A State-Decomposition DDPG Algorithm for UAV Autonomous Navigation in 3-D Complex Environments

Lijuan Zhang, Jiabin Peng, Weiguo Yi, Hang Lin, Lei Lei, and Xiaoqin Song

***Abstract*—Over the past decade, unmanned aerial vehicles (UAVs) have been widely applied in many areas, such as goods delivery, disaster monitoring, search and rescue etc. In most of these applications, autonomous navigation is one of the key techniques that enable UAV to perform various tasks. However, UAV autonomous navigation in complex environments presents significant challenges due to the difficulty in simultaneously observing, orientation, decision and action. In this work, an efficient state-decomposition deep deterministic policy gradient algorithm is proposed for UAV autonomous navigation (SDDPG-NAV) in 3-D complex environments. In SDDPG-NAV, a novel state-decomposition method that uses two subnetworks for the perception-related and target-related states separately is developed to establish more appropriate actor networks. We also designed some objective-oriented reward functions to solve the sparse reward problem, including approaching the target, and avoiding obstacles and step award functions. Moreover, some training strategies are introduced to maintain the balance between exploration and exploitation, and the network is well trained with numerous experiments. The proposed SDDPG-NAV algorithm is capable of adapting to surrounding environments with generalized training experiences and effectively improves UAV's navigation performance in 3-D complex environments. Comparing with the benchmark DDPG and TD3 algorithms, SDDPG-NAV exhibits better performance in terms of convergence rate, navigation performance, and generalization capability.**

***Index Terms*—Autonomous navigation, decision making, deep reinforcement learning (DRL), path planning, unmanned aerial vehicle (UAV) autonomy.**

I. Introduction

OVER the past decade, unmanned aerial vehicles (UAVs) have been widely applied in many areas to perform transporting, cruising, and communication tasks, such as goods delivery, disaster monitoring, search and rescue, border surveillance, air base stations, and so on [1], [2]. In most of these applications, autonomous navigation is one of the key techniques that enable UAV to performance various tasks. In fact, there is a huge demand for UAV navigation without much human intervention, especially in harsh environments where human exploration is not possible. As is shown in Fig. 1, UAV autonomous navigation is a basic technique that enables UAV to find a suitable path from the departure position to the destination without colliding with obstacles. The problem is challenging since it is difficult for the UAV to perform observing, orientation, decision and action operations simultaneously. The complexity further increases in uncertain 3-D environments with dense obstacles and dynamic objects.

Autonomous navigation is a fundamental problem in UAV autonomy, and a good number of solutions have been proposed in [2], [3], [4], [5], [6], and [7]. Classic planning-based methods, including geometric methods, graph theory methods, and probabilistic roadmap methods, mainly focus on providing a preplanned path with the help of accurate global maps where the positions of obstacles are known in prior [3], [4]. For uncertain environments, some simultaneously localization and mapping (SLAM) algorithms [5] are presented to facilitate autonomous navigation of UAV. However, the map regeneration process of SLAM requires high computation and power consumption which can create communication delays for real-time navigation [6]. In the last decade, many intelligent optimization algorithms are proposed for UAV autonomous navigation, such as anti-colony, particle swarm, differential evolution, fuzzy logic, artificial potential field, and so on [7]. These algorithms aim at searching for an optimal path to guide UAV, but the computational complexity of these algorithms is high and the performance relies on the rationality of parameter settings. In fact, the behaviors of UAV rely on its flight controller, dynamics, actuators, and sensor information. UAV needs to make decisions for collision avoidance and dynamically adjust its trajectory based on the collected sensor information. In complex environments, UAV must cope with the uncertainties of both physical hardware and operational environment. Traditional planning and intelligent optimization methods may not provide a satisfactory solution. Generally, Table I compares three types of navigation algorithms.

Recent advances in deep reinforcement learning (DRL) provide the possibility of adapting to surrounding environments

Manuscript received 3 April 2023; revised 30 September 2023; accepted 20 October 2023. Date of publication 26 October 2023; date of current version 7 March 2024. This work was supported in part by the National Natural Science Foundation of China under Grant 62371232 and Grant 61902182; in part by the Future Network Scientific Research Fund Project under Grant FNSRFP-2021-ZD4; in part by the National Key Research and Development Program of China under Grant 2020YFB1600104; in part by the Key Research and Development Plan of Jiangsu Province under Grant BE2020084-2; in part by the Natural Science Foundation of Jiangsu Province of China under Grant BK20190409; and in part by the China Postdoctoral Science Foundation under Grant 2019TQ0153. *(Corresponding author: Lei Lei.)*

The authors are with the College of Electronic and Information Engineering/College of Integrated Circuits, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China (e-mail: lijuanzhang6@gmail.com; pengjiabin98@163.com; ywg13879727055@163.com; a1396087825555@163.com; leilei@nuaa.edu.cn; xiaoqin.song@163.com).

Digital Object Identifier 10.1109/JIOT.2023.3327753

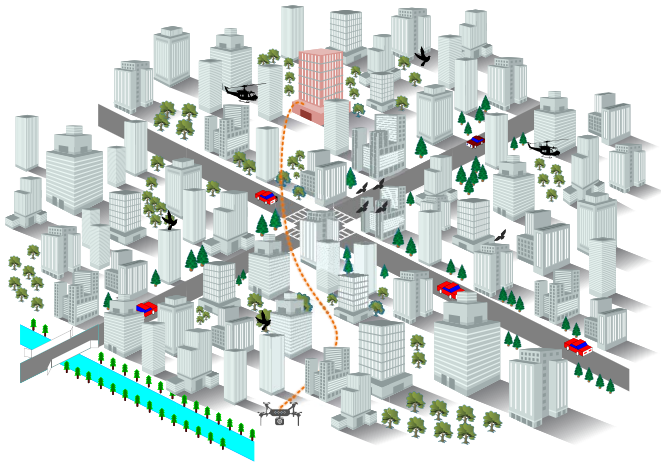


Fig. 1. UAV autonomous navigation in complex environments.

TABLE I
COMPARISON OF THREE TYPES OF NAVIGATION ALGORITHMS

	classical planning algorithms	intelligent optimization algorithms	reinforcement learning algorithms
Features	use geometric, graph theory or probabilistic roadmap methods for path planning	Imitate biological group behavior or evolutionary process to search for optimal path	Learn optimal policy from experiences through interaction between the agent and environment
drawbacks	Rely on accurate global map information	relies on the rationality of parameter settings	Require a lot of training and experiences
complexity	simple ⟶ complicated		
efficiency	low ⟶ high		
real-time navigation	x	✓	✓

by generalizing from training experiences [8]. Liaq and Byun [9] provided a framework to apply traditional Q-learning method for UAV navigation in unknown environments. Zhang et al. [10] proposed a successor-feature-based DRL algorithm by using two-stream Q-network to perform navigation tasks in a dynamic environment. Singla et al. [11] created a deep Q-network (DQN)-based obstacle avoidance technique, which integrated a recurrent neural network with temporal attention, to maintain important information for obstacle avoidance in lengthy observation sequences. Nilwong and Capi [12] combined the training of DQN with the first-person shooter game-based simulation environment for outdoor robot navigation. Marchesini and Farinelli [13] optimized the double DQN network by employing parallel asynchronous training and multibatch priority experience replay to reduce the training time for map-less navigation. However, the considered environment is so simple that the state space is small. In general, DQN-based algorithms are restricted to a discrete action space that it is hard to plan a smooth trajectory in complex environments. It is not suitable for continuous control of UAV especially in dynamic environments.

To facility continuous control, some policy-gradient-based reinforcement learning (RL) algorithms are induced to solve the autonomous navigation problem [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27]. Chiang et al. [18] proposed a shaped deep deterministic policy gradient (shaped-DDPG) algorithm with optimized reward and network architecture to execute navigation behaviors. However, they focused on indoor navigation of Fetch robot which may not suitable for UAV applications. Walker et al. [19] also modeled UAV navigation in indoor environments into two separate problems, i.e., high-level planning and low-level action under uncertainty. Shi et al. [20] proposed an intrinsic curiosity model-based asynchronous advantage actor–critic (ICM-A3C) method for end-to-end navigation of mobile robots in map-less environments. Castellini et al. [21] used partially observable Monte Carlo planning method to navigate mobile robot in indoor environment. Xi and Liu [22] investigated to combine human knowledge with DDPG to train the UAV. They used a stage training scheme to obtain the final reward, i.e., reaching target phase, no-fly-zone avoidance phase, and efficient flying phase. Wan et al. [23] introduced three learning tricks in Robust-DDPG to improve the efficiency and stability, including delayed-learning, adversarial attack, and mixed exploration tricks. Although these algorithms can solve the continuous control problem in UAV navigation, they only considered a 2-D simple environment that the state and action spaces are small. In 3-D complex environments, UAV navigation is more challenging with more complicated state and action spaces.

In recent research, a few works considered UAV navigation in 3-D environments. Wang et al. [24] proposed a fast recurrent deterministic policy gradient (fast-RDPG) algorithm for UAV autonomous navigation. They developed a 3-D complex environment based on OpenAI gym to train the UAV and achieved good navigation performance. Next, they further solved the sparse reward problem by introducing a DRL with nonexpert helpers algorithm in [25]. Later on, Xue and Chen [26] extended their work with a recurrent stochastic valued gradient algorithm based on the actor–critic framework. However, these works considered to control UAV in X-Y plane by assuming that UAV flies at a fixed height, which is far from the real 3-D navigation. Bouhamed et al. [27] proposed a 3-D autonomous UAV navigation framework using the DDPG method to employ a self-trained UAV as an airborne Internet of Things, whereas they used the naive DDPG algorithm in simple environments with a few number of obstacles. Recently, Li et al. [28] combined range finders and depth image information to realize 3-D navigation in simple environment. So far, UAV autonomous navigation in 3-D complex environments still lacks of in-depth exploration, and the navigation performance of existing works needs further improvement to adapt to such environments.

In this work, we propose an improved state-decomposition DDPG algorithm for UAV autonomous navigation (SDDPG-NAV) in 3-D complex environments. The main contributions are in threefold as follows.

1) A novel state-decomposition method which uses two subnetworks for the perception-related and target-related states separately is developed to establish more appropriate actor networks, and output better navigation policy;

2) Some objective-oriented reward functions, including approaching the target, and avoiding obstacles and step rewards functions, are designed to solve the sparse-reward problem and to accelerate the convergence rate; and
3) Some training strategies are introduced to maintain an appropriate balance between exploration and exploitation. Numerous experiments are conducted to fine-tune the network structure and evaluate the performance of our proposed algorithm.

Moreover, various simulation results are presented to evaluate the convergence, navigation and generalization performance of our algorithm. Comparing with the benchmark DDPG and TD3 algorithms, SDDPG-NAV has higher convergence rate and achieves better rewards. The navigation performance in terms of success rate, collision rate, stray rate, and average path length is demonstrated to be better than other comparative algorithms. Finally, the comparison results in three generalized environments, including dense obstacles, large test area, and dynamic environments, are presented to validate the effectiveness of SDDPG-NAV.

The reminder of this work is as follows. Section II formulates the UAV autonomous navigation problem and models it as a Markov decision process (MDP). Then, the proposed SDDPG-NAV algorithm is described in detail in Section III. In Section IV, numerous simulation results are presented to demonstrate the effectiveness of the proposed algorithm. Finally, Section V gives some conclusion remarks and future works.

II. Preliminaries

This section first formulates UAV navigation problem in 3-D complex environments and gives dynamic functions of UAV. Next, the targeting problem is modeled as a MDP with continuous observations and actions.

A. Problem Statement

This work focuses on autonomous navigation of UAV in complex environments, e.g., goods delivery in crowded urban area or bulk warehouses. As is shown in Fig. 1, the environment is crowded with buildings and public infrastructures that have different shapes and sizes. There are also dynamic objects moving in the air, such as helicopters, other UAVs, flying birds, etc. Fig. 2 gives the sensing model of UAV used in this work. Assume that UAV is located at a random departure position $(x_0,\ y_0,\ z_0)$ in Cartesian geographical location, and targets at flying to the destination position $(x_g,\ y_g,\ z_g)$. Both deterministic and dynamic obstacles are randomly generated in the environment in each navigation task. UAV is assumed to have access to its sensor readings of local environment with range finders and its relative position to the destination with GPS.

A UAV system typically consists of six degrees of freedom and is controlled by an onboard autopilot, such as Pixhawk, which provides low-level flight controls and helps maintain roll, pitch, and yaw stability, as well as velocity tracking and altitude holding functions [23]. To simplify UAV's control in

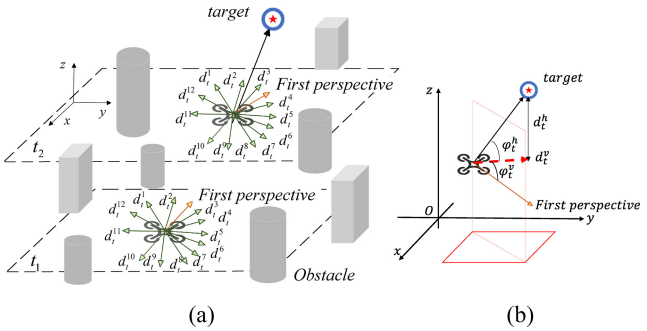


Fig. 2. Sensing model of UAV in complex environments: (a) obstacle-related observations S_o and (b) target-related observations S_g. Note that the orange line represents UAV's first perspective direction and the green lines represent signals emitted from range finders.

complex environments, most previous works only considered to control the velocity and direction of UAV in X-Y plane by assuming a constant flying altitude [23], [24], [25]. However, in 3-D environments, it is hard to obtain a smooth and feasible trajectory without considering the control of flying altitude. In this work, we explore more realistic scenarios with the following dynamic functions of UAV, i.e.,

$$\begin{cases} x_{t+1} = x_t + v_{t+1}\cos(\theta_{t+1}) \\ y_{t+1} = y_t + v_{t+1}\sin(\theta_{t+1}) \\ z_{t+1} = z_t + \omega_{t+1} \\ \theta_{t+1} = \theta_t + \phi_{t+1} \\ v_{t+1} = v_t + a_{t+1} \\ \omega_{t+1} = \omega_t + \kappa_{t+1} \end{cases} \tag{1}$$

where $\boldsymbol{p}_t = [x_t, y_t, z_t]$ is the location of UAV at time t, and θ_t, v_t, and ω_t refer to the heading direction, speed, and flight height change of UAV, respectively. Note that ϕ_t, a_t, and κ_t are used to control yaw angle, speed, and flight height, separately.

The main *objective* is to find a feasible and collision-free path that navigate UAV from the departure position to the destination in 3-D complex environments.

B. Modeling UAV Navigation as MDP

In this work, UAV autonomous navigation is modeled as a MDP with continuous observation and action spaces. Formally, an MDP is represented by a 5-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \gamma, \mathcal{R})$, where $\mathcal{S}$ is the state space; $\mathcal{A}$ is the action space; $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function that returns the conditional probability $\mathcal{P}(s'|s, a)$, i.e., the probability from current state s to the next state s' after taking action a; $\gamma \in (0, 1)$ is a scalar discount factor; and $\mathcal{R} : \mathcal{S} \times \mathcal{A}$ is the reward function.

State space represents the environmental information perceived by UAV and the changes brought about after each action. It is the basis for UAV to make decisions and evaluate its long-term cumulative rewards. In the formulation, the state space is composed of three parts, i.e., the internal states of UAV, the perceptual information of nearby obstacles, and the target information.

1) The internal states of UAV consist of its heading direction θ_t (relative to the north), the speed v_t, and flight height ω_t at time step t. It is denoted as $s_u = [v_t,\ \theta_t,\ \omega_t]$. Note that the absolute position of UAV is not added into

the state space since it would weaken the generalization ability to different environments [22];

2) The perceptual information of nearby obstacles is obtained from the distances returned by range finders. As is shown in Fig. 2(a), 12 range finders are used to characterize the surrounding environment. At each time step, the range finders reflect information of surrounding environment. The obstacle-related observations are denoted as $s_o = [d_t^1, d_t^2, \ldots, d_t^{12}]$.
3) The target information is represented by the relative distance and angle of UAV to the destination. Let d_t^v and d_t^h be the relative distances of UAV to the destination in horizontal and vertical directions, respectively. φ_t^v and φ_t^h be the horizontal and vertical angle relationship separately. As is shown in Fig. 2(b), the target-related observations are denoted as $s_g = [d_t^v, d_t^h, \varphi_t^v, \varphi_t^h]$.

The obstacle-related observations mainly prevent the UAV from colliding with nearby obstacles and keep safe navigation. The target-related observations help to control flight height of the UAV. Finally, the action space is denoted as $A = [a_t,\ \phi_t,\ \kappa_t]$. Here, $a_t \in [-1, 1]$, is used to modulate the speed of UAV, $\phi_t \in [-1, 1]$ controls UAV's heading direction and $\kappa_t \in [-1, 1]$ decides the change of flight height.

III. Methods

This section describes the method used to solve UAV autonomous navigation in 3-D complex environments. First, the RL model for UAV navigation is introduced. Next, the proposed improved state-decomposition DDPG algorithm is described in detail. Then, the objective-oriented reward functions are developed. Finally, the network architecture and algorithm pseudo-code are also presented.

A. RL for UAV Navigation

In this work, we develop an RL-based UAV autonomous navigation model to transfer inputs of sensor observations and relative position of destination to commands of UAV's movement. In the model, UAV observes current state s_t at time step t, and chooses an action a_t according to policy π. Next, it executes the action and transits environment into a new state s_{t+1} based on the state transition probability $\mathcal{P}$. Then, it receives an immediate reward $r(s_t, a_t)$. The goal is to learn an optimal policy π^* that maximizes the cumulative discounted reward.

The reward function usually has two forms, one is the state-value function, which is defined as

$$V_\pi(s_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t\right]. \tag{2}$$

Another is the action-value function that denotes the expected return after taking action a_t in state s_t and thereafter following policy π:

$$Q_\pi(s_t, a_t) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r(s_{t+k}, a_{t+k}) | s_t, a_t\right]. \tag{3}$$

For UAV navigation problem, both the state and action spaces are tremendous. The action-value function cannot be accurately obtained and needs parameterizing, i.e., $\hat{Q}(s, a, \omega) \approx Q_\pi(s, a)$. The parameterized action-value function can be updated by applying time-difference (TD) errors as follows:

$$\hat{Q}(s, a, \omega) \leftarrow \hat{Q}(s, a, \omega) + \alpha\left[r + \gamma \hat{Q}(s', a', \omega) - \hat{Q}(s, a, \omega)\right] \tag{4}$$

where α is the learning rate used to control the updating speed; and $r + \gamma \hat{Q}(s', a', \omega)$ is TD-target and $r + \gamma \hat{Q}(s', a', \omega) - \hat{Q}(s, a, \omega)$ is TD-error. Then the optimal policy can be determined after (4) converges

$$\pi^* = \arg\max_{\pi} \hat{Q}(s, a, \omega). \tag{5}$$

B. Efficient State-Decomposition DDPG Algorithm

In UAV autonomous navigation, the state transition probability $\mathcal{P}$ is unknown in prior. Agent needs to approximate the action-value function to learn an optimal policy. The traditional DQN algorithm uses a deep neural network as the Q function appropriator and applies gradient descent to minimize the objective function. It only handles discrete and low-dimensional action space. In complex environments, DQN has to go through an expensive discretization process. In this work, we propose a novel state-decomposition deep deterministic policy gradient algorithm for efficient UAV autonomous navigation with continuous action space, i.e., SDDPG-NAV for short. The framework of SDDPG-NAV is illustrated in Fig. 3.

As is shown in Fig. 3, SDDPG-NAV uses actor and critic neural networks to concurrently learn Q-function and policy, respectively. In the actor–critic framework, actor decides which action should be taken and critic informs actor how good was the action and how it should adjust. Critic evaluates an action produced by actor through computing the state value function with minimized TD errors

$$L\left(s, a\middle|\theta^Q\right) = \left[r(s, a) + \gamma Q'\left(s', a'\middle|\theta^{Q'}\right) - Q\left(s, a\middle|\theta^Q\right)\right]^2. \tag{6}$$

This is also defined as loss function. Then parameters of the critic network are updated by applying policy gradient

$$\theta^Q \leftarrow \theta^Q + \alpha \nabla_\theta L\left(s, a\middle|\theta^Q\right). \tag{7}$$

The actor function is parameterized by $\mu(s|\theta^\mu)$ and updated by applying the chain rule to the expected return from state distribution J [14]

$$\nabla_{\theta^\mu} J = \mathbb{E}\left[\nabla_a Q\left(s, a\middle|\theta^Q\right)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta^\mu} \mu\left(s\middle|\theta^\mu\right)|_{s_t}\right]. \tag{8}$$

To accelerate the convergence rate and increase the generalization ability of SDDPG-NAV, three main improvements are proposed as follows.

1) Decomposition of State Space: The state space of UAV is governed by three main components: 1) internal state s_i, including current velocity, direction, and flight hight, that determines UAV's flying state; 2) target-related state s_g that guides UAV toward the target position; and 3) perception-related state s_o that ensures UAV's avoidance of

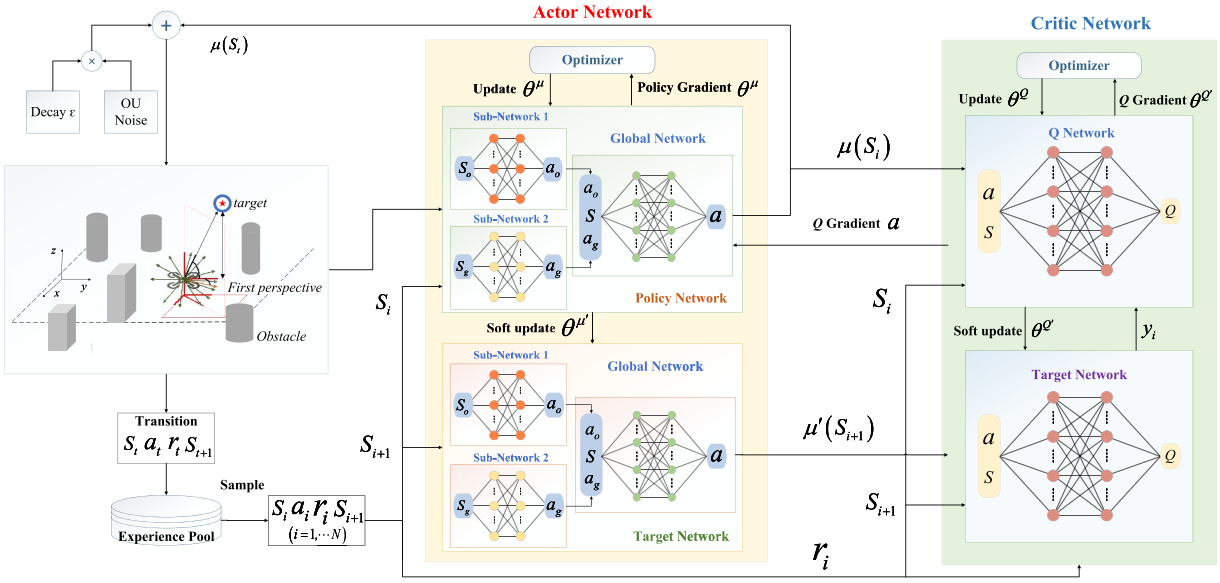


Fig. 3. Framework of the proposed SDDPG-NAV algorithm.

nearby obstacles timely. Therefore, to enable UAV to gain a comprehensive understanding of the environment and generate a high-quality policy, a state-decomposition strategy is developed to establish actor networks in a step-by-step manner. Specifically, this strategy involves the implementation of two subnetworks for perception-related and target-related states separately, followed by the use of a global network for the merged states and actions.

As can be observed in the middle of Fig. 3, subnetworks 1 and 2 independently take s_o and s_g as input, and output actions a_o and a_g accordingly. The former is responsible for obstacle avoidance, while the latter guides UAV toward the target. Subsequently, the global network combines a_o, a_g and s as new input, and outputs final action a, where s is the overall state space, i.e., $s = [s_i, s_o, s_g]$. The state-decomposition strategy is superior to the conventional approach that employs a single network, which inputs all state information into one actor network. This is because our method effectively distinguishes the impact of different trigger states to generate good decisions.

2) Target Soft Update and Smooth L_1 Loss: With high complexity of UAV autonomous navigation in complex environments, neural network function approximators are used to learn in large state and action spaces. However, directly implementing Q learning (6) with neural networks is proved unstable and the Q-update is prone to divergence. Therefore, the "soft" target update method is used by creating copies of actor and critic networks, i.e., target actor network and target critic network as shown in the middle and right bottom of Fig. 3. The weights of these target networks are then updated by having them slowly track the learned networks $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$, where $\tau \gg 1$ [14]. Moreover, the experience relay buffer is also used to enable the actor and critic updates by sampling a minibatch uniformly from the buffer as in the left bottom of Fig. 3. With a large relay buffer, the algorithm can benefit from learning across a set of uncorrelated transitions.

Meanwhile, the update may be unstable if the policy network changes radically. To enhance the stability of our model, a smooth L_1 loss is implemented in the objective function when sampling from the experience pool and updating the parameters of critic network. Specifically

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise.} \end{cases} \tag{9}$$

3) Exploration With Decayed OU Noise: Since the major challenge of learning in continuous action space is exploration, an decayed exploration policy is developed by adding noise sampling from a noise process $\mathcal{N}$ to the actor policy as is shown in the left top of Fig. 3

$$\mu'(s_t) = \mu\left(s_t|\theta_t^{\mu}\right) + \varepsilon\mathcal{N} \tag{10}$$

where $\mathcal{N} \sim \mathcal{OU}(\mu, \theta, \sigma)$ is an Ornstein–Uhlenbeck (OU) process and ε is the dynamic decay factor. As is demonstrated in [29], the use of OU noise can generate temporally correlated exploration for exploration efficiency in physical control problems with inertia. This stochastic process encourages the agent keep exploring state and action spaces. Besides, a dynamic ε decay is added to the OU noise to make tradeoff between exploration and exploitation. In the early stage of training, exploration noise plays a dominant role to sufficiently explore the environment. With the number of training episodes increases, ε gradually decays and the proportion of noise decreases. With the decayed OU noise, UAV can take actions based on what it learned from the environment with exploitation.

In general, the aforementioned three strategies used in SDDPG-NAV stabilize the learning process while maintain a good balance between exploration and exploitation.

C. Objective-Oriented Reward Functions

In RL, the design of reward function is critical since a rational reward can reinforce the action of agent. Reward acts as a signal evaluating how good it is when taking an action at a

state [24]. In UAV navigation, a straightforward design is using sparse reward, in which agent gets reward only when it arrives at the target position. However, in complex environment with dense obstacles, it is almost impossible for UAV to reach the destination. It is easy to get lost or stuck in such an environment with sparse reward. Fortunately, reward shaping provides agent with a specific form of nonsparse reward with the guarantee of policy invariance [24], [30].

In this work, we propose an objective-oriented reward shaping method to design the nonsparse reward for UAV navigation in complex environments. The reward functions consist of three parts.

1) *Approaching the Target:* To encourage UAV head for the target position, any action that makes UAV fly close to the destination should be awarded, and other actions should be penalized. Let d_t^g be the distance between UAV and its destination at time step t. The reward function for approaching the destination is defined as

$$r_{\text{tag}} = \begin{cases} -30, & d_t^g > d_0 \\ \dfrac{d_{t-1}^g - d_t^g}{\left|d_{t-1}^g - d_t^g\right|} e^{0.05\left(\left|d_{t-1}^g - d_t^g\right|\right)}, & \text{otherwise.} \end{cases} \tag{11}$$

Note that d_0 is a positive constant that prevents UAV from deviating too far from the target position.

2) *Avoiding Obstacles:* For safe flight of UAV, a colliding penalty should be given to ensure that UAV will not get too close to obstacles. In this work, obstacle penalty is modeled as a continuous function of distance, which is proved to be more efficient than the discrete penalty [27]. The function is given by

$$r_{\text{obs}} = \sigma\left(\frac{d}{D_r} - 1\right) \quad d = \min(d_1, \ldots, d_{12}) \tag{12}$$

where σ is a positive constant representing the scale of penalty, D_r is the detective range, and d_i is the distance returned by the ith range finder.

3) *Step Award:* To guide UAV approaching the destination as soon as possible, a constant step award r_{step} is given after each transition.

To summarize, the final reward function for UAV navigation at time step t is given by

$$r = r_{\text{tag}} + r_{\text{obs}} + r_{\text{step}}. \tag{13}$$

With the appropriately designed reward functions, the convergence rate of SDDPG-NAV is further accelerated.

D. Training

With the novel state decomposition strategy, SDDPG-NAV employs two subnetworks in the actor network which requires additional adjustments. In general, there are two hyperparameters must be determined, i.e., the number of hidden layers and that of neurons in each layer. However, one cannot analytically calculate these hyperparameters in an artificial neural network. On one hand, a theoretical finding in [32] suggests that a multilayer perception with two hidden layers is sufficient for creating classification regions of any desired shape. Therefore, following most settings in [14] and [25], we use two hidden layers in the subnetworks. On the other hand, the number of neurons in each layer is also challenging. Choosing too few neurons will result in under-fitting and high-statistical bias, whereas choosing too many neurons may lead to over-fitting, high variance and also increase the time it takes to train the network. Therefore, a systematic experimentation is needed to discover what works best.

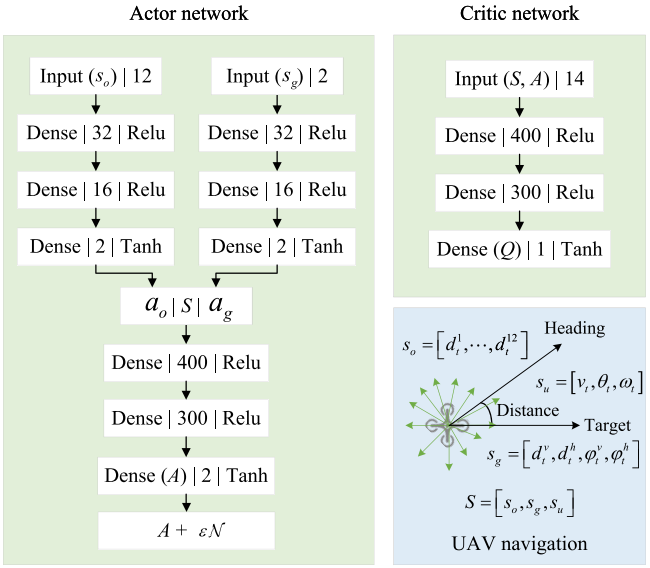


Fig. 4. Network architecture of SDDPG-NAV.

The parameters of hidden-layers are optimized through a good deal of experiments with different random seeds and network sizes. In specific terms, we employed 10 different random seeds to train the neural network over 2000 episodes. Subsequently, each of the 10 models was subjected to 100 performance tests. By comparing the performance of the hidden layers with varying numbers of neurons in the two subnetworks, we ultimately determined the optimal number of neurons for the subnetworks. During training, the subnetworks' hidden layers were implemented as fully connected layers with ReLU activation functions. Additionally, we utilized Adam optimizers to update network parameters. The optimal network structure is selected when SDDPG-NAV achieves the highest success rate in various scenarios. In detail, Fig. 4 illustrates the network architecture of SDDPG-NAV and gives the number of neurons used. In the actor network, two subnetworks are used for the decomposed state information as in Fig. 4(a). In subnetwork 1, the input layer consists of 12 range finder feedbacks, and the hidden layers use 32 × 16 neurons with Relu function. In subnetwork 2, the inputs are the heading direction and speed of UAV, and the hidden layers also take 32 × 16 neurons. Next, outputs of the two subnetworks combined with the overall state information are put into the global network for final decision. The critic network in Fig. 4(b) takes the same structure with the global network, and there are 14 inputs. Finally, Fig. 4(c) gives the decomposed state information used in the actor and critic networks.

To summarize, Algorithm 1 gives the pseudo-code of SDDPG-NAV. It should be noted that to train a more

Algorithm 1: Proposed SDDPG-NAV Algorithm

1 Initialize critic $Q(s,a|\theta^Q)$ and actor $\mu(s|\theta^\mu)$ networks with parameters θ^Q and θ^μ;
2 Initialize target networks $Q\left(s,a\middle|\theta^{Q'}\right)$ and $\mu\left(s\middle|\theta^{\mu'}\right)$ with $\theta^{Q'} \leftarrow \theta^Q$, $\theta^{\mu'} \leftarrow \theta^\mu$;
3 Initialize Experience Pool D;
4 **for** *episode* $= 1, E$ **do**
5 Initialize the environment and navigation task;
6 Receive first observation s_1;
7 Decompose s_1 into s_i, s_o and s_g;
8 Initialize random process $\mathcal{N}$ for action exploration;
9 **for** $t = 1, T$ **do**
10 Input s_o, s_g to sub-actor network 1, 2 and output a_o, a_g;
11 Concatenate (a_o, a_g, s_1) and input it to global actor network, output a;
12 Obtain action $a_t = a + \varepsilon \times \mathcal{N}_t$;
13 Execute action a_t, obtain reward r_t and next state s_{t+1};
14 Push transition (s_t, a_t, r_t, s_{t+1}) into D;
15 Randomly Sample a minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from D;
16 Set $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1}\middle|\theta^{\mu'}\right)\middle|\theta^{Q'}\right)$;
17 Update critic by minimizing the loss:
18 $L = \frac{1}{N}\sum_i smooth_{L_1}\left(y_i - Q\left(s_i, a_i|\theta^Q\right)\right)$;
19 Update the actor policy using policy gradient:
 $\nabla_{\theta^\mu} J \approx$
 $\frac{1}{N}\sum_i \nabla_a Q\left(s, a|\theta^Q\right)\big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu}\mu(s|\theta^\mu)\big|_{s_i}$;
20 Soft update the target networks:
 $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
21 $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$;
22 **end**
23 $\varepsilon = \max(\varepsilon - 0.002, 0.001)$
24 **end**

generalized model, the environment is reset with random obstacles, and the navigation task is also randomly chosen in each episode, i.e., the starting and destination positions of UAV are randomly generated (Algorithm 1 line 5). Note that the selected starting and destination positions are out of the obstacle area. The policy is executed for E episodes, and each episode terminates when either UAV arrives at the destination or collides with obstacles, or the maximum time step T is reached.

IV. Experimental Results

In this section, numerous experiments are conducted to evaluate the performance of SDDPG-NAV, and compare it with some benchmark works.

A. Experiment Settings

The experiment environment is constructed based on literature [25] and OpenAI gym [31] to implement UAV autonomous navigation in 3-D complex environments. The

TABLE II
Hyperparameter Settings

Hyperparameters	Values
Learning rate of actor network	0.0004
Learning rate of critic network	0.004
Discount factor γ	0.99
Batch size N	128
Size of experience pool	1,000,000
Soft update factor τ	0.001
Dynamic ϵ decay	0.7
Target distance d_0	10,000m
Scale of penalty σ	50
Step reward	-5
Detection range of range finders D_r	100m

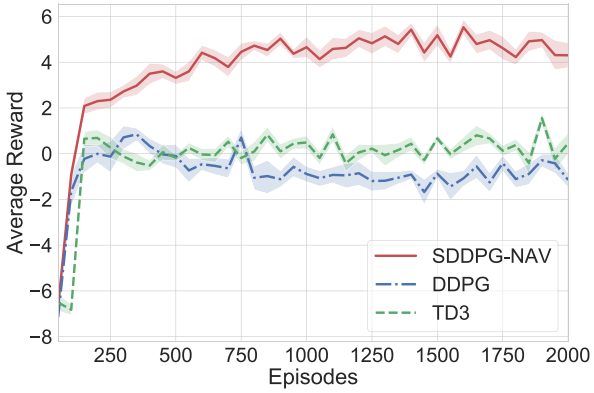
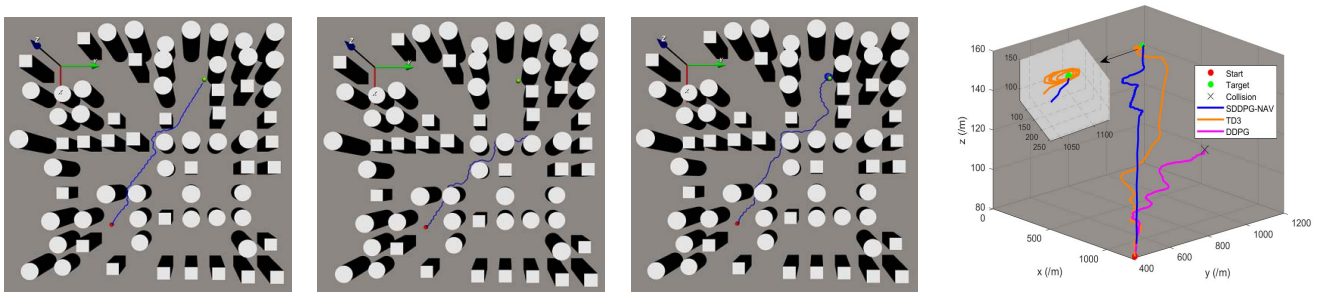


Fig. 5. Convergence performance of SDDPG-NAV, DDPG, and TD3.

training environment covers a square area of 1800×1800 m^2. In the simulation, two types of obstacles are considered, i.e., cylinder and cube shaped obstacles. The radius (or length) of these obstacles are set to be 60 m, and their heights vary between 30 and 240 m. Moreover, for safe navigation, the minimum distance between obstacles is 180 m. The training process consists of a total number of 2000 episodes, and the maximum step size of each episode is 300. At the beginning of each episode, the environment is reset with randomly scattered obstacles and random starting and destination positions. This ensure that UAV can navigate in a variety of complex scenarios, leading to effective training and robust autonomous navigation capabilities. Consider practical applications, the distance between the starting and destination positions should be no less than 200 m as in [25]. The hyperparameters used in SDDPG-NAV are summarized in Table II.

In the experiment, the Monte Carlo method is employed to evaluate the convergence rate, navigation performance, and generalization capability of the autonomous navigation algorithm. In addition, the performance of SDDPG-NAV is compared with two benchmark works, i.e., DDPG [14] and TD3 [16]. It should be noted that DDPG and TD3 are among the most commonly used machine learning algorithms for autonomous decision making in continuous environments. The utilization of Monte Carlo method allows for rigorous evaluation of the proposed autonomous navigation system in terms of its ability to learn and generalize to new scenarios.



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

Fig. 6. Top view of navigation trajectories in training environment: (a) proposed SDDPG-NAV; (b) DDPG; (c) TD3; and (d) 3-D trajectories. Note that the starting and destination positions of UAV are denoted by red and green spheres, respectively.

B. Convergence Rate

Convergence curves of SDDPG-NAV, DDPG and TD3 are given in Fig. 5. During the training process, we train the model with different random seeds for 2000 episodes and record the average rewards obtained for each episode. As is demonstrated in Fig. 5, all the comparative algorithms converge quickly, and SDDPG-NAV achieves the highest average reward, DDPG and TD3 achieve similar rewards. This can be attributed to the objective-oriented reward functions and decayed OU noise used in SDDPG-NAV. First, by jointly considering three types of rewards, UAV's actions are appropriately evaluated, which effectively accelerates the convergence rate. Second, with dynamic ε decay in exploration, the efficiency of data utilization is guaranteed. At early stage of the training process, more explorations are attempted. While with more training episodes, the proportion of exploration noise decreases, and UAV takes decisions mainly based on the trained policy with exploitation. As a result, SDDPG-NAV obtains more rewards than its comparative counterparts.

C. Navigation Performance

In this section, we first give flight trajectories of UAV by executing three comparative algorithms in test environment. Then, the navigation performance is evaluated in four aspects.

1) *Average Success Rate:* The fundamental objective of UAV navigation is to guide UAV from the starting position to the destination successfully. The average success rate is calculated as the ratio of successfully completed navigation missions to the total number of navigation missions.
2) *Average Collision Rate:* This metric evaluates the collision avoidance ability of the trained model. It is defined as ratio of number of failed navigation missions due to colliding with obstacles to the total number of navigation missions.
3) *Average Stray Rate:* In complex environments, UAV may encounter difficulties, such as getting lost or becoming trapped in specific positions. This metric evaluates the ratio of number of failed missions because of being trapped in environments to the total number of navigation missions.
4) *Average Path Length:* In practical applications, UAV navigation should take less time and shorter path length to complete a specific task. This metric measures the average path length traversed by the UAV during successful navigation missions, considering the same starting and destination positions.

Fig. 6 illustrates trajectories of three comparative algorithms in the same test environment. It can be observed that with SDDPG-NAV and TD3, UAV can complete the navigation mission and successfully reach the expected destination, whereas with DDPG, the navigation mission is failed because of collision. In Fig. 6(a), UAV directly flies to the destination with a short path, and the trajectory is smooth without taking any spin. In Fig. 6(c), although UAV can avoid obstacles and successfully reach the destination by implementing TD3, it heavily spins around the destination which results in a long path length. In the test environment, the position of destination is very close to an obstacle. With TD3, the agent cannot effectively understand the environment that it may be confused in such situation. However, SDDPG-NAV uses two subnetworks for perception-related and target-related states separately, so that the agent can generate a better policy. Fig. 6(d) gives 3-D trajectories of the comparative algorithms. One can observe that the UAV climbs quickly at the departure stage, slightly changes its flight height after it reaches some altitude, and gently approaches the destination. It also demonstrates that our proposed SDDPG-NAV algorithm can plan a good 3-D trajectory without any spin, TD3 plans a long trajectory and spins around the destination, while DDPG failed to generate an appropriate path.

Next, the navigation performance in training environment is given in Fig. 7 where the total number of navigation missions is 100. All the results are averaged from tests in 10 different environments with randomly distributed obstacles and the density of obstacles is 0.4. As can be observed from Fig. 7(a)–(c) that the proposed SDDPG-NAV algorithm achieves the highest success rate and the lowest collision and stray rates than the comparative DDPG and TD3 algorithms in 3-D training environment. DDPG takes similar collision rate with TD3, but much higher stray rate than TD3, so that it shows the worst navigation performance. Fig. 7(d) also demonstrates that the average path length of SDDPG-NAV is smaller than that of DDPG and TD3. In general, the proposed SDDPG-NAV shows

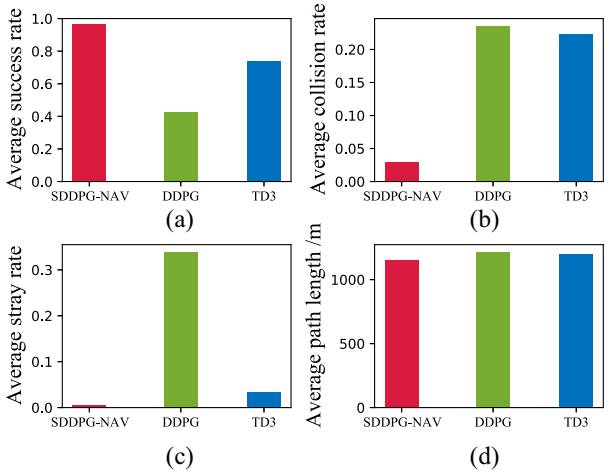


Fig. 7. Navigation performance in training environment: (a) average success rate; (b) average collision rate; (c) average stray rate; and (d) average path length.

TABLE III
NAVIGATION PERFORMANCE IN THREE TYPICAL ENVIRONMENTS

Environment	Performance	SDDPG-NAV	DDPG	TD3
ENV-II	Success rate	**93.6%**	35.8%	55.8%
	Collision rate	**5.80%**	36.0%	40.4%
	Stray rate	**0.60%**	28.2%	3.80%
	Path length (m)	**1176.11**	1217.89	1221.17
ENV-III	Success rate	**89.3%**	55.6%	74.8%
	Collision rate	**9.00%**	34.4%	25.2%
	Stray rate	**1.70%**	10.0%	0.00%
	Path length (m)	**2387.69**	2425.31	2544.76
ENV-IV	Success rate	**90.5%**	31.0%	69.6%
	Collision rate	**7.90%**	32.9%	27.8%
	Stray rate	**1.60%**	36.1%	2.60%
	Path length (m)	**847.74**	900.59	905.34

better navigation performance than the comparative algorithms in 3-D complex environments.

D. Generalization Capability

Generalization capability is evaluated by complexing the test environment in three aspects.

1) *With Denser Obstacles:* The density of obstacles x_d can be varied by controlling two parameters: the number of obstacles and the distance between them. By increasing these two parameters, the density of obstacles increases accordingly resulting a more complex environment. Thus, the navigation mission is more challenging.
2) *With Larger Test Area:* UAV may operate in environments with different sizes. To assess the feasibility of UAV navigation algorithms in larger areas, we expand the area of training environment by x_a times in the test, where $x_a \in [1, 8]$. Note that the original training environment is 1800×1800 m^2.
3) *With Dynamic Obstacles:* To evaluate the navigation performance in dynamic environments, we reset the positions of obstacles every x_f steps, where $x_f \in [10, 50]$. With frequently refreshed random obstacles, the environment keeps changing and unpredictable.

Fig. 8 illustrates the trajectories of SDDPG-NAV in four typical environments, i.e., 1) *ENV-I:* training environment with $x_d = 0.4$, $x_a = 1$, and $x_f = +\infty$; 2) *ENV-II:* environment with dense obstacles and $x_d = 0.8$, $x_a = 1$, $x_f = +\infty$; 3) *ENV-III:* environment with enlarged area and $x_d = 0.6$, $x_a = 8$, $x_f = +\infty$; and 4) *ENV-IV:* environment with dynamic obstacles and $x_d = 0.4$, $x_a = 1$, $x_f = 45$. It should be noted that $x_f = +\infty$ refers to the static environment. As is demonstrated, SDDPG-NAV can effectively navigate UAV from the starting position to the destination in various complex environments and the trajectory is smooth.

Moreover, the navigation performances of SDDPG-NAV, DDPG and TD3 are evaluated with 100 repeated missions in environments ENV-II, ENV-III, and ENV-IV, respectively. The averaged results are given in Table III, and the bold values indicate the best performance among the comparative algorithms. As can be observed that SDDPG-NAV achieves the best performance in terms of higher success rate, lower collision and stray rates in the three test environments, while DDPG and TD3 have much higher collision and stray rates. The average path length of SDDPG-NAV is smaller than other comparative algorithms as well.

To clearly show the generalization capabilities of the comparative algorithms, numerous results are presented in different environments by changing density of obstacles, time of the training environment, and refresh interval of obstacles.

First, Fig. 9 gives the navigation performance of three comparative algorithms in environments with varying densities of obstacles. Note that the size of test environment is the same as the training environment. As is shown in Fig. 9(a), SDDPG-NAV achieves the highest success rate of approximately 98% at a density of 0.4, which is at least 13.7% and 42.3% higher than that of the TD3 and DDPG algorithms, respectively. Furthermore, as the density of obstacles increases, the success rate of SDDPG-NAV does not decrease as significantly as that of the other two algorithms, remaining above 93% even with high density of obstacles. This is because SDDPG-NAV utilizes two subactor networks, which enables it to gain a more profound understanding of the state space of the environment. Additionally, with efficient training strategies and appropriate reward functions, policies generated by SDDPG-NAV can guide the UAV to execute smooth actions.

As is shown in Fig. 9(b), SDDPG-NAV has much smaller collision rate than other two comparative algorithms. Comparing with TD3 and DDPG, SDDPG-NAV reduces collision rate by at least 85% and 88%, respectively. As the density of obstacles increases, collision rate of SDDPG-NAV increases gently, while the corresponding values of TD3 and DDPG increases sharply. This demonstrates that our proposed SDDPG-NAV algorithm is more robust in dense environments. Then, Fig. 9(c) gives stray rates of the comparative algorithms when density of obstacles increases. It shows that both SDDPG-NAV and TD3 take very low-stay rate and rarely get stuck, but DDPG gets stuck more easily with a stray rate higher than 25%. In RL algorithms, agent keeps exploring environment to gather more information for a better decision.

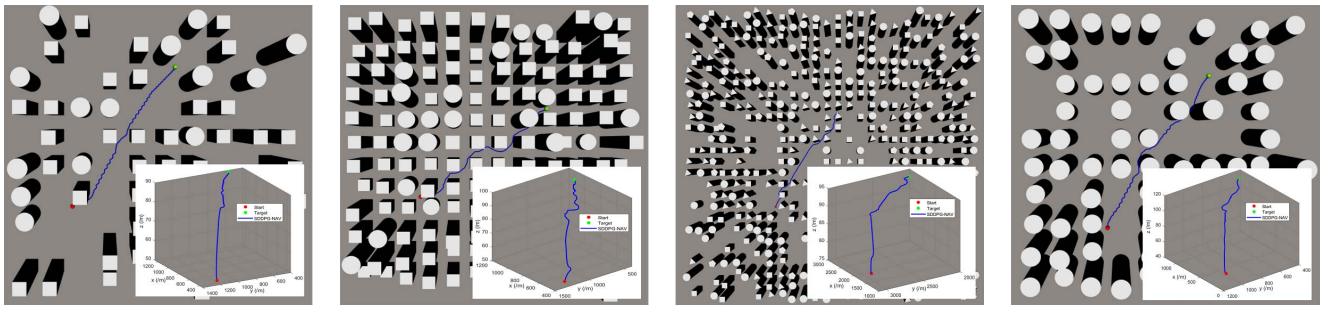


Fig. 8. Trajectories of SDDPG-NAV in 3-D complex environments: (a) training environment; (b) environment with dense obstacles; (c) environment with enlarged area; and (d) environment with dynamic obstacles. Note that obstacles in different types are distinguished from shapes, sizes, and mutual space in the test environment.

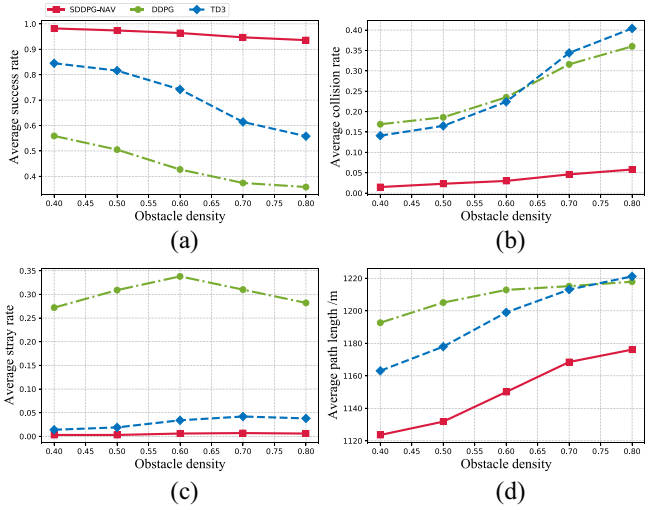


Fig. 9. Navigation performance in the environment of different obstacle densities: (a) average success rate; (b) average collision rate; (c) average stray rate; and (d) average path length.

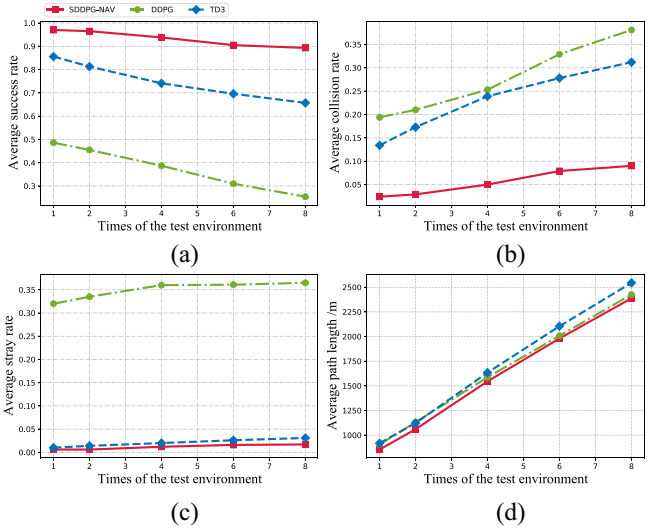


Fig. 10. Navigation performance in the environment of different environmental areas: (a) average success rate; (b) average collision rate; (c) average stray rate; and (d) average path length.

With efficient explorations in SDDPG-NAV and TD3, UAV can effectively escape from traps and hardly get stuck.

Fig. 9(d) exhibits the average path length when distance between the starting position and destination is 1000 m. As can be observed, average path lengths of the three comparative algorithms increases with the density of obstacles. With denser obstacles, the environment is more complex that UAV needs longer path to avoid obstacles. As is demonstrated, SDDPG-NAV takes the shortest path length than other algorithms. Comparing with TD3 and DDPG, SDDPG-NAV reduces the average path length by at least 3.4%. Consequently, SDDPG-NAV makes better decisions than the other comparative algorithms.

Second, Fig. 10 gives the navigation performance in environments with larger areas where the density of obstacles is 0.4. As is shown in Fig. 10(a), success rate of SDDPG-NAV keeps above 89.9% which is much higher than these of other comparative algorithms. When the test environment becomes larger, success rate of SDDPG-NAV decreases very flatly, while these of TD3 and DDPG drop quickly. In detail, the rates of decent of TD3 and DDPG are about 23.3% and 51%, respectively. The rate of decent of SDDPG-NAV is 7.3% which is much lower than the comparative algorithms. Similarly, in Fig. 10(b), collision rate of SDDPG-NAV is lower than 0.1 which is much smaller than TD3 and DDPG. The collision rates of SDDPG-NAV increases much slower than other algorithms when test environment becomes larger. Fig. 10(c) and (d) also demonstrate that SDDPG-NAV has lower stray rate and can plan a shorter path. Consequently, SDDPG-NAV shows better performance than the comparative counterparts when implementing in environments with larger areas.

As is demonstrated in Figs. 9 and 10, DDPG exhibits the poorest navigation performance in both dense and large environments. This can be attributed to the fact that DDPG employs only one actor network to handle a large state space. Consequently, DDPG may struggle to comprehend the relevant information in the state space of the UAV. In contrast, TD3 utilizes two critic networks to evaluate the action value, resulting in a reduction in the occurrence of incorrect

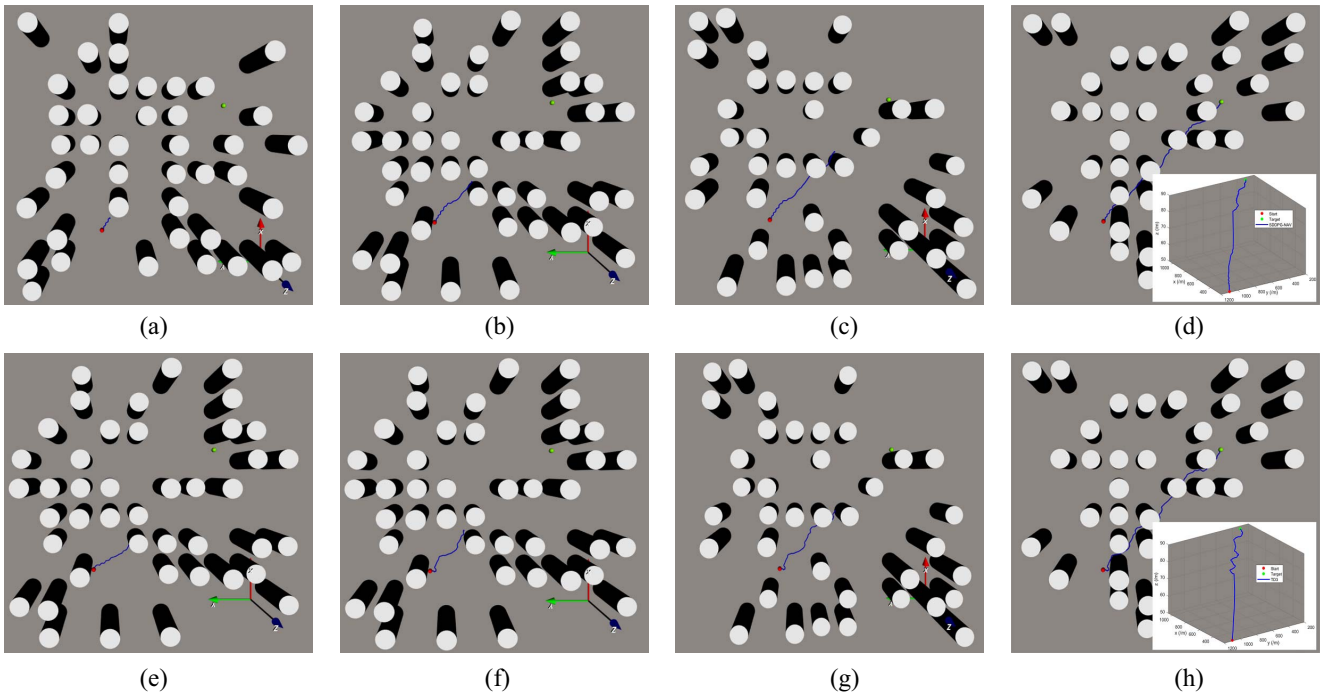


Fig. 11. Navigation trajectories of three algorithms under dynamic environments: (a) initial state at t_0; (b) UAV's trajectory at t_1 using SDDPG-NAV; (c) UAV's trajectory at t_2 using SDDPG-NAV; (d) UAV's trajectory at t_3 using SDDPG-NAV; (e) UAV's trajectory at t_1 using DDPG; (f) UAV's trajectory at t_1 using TD3; (g) UAV's trajectory at t_2 using TD3; and (h) UAV's trajectory at t_4 using TD3. Note that $t_0 < t_1 < t_2 < t_3 < t_4$, and UAV collides with obstacles at t_1 using DDPG that the mission is failed.

TABLE IV
NAVIGATION PERFORMANCE OF THREE COMPARATIVE ALGORITHMS IN DYNAMIC ENVIRONMENTS

Refresh interval	Success rate			Collision rate			Stray rate			Average path length (m)		
	SDDPG-NAV	DDPG	TD3	SDDPG-NAV	DDPG	TD3	SDDPG-NAV	DDPG	TD3	SDDPG-NAV	DDPG	TD3
50	**94.0%**	57.0%	82.0%	**6.0%**	30.0%	18.0%	**0.0%**	13.0%	0.0%	**870.69**	921.88	890.98
40	**89.2%**	55.6%	74.8%	**10.8%**	34.0%	25.2%	**0.0%**	10.4%	0.0%	**824.64**	939.68	898.29
30	**85.8%**	57.6%	75.6%	**14.0%**	35.0%	24.4%	0.2%	7.4%	**0.0%**	**841.95**	878.47	896.57
20	**78.0%**	51.4%	67.2%	**21.8%**	42.6%	32.8%	0.2%	6.0%	**0.0%**	**830.67**	920.87	858.50
10	**57.2%**	41.6%	51.4%	**42.6%**	55.2%	48.6%	0.2%	3.2%	**0.0%**	**803.50**	851.78	850.02

actions. Additionally, the proposed SDDPG-NAV decomposes state information and trains them individually, leading to the generation of more robust policies. Therefore, SDDPG-NAV shows the best performance, and DDPG exhibits the worst performance. In general, one can infer that SDDPG-NAV is more robust and shows better performance than the comparative counterparts in both environments with dense obstacles and large sizes. It also shows very good generalization ability in complex environments.

Third, navigation performance of the comparative algorithms in dynamic environments are expressed in Table IV. In each mission, the environment is refreshed with random obstacles every x_f ($x_f \in [10,\ 50]$) steps. Lower x_f refers to higher refresh frequency and positions of obstacles change more frequently. Thus, the environment is more dynamic and the navigation mission is more challenging. As is shown in Table IV, affected by dynamic obstacles, success rates of the compared algorithms decreases with refresh interval, and the collision and stray rates increase accordingly. Note that the bold values give the best performance among the three comparative algorithms. Nevertheless, SDDPG-NAV always exhibits better performance than other comparative counterparts. Comparing with TD3 and DDPG, SDDPG-NAV increases the success rate by at least 10.64% and 28.2%, respectively. One can also observe that the average path length increases with refresh interval, and SDDPG-NAV always takes the shortest path. Therefore, SDDPG-NAV shows better capability in adapting to changing circumstances and is more effective in navigating UAV under dynamic environments.

Finally, to clearly demonstrate the navigation process in dynamic environments, Fig. 11 depicts the navigation trajectories of three comparative algorithms starting from the same environment. In the test, $x_d = 0.4$, $x_a = 1$, and $x_f = 20$. Fig. 11(a) presents the snapshot of the initial state at t_0. Next, Fig. 11(b)–(d) illustrate the navigation trajectories at t_1, t_2 and t_3 in dynamic situation. As can be observed, SDDPG-NAV can effectively avoid dynamic obstacles and navigate UAV to the destination with a short and smooth trajectory. Then, Fig. 11(e) gives the navigation trajectory of

DDPG at t_1, where the UAV quickly collides with obstacle. This is mainly due to DDPG utilizing a single actor network that its ability to extract sufficient state information from the surrounding environment is limited. Thus, DDPG is less suitable for complex environments with dynamic obstacles. Fig. 11(f)–(h) show the navigation trajectories of TD3 at t_1, t_2 and t_4, separately. Note that $t_4 > t_3$. One may notice that the locations of obstacles at time steps t_3 and t_4 are the same. This is because t_3 and t_4 are in the same refresh interval that the environment keeps unchanged. As is shown, with TD3, UAV can also avoid dynamic obstacles, but the trajectory is rough, and the path is longer than that of SDDPG-NAV. Consequently, SDDPG-NAV is better suited for complex and dynamic environments.

To sum up, SDDPG-NAV shows better generalization capability than the comparative benchmarks. The main reasons are in threefold.

1) With novel state decomposition strategy, the two subnetworks can understand environment in a better way. Thus, the actor network can output better policy;
2) The use of smooth L_1 loss makes the network more robust to outliers and enables the update of network parameters more smoothly. This facilitates the training process;
3) The dynamic ε decay method helps UAV explore environment more efficiently in the early stage of training.

As model training progresses, the influence of explore noise gradually reduces. Then, the agent can make decisions based on what it learned from the environment gradually. Therefore, the trained model of SDDPG-NAV has good generalization capability and can make optimal decisions in complex environments.

V. Conclusion

In this work, UAV autonomous navigation is modeled as a MDP and RL technique is introduced for efficient decision-making. To improve the navigation efficiency of UAV in 3-D complex environments, an efficient state-decomposition DDPG algorithm (SDDPG-NAV) is proposed. The novel state-decomposition strategy and the objective-oriented reward functions are developed to facilitate the training progress. Numerous simulation results are presented to demonstrate the effectiveness of SDDPG-NAV. More significantly, in generalized environments, such as with denser obstacles, larger test areas, and dynamic objects, SDDPG-NAV improves the success rate by at least 10.64% when comparing with the comparative benchmark algorithms. In most works, model training is conducted in simulation environments that may not exactly reflect the real flying situation. In the future work, we will improve the sensing model by inducing range sensors in vertical plane or using depth information of the environment, and design new network structures to better sense the environment. We will also consider to extend this work to solve cooperative control of multi-UAVs, such as trajectory planning, searching, rescuing, etc. More specifically, the effect of dynamically changing number of UAVs need further investigate.

References

[1] S. Hayat, E. Yanmaz, and R. Muzaffar, "Survey on unmanned aerial vehicle networks for civil applications: A communications viewpoint," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2624–2661, 4th Quart., 2016.

[2] Z. Wei, Z. Meng, M. Lai, H. Wu, J. Han, and Z. Feng, "Anti-collision technologies for unmanned aerial vehicles: Recent advances and future trends," *IEEE Internet Things J.*, vol 9, no, 10, pp. 7619–7638, May 2022.

[3] H. T. L. Chiang, J. Hsu, M. Fiser, L. Tapia, and A. Faust, "RL-RRT: Kinodynamic motion planning via learning reachability estimators from RL policies," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 4298–4305, Oct. 2019.

[4] A. Francis et al., "Long-range indoor navigation with PRM-RL," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1115–1134, Aug. 2020.

[5] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[6] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa, "Deep neural network for autonomous UAV navigation in indoor corridor environments," *Procedia Comput. Sci.*, vol. 133, pp. 643–650, Jan. 2018.

[7] S. Rezwan and W. Choi, "Artificial intelligence approaches for UAV navigation: Recent advances and future challenges," *IEEE Access*, vol. 10, pp. 26320–26339, 2022.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Hoboken, NJ, USA: MIT Press, 2018.

[9] M. Liaq and Y. Byun, "Autonomous UAV navigation using reinforcement learning," *J. Mach. Learn. Comput.*, vol. 9, no. 6, pp. 756–761, 2019.

[10] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2017, pp. 2371–2378.

[11] 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.

[12] S. Nilwong and G. Capi, "Outdoor robot navigation system using game-based DQN and augmented reality," in *Proc. 17th Int. Conf. Ubiquitous Robot. (UR)*, 2020, pp. 74–80.

[13] E. Marchesini and A. Farinelli, "Discrete deep reinforcement learning for mapless navigation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2020, pp. 10688–10694.

[14] T. P. Lillicrap et al., "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.

[15] J. Schulman, S. Levine, P. Moritz, M. L. Jordan, and P. Abbeel, "Trust region policy optimization," 2015, *arXiv:1502.05477*.

[16] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. 35th Int. Conf. Mach. Learn. (ICML)*, 2018, pp. 1587–1596.

[17] J. Peng, B. Lv, L. Zhang, L. Lei, and X. Song, "An improved DDPG algorithm for UAV navigation in large-scale complex environments," in *Proc. IEEE Aerosp. Conf.*, 2023, pp. 1–11.

[18] H.-T. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with AutoRL," 2019, *arXiv:1809.10124*.

[19] O. Walker, F. Vanegas, F. Gonzalez, and S. Koenig, "A deep reinforcement learning framework for UAV navigation in indoor environments," in *Proc. IEEE Aerosp. Conf.*, 2019, pp. 1–14.

[20] H. Shi, L. Shi, M. Xu, and K.-S. Hwang, "End-to-end navigation strategy with deep reinforcement learning for mobile robots," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2393–2402, Apr. 2020.

[21] A. Castellini, E. Marchesini, and A. Farinelli, "Partially observable monte carlo planning with state variable constraints for mobile robot navigation," *Eng. Appl. Artif. Intell.*, vol. 104, Sep. 2021, Art. no. 104382.

[22] C. Xi and X. Liu, "Unmanned aerial vehicle trajectory planning via staged reinforcement learning," in *Proc. Int. Conf. Unmanned Aircr. Syst. (ICUAS)*, 2020, pp. 246–255.

[23] K. Wan, X. Gao, Z. Hu, and G. Wu, "Robust motion control for UAV in dynamic uncertain environments using deep reinforcement learning," *Remote Sens.*, vol. 12, no. 4, pp. 640–661, 2020.

[24] 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.

[25] C. Wang, J. Wang, J. Wang, and X. Zhang, "Deep-reinforcement-learning-based autonomous UAV navigation with sparse rewards," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6180–6190, Jul. 2020.
[26] Y. Xue and W. Chen, "A UAV navigation approach based on deep reinforcement learning in large cluttered 3D environments," *IEEE Trans. Veh. Technol.*, vol. 72, no. 3, pp. 3001–3014, Mar. 2023.
[27] O. Bouhamed, X. Wan, H. Ghazzai, and Y. Massoud, "A DDPG-based approach for energy-aware UAV navigation in obstacle-constrained environment," in *Proc. IEEE 6th World Forum Internet Things (WF-IoT)*, 2020, pp. 1–6.
[28] X. Li, J. Fang, K. Du, K. Mei, and J. Xue, "UAV obstacle avoidance by human-in-the-loop reinforcement in arbitrary 3D environment," 2023, *arXiv:2304.05959v1*.
[29] G. E. Uhlenbeck and L. S. Ornstein, "On the theory of the Brownian motion," *Phys. Rev. J. Arch.*, vol 36, no. 5, pp. 823–841, 1930.
[30] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. ICML*, Jun. 1999, pp. 278–287.
[31] G. Brockman et al., "OpenAI gym," 2016, *arXiv:1606.01540*.
[32] R. Lippmann, "An introduction to computing with neural nets," *IEEE ASSP Mag.*, vol. 4, no. 2, pp. 4–22, Apr. 1987.

Lijuan Zhang received the B.Eng. degree in information security from the University of Southwest Jiaotong University, Chengdu, China, in 2010, and the Ph.D. degree in electronic engineering from James Cook University, Cairns, QLD, Australia, in 2018.

She is currently an Associate Professor with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing, China. Her research interests include cooperative control of UAV swarms, reinforcement learning algorithms, and IoT networks.

Jiabin Peng received the B.S. degree in electronic and information engineering from Jiangsu University of Science and Technology, Zhenjiang, China, in 2021. He is currently pursuing the M.S. degree in communication and information engineering with Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His research interests include deep reinforcement learning, UAV autonomous navigation, and cooperative control of UAV swarms.

Weiguo Yi received the B.S. degree in electronic and information engineering from Civil Aviation University of China, Tianjin, China, in 2022, where he is currently pursuing the M.S. degree in electronic and information engineering with Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His research interests are in reinforcement learning and cooperative control of UAV swarms.

Hang Lin received the B.S. degree in information engineering from Huaqiao University, Quanzhou, China, in 2023. He is currently pursuing the M.S. degree in electronic and information engineering with Nanjing University of Aeronautics and Astronautics, Nanjing, China.

His research interests are in reinforcement learning and cooperative control of UAV swarms.

Lei Lei received the B.S. degree in electronic and information engineering from Northwestern Polytechnical University, Xi'an, China, in 2002, and the Ph.D. degree in communication and information system from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2008.

He is currently a Professor with the College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics. His current research interests are in wireless ad hoc networks and UAV swarms.

Xiaoqin Song received the bachelor's, master's, and Ph.D. degrees in communication engineering from Southeast University, Nanjing, China, in 1995, 1998, and 2008, respectively.

She was engaged in mobile communication network technology research with Siemens, Beijing, China, in 1999. From 2018 to 2019, she was a Visiting Scholar with the School of Engineering, University of Toronto, Toronto, ON, Canada. She is currently an Associate Professor with the College of Electronic and Information Engineering/College of Integrated Circuits, Nanjing University of Aeronautics and Astronautics, Nanjing, China. She authored six books and more than 60 papers. She holds more than 50 granted patents. Her current research interests include resource allocation, computation offloading, Internet of Vehicles, and wireless networks.