Deep-Reinforcement-Learning-Based Autonomous UAV Navigation With Sparse Rewards

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

***Abstract*—Unmanned aerial vehicles (UAVs) have the potential in delivering Internet-of-Things (IoT) services from a great height, creating an airborne domain of the IoT. In this article, we address the problem of autonomous UAV navigation in large-scale complex environments by formulating it as a Markov decision process with sparse rewards and propose an algorithm named deep reinforcement learning (RL) with nonexpert helpers (LwH). In contrast to prior RL-based methods that put huge efforts into reward shaping, we adopt the sparse reward scheme, i.e., a UAV will be rewarded if and only if it completes navigation tasks. Using the sparse reward scheme ensures that the solution is not biased toward potentially suboptimal directions. However, having no intermediate rewards hinders the agent from efficient learning since informative states are rarely encountered. To handle the challenge, we assume that a prior policy (nonexpert helper) that might be of poor performance is available to the learning agent. The prior policy plays the role of guiding the agent in exploring the state space by reshaping the behavior policy used for environmental interaction. It also assists the agent in achieving goals by setting dynamic learning objectives with increasing difficulty. To evaluate our proposed method, we construct a simulator for UAV navigation in large-scale complex environments and compare our algorithm with several baselines. Experimental results demonstrate that LwH significantly outperforms the state-of-the-art algorithms handling sparse rewards and yields impressive navigation policies comparable to those learned in the environment with dense rewards.**

***Index Terms*—Deep reinforcement learning (RL), prior information, sparse reward, unmanned aerial vehicle (UAV) navigation.**

I. Introduction

OVER the past few years, we have seen an unprecedented growth of unmanned aerial vehicles (UAVs) applications, such as goods delivery, emergency aid, and intelligent transportation [1]–[3]. When combined with techniques, such as 5G networks [4]–[6] and vehicular *ad hoc* networks [7]–[10], UAVs have the potential in delivering Internet-of-Things (IoT) services from a great height, creating an airborne domain of the IoT. One of the key techniques behind these applications is the autonomous navigation of UAVs in large-scale complex environments. Conventional methods address the problem based on simultaneously localization-and-mapping techniques [11]–[13] and sensing-and-avoidance techniques [14]–[17]. Though these nonlearning-based methods have shown satisfactory performance, they are generally limited to simple and small environments, such as the countryside or indoor areas. It is difficult to directly generalize them to large-scale complex environments since passively avoiding dense obstacles or actively constructing maps are inefficient when the environment is large and complex.

Observing that the UAV navigation problem is a sequential decision-making problem, researchers turn to reinforcement learning (RL)-based methods [18]–[21]. For example, Imanberdiyev *et al.* [18] developed a model-based RL algorithm as a high-level control method for UAV navigation in a grid map. Ross *et al.* [20] built an imitation learning (IL)-based controller using a small set of human expert demonstrations and achieved a good performance in natural forest environments. This article is mainly based on [21] that formulates UAV navigation in large-scale complex environments as a partially observable Markov decision process (MDP). While [21] focuses on addressing the problem of partial observability of states, in this article, we aim at solving the reward shaping problem raised by [21], where manually designed reward functions are prone to yielding unexpected control policies. Specifically, the navigation problem is formulated as an MDP with sparse rewards. In contrast to the dense reward setting, where the UAV can get a reward each time it interacts with the environment by executing an action (i.e., control command), in the sparse-reward setting, the UAV can get a nonzero reward if and only if it successfully completes the navigation task. While defining sparse rewards is simple and straightforward, the underlying learning problem becomes hard to solve. As we all know, effective RL relies on collecting informative reward signals, so the agent has to discover a long sequence of "correct" actions so as to reach the target that yields sparse rewards. While the environment is scattered with dense obstacles (e.g., high buildings) and has a large coverage area, discovering sparse reward signals via random search is highly unlikely.

A magnitude of methods has been proposed to deal with the challenge raised by sparse rewards. The most basic approach is to reshape the reward function by utilizing intrinsic curiosity [22], [23] or information gain [24] that encourages the agent to explore states that are rarely seen before.

Manuscript received October 3, 2019; revised January 10, 2020; accepted January 30, 2020. Date of publication February 11, 2020; date of current version July 10, 2020. This work was supported by the National Key Research and Development Program of China under Grant 2016YFB0100902. The work of Jingjing Wang was supported by the Shuimu Tsinghua Scholar Program. *(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; chinaeephd@gmail.com; zhangxd@tsinghua.edu.cn).

Digital Object Identifier 10.1109/JIOT.2020.2973193

These methods often require fitting models of the environment dynamics so as to measure curiosity. However, either a poorly fitted environment model or less predictable environment dynamics can significantly degenerate their performance. Besides, since these intuitively designed rewards generally do not satisfy the policy invariance condition [25] that the optimal policy of an MDP is preserved if the added rewards for transitions between states can be expressed as the difference in the value of an arbitrary potential function, they may lead to suboptimal policies. A more common way is to use expert-demonstrated trajectories to guide the learning procedure by IL or Learning from Demonstrations (LfD).[1] IL aims at mimicking expert behaviors. As an IL method, inverse RL [26] derives control policies by recovering the reward function from expert demonstrations. The most recent progress of IL is generative adversarial IL [27], which uses a discriminator to distinguish whether a state-action pair is similar to those given by experts. The confidence score is then utilized by the generator to learn a policy to confuse the discriminator. The major drawback of IL is that the performance of the learned policy is upper bounded by the expert policy and usually suffers from drastic performance degradation if expert demonstrations are imperfect. LfD overcomes such limitations by leveraging valuable feedback given by expert demonstrations as well as the environment. Deep Q-LfDs [28] and deep deterministic policy gradient from demonstrations (DDPGfD) [29], for example, add potentially imperfect demonstrations into a replay buffer and never flush them out so that the learning agent can continuously learn from the data sampled from both demonstrations and the environment. Policy optimization from demonstrations (POfD) [30], instead, leverages demonstrations by enforcing stationary state distribution matching between the learned policy and expert data. Though these LfD methods have shown state-of-the-art results in several simple Atari games [31], their performance in environments with extremely sparse rewards is unclear. Besides, for real applications, we generally have strong prior information on the problem to be solved. However, such knowledge is rarely explored and exploited.

In this article, we formulate the problem of UAV navigation in large-scale complex environments as an MDP with extremely sparse rewards. UAVs would be rewarded if and only if it completes the navigation task. To overcome the challenge raised by sparse rewards, we propose an algorithm named DRL with nonexpert helpers (LwH). It is mainly based on three principles.

1) A prior control policy (nonexpert helper) is assumed available to the learning agent. The policy could be of poor performance as long as it can assist the agent in achieving goals (completing navigation tasks) with some probability.
2) The prior policy plays the role of helping the agent explore the environment as well as assisting the agent in achieving goals.
3) The influence of the prior policy is reduced as learning progresses for the purpose of enabling the agent to achieve goals on its own.

LwH operates in a way similar to a baby learning to run. First, he learns to stand up with his mom's help, then he struggles to walk by himself, and finally, he can walk and even run by himself.

It is reasonable to assume that a prior policy is available since for real applications, such as self-driving cars and UAV delivery, human-crafted policies (e.g., planning, sensing, and avoidance) have been applied at scale. In our UAV navigation scenario, supposing the destination is northeast of the UAV, "heading north while avoiding obstacles until the target is on the east, then heading east while avoiding obstacles until arriving at the destination" could be a feasible prior policy.

LwH deviates from previous methods in two aspects: 1) LwH requires a prior control policy (which could be of poor performance) instead of a set of demonstrations and 2) the learning objective of LwH changes over time while most of the existing work only focuses on static objectives. We investigate the LwH algorithm in the context of UAV navigation in large-scale complex environments. Our main contributions can be summarized as follows.

1) To the best of our knowledge, this is the first work modeling UAV navigation in large-scale complex environments as an MDP with extremely sparse rewards.
2) To address the sparse-reward problem, a nonexpert prior policy is used to guide the agent in exploring the state space and to set dynamical learning goals. Motivated by this, a learning algorithm is developed based on rigorous mathematical formulation and derivations.
3) Extensive experimental results demonstrate that our method yields excellent navigation policies and significantly outperforms the state-of-the-art RL algorithms in terms of solving MDPs with sparse rewards.

The remainder of this article is outlined as follows. The problem formulation is detailed in Section II. Several baseline algorithms along with our proposed LwH addressing RL problems with sparse rewards are elaborated in Section III. In Section IV, extensive simulation results are provided for evaluating our proposed LwH algorithm followed by our conclusion and future work in Section V.

II. Problem Formulation and MDP Modeling

A. Preliminaries: MDP and DRL

We model the UAV navigation problem as an MDP. An MDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma, \rho_0)$ [32], where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the state transition probability, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in (0, 1)$ is the discount factor, and $\rho_0 : \mathcal{S} \to \mathbb{R}$ is the distribution of the initial state s_0. The goal of RL is to learn a parameterized control policy $a \sim \pi_\theta(\cdot|s)$ (for ease of denotation, we make it implicit in all cases that π in fact represents π_θ) that maximizes an objective (or its surrogates)

[1]Demonstrations refer to data sets generated by humans. Taking the UAV navigation problem as an example, demonstrations can be collected by letting a person manually maneuver a UAV and recording the operation instructions (i.e., actions) along with the UAV's sensor outputs. If the demonstrations are generated by an expert, they are expert demonstrations. Otherwise, they are nonexpert demonstrations.

defined by long-term discounted cumulative rewards

$$\max_{\theta} \eta(\pi) = \mathbb{E}_{s_0}[V_\pi(s_0)] \tag{1}$$

where $V_\pi(s)$ is referred as the value function of state s and defined as

$$V_\pi(s) = \mathbb{E}_{s_{t\geq 1}\sim\mathcal{P}, a_{t\geq 0}\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)|s_0 = s, \pi\right] \tag{2}$$

RL sometimes involves in estimating the action-value function

$$Q_\pi(s, a) = \mathbb{E}_{s'\sim\mathcal{P}}\left[r(s, a) + \gamma V_\pi(s')\right] \tag{3}$$

which describes the expected return of taking action a_t at state s_t and thereafter following policy π. By definition, $V_\pi(s) = \sum_a \pi(a|s)Q_\pi(s, a)$. Besides, their difference $A_\pi(s, a) = Q_\pi(s, a) - V_\pi(s)$ is referred as the advantage function. Apparently, we have

$$\mathbb{E}_{a\sim\pi(a|s)}[A_\pi(s, a)] = 0. \tag{4}$$

Equation (1) is generally solved within the actor–critic framework [32]. The gradient of the objective function with respect to policy parameters θ is given by [32]

$$\nabla_\theta \eta(\pi) = \mathbb{E}_{s\sim d_\pi(s), a\sim\pi(\cdot|s)}\left[\nabla_\theta \log \pi(a|s)\hat{Q}_\pi(s, a)\right] \tag{5}$$

where $d_\pi(s)$ is the unnormalized state distribution induced by the policy π, and $\hat{Q}_\pi(s, a)$ is an estimate of the action–value function $Q_\pi(s, a)$, which is often parameterized by a function $Q_\omega(s, a)$ [or $V_\omega(s)$]. The critic is adjusted by minimizing the n-step time-difference (TD) error [32]

$$\mathbb{E}_{a\sim\pi(\cdot|s)}\left\|\sum_{\tau=0}^{n} r(s_{t+\tau}, a_{t+\tau}) + \gamma Q_\omega(s_{t+\tau+1}, a_{t+\tau+1}) - Q_\omega(s_t, a_t)\right\|_2 \tag{6}$$

where $Q_\omega(s, a)$ can be replaced with $V_\omega(s)$ if we use $V_\omega(s)$ to estimate $Q_\pi(s, a)$. In the actor–critic framework, the policy and the action–value function are also called the actor and the critic.

B. MDP Modeling of UAV Navigation With Sparse Rewards

Here, we give a formal description of the problem of UAV navigation in large-scale complex environments. Suppose a UAV is located at some point (departure position) in a 3-D environment, which is denoted as (x_0, y_0, z_0) in the Earth-fixed coordinate frame, and targets at flying to a destination that is denoted as (x_d, y_d, z_d). Dense obstacles are scattered in the environment. The environment map, departure positions, and destinations are randomly generated across different navigation tasks. The UAV can only have access to its sensor readings of observations of the local environment as well as its relative position to the destination. The goal of the UAV is to fly from the departure place to the destination by controlling its speed, direction, and height based on limited sensor observations.

For simplicity, we assume that the UAV flies at a fixed height (i.e., $z_0 = z_d = H$, where H is a positive constant), and

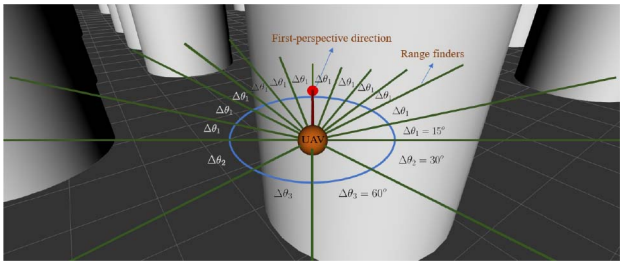


Fig. 1. Illustration of a UAV's observation of the local environment. The UAV and buildings (obstacles) are represented as an orange sphere and gray cylinders, respectively. The red arrow represents the UAV's first-perspective direction and the green lines represent signals emitted from range finders.

suppose that control commands can take effect in no time. Then, the dynamics of the UAV can be formulated as

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

where v_t and ϕ_t denote the speed and the first-perspective direction of the UAV at time step t, and ρ_t and φ_t denote the throttle and the steering commands. In this article, we assume that the UAV's observation of the local environment is achieved by range finders, as illustrated in Fig. 1, and its relative position to the destination is measured by the GPS. The overall observations can be denoted as $o_t = [d_1^t, \ldots, d_{16}^t, d_r^t, \xi^t, v^t, \phi^t]$, where d_r^t and ξ^t denote the distance and the angle (relative to the north direction) between the UAV's current position and the destination at time step t. To model the navigation problem, we regard the UAV's overall observations as the system's states and the throttle as well as the steering commands as users' actions. The state transition probability is automatically determined by the environment, which is unknown.

In this article, we use the sparse reward scheme, i.e., the UAV would be rewarded if and only if its distance to the destination is less than a prefixed threshold. Specifically, we define the sparse reward as

$$r_{sp}(s_t, a_t) = \begin{cases} 1, & \|(x_t, y_t) - (x_d, y_d)\|_2 \leq r_d \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

where r_d is a prefixed constant and (x_t, y_t) denotes the UAV's horizontal position at time step t.

III. Method

A. Baselines: A3C, DDPGfD, and POfD

A3C: A3C [33] constructs approximations to the policy $\pi(a|s)$ and the value $V_\pi(s)$ using two neural networks with parameters θ and ω. Besides, the policy network and the value network generally share the same hidden architecture and parameters. Parameters θ are adjusted by

$$\Delta\theta = \sum_t \nabla_\theta \log(\pi(a_t|s_t))(R_t(n) - V_\omega(s_t)) \tag{9}$$

and parameters ω are adjusted by minimizing

$$J_V = \sum_t \left[(R_t(n) - V_\omega(s_t))^2\right] \tag{10}$$

where

$$R_t(n) = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_i + \gamma^n V_\omega(s_{t+n}). \tag{11}$$

In A3C, many instances of the agent interact in parallel with many instances of the environment and update parameters of the policy network and the value network asynchronously, which stabilizes and accelerates the learning procedure.

DDPGfD: DDPGfD [29], built upon DDPG [34], settles DRL problems with sparse rewards by taking advantage of potentially imperfect demonstrations. DDPG aims at learning a deterministic policy that directly maps states to actions. The actor and the critic are modeled as two neural networks $a = \mu_\theta(s)$ and $Q_\omega(s, a)$. The actor parameters θ are updated by

$$\Delta_\theta = \mathbb{E}_{s\sim d_{\mu(s)}}\left[\nabla_\theta \mu_\theta(s)\, \nabla_a Q_\omega(s, a)|_{a=\mu_\theta(s)}\right] \tag{12}$$

and the critic parameters ω are updated by

$$J_Q = \mathbb{E}_{\mathcal{D}}\left[(R_t(n) - Q_\omega(s_t, a_t))^2\right] \tag{13}$$

where

$$R_t(n) = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_i + Q_\omega(s_{t+n}, \mu_\theta(s_{t+n})) \tag{14}$$

and n generally takes the value 1. The critic parameters ω and the actor parameters θ are updated by sampling a minibatch of transition tuples (s_t, a_t, s_{t+1}, r_t) from a replay memory $\mathcal{D}$, which, as a tool of stabilizing the learning procedure, caches history transitions within a prefixed time window. To handle the sparse-reward problem, DDPGfD puts demonstrations into a separate replay memory $\mathcal{D}^E$ and keeps them throughout the entire learning process. Compared to DDPG, DDPGfD updates the actor and the critic by sampling data from both $\mathcal{D}$ and $\mathcal{D}^E$. The objective for optimizing Q_ω becomes

$$\begin{aligned} J_Q = {} & \mathbb{E}_{\mathcal{D}}\left[(R_t(n) - Q_\omega(s_t, a_t))^2\right] \\ & + \alpha \mathbb{E}_{\mathcal{D}^E}\left[(R_t(n) - Q_\omega(s_t, a_t))^2\right] \end{aligned} \tag{15}$$

where α denotes the ratio of samples from $\mathcal{D}$ and $\mathcal{D}^E$.

POfD: In contrast to DDPGfD that places demonstrations into a replay memory, POfD [30] leverages demonstrations through enforcing occupancy measure (also known as state distribution) matching between the learned policy and demonstrations. This gives a surrogate learning objective

$$\begin{aligned} \min_\theta \max_\nu & -\mathbb{E}_\pi\left[r'(s, a)\right] - \lambda_2 H(\pi) \\ & + \lambda_1 \mathbb{E}_{\pi_E}\left[\log(1 - D_\nu(s, a))\right] \end{aligned} \tag{16}$$

where π_E is the policy generating demonstrations, $D_\nu(s, a)$ is a discriminator judging whether a state–action pair (s, a) is from demonstrations, $r'(s, a) = r(s, a) - \lambda_1 \log(D_\nu(s, a))$ is a virtual reshaped reward function, which augments the sparse reward $r(s, a)$ with demonstration information, $H(\pi)$ is the policy entropy, and λ_1 and λ_2 are two hyperparameters balancing the strength of different parts of the loss function. The above objective is similar with generative-adversarial networks (GANs) [35] and is optimized by alternately updating policy parameters θ and discriminator parameters ν. First, the discriminator is optimized using demonstration data as well as trajectories given by the current policy $\pi(a|s)$. The gradient is computed by

$$\mathbb{E}_\pi\left[\nabla_\nu \log[D_\nu(s, a)]\right] + \mathbb{E}_{\pi_E}\left[\log(1 - D_\nu(s, a))\right]. \tag{17}$$

Given the virtual reshaped reward, the policy parameters θ can be optimized by standard DRL methods, such as A3C [33] and TRPO [36].

B. DRL With Nonexpert Helpers

LwH assumes that the learning agent can have access to a prior control policy $\pi_h(a|s)$ that may be of poor performance. The prior policy can help the agent in achieving goals with some probability and is used for exploring the state space and reshaping the learning objective.

First, the prior policy plays the role of guiding the agent in efficiently exploring the state space, which is achieved by

$$\pi_b(a|s) \propto \pi(a|s) \cdot \pi_h(a|s). \tag{18}$$

where $\pi_b(a|s)$ refers to the behavior policy that is used by the agent for environment interaction and $\pi(a|s)$ is the policy learned by the agent (see Appendix A). Equation (18) encodes that the behavior policy π_b explores the state space by synthesizing information from two independent sources: one from the prior policy π_h and the other from the learned policy π. With such a relationship, it is apparent that the behavior policy is mainly controlled by the source policy with lower variance. If the variance of the learned policy is smaller than that of the prior policy, the behavior policy is mainly controlled by the learned policy. On the contrary, if the variance of the prior policy is smaller, the behavior policy is mainly determined by the prior policy. As a consequence, though the learned policy is similar to a uniform distribution (because it is randomly initialized) in the early training stage, the behavior policy can still perform efficient exploration of the state space since the prior policy has a lower variance.

The prior policy also plays the role of assisting the learning agent in achieving goals. We see that if the behavior policy $\pi_b(a|s)$ is used for environment interaction, importance sampling is needed for an unbiased estimate of the stochastic gradient in (5) of the original DRL objective in (1), since the expectation in (5) is with respect to the state distribution $d_\pi(s)$ induced by the learned policy $\pi(a|s)$ instead of the behavior policy $\pi_b(a|s)$ and the expectation in (6) is also with respect to the learned policy. While in the early learning stage, the behavior policy may significantly deviate from the learned policy, the variance of the importance sampling ratio could be tremendously large and thus might hinder the agent from learning useful policies. In light of this, we propose to use $\eta(\pi_b)$ as the learning objective to avoid using importance sampling. Changing the learning objective from $\eta(\pi)$ to $\eta(\pi_b)$ indicates that we require the learned policy together with the prior policy instead of only the learned policy to achieve goals.

Given the behavior policy is computed by (18), the policy gradient of $\eta(\pi_b)$ is given by

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

where $d_{\pi_b}(s)$ denotes the state distribution induced by the behavior policy $\pi_b(a|s)$, and $A_{\pi_b}(s,a)$ is the corresponding advantage function (see Appendix B). Apparently, data [i.e., state transition tuples (s_t, a_t, s_{t+1}, r_t)] collected by the behavior policy can be directly utilized to estimate the gradient in (19) without importance sampling, since the expectation in (19) is exactly with respect to the state distribution $d_{\pi_b}(s)$ induced by the behavior policy $\pi_b(a|s)$ instead of the state distribution $d_\pi(s)$ induced by the learned policy $\pi(a|s)$. Additionally, the advantage function $A_{\pi_b}(s,a)$ is also with respect to the behavior policy π_b rather than the learned policy π.

Compared to (5) that uses the action–value function $Q_\pi(s,a)$ of the learned policy to guide the policy learning, (19) uses the advantage function $A_{\pi_b}(s,a)$ of the behavior policy. In this respect, the learned policy can leverage the prior policy to improve its performance (since the behavior policy is constructed by both the learned policy and the prior policy) even though the environment gives sparse feedback.

As the ultimate goal is to learn a policy that can achieve goals independently, we need to bridge the gap between $\eta(\pi_b)$ and $\eta(\pi)$. The two objectives hold the following relationship (in fact, it holds true for any two policies [36]):

$$\eta(\pi_b) = \eta(\pi) + \mathbb{E}_{s_{t\geq 0}, a_{t\geq 0}, \ldots, \sim \pi_b}\left[\sum_{t=0}^{\infty} \gamma^t A_\pi(s,a)\right]. \quad (20)$$

By (4), the second term in the left-hand side of (20) would be equal to zero if the behavior policy π_b approaches the learned policy π, which gives

$$\lim_{\pi_b \to \pi} \eta(\pi_b) = \eta(\pi). \quad (21)$$

In view of this, we propose to reduce the influence of the prior policy on the behavior policy as learning progresses (which would be detailed in Section IV-A). This is equivalent to gradually shifting the learning objective from $\eta(\pi_b)$ to $\eta(\pi)$, or in other words, we require the learned policy to achieve goals together with the prior policy in the early learning stage, and as learning progresses, we hope the learned policy can achieve goals on its own.

Although our proposed LwH is compatible with most DRL algorithms handling stochastic policies (e.g., A3C [33], PPO [37], and TRPO [36]), we implement it based on A3C in this article. The pseudocode of the algorithm is demonstrated in Algorithm 1.

Algorithm 1 LwH—Pseudocode for Each Thread

Assuming parameters of the policy and the value nets π_θ and V_ω and the counter $T = 0$ are shared across different threads; Assuming the tread ID is $j \in [1, \cdots, J]$
Initialize thread-specific step counter $t_j \leftarrow 1$
repeat
  Synchronize thread-specific parameters $\theta_j \leftarrow \theta$ and $\omega_j \leftarrow \omega$
  Set $t_{\text{init}=t}$ and observe state s_t
  repeat
    Reduce the influence of the prior policy (by (25))
    Compute the behavior policy $\pi_b(a|s)$ (by (22))
    Execute the action $a_t \sim \pi_b(\cdot|s)$
    Receive reward r_t and new state s_{t+1}
    Update step counters: $t \leftarrow t+1$, $T \leftarrow T+1$
  until terminal s_t or $t - t_{\text{init}} == t_{\max}$
  for $i = t_{\text{init}}, t-1$ **do**
    Estimate the expected return $\hat{R}_i$ of π_b in each state s_i using generalized advantage estimation [38]
  end for
  Compute the gradient with respect to θ by
$$\Delta\theta_j = \sum_{i=t_{\text{init}}}^{t-1}\left[\nabla_{\theta_j} \log \pi_{\theta_j}(a_i|s_i)(\hat{R}_i - V_{\omega_j}(s_i))\right]$$
  Compute the gradient with respect to ω by
$$\Delta\omega_j = \sum_{i=t_{\text{init}}}^{t-1}\left[\nabla_{\omega_j} \partial (R_i - V_{\omega_j}(s_i))^2 / \partial \omega_j\right]$$
  Asynchronously update θ and ω using Δ_{θ_j} and Δ_{ω_j}
until $T \geq T_{\max}$

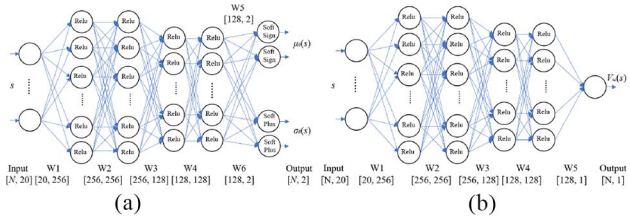


Fig. 2. Network structures of the policy $\pi_\theta(\cdot|s)$ and the value function $V_\omega(s)$. The policy network takes a state as input and outputs the mean and the standard deviation of the policy, and the value network takes a state as input and outputs its value. (a) Policy network. (b) Value network.

IV. Experiments

A. Virtual Environment and Experimental Setting

We construct a virtual environment by the simulation to implement UAV navigation in large-scale complex environments as well as evaluate the performance of our proposed method. The virtual environment covers around 4 km^2. Buildings are represented as cylinders with random heights and the UAV is represented as a sphere. As a stochastic environment, each time a navigation task ends, the environment map, the departure position, and the destination are randomly reinitialized again. The maximum speed of the UAV is restricted to 50 m/s. Details of the virtual environment can be found in Appendix C. Source codes for the virtual environment as well as our proposed algorithm LwH has been open sourced at https://github.com/DennisWangCW/LwH and https://github.com/DennisWangCW/gym-uav, respectively.

The implementation of LwH is based on A3C [33]. The hyperparameters of LwH are summarized in Table II in Appendix D. The policy and the value function are approximated by two neural networks $\pi(\cdot|s)$ and $V_\omega(s)$ with parameters θ and ω, as illustrated in Fig. 2. As most existing methods do, the stochastic policy is approximated by Gaussian distributions. The policy network outputs the mean and the standard deviation, i.e., $\mu_\theta(s)$ and $\sigma_\theta(s)$. Given the relationship demonstrated in (18), if we denote the mean and the standard deviation of the prior policy as $\mu_h(s)$ and $\sigma_h(s)$, then

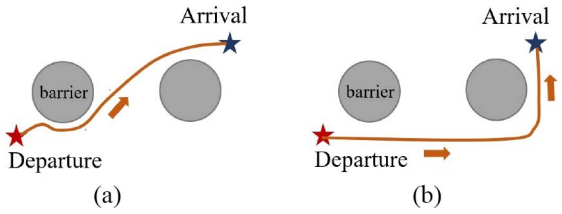


Fig. 3. Schematic view of the two prior control policies. (a) SenAvo-Pri. (b) Naive-Pri.

the behavior policy can be computed by

$$\pi_b(a|s) = \frac{1}{\sqrt{2\pi}\sigma_b(s)} e^{-\frac{(a-\mu_b(s))^2}{2\sigma_b^2(s)}} \tag{22}$$

where

$$\sigma_b^2(s) = \frac{\sigma_h^2(s)\sigma_\theta^2(s)}{\sigma_h^2(s) + \sigma_\theta^2(s)} \tag{23}$$

and

$$\mu_b(s) = \frac{\sigma_\theta^2(s)\mu_h(s) + \sigma_h^2(s)\mu_\theta(s)}{\sigma_\theta^2(s) + \sigma_h^2(s)}. \tag{24}$$

In this article, the influence of the prior policy is reduced by linearly increasing the standard deviation of the prior policy as learning progresses, i.e.,

$$\sigma_h^{t+1}(s) = \sigma_h^t(s) \times \beta T \tag{25}$$

where T refers to the global time step and β is the decay rate.

Two prior policies are used to evaluate the performance of LwH. The first prior policy follows a sensing-and-avoidance strategy while the second follows a relatively naive strategy. For ease of reference, we denote the two prior policies as SenAvo-Pri and Naive-Pri, respectively (schematic views of the two prior policies appear in Fig. 3). For each prior policy, we restrict the expected speed to 2.5 m/s. The two prior policies are in poor performance since: 1) the speed is low and cannot be dynamically adjusted according to the structure of the local environment and 2) the direction is controlled by simple handcrafted policies and cannot be generalized to complex environments. Codes for the two prior policies have been packed into the source code.

B. Comparison With the Baselines

Below, we evaluate the effectiveness of LwH in the virtual UAV navigation environment with sparse rewards. The standard deviation of the two prior policies is set to $\sigma_h(s) = 0.4$ and the prior decay rate is set to $\beta = 5e^{-5}$. For a better understanding of LwH, we first evaluate the performance of the two prior policies by running each for 100 navigation tasks. The results show that the success rates (of navigation tasks) of the two prior policies are only 0.36 and 0.42, respectively, suggesting that they are indeed in poor performance. For purposes of comparisons, we also reimplement the standard A3C algorithm, DDPGfD, and POfD in the virtual environment. As DDPGfD and POfD require demonstrations instead of prior policies, we generate $1e^6$ transition tuples (around 1500 episodes) by running the two prior

TABLE I
SUCCESS RATES OF POLICIES GIVEN BY DIFFERENT ALGORITHMS

Prior \ Algo.	LwH (Ours)	POfD	DDPGfD	A3C
SenAvo-Pri	**0.97**	0.76 (max)	0.00	0.00
Naive-Pri	**0.96**	0.41 (max)	0.00	

TABLE II
HYPERPARAMETERS OF LWH

Hyperparameters	Value	Hyperparameters	Value
Learning rate	$1e^{-4}$	Training steps ($T_{\max}$)	$1.4e^7$
Batch size ($t_{\max}$)	20	Prior decay rate (β)	$5e^{-5}$
Num. threads (J)	16	Prior policy variance $\sigma_h(s)$	0.4
Discount factor (γ)	0.99		

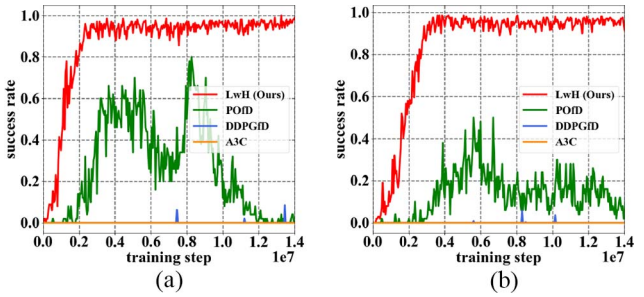


Fig. 4. Mean success rate versus the training step curves for different algorithms. The success rate is obtained by evaluating each learned navigation policy over 100 randomly generated navigation tasks. The mean success rate is estimated by taking the average of the success rates of three independent trials of each algorithm with different random seeds. (a) Trained with SenAvo-Pri. (b) Trained with Naive-Pri.

policies ([30] claims that POfD can learn with only a single demonstrated episode). The success rates of navigation tasks are given in Table I. We see that no matter which prior policy is employed, our proposed LwH can achieve a success rate of more than 0.96. In contrast, POfD gives a success rate of 0.76 when trained with demonstrations given by SenseAvo-Pri and 0.41 when trained with demonstrations given by Naive-Pri. DDPGfD and A3C just fail to complete any navigation tasks. The learning curves are plotted in Fig. 4. Our LwH again converges much faster than the baseline algorithms in all cases. Interestingly, POfD does learn to navigate to some extent in the early training stage, but as learning progresses, the performance degrades unexpectedly. In contrast, both DDPGfD and A3C fail to bring any performance improvement to the learned policy during the entire training process.

We believe the success of LwH contributes to three aspects: 1) the structure of the behavior policy enables the agent to explore the state space by synthesizing information from different sources; 2) the prior policy helps the agent efficiently explore the state space especially in the early training stage, which enables the agent to effectively update policy parameters even though the gradient is biased [in terms of the ultimate learning objective (1)]; and 3) the mechanism of reducing the influence of the prior policy as learning progresses set dynamic learning objectives with increasing difficulty for the agent in different training stages. It eliminates the situation that the learning objective is too difficult while the learning agent is

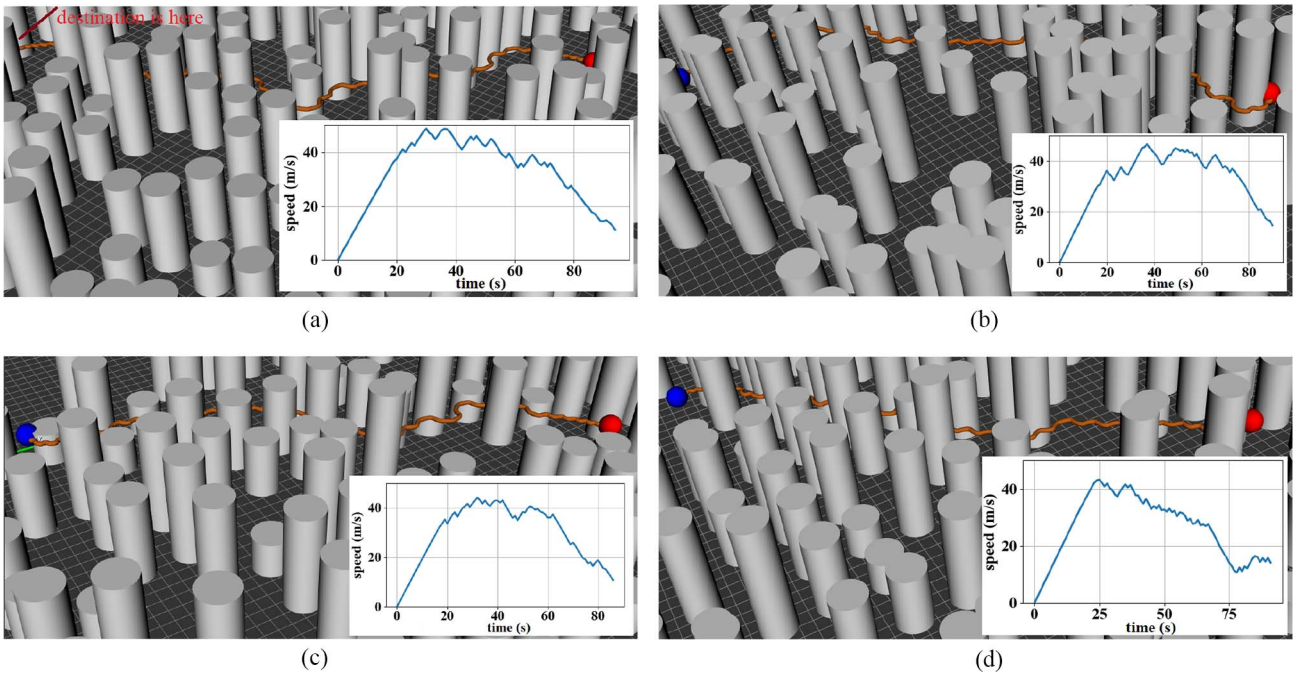


Fig. 5. Navigation trajectories and the corresponding speed versus time curves. Red and blue spheres denote departure places and destinations. The navigation trajectory is denoted as a sequence of orange spheres. Building lower than the UAV's flight altitude is not shown in the figure. (a) Trajectory I (LwH trained with SenAvo-Pri). (b) Trajectory II (LwH trained with SenAvo-Pri). (c) Trajectory III (LwH trained with Naive-Pri). (d) Trajectory IV (LwH trained with Naive-Pri).

too weak to achieve such goals and makes it possible for the agent to derive useful policies step by step.

C. Evaluating the Learned Policy

Below, we evaluate the performance of the learned policy given by our LwH trained with different prior policies in detail. The trajectories of several navigation tasks along with the corresponding speed versus time curves appear in Fig. 5 and videos can be found at https://youtu.be/m575rkcacio. We can observe from Fig. 5 that: 1) the policy can dynamically adjust the UAV's speed and direction according to the structure of the local environment as well as its distance to the destination, which satisfies our expectation that the UAV can learn to autonomously control its speed and direction to adapt to the complex environment and in the meantime perform navigation tasks and 2) the learned policy significantly deviates from the prior policy. For example, the learned policies can adjust the UAV's speed within the range of [0 m/s, 50 m/s] while the expected speed is prefixed to 2.5 m/s if the prior policy is employed. Besides, when the Naive-Pri policy is used to guide the learning procedure, LwH yields a policy that can control the UAV's direction based on its relative position to the destination as well as the structure of the local environment rather than simply mimic the Naive-Pri policy. In summary, LwH is able to learn human-like navigation behaviors by leveraging prior policies to boost self-learning in environments with sparse rewards and avoids overfitting to the prior policy.

D. Robustness to Hyperparameter Configurations

The performance of LwH can be affected by the standard deviation of the prior policy $\sigma_h(s)$ and its decay rate β. To see whether the LwH is robust to these hyperparameters, we implement LwH with a range of hyperparameter configurations. Specifically, $\beta \in [5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}]$ and $\sigma_h(s) \in [0.2, 0.3, 0.4, 0.5]$. Fig. 6 illustrates the success rates of policies given by LwH with different hyperparameter configurations. As we can see, though LwH does not converge under several hyperparameter configurations, in general, it shows robustness to the two hyperparameters. We also notice that if the combination of the standard deviation of the prior policy and its decay rate yields fast decay of the influence of the prior policy, LwH converges much slower (especially when the Naive-Pri is employed). That is because the learning agent can rarely benefit from the prior policy in efficiently exploring the state space. Besides, fast decay of the influence of the prior policy also makes the dynamic learning objective too difficult for the agent to achieve. In contrast, we see that if the combination of the two hyperparameters yields too slow decay of the influence of the prior policy, the agent may fail to learn useful behaviors. The reason is that the objective of LwH is to learn a policy that can complete tasks under the assistant of the prior policy. If the influence of the prior policy decays too slowly during the training phase, the learned policy will have little incentive to learn useful behaviors on its own.

E. Ablation Study: Learning Without Prior Decay

We have seen in Section IV-B that without the assistance of the prior policy, the agent (i.e., the A3C baseline) cannot derive any useful navigation behaviors. Below, we investigate another extreme case that implementing LwH without reducing the influence of the prior policy. In this case, LwH implemented

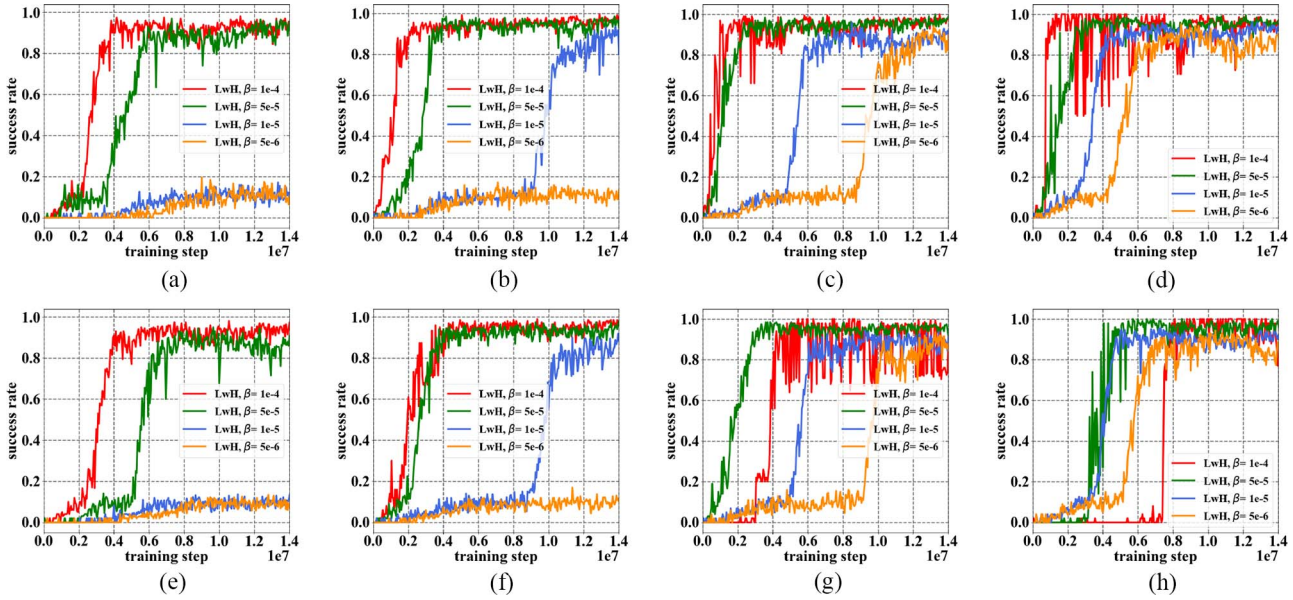


Fig. 6. Mean success rate versus the training step curves for our proposed algorithm LwH with different configurations of the standard deviation of the prior policy $\sigma_h(s)$ and the prior decay rate β. The success rate is obtained in the same way as those in Fig. 4. (a) SenAvo-Pri, $\sigma_h(s) = 0.2$. (b) SenAvo-Pri, $\sigma_h(s) = 0.3$. (c) SenAvo-Pri, $\sigma_h(s) = 0.4$. (d) SenAvo-Pri, $\sigma_h(s) = 0.5$. (e) Naive-Pri, $\sigma_h(s) = 0.2$. (f) Naive-Pri, $\sigma_h(s) = 0.3$. (g) Naive-Pri, $\sigma_h(s) = 0.4$. (h) Naive-Pri, $\sigma_h(s) = 0.5$.

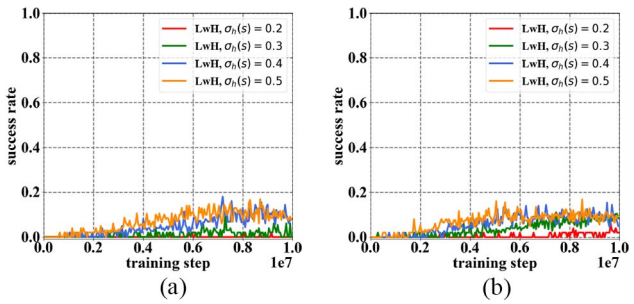


Fig. 7. Mean success rate versus the training step curves for the algorithm LwH implemented without prior decay. The success rate is obtained in the same way as those in Fig. 4. (a) SenAvo-Pri, no prior decay. (b) Naive-Pri, no prior decay.

with the two prior policies with different standard deviation configurations $\sigma_h(s) \in [0.2, 0.3, 0.4, 0.5]$. The simulation results are demonstrated in Fig. 7. As we see, no matter which prior policy is employed, the learned policy only achieves a maximum success rate of around 0.15, significantly lower than those in the standard setting in Section IV-B. Additionally, similar to the phenomenon observed in Section IV-D, we see that the stronger the influence of the prior policy on the behavior policy, the lower the final performance of the learned policy. The reason is that if the influence of the prior policy is not reduced, the learning objective cannot shift from $\eta(\pi_b)$ to $\eta(\pi)$, the ultimate goal of DRL. As a consequence, the agent only learns to achieve goals under the assistance of the prior policy instead of on its own. This demonstrates that the mechanism of reducing the influence of the prior policy as learning progresses is key to the success of LwH.

F. Learning in Environments With Nonsparse Rewards

To have a more comprehensive evaluation of LwH and to investigate whether it can yield policies comparable to those

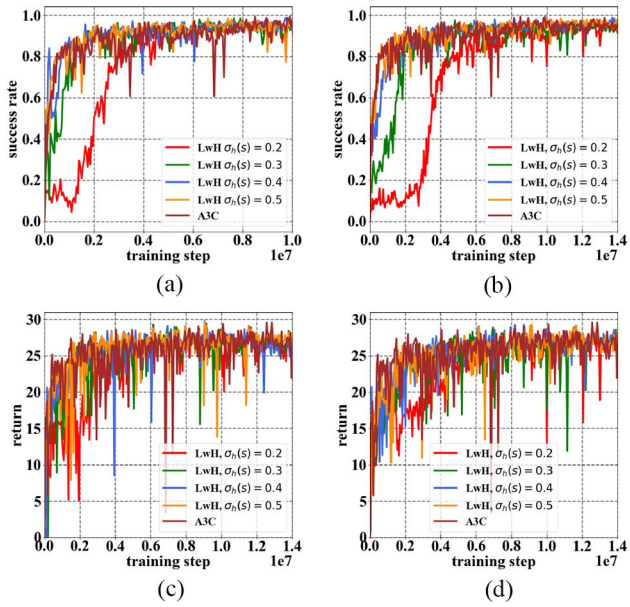


Fig. 8. Mean success rate and expected return versus the training step curves for the algorithm LwH and the baseline algorithm A3C implemented in the environment with dense rewards. (a) and (b) Success rates. (c) and (d) Corresponding expected returns. (a) Success rate, SenAvo-Pri. (b) Success rate, Naive-Pri. (c) Expected return, SenAvo-Pri. (d) Expected return, Naive-Pri.

learned from the environment with dense rewards, we reimplement it in the environment that gives dense feedback (for a meaningful comparison, we have made tremendous efforts on fine-tuning the nonsparse reward function so that it can yield good control policies; details of the nonsparse reward scheme see Appendix E). Besides, we also reimplement A3C in the environment as a baseline. The simulation results appear in Fig. 8. We can see that in most cases, the learning dynamics of LwH in the environment with dense rewards is similar to that

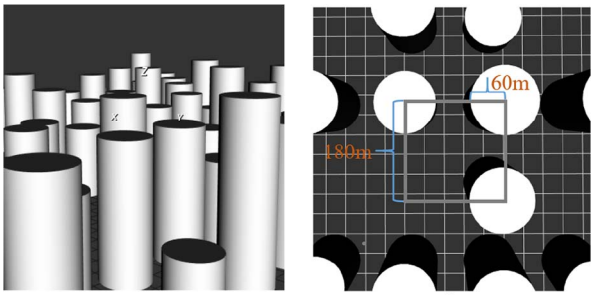


Fig. 9. Overview of the (a) virtual large-scale complex environment and (b) building layout in the environment.

of the baseline algorithm A3C. But if the influence of the prior policy is too strong, LwH converges relatively slower than A3C. That is because in environments with dense rewards, the rewards already provide the agent with rich information in deriving optimal control policies. If the prior policy is employed, it may confuse the agent since the prior policy is suboptimal. We can also notice from the learning curves that, in most cases, LwH is more stable than A3C. We think that contributes to the fact that LwH can set dynamic learning objectives with increasing difficulties for the agent. When compared with the results in the sparse-reward environment (as shown Fig. 4), LwH shows even more stable learning dynamics and yields policies with high success rates comparable to those given by A3C implemented in the dense-reward environment.

In summary, our LwH outperforms the state-of-the-art baselines by a large margin in environments with sparse rewards in terms of convergence speed, success rate, and final performance, and is robust to hyperparameter configurations. Additionally, it exhibits more stable learning dynamics and gives impressive results comparable to those of the baseline algorithm implemented in the environment with dense rewards.

V. Conclusion

We formulate the problem of autonomous navigation of UAVs in large-scale complex environments as an MDP with sparse rewards and propose an efficient learning algorithm LwH, which guides the agent in exploring informative states especially in the early training stage and enables the agent to learn to achieve goals with different difficulty in the different training stage. Specifically, a prior control policy with potentially poor performance is assumed available to the agent. A behavior policy, which is constructed by synthesizing information from both the currently learned policy and the prior policy, is used for environment interaction. As learning progresses, the prior policy exerts less influence on the learned policy by linearly decaying the standard deviation of the prior policy, which is equivalent to setting dynamic learning objectives with increasing difficulty for the agent as learning progresses. Extensive experimental results demonstrate that LwH is efficient in learning high-performance control policies for UAV navigation in large-scale complex environments, robust to a range of hyperparameter configurations and significantly outperforms several state-of-the-art methods. Nonetheless, there are still some works to do in the future. For example, the influence of the prior policy is reduced by linearly decreasing its standard deviation. As a result, we must fine-tune the initial value of the standard deviation and the decay rate, though experimental results demonstrate that LwH is robust to them to some extent. We believe a more principled way is to let the learning algorithm itself determine the extent of influence of the prior policy in different training stages. This may be achieved by automated machine learning methods [39] and we leave it as our future work.

Appendix A
Derivation of (18)

To obtain (18), we first introduce a lemma [40].

Lemma 1: Consider that we have two sets of information D_A and D_B, and we use each set to independently estimate some parameter C. Denote the two estimators as $P(C|D_A)$ and $P(C|D_B)$, then the best estimate of C based on the two sets is given by

$$P(C|D_A, D_B) \propto \frac{P(C|D_A)P(C|D_B)}{P(C)} \tag{26}$$

where $P(C)$ denotes the prior distribution of C.

The construction of the behavior policy (18) is inspired by Lemma 1. By rewriting $\pi_h(a|s)$ as $\pi_h(a|s, H)$ and $\pi_\theta(a|s)$ as $\pi_\theta(a|s, L)$, where $\pi_\theta(a|s, h)$ denote the human estimator that estimates the action in state s and $\pi_\theta(a|s, l)$ is the estimator learned by the agent, the final estimate of the action a at state s is given by

$$\pi_b(a|s, L, H) \propto \frac{\pi_\theta(a|s, L)\pi_h(a|s, H)}{\pi_p(a|s)} \tag{27}$$

where $\pi_p(a|s)$ is the prior action distribution. Since, generally, we have no prior information of the action at each state, we assume that $\pi_p(a|s)$ complies with a uniform distribution. Then, (27) can be reformulated as

$$\pi_b(a|s, L, H) = \frac{\pi_\theta(a|s, L)\pi_h(a|s, H)}{Z_\theta(s)} \tag{28}$$

where $Z_\theta(s)$ is a normalization constant independent of the action a. To keep consistency, we can rewrite (28) as

$$\pi_b(a|s) = \frac{\pi_\theta(a|s)\pi_h(a|s)}{Z_\theta(s)} \propto \pi_\theta(a|s)\pi_h(a|s). \tag{29}$$

Appendix B
Derivation of (19)

To obtain (19), we first introduce an equality (which has been proved true by [32] and [41]) that

$$\begin{aligned}\nabla_\theta \eta(\pi_b) &= \mathbb{E}_{s\sim d_{\pi_b}(s), s\sim\pi_b}\left[\nabla_\theta \log \pi_b(a|s) Q_{\pi_b}(s, a)\right] \\ &= \mathbb{E}_{s\sim d_{\pi_b}(s), s\sim\pi_b}\left[\nabla_\theta \log \pi_b(a|s) A_{\pi_b}(s, a)\right].\end{aligned} \tag{30}$$

The relation holds true since we have

$$Q_{\pi_b}(s, a) = A_{\pi_b}(s, a) + V_{\pi_b}(s) \tag{31}$$

and

$$\begin{aligned}&\mathbb{E}_{a\sim\pi_b}\big[\nabla_\theta \log \pi_b(a|s) V_{\pi_b}(s)\big]\\&= V_{\pi_b}(s)\mathbb{E}_{a\sim\pi_b}\big[\log \pi_b(a|s)\big]\\&= V_{\pi_b}(s)\cdot 0 = 0.\end{aligned} \tag{32}$$

By substituting (18) into the right-hand side of (30), the gradient of η_{π_b} can be rewritten as

$$\begin{aligned}&\nabla_\theta \eta(\pi_b)\\&= \mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log \pi_b(a|s) A_{\pi_b}(s,a)\big]\\&= \mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\left[\nabla_\theta \log\left(\frac{1}{Z_\theta(s)}\pi_\theta(a|s)\pi_h(a|s)\right)A_{\pi_b}(s,a)\right]\\&= \mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log \pi_\theta(a|s) A_{\pi_b}(s,a)\big]\\&\quad+\ \mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log \pi_h(a|s) A_{\pi_b}(s,a)\big]\\&\quad-\ \mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log Z_\theta(s) A_{\pi_b}(s,a)\big].\end{aligned} \tag{33}$$

It is clear that

$$\mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log \pi_h(a|s) A_{\pi_b}(s,a) = 0\big]. \tag{34}$$

Besides, since the normalization term $Z_\theta(s)$ is independent of the action a, we have

$$\begin{aligned}&\mathbb{E}_{s\sim d_{\pi_b}(s),a\sim\pi_b}\big[\nabla_\theta \log Z_\theta(s) A_{\pi_b}(s,a)\big]\\&= \mathbb{E}_{s\sim d_{\pi_b}(s)}\left[\nabla_\theta \log Z_\theta(s)\sum_a \pi_b(a|s) A_{\pi_b}(s,a)\right]\\&= 0.\end{aligned} \tag{35}$$

Substituting (34) and (35) into (33) gives us (19), the final result.

APPENDIX C
THE VIRTUAL ENVIRONMENT

We build a simulator to implement UAV navigation in large-scale complex environments. For simplicity, we omit physical constraints on the UAV dynamics in the real situation and assume that control commands can take effect in no time. Buildings (obstacles) in the environment are abstracted as cylinders and the UAV is abstracted as a sphere. Overview of the virtual environment and the layout of buildings are demonstrated in Fig. 9. Heights of the cylinders comply with a discrete uniform distribution $\text{unif}\{30\ \text{m} + i \times 23\ \text{m}\}_{i=0}^{10}$. To ensure that the reward is sparse, the minimum distance between the departure position and the destination is larger than 200 m. Besides, a navigation task is completed if the distance between the UAV's current position and the destination is less than 10 m [i.e., $r_d = 10$ m in (8)]. The UAV's flight altitude is set to 100 m and the maximum speed is 50 m/s. UAV's observation of the environment is achieved by range finders, as depicted in Fig. 1. The simulator supports the flexible setup of the number and the direction of range finders, the layout of buildings, and the flight altitude of the UAV. The simulator is built upon the gym environment [42] and thus compatible with most DRL algorithms. Therefore, anyone who wants to test their RL algorithms handling the sparse-reward challenge, our simulator can serve as an alternative.

APPENDIX D
HYPERPARAMETERS OF LWH

The hyperparameters of LwH are shown in Table II.

APPENDIX E
NONSPARSE REWARD SCHEME FOR UAV NAVIGATION

The reward function evaluates *how good* is it when executing a certain action at a certain state. So we design the nonsparse reward function as follows. First, the UAV would be rewarded if it approaches the destination and be penalized in reverse, which can be formulated as

$$\begin{aligned}r_d(s_t,a_t) = &\|(x_{t-1},y_{t-1})-(x_d,y_d)\|_2\\&-\|(x_t,y_t)-(x_d,y_d)\|_2.\end{aligned} \tag{36}$$

The UAV would be penalized if it is too close to any obstacles. The penalty is formulated as

$$r_o(s_t,a_t) = \max\big(\min\big(d_1^t,\ldots,d_{16}^t\big)-d_s,0\big) \tag{37}$$

where d_s denotes the minimum safety distance between the UAV and any obstacles. We set its value to 10.0 m in this article. The UAV would also get a reward $s_{sp}(s_t,a_t)$ [formulated as (8)] if it arrives at the destination. Finally, to encourage the UAV to move to the destination as soon as possible, it would get a constant penalty $r_s(s_t,a_t) = -1$. The overall nonsparse reward function can be formulated as

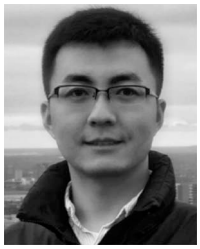
$$\begin{aligned}r_{nsp}(s_t,a_t) = &\beta_d r_d(s_t,a_t)+\beta_o r_o(s_t,s_t)\\&+\beta_s r_{sp}(s_t,a_t)+\beta_c r_c(s_t,a_t)\end{aligned} \tag{38}$$

where β_d, β_o, β_s, and β_c are four positive hyperparameters that take the value 1.0, 10.0, 5.0, and 1.0, respectively. (We have fine-tuned the four hyperparameters so that the baseline algorithm yields the best results.)

REFERENCES

[1] E. T. Bokeno *et al.*, "Package delivery by means of an automated multicopter UAS/UAV dispatched from a conventional delivery vehicle," U.S. Patent 9 915 956, Mar. 13, 2018.
[2] P. Grippa, D. A. Behrens, C. Bettstetter, and F. Wall, "Job selection in a network of autonomous UAVs for delivery of goods," in *Robotics: Science and Systems (RSS)*. Cambridge, MA, USA: MIT Press, 2017.
[3] T. Lagkas, V. Argyriou, S. Bibi, and P. Sarigiannidis, "UAV IoT framework views and challenges: Towards protecting drones as 'things'," *Sensors*, vol. 18, no. 11, pp. 4015–4025, 2018.
[4] B. Li, Z. Fei, and Y. Zhang, "UAV communications for 5G and beyond: Recent advances and future trends," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2241–2263, Apr. 2019.
[5] T. Han, X. Ge, L. Wang, K. S. Kwak, Y. Han, and X. Liu, "5G converged cell-less communications in smart cities," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 44–50, Mar. 2017.
[6] J. Wang, C. Jiang, K. Zhang, X. Hou, Y. Ren, and Y. Qian, "Distributed Q-learning aided heterogeneous network association for energy-efficient IIoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 4, pp. 2756–2764, Apr. 2020.
[7] P. Liu, C. Wang, T. Fu, and Y. Ding, "Exploiting opportunistic coding in throwbox-based multicast in vehicular delay tolerant networks," *IEEE Access*, vol. 7, pp. 48459–48469, 2019.
[8] P. Liu, Y. Ding, T. Fu, X. Shen, and J. Li, "On multi-copy forwarding protocols for large data chunk dissemination in vehicular sensor networks," *EURASIP J. Wireless Commun. Netw.*, vol. 130, no. 1, pp. 1–14, 2018.
[9] P. Liu, Y. Ding, and T. Fu, "Optimal throwboxes assignment for big data multicast in VDTNs," *Wireless Netw.*, to be published.

[10] J. Wang, C. Jiang, Z. Wei, C. Pan, H. Zhang, and Y. Ren, "Joint UAV hovering altitude and power control for space-air-ground IoT networks," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 1741–1753, Apr. 2019.
[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 (IVCNZ)*, 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 DGON Inertial Sensors Syst. (ISS)*, Karlsruhe, Germany, 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, nos. 1–4, pp. 513–533, 2014.
[14] 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. Autom. (ICRA)*, Hong Kong, 2014, pp. 6638–6643.
[15] K. Chee and Z. Zhong, "Control, navigation and collision avoidance for an unmanned aerial vehicle," *Sensors Actuators A, Phys.*, vol. 190, pp. 66–76, Feb. 2013.
[16] 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.
[17] 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 Autom. (ICCA)*, Kathmandu, Nepal, 2016, pp. 984–989.
[18] N. Imanberdiyev, C. Fu, E. Kayacan, and I.-M. Chen, "Autonomous navigation of UAV by using real-time model-based reinforcement learning," in *Proc. Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Phuket, Thailand, 2016, pp. 1–6.
[19] 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, Jun. 2017.
[20] S. Ross *et al.*, "Learning monocular reactive UAV control in cluttered natural environments," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Karlsruhe, Germany, 2013, pp. 1765–1772.
[21] 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.
[22] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros, "Large-scale study of curiosity-driven learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, 2019, pp. 1–15.
[23] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Honolulu, HI, USA, 2017, pp. 16–17.
[24] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "VIME: Variational information maximizing exploration," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Barcelona, Spain, 2016, pp. 1109–1117.
[25] 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. (ICML)*, vol. 99. Bled, Slovenia, 1999, pp. 278–287.
[26] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, vol. 1. Stanford, CA, USA, 2000, pp. 663–670.
[27] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, Barcelona, Spain, 2016, pp. 4565–4573.
[28] T. Hester *et al.*, "Deep *Q*-learning from demonstrations," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3223–3230.
[29] M. Večerík *et al.*, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," 2017. [Online]. Available: arXiv:1707.08817.
[30] B. Kang, Z. Jie, and J. Feng, "Policy optimization with demonstrations," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Stockholm, Sweden, 2018, pp. 2474–2483.
[31] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *J. Artif. Intell. Res.*, vol. 47, no. 1, pp. 253–279, 2013.
[32] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Hoboken, NJ, USA: MIT Press, 2018.
[33] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn. (ICML)*, New York, NY, USA, 2016, pp. 1928–1937.
[34] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, San Juan, Puerto Rico, 2016, pp. 1–142.
[35] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2014, pp. 2672–2680.
[36] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *Proc. Int. Conf. Mach. Learn. (ICML)*, Lille, France, 2015, pp. 1889–1897.
[37] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: arXiv:1707.06347.
[38] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," 2015. [Online]. Available: arXiv:1506.02438.
[39] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning-Methods, Systems, Challenges*. Cham, Switzerland: Springer, 2019.
[40] C. A. Bailer-Jones and K. Smith, "Combining probabilities," Max Planck Inst. Astronomy, Heidelberg, Germany, Rep. GAIA-C8-TN-MPIA-CBJ-053, 2011. [Online]. Available: https://www.mpia.de/3432751/probcomb_TN.pdf
[41] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *J. Mach. Learn. Res.*, vol. 5, pp. 1471–1530, Nov. 2004.
[42] G. Brockman *et al.*, "OpenAI gym," 2016. [Online]. Available: arXiv:1606.01540.



Chao Wang (Student Member, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2015, where he is currently pursuing the Ph.D. degree under the supervision of Prof. X. Zhang.

His research interests include signal processing, statistical machine learning, and reinforcement learning.

Jian Wang (Senior Member, IEEE) 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, machine learning, privacy enhancing technology, and signal processing in the encrypted domain and wireless networks.

Jingjing Wang (Member, IEEE) received the B.S. degree (Highest Hons.) in electronic information engineering from the Dalian University of Technology, Dalian, China, in 2014, and the Ph.D. degree (Highest Hons.) in information and communication engineering from Tsinghua University, Beijing, China, in 2019.

From 2017 to 2018, he visited the Next Generation Wireless Group chaired by Prof. L. Hanzo, University of Southampton, Southampton, U.K. He is currently a Postdoctoral Researcher with the Department of Electronic Engineering, Tsinghua University. His research interests include resource allocation and network association, learning theory-aided modeling, analysis and signal processing, as well as information diffusion theory for mobile wireless networks.

Dr. Wang was a recipient of the Best Journal Paper Award of the IEEE ComSoc Technical Committee on Green Communications & Computing in 2018 and the Best Paper Award from IEEE ICC and IWCMC in 2019.

Xudong Zhang (Member, IEEE) 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.