Autonomous Navigation of UAVs in Large-Scale Complex Environments: A Deep Reinforcement Learning Approach

Chao Wang, *Student Member, IEEE*, Jian Wang, *Senior Member, IEEE*, Yuan Shen, *Member, IEEE*, and Xudong Zhang, *Member, IEEE*

***Abstract*—In this paper, we propose a deep reinforcement learning (DRL)-based method that allows unmanned aerial vehicles (UAVs) to execute navigation tasks in large-scale complex environments. This technique is important for many applications such as goods delivery and remote surveillance. The problem is formulated as a partially observable Markov decision process (POMDP) and solved by a novel online DRL algorithm designed based on two strictly proved policy gradient theorems within the actor-critic framework. In contrast to conventional simultaneous localization and mapping-based or sensing and avoidance-based approaches, our method directly maps UAVs' raw sensory measurements into control signals for navigation. Experiment results demonstrate that our method can enable UAVs to autonomously perform navigation in a virtual large-scale complex environment and can be generalized to more complex, larger-scale, and three-dimensional environments. Besides, the proposed online DRL algorithm addressing POMDPs outperforms the state-of-the-art.**

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

I. Introduction

OVER the past few years we have seen an unprecedented growth of unmanned aerial vehicles (UAVs) applied to various areas such as aerial photography, rescuing and crop protection. A long term goal of UAV applications is to build intelligent systems that can implement various tasks without human intervention. In this work we seek to develop a technique that allows UAVs to autonomously navigate from arbitrary departure places to destinations in large-scale complex environments (e.g., urban areas crowded with skyscrapers). The technique is important for many applications such as goods delivery, emergency aid and remote surveillance, creating safer and smarter cities [1].

A multitude of methods, ranging from non-learning-based to learning-based, have been proposed to address UAV navigation problems. The most common non-learning-based approach is sensing and avoidance, which avoids collisions by steering vehicles in the opposite direction and performs navigation by path planning [2]–[9]. Peng [6], for instance, uses an optical flow technique to detect obstacles and a rapidly exploring random tree algorithm to implement path planning. Jason [2] combines a reactive quad-directional algorithm with a user-defined waypoint-tracking algorithm to implement navigation. Another class of non-learning based methods make use of simultaneously localization and mapping (SLAM). The intuition is that, by constructing a map of the environment using SLAM, navigation is achieved through path planning [10]–[16]. Cui [10] combines GrahpSLAM [14] with an online path planning module, creating an approach enabling UAVs to find obstacle-free trajectories in foliage environments. Apart from the two classes of methods, navigation can also be done by retracing pre-planned trajectories [17]–[22]. Methods in this category generally fall into precisely localizing UAVs using various position techniques such as Kalman filtering and its variations [17]. Navigation is achieved by minimizing the error between the estimated position and the pre-planned position. A common feature of non-learning-based methods is that they require explicit path planning, which may lead to unexpected failure when environments are highly dynamic and complex. In this respect, some works resort to machine learning approaches such as imitation learning and reinforcement learning (RL) [23]–[25]. Nursultan [25], for example, develops a model-based RL algorithm named TEXPLORE as a high level control method for UAV navigation in a grid map with no barriers. Stephane [23] builds an imitation learning-based controller using a small set of human demonstrations and achieves good performance in natural forest environments.

New challenges arise when UAVs perform navigation tasks in large-scale complex environments: 1) The environment is large-scale. In this case, SLAM-based methods will lose efficiency since it is infeasible to build a map of the environment; 2) The environment is complex (crowded with dense obstacles). As sensing and avoidance-based approaches are generally designed to address navigation problems in environments with sparse obstacles, they may lose efficiency when applied in complex environments; 3) The environment is often dynamic. It is evident that pre-path planning is incapable of handling such a situation. RL adapts to dynamic environments but needs improvement in order to cope with large-scale complex environments; 4) Sensors have limited sensing capacity. UAV navigation fully depends on

Manuscript received May 13, 2018; revised September 22, 2018 and November 24, 2018; accepted December 28, 2018. Date of publication January 3, 2019; date of current version March 14, 2019. This work was supported by the National Key Research and Development Program of China under Grant 2016YFB0100902. The review of this paper was coordinated by Dr. A. Chatterjee. *(Corresponding author: Jian Wang.)*

The authors are with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: w-c15@mails.tsinghua.edu.cn; jian-wang@tsinghua.edu.cn; shenyuan_ee@tsinghua.edu.cn; zhangxd@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TVT.2018.2890773

observations returned by sensors. If sensors have limited sensing capacity, UAVs might be confused about the situations they are encountering.

To overcome these challenges, UAVs need to resort to the entire history observations to help them distinguish the encountered situation and decide the action to take. In this respect, the decision making process fits in the framework of partially observable Markov decision process (POMDP), which characterizes a process that an agent at some hidden state obtains an observation of the state, takes an action, transits to another hidden state and obtains some reward. Therefore, in this paper, we model the navigation problem as a POMDP and design a deep reinforcement learning (DRL)-based algorithm to settle it [26].

The POMDP structuring the UAV navigation problem must meet with several requirements. Firstly, it should be model-free since it is infeasible to know a priori its state transition probabilities and observation probabilities. Secondly, its action space should be continuous since UAVs' control signals are continuous. Lastly, the observation space and reward function should be informative enough in order to achieve certain goals. Correspondingly, we design an observation space that synthesizes multiple sensory outputs characterizing UAVs' internal states, relationship with the environment and the destination. Besides, a domain knowledge-inspired reward function is designed to award or penalize certain actions taken at certain states.

RL solves model-free POMDPs by maintaining memories of past observations and actions [27]–[31], from which stochastic or deterministic control policies are derived (stochastic policies map memories of past observations and actions into distributions over the action space while deterministic policies map memories into actions). Of these methods, policy gradient methods such as likelihood-ratio methods [29], [30] and actor-critic methods [31], [32] derive parameterized stochastic policies for model-free POMDPs with continuous action spaces by performing gradient descent in the parameter space. These algorithms suffer from a major drawback that they cannot be implemented in an end-to-end fashion. Besides, RL is known to be unstable when nonlinear function approximators are used to approximate value functions [33]. In this respect, DRL, a combination of deep learning and RL that directly projects raw sensory outputs into control signals, is developed to provide an end-to-end solution concept to Markov decision processes (MDPs) [34] and POMDPs. Mnih [33] firstly utilizes DRL to play electronic games with discrete control profiles. Afterwards, Lillicrap [35] applies its insight to problems with continuous action spaces and designs a deep deterministic policy gradient algorithm (DDPG) within the actor-critic framework [36]. Heess [37] takes partial observability of states of MDPs into consideration and designs a recurrent deterministic policy gradient algorithm (RDPG). He further shows that there is little performance difference between stochastic and deterministic policies controlling POMDPs.

Because DRL obtains optimal control policies through trial-and-error learning that might lead to irreversible damages to agents, it shall be capable of learning from small samples in the real world. RDPG is not sample-efficient since it will not update policy parameters until an episode's end (in each episode the agent interacts with the environment by executing actions determined by the learned policy, and an episode ends if the agent reaches any terminal states). Obviously, updating the parameters of the policy step-by-step can make immediate use of the newest experience, facilitating the agent to better explore the unknown environment and to learn faster. Besides, unlike DDPG that conducts parameter optimization based on state transition tuples (state, action, reward and the next state), RDPG updates parameters based on entire episodes as we cannot break an episode into state transition tuples given that POMDPs do not hold the Markov property. We will show that such an optimization procedure is inefficient when stochastic gradient descend (SGD) is applied to conducting parameter optimization. In this regard, we propose a more efficient online DRL algorithm named Fast-RDPG to solve POMDPs.

The main contributions of this paper are as follows:

1) We develop a DRL framework for UAV navigation in large-scale complex environments, where the navigation problem is modeled as a POMDP, which is solved by a novel online DRL algorithm.

2) We prove that in the POMDP setting, policy gradients within the actor-critic framework are determined by history observations and actions instead of entire episodes, and the policy gradient of a deterministic policy is just a special case of that of a stochastic policy without preconditions.

3) We design an efficient online DRL algorithm approaching POMDPs with continuous action spaces based on the two theorems and build a stochastic large-scale complex environment to validate the effectiveness of the proposed method.

The remainder of this manuscript is structured as follows. Section II introduces some background knowledge of MDPs, POMDPs and two DRL algorithms solving POMDPs with continuous control spaces. Section III formulates the navigation problem. Section IV elaborates on the POMDP modeling process of the navigation problem. The proposed algorithm is demonstrated in Section V. Simulation results and discussions are presented in Section VI. Section VII concludes this paper and envisages some future work.

II. Background

In this section we first give a brief introduction to MDPs and POMDPs, followed by two DRL algorithms solving POMDPs with continuous action spaces.

A. MDPs and POMDPs

POMDPs, as extensions of MDPs to partially observable states, allows for decision making under uncertain conditions. A MDP is composed of a state space $\mathcal{S}$, an initial state space $\mathcal{S}_0$ having an initial state distribution $p(s_0)$, an action space $\mathcal{A}$, a state transition probability distribution $p(s_{t+1}|s_t, a_t)$ satisfying the Markov property, and a reward function $r(s_t, a_t)$: $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$ characterizing the environment feedbacks when executing action a_t at state s_t. The state space and action space of a MDP could be either discrete or continuous. Since UAVs' control signals are continuous, in this paper we focus on MDPs and POMDPs with continuous state and action spaces. For ease of denotation, we leave it implicit in the following context that all summation operations are actually integral operations.

MDPs is generally addressed via RL, which learns an optimal policy π through trial-and-error learning. The policy could be either deterministic or stochastic, where a deterministic policy, denoted as $\mu(s)$: $\mathcal{S} \rightarrow \mathcal{A}$, projects states to actions, and a stochastic policy, denoted as $a \sim \pi(\cdot|s)$: $\mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$, where $\mathcal{P}(\mathcal{A})$ is the set of probability measures on $\mathcal{A}$, returns the probability density of available state and action pairs (s, a). If the action space is continuous, the stochastic (or deterministic) policy is parameterized as a function $a \sim \pi(\cdot|s,\theta)$ (or $a = \mu(s,\theta)$), where θ refers to the parameters of the function. To keep the notation simple, we leave it implicit in all cases that policy π (or μ) is a function of θ and all the gradients are with respect to θ.

RL involves in estimating value functions of states and action-value functions of state-action pairs. For stochastic policies, value function is defined as

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

where γ is a discount factor ranging from 0 to 1, and action-value function

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

A stochastic policy is optimal if an agent receives the maximum expected future discounted reward (which is also referred as target function) by executing it:

$$\eta(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]. \tag{3}$$

For a deterministic policy, the value functions, action-value functions and target function can be obtained by replacing $a_t \sim \pi(\cdot|s_t)$ by $a_t = \mu(s_t)$ in (1), (2) and (3).

A MDP becomes a POMDP when an agent could not observe the state s_t but instead receives an observation o_t having a distribution $p(o_t|s_t)$. It is immediate that the observation sequence no longer satisfies the Markov property: $p(o_{t+1}|a_t, o_t, a_{t-1}, o_{t-1}, \ldots, o_0) \neq p(o_{t+1}|o_t, a_t)$. As a consequence, an agent needs to access to the entire history trajectory $h_t = (o_t, a_{t-1}, o_{t-1}, \ldots, o_0)$ to infer the current state s_t and makes decisions based on it.

The goal of RL in partially observable settings is thus to learn an optimal policy $a_t \sim \pi(\cdot|h_t)$ that projects history trajectories to action distributions by maximizing (3).

B. Actor-Critic, DDPG, and RDPG

Actor-critic [36], as a category of policy gradient methods addressing MDPs with continuous control spaces, seeks to learn an optimal policy $a_t \sim \pi(\cdot|s_t,\theta)$ (or $a_t = \mu(s,\theta)$) by implementing policy gradient in the parameter space:

$$\frac{\partial V_\pi(s_0)}{\partial \theta} = \mathbb{E}_{s\sim d_{s_0,\pi}(s)}\mathbb{E}_{a\sim\pi(a|s))}\left[\nabla \log \pi(a|s) Q_\pi(s,a)\right], \tag{4}$$

where $d_{s_0,\pi}(s)$ is the un-normalized state distribution in a randomly generated episode starting from the initial state s_0 and then following π.

In the actor-critic framework, policy $\pi(a|s)$ and action-value function $Q_\pi(s,a)$ are called actor and critic, respectively, and the critic is parameterized by a function $Q_\omega(s,a)$ and estimated by Time-difference (TD) method:

$$\begin{aligned}\omega_{t+1} = \omega_t + \epsilon(r_{t+1} + \gamma Q_{\omega_t}(s_{s+1}, a_{t+1}) \\ - Q_{\omega_t}(s_t, a_t))\nabla Q_{\omega_t}(s_t, a_t),\end{aligned} \tag{5}$$

where $r_{t+1} + \gamma Q_\omega(s_{t+1}, a_{t+1})$ is referred as target values and ϵ learning rate.

As an actor-critic learning method, DDPG is derived from the *deterministic policy gradient theorem for MDPs* [38], which says that for MDPs with continuous action spaces, the deterministic policy gradient exists and can be computed by

$$\nabla\eta(\mu) = \mathbb{E}_{s\sim d_\mu(s)}\left[\nabla_\theta \mu(s) \nabla_a\ Q_\mu(s,a)|_{a=\mu(s)}\right], \tag{6}$$

where $d_\mu(s)$ denotes the un-normalized state distribution in a randomly generated episode.

RDPG extends the framework of DDPG to POMDPs by updating policy parameters using the following policy gradient:

$$\nabla\eta(\mu) = \mathbb{E}\left[\sum_t \gamma^{t-1} \nabla_\theta \mu(h_t) \nabla_a\ Q_\mu(h_t,a)|_{a=\mu(h_t)}\right], \tag{7}$$

During learning, a RDPG agent interacts with the environment using its currently learned policy, and at the end of an episode, the agent caches the entire episode into a replay memory (which is designed to randomize over observations so as to stabilize the learning process [33]) and samples a batch of episodes to conduct parameter optimization.

While RDPG shows its effectiveness in many simulated continuous control tasks with partially observable states, it has two limitations. 1) RDPG is an offline learning algorithm. It only executes parameter optimization at the end of an episode, contributing to insufficient utilization of the experience and exploration of the environment. 2) RDPG converges slowly. As will be demonstrated in Section V, the parameters should be updated based on history trajectories instead of entire episodes. Observing that the stochastic gradient in (7) is actually the sum of a sequence of history trajectory-related gradients in the same episode, RDPG should converge slowly when using SGD optimization method. In this respect, we propose a new approach to address POMDPs.

III. Problem Formulation

In this section we formulate the UAV navigation problem in large-scale complex environments.

A. Coordinates Frames of UAVs

Typically, the control profile of UAVs has three degrees of freedom, including throttle, heave and steering, which refer to speed change, vertical displacements and rotations around transverse axes, respectively. Correspondingly, there are altogether

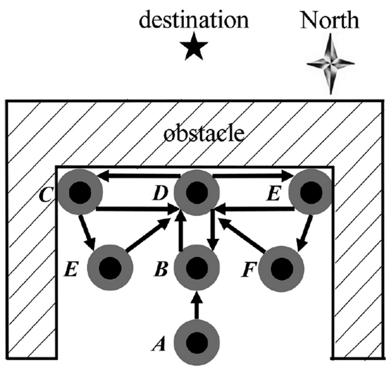


Fig. 1. Illustration of partial observability of UAV navigation in large-scale complex environments. Solid black circles and gray circles represent a UAV and its perceptron range. Arrows denote possible trajectories generated by the UAV.

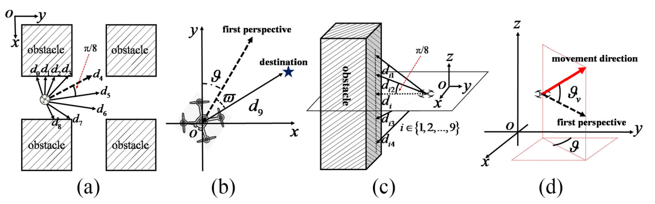


Fig. 2. (a) and (b). Observations of UAVs in two-dimensional environments. $d_0 \sim d_8$ denote distances returned by the nine virtual range finders; ϖ and d_9 represent the angle and the distance between UAVs' current position and the destination; ϑ denotes the angle between UAVs' first-perspective direction and the north direction. (c) and (d). Deployment of additional sensors in three-dimensional environments. $d_{i1} \sim d_{i4}$ denote distances returned by range finders outside the horizontal plane given in the i-th horizontal detection direction in (a); ϑ_z denotes the angle between the UAV's movement direction and first-perspective direction.

six independent coordinates determining the position, orientation and motion of UAVs. An earth-fixed coordinate frame can be used to describe UAVs' absolute position and orientation, denoted as $\varphi = [x, y, z, \vartheta, \vartheta_y, \vartheta_z]$. Besides, a body-fixed coordinate frame can be used to describe UAVs' linear and angular velocities, denoted as $\mu = [a, b, c, \vartheta_a, \vartheta_b, \vartheta_c]$. The earth-fixed coordinate frame together with the body-fixed coordinate frame characterize UAVs' position, orientation and motion, which can be referred as UAVs' internal state descriptions.

B. UAV Navigation

For simplicity, below we assume UAVs fly at a fixed speed and a fixed height unless stated. Furthermore, we omit UAVs' momentum while flying and suppose that the steering action takes effect in no time. As a consequence, activities of UAVs are restricted to the $x - y$ plane, the vector describing UAVs' position, orientation and motion are simplified to $\zeta = [x, y, \vartheta]$, and the dynamic of UAVs is formulated as:

$$\begin{aligned} \vartheta_{t+1} &= \vartheta_t + a_t \\ x_{t+1} &= x_t + v \times \cos(\vartheta_{t+1}) \\ y_{t+1} &= y_t + v \times \sin(\vartheta_{t+1}) \end{aligned} \tag{8}$$

where a_t is the steering signal and v the constant speed.

The goal is to control a UAV to fly from arbitrary departure places ϕ_{dep} to destinations ϕ_{des} in the earth-fixed coordinate frame (the large-scale complex environment) without being trapped or colliding with obstacles.

IV. POMDP Modeling

In this section we first explain by an example the reason why states in the navigation problem are partially observable. Then we present the details of the POMDP modeling procedure.

A. UAV Navigation as a POMDP

As depicted in Fig. 1, supposing a UAV locates at point A, to approach the target position, it shall head north for point D. At this point, the UAV finds out that its path to the target is blocked by the obstacle and shall turn back, turn left or turn right. By turning back, it reaches point B. With limited sensing capacity, it immediately finds out that the surrounding is obstacle-free due to its limited sensing capacity. As a consequence, it turns back again and eventually strays between A and B. Similarly, by turning left or right, the UAV also ends up cycling in the trap. In contrast, if the UAV could remember what it has experienced, it shall gradually construct the structure of the local environment based on its experience and accordingly makes better decisions. While MDPs assume that states are fully observable, they would fail to capture the complex structures of the environment. By comparison, POMDPs are able to obtain more information from history trajectories, and therefore shall work more efficiently.

B. Observation Space and Action Space Specification

For the purpose of navigation, UAVs shall at least be capable of receiving information from three sources, i.e., information indicating their internal states, their relationship with the environment and their relationship with the destination. Firstly, since the environment is often dynamic, we abandon UAVs' absolute position $[x, y]$ and only use $[\vartheta]$, the first-perspective orientation, to describe their internal state, as illustrated in Fig. 2(b). In practice, $[\vartheta]$ can be measured by gyroscope-like devices. Secondly, UAVs' relationship with the surrounding can be characterized by images returned by cameras, radar signals returned by radars or distances returned by range finders. In this work we use nine range finders to characterize the relation, denoted as $\psi = [d_0, \ldots, d_8]$ and depicted in Fig. 2(a). Lastly, a vector $\xi = [d_9, \varpi]$ that represents the distance and angle between UAVs' current position and the target is used to describe their connection with the destination, as depicted in Fig. 2(b). In practice, ξ can be measured by GPS-like devices. Composing the three kinds of information brings us to the final description of the observation space: $o = [\vartheta, d_0, \ldots, d_8, d_9, \varpi]$, where $\vartheta, \varpi \in [-\pi, \pi]$, $d_0 \sim d_8 \in [0, 100]$ and $d_9 \in [0, +\infty]$.

Given that UAVs' flight altitude and speed have been prefixed as constants, we use $a = [\rho]$, ranging from $-\pi$ to π, to denote the steering signal.

C. Reward Design: Incorporating Domain Knowledge

Reward acts as a signal evaluating *how good* it is when taking an action at a state. For UAV navigation in large-scale complex environments, a straightforward way is sparse reward, which means that UAVs would be rewarded if and only if they arrive at the target position. However, as the initial policy is randomly generated, UAVs will strike the destination with probability zero in large-scale environments full of barriers. Accordingly, learning algorithms would take extremely long time to converge. An alternative approach is non-sparse reward. Reward shaping [39] provides learning agents with a specific form of non-sparse reward with the guarantee of policy invariance. Nonetheless, since it only makes sense under the condition that the shaped reward is the difference of a state-dependent potential function, it is hard to implement in practice and tends to block the way of incorporating domain knowledge [40].

In this paper, we design a non-sparse reward that incorporates our domain knowledge about the navigation problem and meanwhile preserving a relatively satisfactory policy. The non-sparse reward consists of four parts, namely transition reward, obstacle penalty, free-space reward and step penalty. The transition reward is designed as:

$$r_{trans} = \sigma d_{dist}, \tag{9}$$

where σ is a positive constant and d_{dist} the reduced distance between UAVs' current position and the destination after one-step transition. The transition reward incorporates the knowledge that we encourage UAVs to head for destinations and penalize them if they distance targets. Besides, in practice, colliding with obstacles could be catastrophic for UAVs. To prevent them from getting too close to any obstacles, we design the obstacle penalty as

$$r_{bar} = -\alpha e^{-\beta d_{min}}, \tag{10}$$

where α and β are two positive constants and d_{min} is the minimum distance between UAVs and obstacles. To further encourage them to avoid barriers, they would obtain a constant free-space reward r_{free} if their first-perspective direction points to obstacle-free places. Lastly, to ensure UAVs arrive at the destination as soon as possible, after each transition they would get a constant penalty r_{step}. To summarize, the final non-sparse reward can be formulated as

$$r_{final} = r_{trans} + r_{bar} + r_{free} + r_{step}. \tag{11}$$

Different instances of (11) may result in different policies. To obtain policies as we expect, the parameters must be fine-tuned. We will make a deep survey of this relationship in Section VI.

V. A New Approach to Addressing POMDPs

In this section we derive a novel online DRL algorithm to solve POMDPs. We begin the derivation by stating some basic definitions. Define the value function of the underlying state s_t after observing h_t as

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

the action-value function of the underlying state-action pair (s_t, a_t) after observing h_t as

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

and the expected reward of taking action a_t at state s_t after observing h_t as

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

Take expectation of (12), (13) and (14) with respect to the state distribution $s_t \sim p(s_t|h_t)$, we obtain the value function of history trajectory h_t as

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

the action-value function of history trajectory and action pair (h_t, a_t) as

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

and the reward of history trajectory and action pair (h_t, a_t) as

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

Based on these definitions, we re-formulate the performance objective in (3) as

$$\eta(\pi) = \mathbb{E}_{h_0}\left[V_\pi(h_0)\right], \tag{18}$$

where the expectation is over the initial history trajectory h_0: $h_0 \sim p(h_0)$, and h_0 equals to o_0, the initial observation.

Taking the gradient of the performance objective with respect to parameter θ brings us to the *stochastic policy gradient theorem for POMDPs*.

Theorem 1: Consider learning a stochastic policy for a POMDP with a continuous action space. The gradient of the stochastic policy exists and can be calculated as

Theorem I: (Stochastic Policy Gradient Theorem for POMDPs).

Consider learning a stochastic policy for a POMDP with a continuous action space. The gradient of the stochastic policy exists and can be calculated as

$$\nabla\eta(\pi) = \mathbb{E}_{h\sim d_\pi(h)}\mathbb{E}_{a\sim\pi(a|h)}\left[\nabla_\theta \log(\pi(a|h))Q_\pi(h, a)\right], \tag{19}$$

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

Remark 1: The stochastic gradient in (19) is related to history trajectories instead of entire episodes. As a result, every time an agent interacts with the environment, it can perform online parameter optimization without waiting until the end of an episode, facilitating the agent to better explore the unknown environment and to learn faster.

The analogous structures of (19) and (4) reflects that POMDPs can be converted to MDPs by informally regarding history trajectories in POMDPs as fully observable states. Silver [38] proves that the deterministic policy gradient of MDPs exists and is a special case of the stochastic policy gradient if the stochastic policy satisfies certain conditions. Nonetheless, we argue

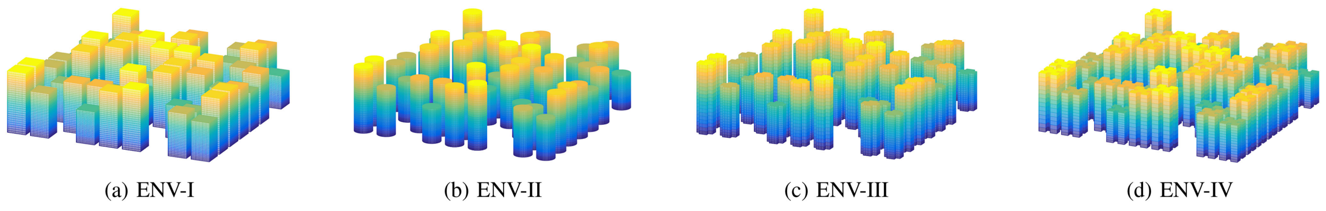


Fig. 3. Schematic view of the simulated stochastic large-scale complex environment. Obstacles (or buildings) in different types of environments are distinguished from shapes, sizes and mutual space. The randomness of the environment manifests in two aspects: 1) The four types of environments are uniformly randomly selected by the learning agent every time before it prepares to interact with an environment; 2) Once an environment is chosen by the agent, buildings in it are immediately randomly regenerated.

that Theorem I can be extended to deterministic policies without making any assumptions. The extension is accomplished by reformulating deterministic policies as Dirac delta functions.

Theorem II: (Deterministic Policy Gradient Theorem for POMDPs).

Consider learning a deterministic policy for a POMDP with a continuous action space. The gradient of the deterministic policy $a = \mu(h)$ exists and can be calculated as

$$\nabla\eta(\mu) = \mathbb{E}_{h\sim d_\mu(h)}\left[\nabla_\theta \mu(h)\nabla_a\, Q_\mu(h,a)|_{a=\mu(h)}\right], \quad (20)$$

where $d_\mu(h)$ denotes the un-normalized history trajectory distribution induced by the deterministic policy. (Proof see Appendix B)

Remark 2: The stochastic gradient in (7) equals to the discounted sum of that in (20) in terms of an entire episode, i.e., parameters of the actor and the critic in RDPG are actually optimized based on a sequence of highly correlated history trajectories. While in the actor-critic framework, both the actor and the critic are approximated by non-linear functions and parameters are updated using SGD, policy gradient algorithms addressing model-free POMDPs based on (20) might be potentially more stable and converge faster than those based on (7) as SGD requires i.i.d. input data sequence. We will validate the conclusion by experiments in Section VI.

Since both the gradients in (19) and (20) involve in taking expectations with respect to the unknown history trajectory distribution $d_\pi(h)$, we must estimate them by sampling the state and action spaces. In this regard, RL algorithms using stochastic policies generally converge much slowly than those using deterministic policies, for that deterministic policies only require samples from the state space while stochastic policies need samples from both the state space and the action space. For faster convergence, we design a DRL algorithm named Fast-RDPG based on Theorem II.

The proposed algorithm is demonstrated in Table I, where the use of two target actor and critic networks and a replay memory aim to stabilize the learning process, and the addition of the stochastic process (exploration noise) to the currently learned policy is to encourage agents keep exploring the state and action spaces [37]. Compared to RDPG, Fast-RDPG performs online learning strategy and updates parameters in terms of history trajectories instead of entire episodes.

TABLE I

```
**Algorithm** Fast-RDPG
Initialize critic $Q_\omega(a,h)$ and actor $\mu_\theta(h)$ with parameters $\omega$ and $\theta$.
Initialize target critic $Q_{\omega'}(a,h)$ and target actor $\mu_{\theta'}(h)$ with weights $\omega' \leftarrow \omega, \theta' \leftarrow \theta$.
Initialize replay buffer $R$.
for $episodes = 1, M$, do
    Initialize a stochastic process $N$ for action exploration
    Receive an initial observation $o_0$ (or history trajectory $h_0$)
    for $t = 1, T$ do
        Obtain action $a_t = \mu_\theta(h_t) + N_t$
        Execute action $a_t$, obtain reward $r_t$ and observation $o_t$
        Store transition $(h_{t-1}, a_t, o_t, r_t)$ into $R$
        Update history trajectory $h_t = [h_{t-1}, a_t, o_t]$
        Sample a minibatch of $L$ transitions $\{(h_i, a_i, o_i, r_i)\}_{i=1}^{L}$
        Set $y_i = r_i + \gamma Q_{\omega'}([h_i, a_i, o_i], \mu_{\theta'}([h_i, a_i, o_i]))$
        Compute critic update
            $\Delta\omega = \frac{1}{L}\sum_i (y_i - Q_\omega(h_i, a_i))\frac{\partial Q_\omega(h_i,a_i)}{\partial\omega}$
        Compute actor update
            $\Delta\theta = \frac{1}{L}\sum_i \frac{\partial Q_\omega(h_i,\mu_\theta(h_i))}{\partial a}\frac{\partial\mu_\theta(h_i)}{\partial\theta}$
        Update actor and critic using Adam
        Update the two target networks ($\varepsilon$ is the soft target update rate)
            $\omega' \leftarrow \varepsilon\omega + (1-\varepsilon)\omega'$
            $\theta' \leftarrow \varepsilon\omega + (1-\varepsilon)\theta'$
    end for
end for
```

VI. Simulation Results

In this section we present the experimental settings and simulation results, along with some discussions.

A. Experimental Settings

We construct four types of large-scale complex environments as a stochastic environment, depicted in Fig. 3. Each environment covers more than one and a half square kilometers and is crowded with buildings with random height. During implementation, the agent uniformly randomly chooses one environment and rollouts an episode by following its currently learned policy corrupted by the exploration noise. Every time before the agent starts interacting with the environment, buildings with height having a uniform distribution $U(30.0\text{ m}, 200.0\text{ m})$ are regenerated.

Analogous to RDPG, the actor and critic in (20) are approximated by two LSTMs summarizing information encoded in

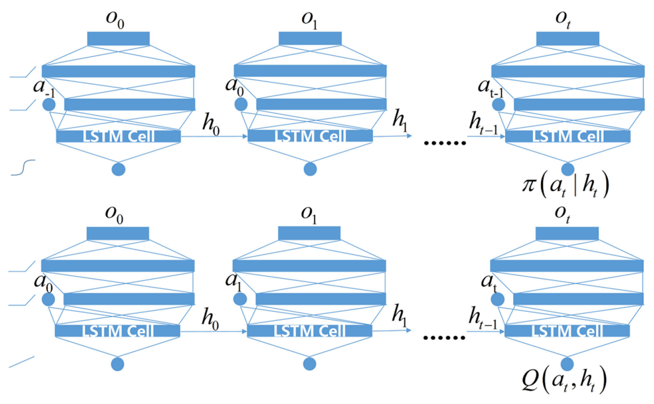


Fig. 4. Network structures of the actor and the critic. The two networks take exactly the same forms, where the size of the first three layers are 11, 400, and 300, respectively, the LSTM cell is 300, and the output layer is 1. In contrast to the critic network, the input action sequence of the actor network begins with a_{-1}, which is designed to keep a uniform structure of the network in folded form and is pre-fixed to zero.

history trajectories. Their network structures are demonstrated in Fig. 4. The observations of the UAV are normalized to $[0,1]$ and the steering signal is normalized to $[-1,1]$. The UAV's speed and flight altitude are set to 2.0 m/s and 90.0 m, respectively. The reward is instantiated as: $\sigma = 2.0$, $\alpha = 8.0$, $\beta = 25.0$, $r_{free} = 0.1$ and $r_{step} = -0.6$. The maximum episode length (time step) is set to 25. Besides, Adam optimizer[41] is employed to learn the network parameters with a learning rate of 10^{-4} and 10^{-3} for the actor and critic, respectively. The parameters of the critic network are regularized with weight decay of 10^{-2}, the discounted factor is $\gamma = 0.99$ and the soft target update rate is $\varepsilon = 0.001$. In addition, an exploration noise with a uniform distribution $U(-0.25, 0.25)$ is used to explore the state and action spaces.

B. Navigation Behavior of Fast-RDPG

To validate the effectiveness of our proposed method, a well-trained Fast-RDPG agent is dispatched to execute multiple navigation tasks in the simulated environment. As depicted in Fig. 5, by projecting history trajectories to control signals, the agent successfully flies from departure places to destinations. The humanlike navigation behavior suggests that DRL is an efficient approach to addressing the UAV navigation problem in large-scale complex environments.

To verify that the memorable Fast-RDPG agent is more efficient in performing navigation tasks than the memoryless DDPG agent, we select two pairs of starting positions and target positions in ENV-IV, the most difficult environment, and let the two agents execute the navigation tasks. As depicted in Fig. 6, both of them have learned to navigate in the complex environment. Nevertheless, since our Fast-RDPG agent makes decisions based on its history observation and action trajectories, it perceives more information about the local structure of the environment and accordingly is capable of escaping from traps. On the contrary, the DDPG agent cannot remember what it has experienced and makes decisions only based on its current perception of the environment. As a consequence, it cannot percept the local structures of the environment and tends to be caught in traps.

The conclusion can also be verified by the convergence curves of DDPG and Fast-RDPG in Fig. 7, which are obtained by evaluating the target function (3) using Monte Carlo methods. Specifically, 200 independent navigation tasks are performed every five hundred training episodes using the learned policy and the target function is approximated by averaging the discounted cumulated reward of the 200 navigation episodes. As can be seen, while both the two algorithms manifest similar convergence speed, the estimated normalized return of DDPG is much lower than that of Fast-RDPG when converge.

We also make a quantitative survey over the success rate, stray rate and crash rate of navigation missions of the DDPG agent as well as our Fast-RDPG agent in each of the four environments. For each agent, the success rate, stray rate and crash rate is obtained by calculating the percentage of successful navigation missions, the percentage of failed missions ending up being trapped into local environment, and the percentage of failed missions due to crashing with obstacles over 300 complete navigation missions in each environment, respectively. The results are illustrated in Table II. As we can see, the success rate of the DDPG agent in each of the four environments is lower than 54% and is only 42.67% by average. By contrast, the Fast-RDPG agent finishes navigation missions with a success rate of more than 94% in each environment and 97.42% by average, which is a significant performance improvement. Additionally, while both the two agents exhibit low crash rate, the stray rate of the DDPG agent reaches up to 69% in ENV IV, the environment with the largest amount of traps, and is much higher than that of the Fast-RDPG agent, validating that the DDPG agent cannot remember what it has experienced and is prone to be trapped and our Fast-RDPG agent makes decisions based on its history experience and has the ability to escape from traps. The significant performance improvement of Fast-RDPG over DDPG comes at the cost of increased computational complexity. DDPG uses two feedforward neural networks to approximate the actor and the critic while Fast-RDPG uses two recurrent neural networks (LSTM in our experiment). The computational complexity of optimizing a recurrent neural network is higher compared with that of optimizing a feedforward neural network.

C. Performance of Fast-RDPG

To validate that Fast-RDPG is more efficient than RDPG, we re-implement RDPG with hyper-parameters equal to those of Fast-RDPG. As illustrated in Fig. 7, Fast-RDPG only uses around $2,500$ episodes to converge while RDPG uses around $8,000$ episodes to reach its convergence point. Since in practice, interaction with real environments could be catastrophic or time-consuming, compared with RDPG, Fast-RDPG requires less times of interaction with the environment and therefore is more efficient in small-sample learning. We believe the performance improvement comes from two aspects: 1) Fast-RDPG is an online learning algorithm. Compared with RDPG, which performs parameter optimization at the end of an entire episode, Fast-RDPG updates parameters every time the agent interacts

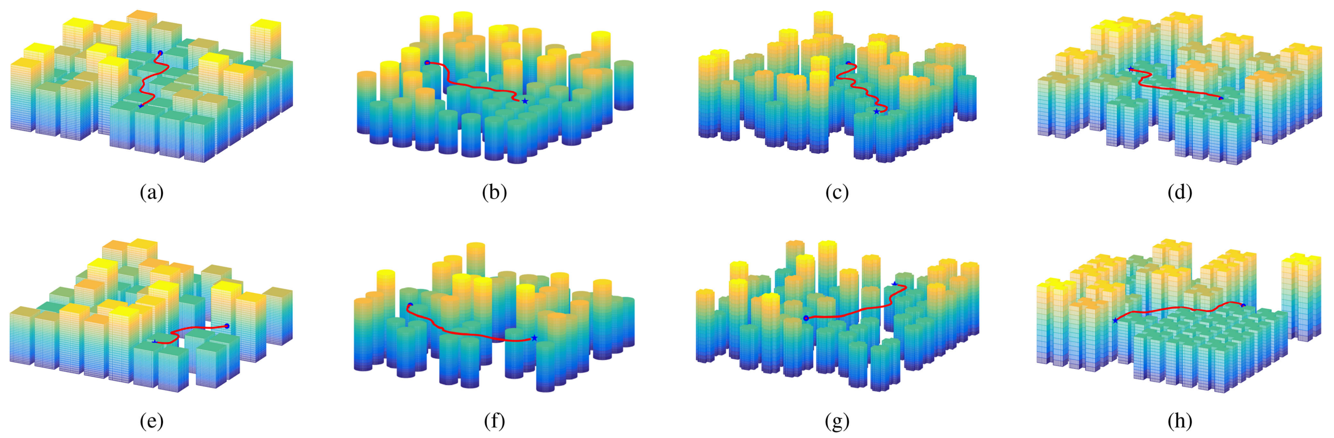


Fig. 5. Navigation trajectories of the Fast-RDPG agent in the simulated environment. The solid circles and stars denote departure places and destinations, respectively. The curves connecting them represent navigation trajectories. For clarity, any buildings lower than the altitude the UAV flies at are not drawn in the images. Besides, for buildings obstructing our view of the trajectories, we clip them to lower ones with height equal to the UAV's flight altitude.

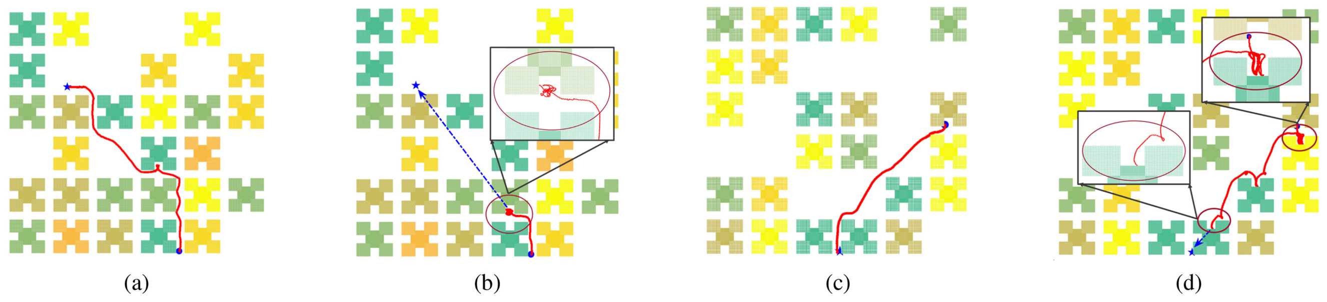


Fig. 6. Schematic view of navigation trajectories of the Fast-RDPG agent and the DDPG agent in ENV-IV. For the sake of clarity, only the top views of the environment are drawn and buildings lower than the UAV's flight altitude are omitted. Subfigures (a) and (c) depict the navigation trajectories of the Fast-RDPG agent, and subfigures (b) and (d) the corresponding trajectories of the DDPG agent.

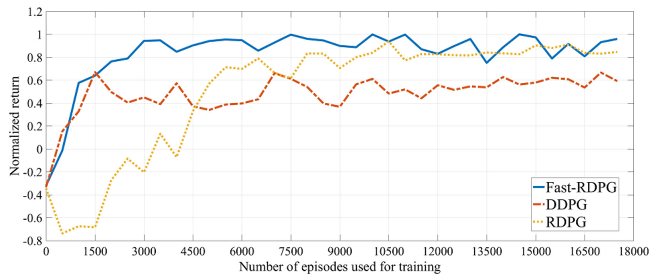


Fig. 7. Convergence curves of DDPG, RDPG, and Fast-RDPG. The x-axis represents the number of episodes used for training, and the y-axis the normalized return. At each training stage, the un-normalized return, formulated by (3), is approximated by averaging the return of 200 navigation trajectories generated by the learned policy.

with the environment, making immediate use of the newest experience. 2) Fast-RDPG is more efficient in data utilization. The high correlation among the history trajectories in the same episode reduces efficiency of RDPG in data utilization. In contrast, Fast-RDPG uses mutually independent history trajectories sampled from the replay memory to perform parameter optimization and therefore works more efficiently.

Although both Fast-RDPG and RDPG achieve similar normalized return when converge, their learned policies deviate from each other a lot. As we can see in Table II, the success rate of the RDPG agent in each of the four environments is lower than 87% and is only 76.25% by average, around 21% lower than that of our Fast-RDPG agent. Besides, the stray rate of the RDPG agent is much higher than that of the Fast-RDPG agent. Consequently, though both the two algorithms are designed to settle POMDPs, our Fast-RDPG is more efficient.

The superiority of our Fast-RDPG over RDPG can also be demonstrated by its strong generalization ability. As the height of the buildings in the stochastic environment has a uniform distribution on the interval $[30\text{ m}, 200\text{ m}]$, the lower the height, the denser the obstacles are. To test the generalization ability of the two algorithms, the well-trained Fast-RDPG agent and the RDPG agent are dispatched to execute navigation missions at different flight altitudes in the complex environment. Besides, we also fix the flight altitude to 90 m and let the two agents execute navigation tasks in even larger-scale complex environments. The results are illustrated in Fig. 8 and Fig. 9.As we see in Fig. 8(a), (b) and (c), with the decrease of flight altitude, the success rate of the Fast-RDPG agent does not degrade at all whereas that of the RDPG agent manifests a relatively significant decrease. Besides, in contrast to our Fast-RDPG agent, the stray rate and crash rate of the DDPG agent tend to increase as

TABLE II
STATISTICAL QUANTITIES OF DDPG, RDPG, AND FAST-RDPG IN DIFFERENT ENVIRONMENTS

Results / Environment	Success rate			Crash rate			Stray rate		
	DDPG	RDPG	Fast-RDPG	DDPG	RDPG	Fast-RDPG	DDPG	RDPG	Fast-RDPG
ENV-I	53.33%	78.33%	**97.67%**	**0.33%**	1.00%	0.67%	46.33%	20.67%	**1.67%**
ENV-II	37.67%	82.67%	**94.33%**	**0.33%**	**0.33%**	5.67%	62.00%	17.00%	**0.00%**
ENV-III	49.67%	86.33%	**98.33%**	0.67%	**0.00%**	0.33%	49.67%	13.67%	**1.33%**
ENV-IV	30.00%	57.67%	**99.33%**	1.00%	23.33%	**0.33%**	69.00%	19.00%	**0.33%**
Average	42.67%	76.25%	**97.42%**	**0.58%**	6.17%	1.75%	55.75%	17.59%	**0.83%**

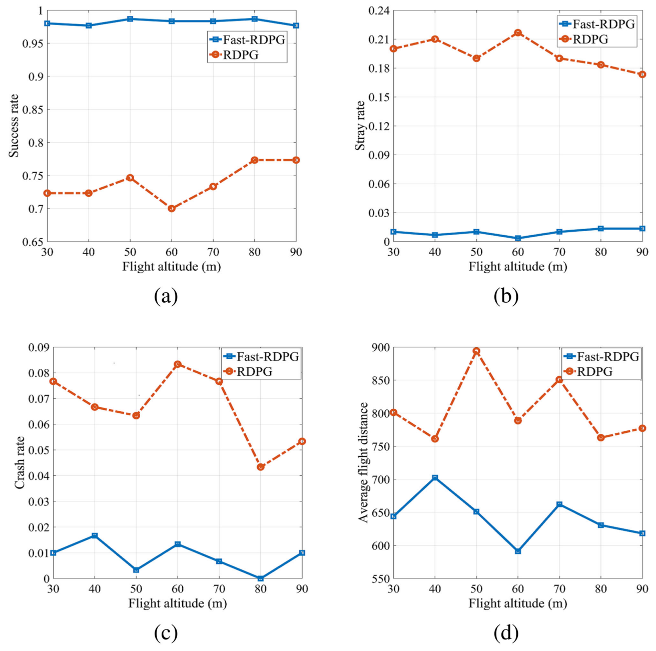


Fig. 8. Success rate, stray rate, crash rate, and average flight distance of navigation missions of the Fast-RDPG agent and the RDPG agent at different flight altitudes. At each flight altitude, the success rate, stray rate and crash rate are obtained in the same way as those in Table II and the average flight distance is estimated by averaging the length of successfully arrived trajectories in each environment.

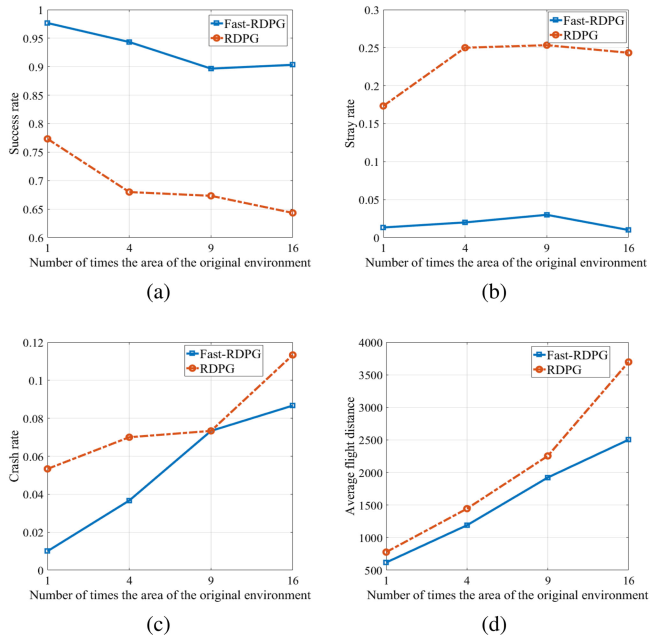


Fig. 9. Success rate, stray rate, crash rate, and average flight distance of navigation missions of the Fast-RDPG agent as well as the RDPG agent in the complex environments of different sizes. Environments of different sizes are obtained by simply extending the original environment shown in Fig. 3 to larger sizes.

TABLE III
STATISTICAL QUANTITIES OF FAST-RDPG TRAINED WITH DIFFERENT REWARD

r_{free};r_{step}	Success rate	Stray rate	Crash rate	Flight distance
$1.5; 0.0$	11.67%	86.67%	1.67%	$1410.20m$
$0.1; -0.6$	97.67%	1.33%	1.00%	$618.23m$

the flight altitude decreases. The higher success rate, lower stray rate and crash rate of the Fast-RDPG agent at different flight altitude demonstrates that it does not simply remember the structure of the complex environment but derives the ability to cope with complex environments. Fig. 8(d) is the average flight distance of the two agents. As we can see, the average flight distance of the RDPG agent is also uniformly longer than that of our Fast-RDPG agent at different flight altitude, illustrating that our proposed Fast-RDPG is more efficient than DDPG in escaping from traps in the complex environment. Besides, we see that the lower the flight altitude, the more steps the two agents spend in heading for the destination. The results are in accordance with our intuition because with the decrease of flight altitude, obstacles in the environment become denser and the path to the destination becomes more twisty. In addition, as illustrated in Fig. 9, with the increase of the area of the complex environment, both the two agents manifest a tendency to decrease in success rate and to increase in stray rate, crash rate and average flight distance. Nonetheless, the success rate of our Fast-RDPG agent in the largest-scale environment is still higher than 90% and is around 13% higher than that of the DDPG agent in the smallest-scale environment. Besides, the RDPG agent's average flight distance is much shorter than our Fast-RDPG agent's especially in the largest-scale environment. Considering that the distance between UAVs and the destination is one dimension of the observation space and the agents are never trained in larger-scale complex environments, the high performance of our Fast-RDPG illustrates that it has strong generalization ability.

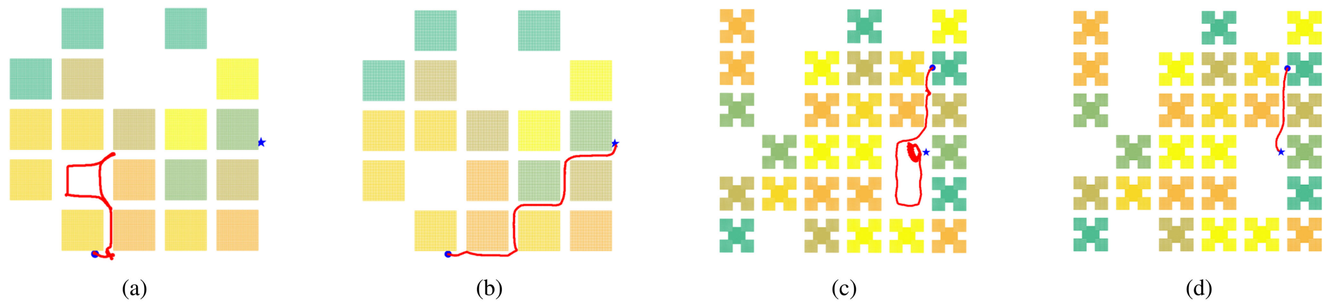


Fig. 10. Schematic view of the influence of reward on navigation behaviors. For brevity, we only choose two pairs of starting points and target points in ENV I and ENV IV. Subfigure (a) and (c) show the navigation trajectories of the Fast-RDPG agent with $r_{free} = 1.5$ and $r_{step} = 0.0$ and (b) and (d) the corresponding navigation trajectories with $r_{free} = 0.1$ and $r_{step} = -0.6$.

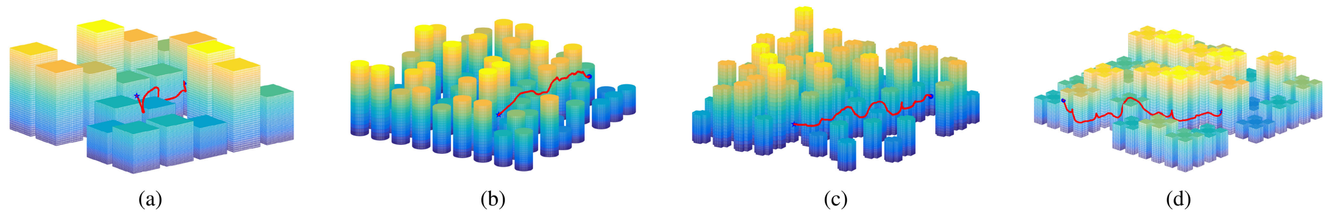


Fig. 11. Navigation trajectories of the Fast-RDPG agent in the 3D environment. For clarity, we do not draw buildings obstructing our view of the trajectories.

D. The Effect of Reward

In addition to partial observability of states, reward also affects the navigation behavior of the Fast-RDPG agent. We investigate its influence on navigation behaviors by varying the free-space reward and step penalty. We reset the free-space reward and step penalty to $r_{free} = 1.5$ and $r_{step} = 0.0$, respectively, keep the other parameters unchanged and retrain the Fast-RDPG agent. Table III shows the success rate, stray rate, crash rate and average flight distance of the agent before and after retraining. As we can see, when trained with low step penalty and high free-space reward, the Fast-RDPG agent shows a drastic success rate decrease at around 85%. Besides, its stray rate and average flight distance significantly increases from 1.33% and 618.23 m to 86.67% and 1410.20 m, respectively, reflecting that the retrained agent tends to stray into free spaces rather than head for the destination. Fig. 10 gives a schematic view of the results. As we see, when fed with higher free-space reward and lower step penalty, the well-trained agent strays into free spaces and fails to strike the destination. The result is consistent with our intuition since the value of free-space reward and step penalty exactly modulates *how much* we want UAVs to head for free spaces so as to avoid obstacles and to head for the destination as soon as possible. As a result, though non-sparse reward is an appropriate way to encode our domain knowledge on the problem and to speed up the learning procedure, it must be designed and fine-tuned carefully. This might be the main drawback of non-sparse reward without reward shaping.

E. Navigation in Three-Dimensional Environments

Here we demonstrate that our method also works in three-dimensional (3D) environments. Since UAVs are generally not permitted to fly neither too high nor too low in urban areas, we restrict its cruising altitude to $[50\text{ m}, 130\text{ m}]$. The ascending and descending processes at the beginning and the end of navigation procedures are not taken into consideration for UAVs can take off and land vertically. Thus as long as the UAV arrives at the overhead space of the target position, the navigation mission completes. For each horizontal detection direction, we add four range finders to sense obstacles outside the horizontal plane, illustrated in Fig. 2(a) and (c) (we assume the UAV's first-perspective direction is in parallel with the horizontal plane).

Adjustments are also made to the reward function, the action and the observation spaces. The UAV will get an extra penalty if its altitude is near either end of the altitude interval. The penalty is the same as the obstacle penalty except that d_{min} here represents the minimum vertical distance to the ends of the altitude interval. Besides, as illustrated in Fig. 2(c) and (d), a new dimension is added to the control vector to maneuver the vertical angle ϑ_z between the UAV's movement direction and first-perspective direction, and the observation space is adjusted by adding the UAV's altitude h_z and vertical angle ϑ_z and the 36 distances $\{d_{i1}, d_{i2}, d_{i3}, d_{i4}\}_{i=1}^{9}$ returned by the additional range finders.

With these adjustments, we re-implement Fast-RDPG. As illustrated in Fig. 11, the UAV can also perform navigation tasks in 3D environments. In contrast to the two-dimensional setting, the UAV in the 3D environment can avoid obstacles by changing its horizontal direction as well as its altitude. The success rate, stray rate and crash rate of the Fast-RDPG agent are 96.0%, 1.0% and 3.0%, respectively. Compared with the two-dimensional setting, there is a crash rate increase at 1.25%. We think the performance degradation is resulted from the poor sensing capacity of range finders. Different from cameras, range finders can only detect obstacles in several pre-specified directions. Therefore, it is possible that there are obstacles very close to the UAV but the UAV "thinks" it keeps a proper distance to them. The situation becomes even worse in 3D environments since 3D environments are even more complex than two-dimensional environments.

VII. Conclusion and Future Work

In this work we develop a DRL framework for UAV navigation in large-scale complex environments. The problem is formulated as a POMDP and a general DRL algorithm is proposed to address it. Without map reconstruction and path planning, our method enables UAVs to fly from arbitrary departure places to destinations in large-scale complex environments. Both theoretical derivation and simulation results demonstrate that Fast-RDPG is more efficient that RDPG. Besides, it could generalize to more complex, larger-scale and 3D environments with only minor performance degradation. Nonetheless, there are still some work to do in the future. For instance, the reward function must be designed and fine-tuned, otherwise it may yield unsatisfactory performance. A potential way to settle this problem is by directly handling sparse rewards.

Appendix A

The proof follows along similar lines of the standard *policy gradient theorem for MDPs* in [42]. As derived in (21), shown at the bottom of this page, the value function $V_\pi(h)$ and action-value function $Q_\pi(h,a)$ satisfy a recursive relationship similar to those of MDPs, where we use $p(h_{t+1}|a_t,h_t)$ to denote the *observation dynamics* of history trajectories, analogous to the environment dynamics. Note that $p(h_{t+1}|a_t,h_t)$ in fact refers to $p(o_{t+1}|a_t,h_t)$. By setting $p(h_{t+1}|a_t,h_t)$ to zero if h_t is not the prefix of h_{t+1}, it can be extended to represent transition probabilities of any pairs of history trajectories.

Since we explicitly reformulate the target function in (18) in terms of value functions of history trajectories, its gradient involves in taking the derivative of the value function $V_\pi(h)$ in terms of the policy weight θ. Equipped with the recursive relationship, the left part of this proof becomes straightforward. By applying (21) and following the lines of derivation in [42], we can write the gradient of the value function $V_\pi(h_0)$ as:

$$\begin{aligned}\frac{\partial V_\pi(h_0)}{\partial\theta} &= \sum_h\sum_{k=0}^{\infty}\gamma^k p(h_0\to h,k,\pi)\sum_a\frac{\partial\pi(a|h)}{\partial\theta}Q_\pi(h,a)\\ &= \sum_h d_{h_0,\pi}(h)\sum_a\frac{\partial\pi(a|h)}{\partial\theta}Q_\pi(h,a)\end{aligned}\tag{22}$$

where $\Pr(h_0\to h,k,\pi)$ denotes the probability of transitioning from history trajectory h_0 to h in k steps following policy π, and h could be any history trajectories since we have extended $p(h_{t+1}|a_t,h_t)$ to the entire history trajectory space. Besides, $d_{h_0,\pi}(h)=\gamma^k p(h_0\to h,k,\pi)$ represents the un-normalized history trajectory distribution in a randomly generated episode starting from an initial history trajectory h_0 and then following policy π. Then it is immediate that the gradient of the target function (18) can be written as:

$$\begin{aligned}\nabla\eta(\pi) &= \mathbb{E}_{h_0\sim p(h_0)}\left[\frac{\partial V_\pi(h_0)}{\partial\theta}\right]\\ &= \mathbb{E}_{h_0\sim p(h_0)}\mathbb{E}_{h\sim d_{h_0,\pi}(h)}\mathbb{E}_{a\sim\pi(a|h)}\\ &\quad[\nabla\log(\pi(a|h))Q_\pi(h,a)]\\ &= \mathbb{E}_{h\sim d_\pi(h)}\mathbb{E}_{a\sim\pi(a|h)}\left[\nabla\log(\pi(a|h))Q_\pi(h,a)\right].\end{aligned}\tag{23}$$

where $d_\pi(h)$ are as defined in Theorem I.

Appendix B

We first introduce a lemma.

Lemma (Derivative of Dirac Delta Functions).

Supposing $f(x)$ is any continuous, well-behaved function and $\delta(x)$ the Dirac delta function, the derivative of $\delta(x)$ holds the following property:

$$\int_{-\infty}^{\infty}\delta'(x)f(x)\,dx=-f'(0).\tag{24}$$

Recall that in the main context we replace the integral operations by summation operations in all cases for ease of denotation. Here we rewrite (19) as

$$\nabla\eta(\pi)=\int_h d_\pi(h)\int_a\pi(a|h)\nabla\log\left(\pi(a|h)\right)Q_\pi(h,a)\,dadh.\tag{25}$$

$$\begin{aligned}Q_\pi(h_t,a_t) &= \mathbb{E}_{s_t\sim p(s_t|h_t)}\left[Q_\pi^{h_t}(s_t,a_t)\right]\\ &= \mathbb{E}_{s_t\sim p(s_t|h_t)}\left\{\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|s_t,a_t)}\left[r_{t+1}\right]\right\}+\gamma\mathbb{E}_{s_t\sim p(s_t|h_t)}\left\{\mathbb{E}_{o_{t+1},s_{t+1}\sim p(s_{t+1}|s_t,a_t)p(o_{t+1}|s_{t+1})}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\}\\ &= R_\pi(a_t,h_t)+\gamma\left\{\mathbb{E}_{s_t,s_{t+1},o_{t+1}\sim p(s_t,s_{t+1}|h_t,a_t,o_{t+1})p(o_{t+1}|a_t,h_t)}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\}\\ &= R_\pi(a_t,h_t)+\gamma\mathbb{E}_{h_{t+1}\sim p(h_{t+1}|a_t,h_t)}\left\{\mathbb{E}_{s_t,s_{t+1}\sim p(s_t,s_{t+1}|h_{t+1})}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\}\\ &= R_\pi(a_t,h_t)+\gamma\mathbb{E}_{h_{t+1}\sim p(h_{t+1}|a_t,h_t)}\left\{\mathbb{E}_{s_{t+1}\sim p(s_{t+1}|h_{t+1})}\left[V_\pi^{h_{t+1}}(s_{t+1})\right]\right\}\\ &= R_\pi(a_t,h_t)+\gamma\sum_{h_{t+1}}p(h_{t+1}|a_t,h_t)V_\pi(h_{t+1}).\end{aligned}\tag{21}$$

$$
\begin{aligned}
\nabla\eta(\pi) &= \int_h d_\pi(h)\int_a \pi(a|h)\nabla\log\left(\pi(a|h)\right)Q_\pi(h,a)\,dadh\\
&= \int_h d_\pi(h)\int_{f(a,h)}\frac{\partial\delta(f(a,h))}{\partial\theta}Q_\pi(h,\mu(h)-f(a,h))\,df(a,h)dh\\
&= \int_h d_\mu(h)\int_{f(a,h)}\frac{\partial\delta(f(a,h))}{\partial f(a,h)}\frac{\partial f(a,h)}{\partial\theta}Q_\pi(h,\mu(h)-f(a,h))\,df(a,h)dh\\
&= -\int_h d_\mu(h)\left.\frac{\partial(\frac{\partial f(a,h)}{\partial\theta}Q_\mu(h,\mu(h)-f(a,h)))}{\partial f(a,h)}\right|_{f(a,h)=0}dh\\
&= -\int_h d_\mu(h)\left.\frac{\partial\frac{\partial f(a,h)}{\partial\theta}}{\partial f(a,h)}Q_\pi(h,\mu(h)-f(a,h))\right|_{f(a,h)=0}dh-\int_h d_\mu(h)\left.\frac{\partial Q_\pi(h,\mu(h)-f(a,h))}{\partial f(a,h)}\frac{\partial f(a,h)}{\partial\theta}\right|_{f(a,h)=0}dh\\
&= \int_h d_\mu(h)\left.\frac{\partial Q_\pi(h,a)}{\partial a}\right|_{a=\mu(h)}\frac{\partial\mu(h)}{\partial\theta}dh
\end{aligned}
\tag{26}
$$

A deterministic policy can be represented as $\delta(\mu(h)-a)$. If we let $f(h,a)=\mu(h)-a$, the policy can be denoted as $\delta(f(a,h))$. Substituting it into (25), employing the change of variable technique and the property present in the lemma, we obtain the final result, shown in (26) at the top of this page.

REFERENCES

[1] F. Mohammed, A. Idries, N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "UAVs for smart cities: Opportunities and challenges," in *Proc. IEEE Int. Conf. Unmanned Aircraft Syst.*, 2014, pp. 267–273.

[2] J. Israelsen, M. Beall, D. Bareiss, D. Stuart, E. Keeney, and J. van den Berg, "Automatic collision avoidance for manually tele-operated unmanned aerial vehicles," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2014, pp. 6638–6643.

[3] K. Chee and Z. Zhong, "Control, navigation and collision avoidance for an unmanned aerial vehicle," *Sens. Actuators A, Phys.*, vol. 190, pp. 66–76, 2013.

[4] P. Agrawal, A. Ratnoo, and D. Ghose, "Inverse optical flow based guidance for UAV navigation through urban canyons," *Aerosp. Sci. Technol.*, vol. 68, pp. 163–178, 2017.

[5] N. Gageik, P. Benz, and S. Montenegro, "Obstacle detection and collision avoidance for a UAV with complementary low-cost sensors," *IEEE Access*, vol. 3, pp. 599–609, 2015.

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

[7] S. Roelofsen, D. Gillet, and A. Martinoli, "Reciprocal collision avoidance for quadrotors using on-board visual detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2015, pp. 4810–4817.

[8] F. Belkhouche, "Modeling and calculating the collision risk for air vehicles," *IEEE Trans. Veh. Technol.*, vol. 62, no. 5, pp. 2031–2041, Jun. 2013.

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

[10] J. Q. Cui, S. Lai, X. Dong, and B. M. Chen, "Autonomous navigation of UAV in foliage environment," *J. Intell. Robot Syst.*, vol. 84, no. 1–4, pp. 259–276, 2016.

[11] T. Gee, J. James, W. van der Mark, P. Delmas, and G. Gimel'farb, "LIDAR guided stereo simultaneous localization and mapping (SLAM) for UAV outdoor 3-D scene reconstruction," in *Proc. IEEE Int. Conf. Image Vis. Comput. New Zealand*, 2016, pp. 1–6.

[12] R. Li, J. Liu, L. Zhang, and Y. Hang, "LIDAR/MEMS IMU integrated navigation (SLAM) method for a small UAV in indoor environments," in *Proc. IEEE Conf. Inertial Sensors Syst. Symp.*, 2014, pp. 1–15.

[13] C. Fu, M. A. Olivares-Mendez, R. Suarez-Fernandez, and P. Campoy, "Monocular visual-inertial SLAM-based collision avoidance strategy for fail-safe UAV using fuzzy logic controllers," *J. Intell. Robot Syst.*, vol. 73, no. 1–4, pp. 513–533, 2014.

[14] A. E. Oguz and H. Temeltas, "On the consistency analysis of A-SLAM for UAV navigation," *Proc. SPIE*, vol. 9084, no. 18, pp. 5450–5453, 2014.

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

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

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

[18] A. M. Zhang and L. Kleeman, "Robust appearance based visual route following for navigation in large-scale outdoor environments," *Int. J. Robot Res.*, vol. 28, no. 3, pp. 331–356, 2009.

[19] S. Karpenko, I. Konovalenko, A. Miller, B. Miller, and D. Nikolaev, "UAV control on the basis of 3-D landmark bearing-only observations," *Sensors*, vol. 15, no. 12, pp. 29802–29820, 2015.

[20] I. A. Konovalenko, A. B. Miller, B. M. Miller, and D. P. Nikolaev, "UAV navigation on the basis of the feature points detection on underlying surface," in *Proc. Eur. Conf. Model. Simul.*, 2015, pp. 499–505.

[21] A. Cerón, I. F. Mondragón, and F. Prieto, "Visual based navigation for power line inspection by using virtual environments," *Proc. SPIE*, vol. 9406, 2015, Art. no. 94060J.

[22] R. Strydom, S. Thurrowgood, and M. V. Srinivasan, "Visual odometry: Autonomous UAV navigation using optic flow and stereo," in *Proc. Australas. Conf. Robot. Automat.*, 2014, pp. 1–10.

[23] S. Ross *et al.*, "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 1765–1772.

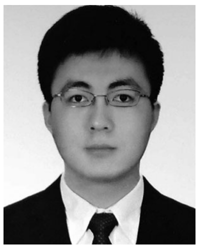
[24] A. Faust, I. Palunko, P. Cruz, R. Fierro, and L. Tapia, "Automated aerial suspended cargo delivery through reinforcement learning," *Artif. Intell.*, vol. 247, pp. 381–398, 2017.

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

[26] C. Wang, J. Wang, X. Zhang, and X. Zhang, "Autonomous navigation of UAV in large-scale unknown complex environment with deep reinforcement learning," in *Proc. IEEE Int. Conf. GlobalSIP*, 2017, pp. 858–862.

[27] L. Chrisman, "Reinforcement learning with perceptual aliasing: The perceptual distinctions approach," in *Proc. Assoc. Advancement Artif. Intell.*, 1992, pp. 183–188.

[28] L. Lin and T. M. Mitchell, "Memory approaches to reinforcement learning in non-Markovian domains," School Comput. Sci., Carnegie Mellon Univ., Pittsburgh, PA 15213, USA, Tech. Rep. CMU-CS-92-138, May 1992.
[29] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 229–256, 1992.
[30] D. Wierstra, A. Foerster, J. Peters, and J. Schmidhuber, "Solving deep memory POMDPs with recurrent policy gradients," in *Proc. Int. Conf. Artif. Neural Netw.*, 2007, pp. 697–706.
[31] F. Jurčíček, B. Thomson, and S. Young, "Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs," *ACM Trans. Speech Lang. Process.*, vol. 7, no. 3, pp. 1–26, 2011.
[32] D. Wierstra and J. Schmidhuber, "Policy gradient critics," in *Proc. Eur. Conf. Mach. Learn.*, 2007, pp. 466–477.
[33] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[34] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, vol. 1, no. 1. Cambridge, MA, USA: MIT Press, 1998.
[35] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, arXiv:1509.02971.
[36] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.
[37] N. Heess, J. J. Hunt, T. P. Lillicrap, and D. Silver, "Memory-based control with recurrent neural networks," 2015, arXiv:1512.04455.
[38] 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.
[39] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. Int. Conf. Mach. Learn.*, 1999, vol. 99, pp. 278–287.
[40] A. Harutyunyan, S. Devlin, P. Vrancx, and A. Nowé, "Expressing arbitrary reward functions as potential-based advice," in *Proc. Assoc. Advancement Artif. Intell.*, 2015, pp. 2652–2658.
[41] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Repr.*, 2015.
[42] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.



Chao Wang (S'16) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2015. He is currently working toward the Ph.D. degree with the Department of Electronic Engineering, Tsinghua University under the supervision of Professor X. Zhang. His research interests include signal processing, machine learning, and intelligent control.

Jian Wang (SM'18) received the Ph.D. degree in electronic engineering from Tsinghua University, Beijing, China, in 2006. In 2006, he joined the faculty of Tsinghua University, where he is currently an Associate Professor with the Department of Electronic Engineering. His research interests include intelligent collaborative systems, information security, privacy-enhancing technology, and signal processing in the encrypted domain and wireless networks.

Yuan Shen (S'05–M'14) received the B.E. degree (with highest honors) in electronic engineering from Tsinghua University, Beijing, China, in 2005, and the S.M. and Ph.D. degrees in electrical engineering and computer science from MIT, Cambridge, MA, USA, in 2008 and 2014, respectively.

He is an Associate Professor with the Department of Electronic Engineering, Tsinghua University. Prior to that, he was a Research Assistant and then a Postdoctoral Associate with the Wireless Information and Network Sciences Laboratory, MIT, during 2005–2014. His research interests include statistical inference, communication theory, information theory, and optimization. His current research focuses on network localization and navigation, inference techniques, resource allocation, and cooperative networks.

Dr. Shen is the Chair (2019–2020), and also served as the Vice Chair (2017–2018), and the Secretary (2015–2016) for the IEEE ComSoc Radio Communications Committee. He is the TPC symposium Co-Chair for the IEEE ICC (2020) and also he was the Co-Chair for IEEE Globecom (2018 and 2016), EUSIPCO (2016), and IEEE ICC Advanced Network Localization and Navigation (ANLN) Workshop (2016–2018). He has been an Editor for the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS since 2018, the IEEE WIRELESS COMMUNICATIONS LETTERS since 2018, the IEEE CHINA COMMUNICATIONS since 2017, and the IEEE COMMUNICATIONS LETTERS (2015–2018) and a Guest-Editor for the *International Journal of Distributed Sensor Networks* (2015). He was a recipient of the IEEE ComSoc Asia-Pacific Board Outstanding Young Scholar Award, Qiu Shi Outstanding Young Scholar Award, the China's Youth 1000-Talent Program, and the Marconi Society Paul Baran Young Scholar Award. His papers received the IEEE ComSoc Fred W. Ellersick Prize and three Best Paper Awards from the IEEE international conferences.

Xudong Zhang (M'04) received the Ph.D. degree from Tsinghua University, Beijing, China, in 1997. He has been with the Department of Electronics Engineering, Tsinghua University since 1997. He has authored or coauthored more than 150 papers and three books in the field of signal processing and machine learning. His research interests include statistical signal processing, machine learning theory, and multimedia signal processing.