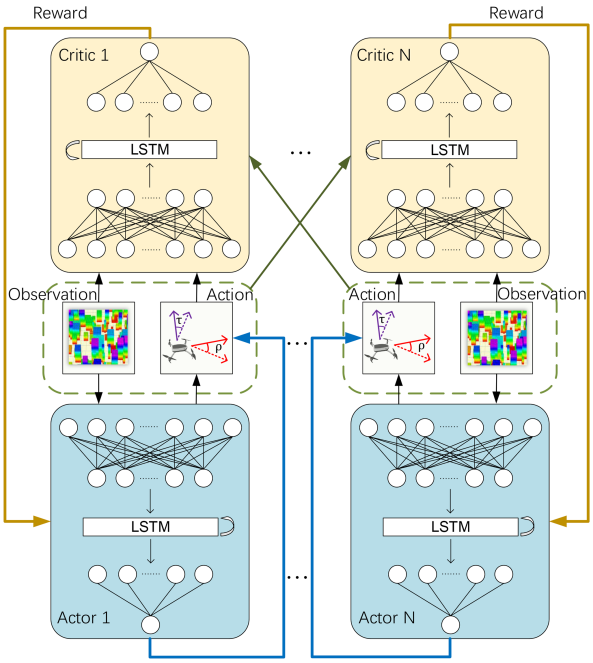


Fig. 4. MARDPG algorithm of navigation control for N UAVs.

other agents' experiences during training, each agent's actor network indirectly incorporates the impact of their actions and behaviors in the environment. This fosters the development of a policy that accommodates the actions and decisions of other agents, thereby facilitating coordinated behaviors and enhancing decision-making in multi-agent settings. In decentralized implementation, each agent acts autonomously based on its own observations and policy, without explicit communication or coordination with other agents. The policies acquired from centralized training are effectively applied in a decentralized manner, where each agent acts independently, relying solely on its local observations. The loss function of the centrally trained critic network is

$$\mathcal{L}(\theta_i) = \mathbb{E}_{U(\mathcal{D})}\left[\left(Q_i^{\mu}(h_{1,t},\ldots,h_{N,t},a_{1,t},\ldots,a_{N,t}) - y_{i,t}\right)^2\right] \tag{12}$$

where

$$y_{i,t} = r_{i,t} + \gamma Q_i^{\mu'}\left(h_{1,t+1},\ldots,h_{N,t+1},a'_{1,t+1},\ldots a'_{N,t+1}\right)|_{a'_j=\mu'_j(h_{1,t+1},\ldots,h_{N,t+1})} \tag{13}$$

The target network policy $\mu' = \{\mu_{\theta'_1},\ldots,\mu_{\theta'_N}\}$ is defined for the delay parameter θ'_N. In practice, similar to the full observable case, Q^{ω} is used as an approximation instead of Q^{μ}. The centrally trained, decentralized execution algorithm with recursiveness in this article outperforms previous algorithms in partially observable multi-UAV navigation, called multi-agent recurrent deterministic policy gradient navigation (MARDPG-NAV) and the algorithm model is shown in Fig. 4.

During the centralized training phase, MARDPG-Nav employs a shared centralized critic network to estimate the joint action-value function using observations and actions from all agents. Communication between UAVs occurs indirectly through the centralized critic network, which acts as a communication channel. Each UAV shares its observations with the centralized critic, enabling the capture of collective information for coordinated action. In the decentralized execution phase, UAVs act independently based on their local policy network and observations. Importantly, there is no direct communication or information exchange among UAVs, allowing for autonomy in decision-making. This characteristic makes MARDPG-Nav suitable for scenarios with limited communication possibilities. Despite the absence of direct communication during execution, coordination and cooperation naturally arise from the learned policies in the training phase.

By combining time dependence, collaborative centralized training, and decentralized execution, our proposed algorithm facilitates collaborative work in multi-UAV path planning. The recurrent network can estimate other UAV's actions by capturing temporal dependencies thus performing collision avoidance in dense obstacle environments. Centralized training aligns the UAVs' decision-making with global objectives, and decentralized execution allows UAVs to navigate collaboratively based on their learned policies.

The main motivation of MARDPG is that if the actions of all agents are known, the environment is stable even in the case of policy changes. And by exploiting the time-continuous character of UAV navigation, more correlations between potential information and state information can be captured. The inclusion of a recurrent network facilitates more potential information from the historical trajectories for better estimation of Q values. The proposed MARDPG algorithm for navigation control of N UAVs is summarized as Algorithm 1.

B. Network Structure and Computational Complexity

Combining the domain knowledge in UAV navigation, the input to the network structure contains the UAV state information ϑ, φ, the relationship with obstacles ϕ and the relative position to the target point ξ. The information in different dimensions is first preprocessed thus extracting features. The perceptual information of the ultrasonic rangefinder is a 5×5 matrix, which is processed by a convolutional layer containing N filters, where each filter has a size of 2×2 and a step size of (1,1). For the angle vector $[\vartheta, \varphi]$ and the relationship vector with the target point ξ are encoded using two simple fully-connected layers, each converted to an 8-dimensional vector. The extracted vectors are combined into a joint vector and passed into the LSTM layer, and finally the output of the LSTM is passed into the fully connected layer and outputs a one-dimensional action. The structure of the actor network is shown in Fig. 5. The critic network contains not only the state information of a single UAV, but also the action and observation information of all UAVs globally, using the fully-connected layer of all agents as input to the hidden LSTM layer, and finally also outputting the action through a fully-connected layer.

We also analyze the computational complexity of the MARDPG algorithm in this part. In the training process, the computational complexity is mainly related to the number of layers of the deep neural network used by each agent and

Algorithm 1: MARDPG-NAV for N Intersections.

Initialize critic network $Q^{\omega_i}(h_{i,t}, a_{i,t})$ and actor $\mu_{\theta_i}(h_{i,t})$ with parameters ω_i and θ_i for all agent $i \in \{1, \ldots, N\}\}$
Initialize target networks $Q^{\omega_i'}$ and $\mu_{\theta_i'}$ with weights $\omega' \leftarrow \omega, \theta' \leftarrow \theta$
Initialize replay buffer $\mathcal{D}$
for episodes $= 1$, M **do**
  Initialize a random process ϵ for action exploration
  Receive initial empty history $h_{i,0}$
  Receive an initial observation o_0 (or history trajectory h_0)
  for $t = 1$ to max-episode-length **do**
    For each UAV i, select navigation action $a_{i,t} = \mu_{\theta_i}(h_{i,t}) + \epsilon$ according to the current policy and exploration noise
    Execute actions $a_{i,t} = \{a_{1,t}, \ldots, a_{N,t}\}$ and receive reward $r_{i,t}$ and new observation $o_{i,t+1}$
    $h_{i,t+1} \leftarrow h_{i,t}, a_{i,t}, o_{i,t+1}$(Append observations and previous actions to the history for all of agents)
    for agent i=1, N **do**
      Store the sequence $(o_{i,1}, a_{i,1}, r_{i,1} \ldots o_{i,T}, a_{i,T}, r_{i,T})$ to replay buffer $\mathcal{D}$
      Sample a minibatch of S samples $(o_{i,1}^j, a_{i,1}^j, r_{i,1}^j \ldots)$ from $\mathcal{D}$
      Construct histories $h_{i,t}^j = (o_{i,1}^j, a_{i,1}^j, \ldots a_{i,t-1}^j, o_{i,t}^j)$
      Compute target values for each sample episode $y_{i,t}^j$ using the recurrent target networks

$$y_{i,t}^j = r_{i,t}^j + \gamma Q_i^{\omega'}\left(h_{i,t+1}, a_{i,t+1}'\right)\Big|_{a_j'=\mu_j'(h_{i,t+1})}$$

      Compute critic update (using BPTT)

$$\mathcal{L}(\omega_i) = \frac{1}{S}\sum_j \left(y_{i,t}^j - Q_i^{\omega}\left(h_{i,t+1}, a_{i,t+1}\right)|_{a_{i,t}=\mu_i(h_{i,t})}\right)^2$$

      Compute actor update (using BPTT)

$$\nabla_{\theta_i} J(\theta_i) \approx \tfrac{1}{S}\textstyle\sum_j \left[\nabla_{\theta_i}\mu_{\theta_i}(h_{i,t}^j)\nabla_{a_{i,t}} Q_i^{\omega}\left(h_{i,t}^j, a_{i,t}^j\,|_{a=\mu_{\theta_i}(h_{i,t}^j)}\right)\right]$$

    end for
  end for
  Update the target networks

$$\omega_i' \leftarrow \tau\omega_i + (1-\tau)\omega_i'$$

$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$

end for

the number of neurons in each layer. We consider that each neural network is a fully connected neural network containing L layers of hidden layers and each layer contains N neurons, b is the size of the training batch. Then the number of edges of the actor and critic containing the recurrent neural network

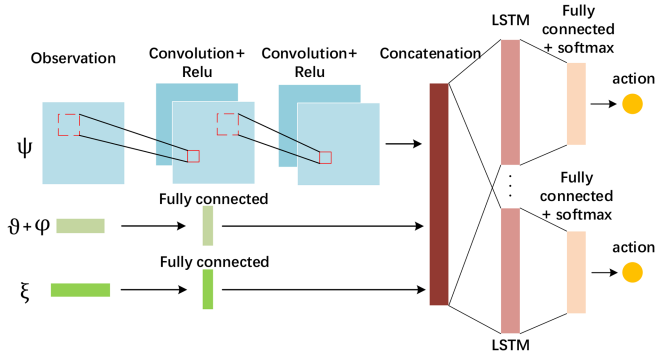


Fig. 5. Architecture of actor networks to select actions.

are: $\underbrace{b \times |S_i| \times N}_{\text{batch size} \times \text{input} \times \text{layer (1)}}, \ldots, \underbrace{b \times N^2}_{\text{batch size} \times \text{layer}(n-1) \times \text{layer}(n)}, \ldots,$
$\underbrace{b \times N \times |A_i|}_{\text{batch size} \times \text{layer}(N-1) \times \text{layer}(N)}$ and $\underbrace{b \times (|S_i| + |A_i|) \times N}_{\text{batch size} \times \text{input} \times \text{layer (1)}}, \ldots,$
$\underbrace{b \times N^2}_{\text{batch size} \times \text{layer}(n-1) \times \text{layer}(n)}, \ldots, \underbrace{b \times N \times |A_i|}_{\text{batch size} \times \text{layer}(N-1) \times \text{layer}(N)},$
where $|S_i|$ and $|A_i|$ represent the size of the agent's state space and action space. Therefore, the computational complexity of the MADDPG algorithm in the training process is $O(bN|S_i|)$, while the computational complexity of the RDPG algorithm and the MARDPG algorithm with a recurrent neural network in the training process is $O(bN^2)$.

C. Parameter Sharing

Multi-agent network training brings a large consumption of computational resources, and sharing the parameters of the neural network among different agents can improve the training efficiency [32]. In this work, the network parameters are shared among different agents to increase the efficiency of the algorithm. However, sharing the parameters can lead to similar behavioral policies among the agents, and even end up with the same policy. This may be beneficial in some cases, but UAVs navigate in unknown environments at different locations, and this spatial difference will be more pronounced if flying in formation. All its policies should be characterized according to the position in which they are located, so in this work only the parameters are shared in the underlying neural network, which will bring efficiency in navigation than sharing parameters completely. The lower layer of the network is responsible for extracting feature information, and sharing the parameters of this part can improve the speed of UAV processing of environmental sensing information and reduce memory usage. Both the individual processing of the upper neural network parameters and the individual observational differences of the agents allow each UAV to generate its unique policy.

D. Optimal Policy

One of the main challenges in continuous action spaces is that each agent cannot fully explore all state spaces. If the agent can only experience a limited number of state combinations, it

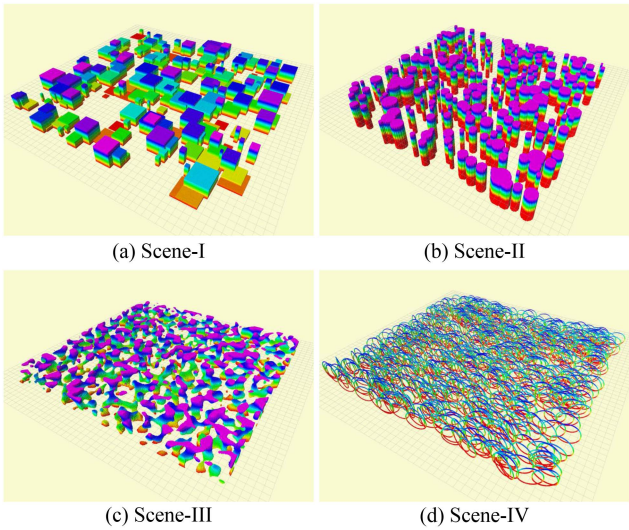


Fig. 6. Navigation algorithm will be tested in the following four environments, by randomly selecting scenes and starting point pairs for training. (a) is a square column obstacle, (b) is a cylindrical obstacle, (c) is a simulated forest obstacle, and (d) is a circular obstacle.

may lead to an inability to accurately estimate the Q value of an unknown scenario. One advantage of the MARDPG algorithm is that it can handle traversing the environment independently of the learning algorithm. By adding the noise ϵ sampled from the noise process to the current learning policy, $a_{i,t} = \mu_{\theta_i}(h_{i,t}) + \epsilon$, the randomness of the learning policy can be increased, thus allowing each UAV to generate a differentiated policy. It can also enable the agent to be encouraged to continue exploring the state space to reach the optimal policy.

VI. Experimental Results

A. Experiment and Parameter Setting

The simulated environment consists of randomly generated maps based on the ROS environment [33]. There are four types of environments: randomly generated square pillars, cylinders, circular rings, and foggy 3D obstacles, as shown in Fig. 6. These environments are used to simulate obstacles such as buildings in cities and trees in forests. The agents will choose a training environment randomly from which to progress episodes. The density of obstacles is measured as the percentage of space occupied by the obstacles. As the volume of obstacles increases, it poses a greater challenge to UAV exploration and navigation. All simulations were run on Ubuntu 16.04 with 32 GB RAM and a GeForce GTX 1080Ti GPU. We trained our UAV for a total of 20,000 time steps, which took approximately 12 hours to execute.

The hyperparameters in the model were adjusted through a series of repeated experiments, and the best hyperparameters were used to deploy the model. The flight altitude of the UAV is confined to $[0.0\,\text{m}, 60.0\,\text{m}]$ in the environment. The control signal is normalized to $[-1, 1]$. The parameters of the reward were set to $\alpha = 3.0$, $\lambda = 5.0$, $\sigma = 15.0$, $r_{free} = 0.1$, and $r_{step} = -0.6$. The scaling factors δ_1, δ_2, δ_3 and δ_4 of the reward function are 0.45, 0.3, 0.15 and 0.1, respectively. The experiments assume that the learning rate of Adam Optimizer is 0.01 and the update rate of the target network is $\tau = 0.01$. The size of the replay buffer is 10^5, which is sampled every 100 time steps. The discount factor is $\gamma = 0.99$. The time step size of LSTM is 80. The update mini-batch size (number of episodes) is 128.

B. Compared Methods

The proposed method will be compared to the following baseline way to evaluate the efficiency of the model in this research. The same states, actions, and rewards are used in all baselines.

a) RDPG: RDPG can be extended to apply to MADRL by an independent learning approach in which each agent perceives all other agents as part of the environment. This approach can be called Ind-RDPG, where the UAVs' navigation policies are trained through a recurrent deterministic policy gradient, and the UAVs in the environment all adopt the same policy, with no information exchange between the individual UAVs.

b) MADDPG: Feedforward neural networks are introduced for learning, and multi-UAV are trained by a centralized training and decentralized execution method, avoiding the problems of environmental non-stationarity and large variance.

c) ORCA-3D: ORCA-3D is a widely used classical algorithm for decentralized collision avoidance in multi-agent systems. It provides a real-time collision avoidance framework by dynamically generating velocity commands for each agent to avoid potential collisions with other nearby agents [22], [23].

C. Evaluation Metric

A set of quantitative evaluation measures that indicate the safety, efficacy, and robustness of the navigation algorithm were designed to evaluate the quality of UAV navigation. The following are the details:

a) Average reward: The learned policy completes 250 navigation tasks in four environments, with the reward being the average of all times and agents. Smaller rewards imply a lower level of performance.

b) Success rate: The percentage of agents who arrive at the target point without colliding within the time constraint.

c) Trapped rate: The percentage of agents that are unable to reach the target point within the time constraint.

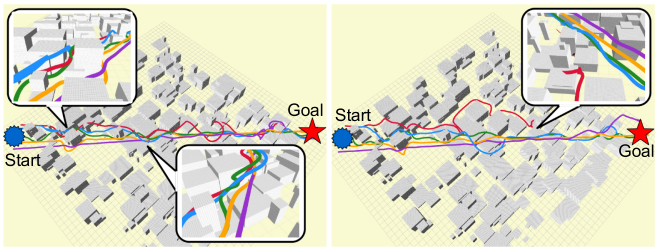
d) Collision rate: The percentage of agents that failed the mission because collided with obstacles.

e) Path efficiency: The ratio of the sum of the straight-line distance between the starting point and the target point of all agents to the sum of the actual trajectory lengths of all agents.

These metrics were averaged over 250 experiments to evaluate the performance of the algorithm.

D. Performance of MARDPG

To verify the effectiveness of the proposed algorithm, trained MARDPG agents were used to complete numerous navigation



(a) MARDPG (b) MADDPG

Fig. 7. Navigation schematic of 5 UAVs in a square column obstacle scene. (a) is the navigation result of the MARDPG algorithm, where the UAV with memory can escape from the dense part of the obstacle; (b) is the navigation result of the MADDPG algorithm, where the UAV gets trapped in the dilemma.

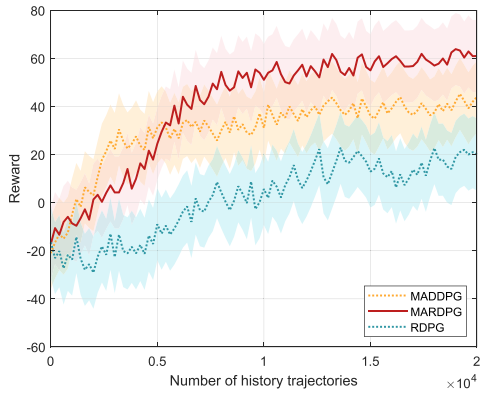


Fig. 8. Reward during all the training episodes.

tasks in a simulated environment. The approach can successfully navigate the UAV to the target point by learning from the historical trajectory to control the UAV, as illustrated in Fig. 8. The efficient performance of the MARL method demonstrates that it is a feasible solution to the problem of navigating multi-UAV in complex environments.

1) Compare With MADDPG: The following experiments were conducted to prove that the MARDPG algorithm with memory is more effective than the MADDPG method without memory in executing multi-UAV navigation tasks. In Environment 1, the start and target points for each of the five UAVs are selected, and the five lines are intersected so that the avoidance behavior of UAVs towards environmental obstacles and other UAVs can be clearly represented. Both approaches, as depicted in Fig. 8, learn to navigate through complex environments. However, since the LSTM unit of the proposed MARDPG algorithm can make decisions based on previous observations and action trajectories, it may sense more information about the local environment's structure and thus escape from local traps.

Similar conclusions can be seen from the average return curves of the MARPDG algorithm and the MADDPG algorithm. The general trend of both curves is the same as seen in Fig. 7. The convergence of the MARDPG algorithm is slower than that of the MADDPG algorithm because learning by backpropagation in the actor network and the critic network takes more time. But the reward after convergence is higher for the MARDPG algorithm than for the MADDPG algorithm, illustrating the importance of recurrent networks to learn optimal policies from partial observables. The partially observable domain is challenging for the MADDPG algorithm because it cannot preserve the previous history and thus cannot estimate the true system state. In contrast, the MARDPG algorithm trained by incorporating LSTM can learn better policy from the previous historical trajectory.

We also performed a quantitative comparison of the success rate, trapped rate and collision rate for both algorithms, then the results are shown in Table I . In all four environments, the success rate of the MADDPG agents is below 82% percent, with an average success rate of 75.75% percent. In contrast, the MARDPG agents had the lowest success rate of 91.75% percent in the four environments, which is nearly 20% higher than the average MARDPG agent success rate, indicating a significant performance improvement. Furthermore, despite the fact that both approaches achieve low collision rates, the MADDPG agents obtain a trapped rate of 53.75% in the most structurally complicated environment III, implying that more than half of the UAVs become trapped in localized dilemmas and are unable to reach the target point. The superior performance of MARDPG is owing to its more complicated computation and, as a result, slower convergence than MADDPG. MADDPG uses two feed-forward neural networks to approximate the actor network and the critic network, while MARDPG uses two LSTMs. However, in the UAV navigation task, reaching the target point safely is more important.

2) Compare With RDPG: Within the first 1100 episodes, the rewards of the RDPG-based navigation algorithm show an upward tendency, although there are large fluctuations, as shown in the simulation results in Fig. 7. The rewards indicate a dramatic declining trend after 1100 episodes, and the rewards for each episode are quite inconsistent. This is because, in the RDPG algorithm, each UAV is simply controlled based on local observations, making it impossible to ensure that all UAVs learn the optimal policy. The success rate of RDPG in all four environments is roughly 10% lower than the MARDPG algorithm, the collision rate is nearly five times greater than the MARDPG algorithm, and the agents are trapped at a higher rate than the MARDPG algorithm, as shown in Table I. Although the RDPG algorithm takes historical trajectories into account, it is designed for a single agent, and when multi-UAV use it for navigation, the instability of the environment causes slow convergence and the navigation capabilities to drop short of expectations.

3) Compare With Variant of MARDPG: After removing the parameter sharing and exploration noise, the algorithm's convergence rate and reward are shown in Table III. It can be seen, the episodes that reach convergence after removing parameter sharing are 450 times later than the case of shared parameters. The results show that sharing parameters in the lower layer network can make the training speed improved, and in the case of high-dimensional input, the lower layer network is used for feature extraction, so sharing parameters can reduce the computational effort. The noise in the training process uses uncorrelated Gaussian noise with a mean value of 0. Table III also shows that the final reward after removing the exploration noise in the training phase is much lower than that with the addition of the exploration noise, implying that the exploration ability of the UAV decreases after removing the exploration

TABLE I
STATISTICS OF THE SUCCESS RATE, LOST RATE AND COLLISION RATE OF MADDPG, RDPG AND MARDPG IN DIFFERENT SCENARIOS

Results / Environments	Success rate			Collision rate			Trapped rate		
	MADDPG	RDPG	MARDPG	MADDPG	RDPG	MARDPG	MADDPG	RDPG	MARDPG
SCENE-I	76.75%	84.75%	**93.25%**	**0.75%**	4.75%	1.25%	22.50%	11.00%	**5.50%**
SCENE-II	82.00%	91.25%	**97.50%**	**0.25%**	2.50%	0.50%	17.75%	6.25%	**2.00%**
SCENE-III	64.50%	79.25%	**91.75%**	2.5%	9.25%	**1.75%**	33.25%	11.50%	**6.50%**
SCENE-IV	79.75%	89.00%	**96.50%**	**1.25%**	5.50%	2.75%	19.00%	5.50%	**0.75%**
Average	75.6875%	86.0625%	**94.75%**	**1.1875%**	5.375%	1.5625%	23.125%	8.5625%	**3.6875%**

The best metrics in each category are shown in bold.

TABLE II
THE NAVIGATION PERFORMANCE OF ORCA-3D AND THE PROPOSED ALGORITHM IS EVALUATED IN DIFFERENT UAV NUMBER AND OBSTACLE DENSITY ENVIRONMENTS

Metric	Obstacle density(10%)				Obstacle density(50%)			
	5 UAVs		10 UAVs		5 UAVs		10 UAVs	
	ORCA-3D	MARDPG	ORCA-3D	MARDPG	ORCA-3D	MARDPG	ORCA-3D	MARDPG
Success rate	**94.25%**	93.5%	**91.75%**	89.75%	82.5%	**90.75%**	76.25%	**84.25%**
Trapped rate	5.25%	**4.75%**	7.5%	**5.75%**	11.25%	**7.5%**	14.75%	**8.25%**
Path efficiency	**89.34%**	87.61%	**86.72%**	84.29%	80.35%	**81.76%**	73.28%	**76.54%**
Calculating time	**1.9 ms**	3.7 ms	**4.3 ms**	7.9 ms	**2.5 ms**	6.7 ms	**4.9 ms**	8.3 ms

The best metrics in each category are shown in bold.

TABLE III
COMPARISION WITH VARIANT OF OUR PROPOSED METHOD

Method	Reward	Convergence episode
Ours	61.3	10000
Without parameter sharing	54.6	14500
Without optimal policy	16.7	12100

noise, and it cannot traverse more observed scenes and thus cannot obtain the optimal policy.

4) Compare With ORCA-3D: We analyze the performance differences between ORCA-3D and MARDPG algorithms for multi-UAV navigation. Table II presents the results of comparing navigation success rate, trapping rate, path efficiency, and calculating time at various obstacle densities and numbers of UAVs. At an obstacle density of 10%, ORCA-3D demonstrates superior performance compared to the MARDPG algorithm in terms of success rate, path efficiency, and calculating time for different numbers of UAVs. The success rates of ORCA-3D are consistently higher, with values of 94.25% and 91.75% for 5 and 10 UAVs, respectively. In contrast, the MARDPG algorithm achieves success rates of 93.5% and 89.75% for the same UAV configurations. However, the trapping rate of the ORCA-3D algorithm is slightly worse than that of the MARDPG algorithm, particularly evident at an obstacle density of 50% in the 10 UAV navigation scenario. The trapping rate of ORCA-3D is 6.5% higher than that of the MARDPG algorithm, highlighting the MARDPG algorithm's ability to leverage a recurrent network for learning from past experiences and avoiding local dilemma. The collaborative nature of the MARDPG algorithm allows UAVs to explore the environment together, exchange obstacle information, and learn from one another, leading to more effective exploration and navigation strategies.

In terms of calculating time, the overall performance of the ORCA-3D algorithm is stronger than that of the MARDPG algorithm. ORCA-3D achieves calculating times of less than 5 ms in various environments, demonstrating high real-time response capability. On the other hand, the MARDPG algorithm requires a significant amount of training data and computational resources during the training phase to learn effective navigation strategies. However, during the decentralized execution phase, it does not require centralized updates or policy optimization. The MARDPG algorithm can maintain computation times within 8.3 ms when operating with 10 UAVs and 50% obstacle density, meeting the real-time requirements of UAV navigation.

In conclusion, the experimental analysis shows that ORCA-3D outperforms the MARDPG algorithm in terms of path efficiency and calculating time at low obstacle densities. However, in higher obstacle density scenarios, the MARDPG algorithm shows advantages in terms of capture rate and success rate. The computational efficiency of ORCA-3D enables real-time processing, while the MARDPG algorithm leverages its collaborative nature for effective exploration. Both algorithms offer strengths in different aspects of multi-UAV navigation, providing valuable insights for future research and applications.

E. Generalization Ability of MARDPG

In this section, initially, we analyze the algorithm's real-time performance and robustness under sensor uncertainty, then consider the impact of changes in obstacle density and number of UAVs on the performance of multi-UAV navigation.

1) Real-Time Performance: The real-time performance of the algorithm is a key indicator of whether it can be applied to reality. We analyze the real-time performance of the three algorithms in Scene-I with obstacle densities of 0.1 and 0.5 by

TABLE IV
IN SCENE-I WITH GAUSSIAN NOISE LEVEL $\sigma = 0.1$ AND 6 UAVS, THE NAVIGATION PERFORMANCE OF THE THREE METHODS IN DIFFERENT OBSTACLE DENSITY ENVIRONMENTS IS EVALUATED

Results / Algorithm	Obstacle density(10%)				Obstacle density(50%)			
	Flight distance	Flight time	Collision rate	Calculating time	Flight distance	Flight time	Collision rate	Calculating time
RDPG	76.58 m	**62.87** s	6.25%	3.1 ms	83.65 m	**75.94** s	11.50%	4.7 ms
MADDPG	83.26 m	102.51 s	1.50%	2.6 ms	95.14 m	132.38 s	2.75%	3.9 ms
MARDPG	**75.31** m	83.68 s	**0.75%**	3.7 ms	**81.89** m	90.98 s	**1.25%**	5.6 ms
ORCA-3D	76.94 m	86.23 s	1.25%	**2.1** ms	84.27 m	93.44 s	7.25%	**3.1** ms

The best metrics in each category are shown in bold.

TABLE V
IN SCENE-I WITH AN OBSTACLE RATIO OF 20% AND 6 UAVS, THE NAVIGATION PERFORMANCE OF THE THREE METHODS IS EVALUATED WHEN THE SENSORS ARE AFFECTED BY NOISE

Results / Algorithm	$\sigma = 0.1$				$\sigma = 0.5$			
	Success rate	Collision rate	Trapped rate	Path efficiency	Success rate	Collision rate	Trapped rate	Path efficiency
RDPG	83.00%	5.75%	11.25%	69.354%	74.50%	13.75%	11.75%	59.432%
MADDPG	69.50%	**1.25%**	29.25%	**81.292%**	57.25%	**3.50%**	39.25%	**77.518%**
MARDPG	**92.75%**	2.50%	**4.75%**	73.951%	**86.75%**	6.50%	**6.75%**	75.189%
ORCA-3D	90.25%	3.25%	6.50%	78.629%	80.75%	4.75%	14.50%	71.243%

The best metrics in each category are shown in bold.

comparing the flight distance, flight time, and calculating time required by the three algorithms to complete the same flight task. The results are shown in Table IV. In scenarios with low obstacle density, the online decentralized control runs 6 UAVs, and the time for the policy network to calculate the next action can be maintained within 4 ms. With the increase of obstacle density, the calculation time of the policy network will reach 6 ms, which can still meet the real-time navigation control of UAVs. Compared with MADDPG algorithm and RDPG algorithm, although our method has slower online calculating time, it can choose a shorter path to reach the target quickly in a safer way.

2) Robust Performance: In order to simulate the impact of sensor uncertainty on the algorithm, we add Gaussian noise to the state information of other agents observed by the agent, $o' = o + g, g \sim \mathcal{N}(0, \sigma^2)$, where o' is the actual observed value after adding standard normal distribution Gaussian noise g to the original observed value o. In the experiment, the noise levels $\sigma = 0.1$ and $\sigma = 0.5$ are considered separately for training and testing to generate different policy models. Table V shows the performance evaluation of different methods under different Gaussian noises, and it can be seen that the proposed MARDPG algorithm yields the best performance in most cases. Especially when the sensor is seriously disturbed by noise, the two multi-agent algorithms adopting the centralized training strategy can share the information of neighbors, which is superior to the decentralized training algorithm in terms of navigation safety. The recurrent network combined with the proposed MARDPG algorithm has the ability to summarize historical trajectory information, and can achieve the highest navigation success rate in the case of sensor uncertainty.

3) Generalization Performance: The density of obstacles in the real world has a significant impact on UAV navigation. As a result, in each of the four environments, the trained MARDPG and MADDPG agents were tested for navigation with various obstacle densities. The experimental results are shown in Fig. 9(a)–(c). Both algorithms perform well in SCENE-I, however, the success rate of MADDPG agents drops dramatically at increasing obstacle densities, whereas the success rate of MARDPG agents can be maintained at a high level. Furthermore, when compared to MARDPG agents, the collision rates and trapping rates of RDPG agents rise in higher density obstacle environments. In environments with varying obstacle densities, the MARDPG algorithm has a high success rate, low trapped

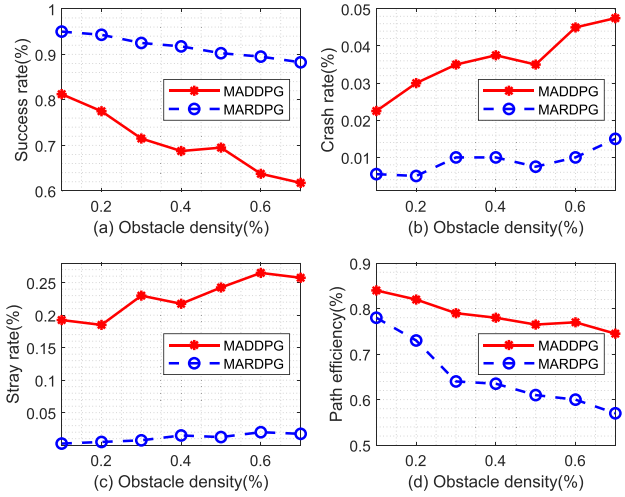


Fig. 9. Performance of MADDPG and MARDPG algorithms with the change of obstacle density. (a), (b), and (c) show the changes of success rate, trapped rate and collision rate, respectively, and (d) show the path efficiency.

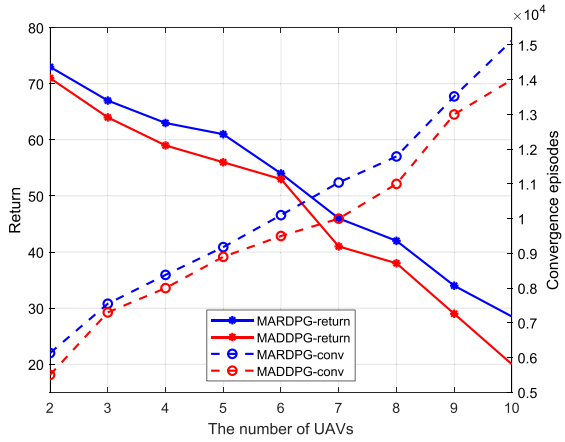


Fig. 10. Change in convergence time and final return as the number of UAVs increases. The solid line is the final return variation, and the dashed line is the change in the time to reach convergence.

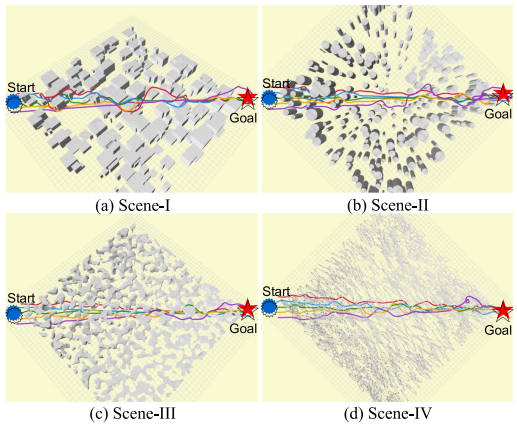


Fig. 11. Navigation performance of MARDPG algorithm in four environments with 5 UAVs and 30% obstacle density.

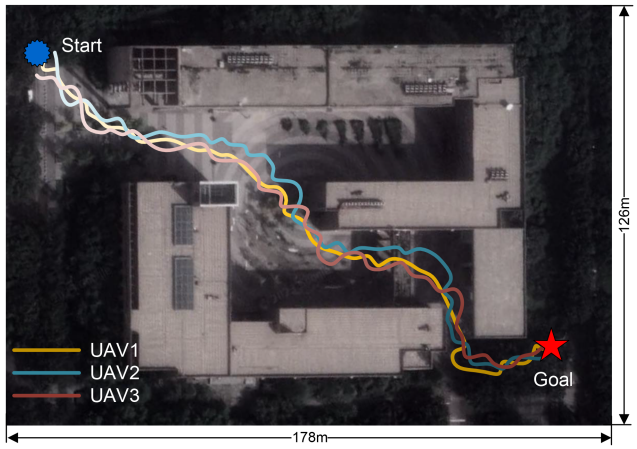


Fig. 12. Top view of real-world experimental trajectory of MARDPG-Nav algorithm.

rate, and low collision rate, suggesting its capacity to memorize not just simple environmental structures but also to deal with complicated environments for navigation. The change in path efficiency as the density of obstacles changes is depicted in Fig. 9(d). In the same obstacle density environment, the path efficiency of the MARDPG algorithm is higher than that of the MADDPG algorithm, indicating that the proposed MARDPG algorithm is more efficient in its navigation action decision making. Furthermore, when the density of the environment rises, both algorithms' path efficiency decreases, and the trajectory to the target point becomes more tortuous.

Furthermore, the number of UAVs available for experiments with the same configuration has been increased. The Fig. 10 shows the time required for the convergence of the proposed method at different number of UAVs and the final reward, where the solid line represents the number of episodes required for approach convergence and the dashed line represents the reward after convergence. The experiments show that the number of episodes required for convergence grows as the number of UAVs increases, but the average reward after convergence decreases. In general, the reward of the proposed method after convergence is higher than that of the MADDPG method. Fig. 11 shows the navigation performance of the trained MARDPG algorithm in four environments with 5 UAVs and 30% obstacle density. The RDPG algorithm is not shown in the figure because its convergence is slow or even impossible in the environment after higher than the number of 5 UAVs, while the proposed method can satisfy up to 20 UAVs for the navigation task, so the MARDPG algorithm is more suitable for multi-UAV navigation.

4) Real-World Experiments: In this section, we present a detailed analysis of the real-world performance and capabilities of multi-UAV based on MARDPG-Nav. The algorithm was initially trained in a simulated environment to generate coordinated policies. The trained policies were then applied to multi-UAV equipped with LIDARs, which measured distances at 20 m intervals with a scanning frequency of 10 Hz, operating in the real world. To simplify computations, the entire experimental flight was conducted at a fixed altitude. The experimental setup is illustrated in Fig. 12, with a focus on UAVs navigating through complex structures without getting stuck in local obstacles, such as the upper-right corner shown in Fig. 12.

The experimental results demonstrate the successful completion of the mission by all three UAVs, as they navigated through the buildings without getting stuck. This achievement validates the effectiveness of the MARDPG-Nav policy in promoting coordinated decision-making and obstacle avoidance. The average time taken by the UAVs to cover a distance of 232.4 m was 317.8 s. The experiments validate the advantages of the proposed MARDPG-Nav algorithm, which leverages historical information for decision-making in real-world scenarios. The integration of historical data enables the multi-UAV system to exhibit efficient collaboration and decision-making capabilities, effectively navigating through complex structures. However, certain limitations were observed during the experiments, such as fluctuations in the forward direction of the UAVs during navigation. Future efforts will focus on addressing these limitations by incorporating robust control methods to ensure stability and accuracy in UAV motion.

VII. Conclusion

In this article, the MARDPG algorithm was proposed to solve the problem of navigating multi-UAV in complex environments.

TABLE VI
MAIN NOTATION MEANING

Notion	Meaning
ϑ	The angle between the first perspective and the x-axis
φ	The angle between the first perspective and the horizontal line
d	The distance from UAV to obstacle
ψ	Ultrasonic rangefinder data
ξ	Relative angle and distance of UAV to goal
ϖ	The angle between the UAV and the goal in the x-y plane
ϖ_z	The angle between the UAV and the goal in the vertical plane
ρ	The steering signal of the UAV in the horizontal plane
τ	The steering signal of the UAV in the vertical plane
N	The number of agents
$\mathcal{S}$	The set of states of all agents
$\mathcal{A}$	The set of actions
Ω	The set of partial observables by agents
h_t^i	The historical trajectory of the agent i at time step t
μ_θ	The deterministic policy with parameter θ
$\mathcal{D}$	The experience replay buffer

The environmental information collected by the sensors on board each UAV was used to develop navigation policy. The control of each UAV is not isolated because its global state can be obtained during the training process. Each UAV can estimate the control strategies of other UAVs during decision making, thus adjusting the local policies and making optimal decisions. Simulation results show that the MARDPG algorithm is more suitable for application to partially observable environments than the MADDPG algorithm, and the proposed method enables multi-UAV to fly from any start point to the target point in a complex and unknown environment. Moreover, the method can be extended to more complex and dynamic unknown environments. In the future, we will focus on MARL-based control algorithms for cooperative navigation of multi-UAV in dynamic environments, which is of great importance to further improve their overall navigation capabilities.

REFERENCES

[1] M. Eskandari, H. Huang, A. V. Savkin, and W. Ni, "Model predictive control-based 3D navigation of a RIS-equipped UAV for los wireless communication with a ground intelligent vehicle," *IEEE Trans. Intell. Veh.*, vol. 8, no. 3, pp. 2371–2384, Mar. 2023.

[2] S. Zhang and J. Liu, "Analysis and optimization of multiple unmanned aerial vehicle-assisted communications in post-disaster areas," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12049–12060, Dec. 2018.

[3] M. Erdelj, M. Król, and E. Natalizio, "Wireless sensor networks and multi-UAV systems for natural disaster management," *Comput. Netw.*, vol. 124, pp. 72–86, 2017.

[4] S. Huang, R. S. H. Teo, and K. K. Tan, "Collision avoidance of multi unmanned aerial vehicles: A review," *Annu. Rev. Control*, vol. 48, pp. 147–164, 2019.

[5] J. Wang, M. K. Lim, Y. Zhan, and X. Wang, "An intelligent logistics service system for enhancing dispatching operations in an IoT environment," *Transp. Res. Part E: Logistics Transp. Rev.*, vol. 135, 2020, Art. no. 101886.

[6] D. Wang, T. Fan, T. Han, and J. Pan, "A two-stage reinforcement learning approach for multi-UAV collision avoidance under imperfect sensing," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3098–3105, Apr. 2020.

[7] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Cooperative path planning for multiple UAVs in dynamic and uncertain environments," in *Proc. IEEE 41st Conf. Decis. Control*, 2002, pp. 2816–2822.

[8] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 477–483.

[9] K. H. Movric and F. L. Lewis, "Cooperative optimal control for multi-agent systems on directed graph topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 769–774, Mar. 2014.

[10] Z. Liu, H. Wang, H. Wei, M. Liu, and Y.-H. Liu, "Prediction, planning, and coordination of thousand-warehousing-robot networks with motion and communication uncertainties," *IEEE Trans. Automat. Sci. Eng.*, vol. 18, no. 4, pp. 1705–1717, Oct. 2021.

[11] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*. Berlin, Germany: Springer, 2011, pp. 3–19.

[12] J. Snape, J. v. d. Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011.

[13] D. Bareiss and J. Van den Berg, "Generalized reciprocal collision avoidance," *Int. J. Robot. Res.*, vol. 34, no. 12, pp. 1501–1514, 2015.

[14] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Hoboken, NJ, USA: Wiley, 2014.

[15] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, Nov. 2017.

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

[17] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *Int. J. Robot. Res.*, vol. 39, no. 7, pp. 856–892, 2020.

[18] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1527–1533.

[19] M. Bojarski et al., "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[20] Y. Chen et al., "DRQN-based 3D obstacle avoidance with a limited field of view," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 8137–8143.

[21] X. Zhu, F. Vanegas, F. Gonzalez, and C. Sanderson, "A multi-UAV system for exploration and target finding in cluttered and GPS-denied environments," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2021, pp. 721–729.

[22] D. Alejo, J. Cobano, G. Heredia, and A. Ollero, "Optimal reciprocal collision avoidance with mobile and static obstacles for multi-UAV systems," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2014, pp. 1259–1266.

[23] S. H. Arul et al., "LSwarm: Efficient collision avoidance for large swarms with coverage constraints in complex urban scenes," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 3940–3947, Oct. 2019.

[24] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 285–292.

[25] H. Qie, D. Shi, T. Shen, X. Xu, Y. Li, and L. Wang, "Joint optimization of multi-uav target assignment and path planning based on multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 146264–146272, 2019.

[26] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6382–6393.

[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] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Math. Operations Res.*, vol. 27, no. 4, pp. 819–840, 2002.

[29] C. J. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3, pp. 279–292, 1992.

[30] 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.

[31] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 1999, pp. 1008–1014.

[32] J. N. Foerster, Y. M. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate to solve riddles with deep distributed recurrent Q-networks," 2016, *arXiv:1602.02672*.

[33] William. Wu, "mockamap," 2017. [Online]. Available: https://github.com/HKUST-AerialRobotics/mockamap

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 the 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 current research interests include multiagent systems, adaptive control, event-triggered control, distributed optimization, learning, and control.