AUTONOMOUS NAVIGATION OF UAV IN LARGE-SCALE UNKNOWN COMPLEX ENVIRONMENT WITH DEEP REINFORCEMENT LEARNING

Chao Wang, Jian Wang, Xudong Zhang, and Xiao Zhang

Department of Electronic Engineering, Tsinghua University, China

ABSTRACT

Unmanned Aerial Vehicles (UAVs) based delivery is thriving. In this paper, we model autonomous navigation of UAV in large-scale unknown complex environment as a discrete-time continuous control problem and solve it using deep reinforcement learning. Without path planning or map construction, our method enables UAVs to navigate from arbitrary departure places to destinations using only sensory information of local environment and GPS signal. We argue the navigation task is a partially observable Markov decision process (POMDP) and extant recurrent deterministic policy gradient algorithm is less efficient. Consequently, we derive a faster policy learning algorithm for POMDP based on actor-critic architecture. To validate our ideas, we simulate five virtual environments and a virtual UAV flying at a fixed altitude with constant speed. Cognition of local environment is achieved by measuring distances from UAV to obstacles in multiple directions. Simulation results demonstrate the effectiveness of our method.

Index Terms— large-scale autonomous navigation, UAV delivery, deep reinforcement learning, partially observable Markov decision process

1. INTRODUCTION

Remote delivery by UAVs, as a burgeoning technique, has attracted many researchers, for it can be used as a last resort in delivery hierarchy [1] and promote the development of smart cities [2]. While this envision is attractive, researchers face two challenges, i.e., autonomous navigation and navigation in large-scale unknown complex environment (e.g., urban areas). Cui [3] and Bachrach [4] employ the well-known SLAM (Simultaneously Localization And Mapping) algorithm [5] to attack these challenges. Its basic idea is incrementally building a map of the unknown environment and using it to simultaneously localize. Nevertheless, we argue that it is unnecessary to learn a 3D map of an unknown environment that stretches for miles and plan paths based on it. For instance, after finishing a delivery task within some area, a UAV may be dispatched to another unknown area. Israelsen [6] and [1] provide another solution concept, sense and avoidance. It empowers UAVs to sense the obstacles on the path to the destination, evade them, and then go back autonomously or manually to the pre-planned trajectory. However, while this technique works in simple environment (e.g., countryside), dense obstacles in complex environment (e.g., urban areas crowded with sky scrapers and signal towers) may tremendously reduce its efficiency, for it must continuously cope with obstructions and put UAVs back to its pre-planned trajectory or re-plan a new trajectory. Zhang [7] proposes an approach that enables robots to retrace routes pre-taught under the control of men based on visual appearance matching. However, it is intractable to pre-teach UAVs to fly from random starting positions to random target positions.

While the aforementioned methods partially solve the navigation problem in large-scale unknown complex environment, most of them include learning a map of the unknown environment, route planning or interaction with human operators. In this paper, we aim to empower UAVs, without constructing local or global maps and path planning, using only GPS signal and sensory information of the local environment, to navigate autonomously and intelligently from arbitrary departure places to arbitrary target positions in large-scale unknown complex environment.

We model the autonomous navigation problem as a discrete-time continuous control problem and employ deep Reinforcement learning (DRL) to solve it. DRL, which applies deep learning to traditional reinforcement learning (RL) with function approximation and provides a solution concept to Markov decision processes (MDP) and partially observable Markov decision processes (POMDP), directly projects sensory inputs into control signals and bridges the gap between cognition of the natural world and control strategy. It is firstly introduced by Mnih [8] to create virtual agents that surpass human-beings in playing multiple electronic games with discrete control. Afterwards, Lillicrap [9] uses two neural networks to approximate actor and critic in actor-critic algorithm [10] based on [11] and designs deep deterministic policy gradient (DDPG) algorithm to solve MDP with continuous control. Heess [12] further extends [9] to POMDP by approximating actor and critic using two recurrent neural networks and develops an algorithm named recurrent deterministic policy gradient (RDPG).

In our work, a UAV regards distances from itself to obstacles in multiple directions, its orientation angle, the

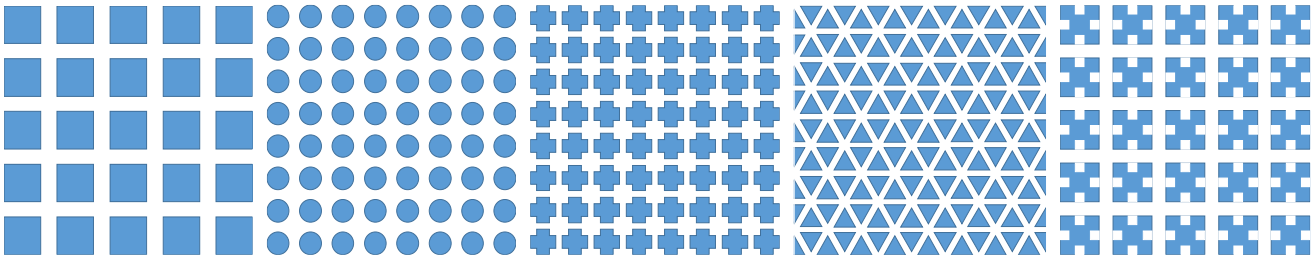


Fig. 1. The five types of virtual environments, each of which covers around one square kilometers

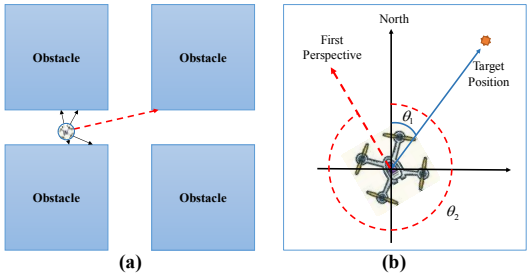


Fig. 2. (a) Range finders sensing obstacles in five different directions. (b) θ_1: the angle between the present position and the target position. θ_2: direction angle of the UAV.

distance and the angle between its present position and the destination (which can be derived from GPS signal) as sensory information. Then it learns to navigate to the target position using DRL. Our proposed method doesn't include learning a local or global map of the unknown complex environment or path planning. Besides, we argue that the navigation problem is a POMDP, and extant RDPG algorithm is not efficient in the framework of learning using memory replay [8]. Consequently, based on actor-critic architecture with function approximation [13], we derive a more efficient policy learning algorithm, i.e., Fast-RDPG, for POMDP with continuous control.

The reminder of this manuscript is organized as follows. Section II elucidates the navigation model, briefly introduces the basic theories of RL and DRL, and derives our Fast-RDPG algorithm. Section III validates our method through simulation. Section IV concludes this paper and envisages our future work.

2. METHOD

2.1. Model description

Autonomously navigate in real environment could be complex. For sake of brevity and without loss of generality, we create a virtual UAV that flies at a fixed altitude with fixed speed, and its control profile only contains turning left or right. Besides, we simulate several types of environments as large-scale unknown complex environment, depicted in Fig. 1. The goal of the virtual UAV is to fly from arbitrary starting positions to arbitrary destinations using GPS signal and its perception of the local environment.

To model the navigation problem as a control problem and employ DRL to solve it, we first describe the profile of sensory information (or state) of the virtual UAV. Supposing the UAV is mounted with five virtual range finders, illustrated in Fig. 2 (a), we regard the distances from the UAV to obstacles in the five directions as the UAV's sensory information of the environment. Nevertheless, it's obvious that the five distances are insufficient for finishing a navigation task. To qualify UAVs for navigation, we combine the five distances with the UAV's orientation angle, and distance and angle between its present position and the destination, depicted in Fig. 2 (b), as the final state.

Another important issue is reward specification. In our work, reward is composed of four parts. The first part is environment penalty. If the UAV is too close to any obstacles during flying, it should be penalized regarding the minimum distance between itself and obstacles. Here we use an exponential function to formulate the penalty, that is, if the minimum distance decreases, the penalty grows exponentially. The second part is transition reward. If the distance from the UAV to its target position decreases after a time step, the UAV should be rewarded proportional to the reduced distance. The third part is direction reward. The UAV would get a constant reward if its first-perspective direction is in accordance with the corresponding direction of the biggest distance among the five distances. The forth part is step penalty. After every time step, the UAV would be given a constant penalty so as to encourage it to reach the target position as soon as possible.

Till now, DDPG can be directly employed to solve the navigation problem. However, as depicted in Fig. 3 (a), the UAV could be easily trapped in some local areas. We argue that the aforementioned state description is just a partial observation of the real state, for the UAV's real state should be determined by its internal state, position information and the local environment, while the UAV only has limited ability to percept the local environment.

2.2. RDPG and Fast-RDPG algorithms

DRL provides a solution concept to MDP and POMDP. A MDP is described by a set of environment states $\mathcal{S}$ and a set of actions $\mathcal{A}$, an initial state distribution $p(s_0)$, a transition function $p(s_{t+1}|s_t, a_t)$ and a reward function $r(s_t, a_t)$. The goal of DRL agents is to learn an optimal stochastic policy $\pi(s_t)$

or deterministic policy $\mu(s_t)$, which maximizes its long-term cumulated discounted reward. A MDP becomes a POMDP when the agent could not directly observe the state s_t and instead obtains an observation o_t, where $o_t \sim p(o_t|s_t)$. Consequently, the policy is a function of trajectory $h_t = (o_1, a_1, o_2, a_2, \ldots a_{t-1}, o_t)$ other than the state s_t. Different from RL, which needs handcrafted features as state representations, DRL automatically abstracts state descriptions from raw sensory inputs using deep network structures.

RDPG, based on actor-critic algorithm, solves POMDP by approximating actor and critic (or policy) using two recurrent neural network functions Q^{μ} and μ^{θ}. During learning, the actor network is updated with the following gradient [12] through Back Propagation Through Time (BPTT):

$$E_{\tau}\left[\sum_{t}\gamma^{t-1}\frac{\partial Q^{\mu}(h_t,a)}{\partial a}\bigg|_{a=\mu^{\theta}(h_t)}\frac{\partial \mu^{\theta}(h_t)}{\partial \theta}\right], \quad (1)$$

where γ is a discounted factor. Note the expectation is explicitly over entire trajectories $\tau = (s_1, o_1, a_1, s_2, o_2, a_2, \ldots)$. We argue that this policy update is not suitable in the framework of learning using memory replay [8], which is designed to reduce the correlation present in the observation sequence so as to stabilize the learning process and speed up the convergence procedure. To be specific, supposing the underlying real state of h_t is s_t', RDPG actually updates actor based on a sequence of highly correlated states (s_0', s_1', $s_2' \cdots$).

To break up the correlation, we make use of actor-critic architecture with function approximation [13], which derives policy update by directly maximizing the expected long-term accumulated discounted reward, and obtain a new policy update for POMDP with continuous control. Our new policy is explicitly updated with history trajectory h_t rather than entire trajectory τ. Below follows the detailed derivation. Define the stochastic policy function parameterized by θ as:

$$a_t \sim \pi_{\theta}(a_t \mid h_t). \quad (2)$$

Denote the underlying real state after observing the history trajectory h_t as s_t, where $s_t \sim p(s_t \mid h_t)$. Define the value function, the action-value function, and the expected reward when observing history trajectory h as:

$$V^{\pi_{\theta}}(h) = \underset{p(s|h)}{E}\underset{\tau_1}{E}\left[\sum_{t=1}^{\infty}\gamma^{t-1}r_t \,\middle|\, s_1 = s, h_t = h, \pi_{\theta}\right], \quad (3)$$

$$Q^{\pi_{\theta}}(h,a) = \underset{p(s|h)}{E}\underset{p(o_{t+1}|s_t,a_t)p(\tau_{t+1})}{E}\left\{\sum_{k=1}^{\infty}\gamma^{k-1}r_{t+k} \,\middle|\, s_t = s, h_t = h, a_t = a, \pi_{\theta}\right\}, \quad (4)$$

$$R_h^a = \underset{p(s|h)}{E}\underset{p(s_{t+1}|s_t=s,a_t=a)}{E}\left[r_{t+1} \,\middle|\, s_t = s, h_t = h, a_t = a, \pi_{\theta}\right], \quad (5)$$

where $\tau_1 \sim \prod_{t=1}^{\infty}\pi(a_t|h_t)p(s_{t+1}|s_t,a_t)p(o_{t+1}|s_{t+1})$.

The relation between value function and action-value function can be described as:

$$Q^{\pi_{\theta}}(h,a) = R_h^a + \gamma\sum_{h'}\left[p(h'|h,a)V^{\pi_{\theta}}(h')\right], \quad (6)$$

where $p(h'|h)$ represents the probability of transition from trajectory h to trajectory $h' = [h, o, a]$. Obviously we can extend this transition probability to any two trajectories by letting $p(h'|h) = 0$ if $h' = [h'', o, a]$ and $h'' \neq h$. Define the target function as:

$$J(\theta) = \underset{p(h_0)}{E}\left[V^{\pi_{\theta}}(h_0)\right], \quad (7)$$

where $p(h_0) = p(o_0)$ represents the initial observation distribution, and:

$$\frac{\partial V^{\pi_{\theta}}(h_0)}{\partial \theta} = \sum_{a}\left[\frac{\partial \pi_{\theta}(h_0,a)}{\partial \theta}Q^{\pi_{\theta}}(h_0,a) + \pi_{\theta}(h_0,a)\left(\gamma\sum_{h}p(h|h_0)\frac{\partial V^{\pi_{\theta}}(h)}{\partial \theta}\right)\right]. \quad (8)$$

Note that Eq. 8 possesses the same structure as Eq. 7 in [13]. Therefore, it can be rewritten as:

$$\frac{\partial V^{\pi_{\theta}}(h_0)}{\partial \theta} = \sum_{h}\rho^{\pi_{\theta}}(h|h_0)\sum_{a}\frac{\partial \pi_{\theta}(h,a)}{\partial \theta}Q^{\pi_{\theta}}(h,a), \quad (9)$$

where $\rho^{\pi}(h|h_0)$ represents history distribution induced by policy π. Then policy update can be derived as:

$$\frac{\partial J(\theta)}{\partial \theta} = \underset{\pi_{\theta}}{E}\underset{p(h_0)\rho^{\pi_{\theta}}(h|h_0)}{E}\left[\nabla_{\theta}\log\left(\pi_{\theta}(h,a)\right)Q^{\pi_{\theta}}(h,a)\right]. \quad (10)$$

As elucidated by Eq. 10, the expectation is over history trajectory h rather than entire trajectory τ. Note that the derived policy is a stochastic one. Lever [11] proves that deterministic policy is just a special case of stochastic policy. Following his idea, we obtain the update of deterministic policy $a_t = \mu^{\theta}(h_t)$ for POMDP with continuous control as:

$$\frac{\partial J(\theta)}{\partial \theta} = \underset{p(h_0)\rho^{\mu^{\theta}}(h|h_0)}{E}\left[\frac{\partial Q^{\theta}(h_t,\mu^{\theta}(h_t))}{\partial a}\frac{\partial \mu^{\theta}(h_t)}{\partial \theta}\right]. \quad (11)$$

As outlined in Table 1, we design a new algorithm based on Eq. 11 and [9], and name it as Fast-RDPG.

3. SIMULATION RESULTS

Table 1. Fast-RDPG algorithm

Fast-RDPG

Initialize critic network $Q^w(a_t, h_t)$ and actor $\mu^\theta(h_t)$ with parameters w and θ.
Initialize target networks $Q^{w'}$ and $\mu^{\theta'}$ with weights $w' \leftarrow w$, $\theta' \leftarrow \theta$.
Initialize replay buffer R.
for episodes=1, M, **do**
  Initialize a process N for action exploration
  Receive initial observation o_0 ($h_0 = o_0$)
  for t=1, T **do**
    Obtain action $a_t = \mu^\theta(h_t) + N_t$
    Execute action a_t, receive reward r_t, and obtain a new 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)
    Set $y_i = r_i + \gamma Q^{w'}([h_i, a_i, o_i], \mu^{\theta'}([h_i, a_i, o_i]))$
    Compute critic update (using BPTT)

$$\Delta\omega = \frac{1}{L}\sum_i \left(y_i - Q^\omega\left(h_i, a_i\right)\right)\frac{\partial Q^\omega\left(h_i, a_i\right)}{\partial \omega}$$

    Compute actor update (using BPTT)

$$\Delta\theta = \frac{1}{L}\sum_i \frac{\partial Q^\omega\left(h_i, \mu^\theta\left(h_i\right)\right)}{\partial a}\frac{\partial \mu^\theta\left(h_i\right)}{\partial \theta}$$

    Update actor and critic using Adam [14]
    Update the two target networks

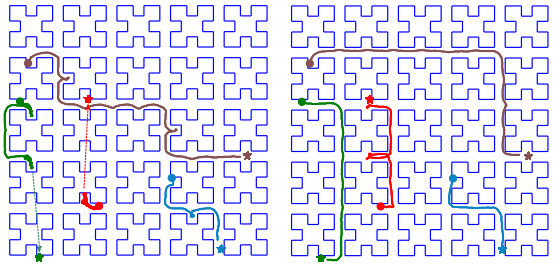
$$w' \leftarrow \varepsilon w + (1-\varepsilon)w'$$
$$\theta' \leftarrow \varepsilon\theta + (1-\varepsilon)\theta'$$

  end for
end for

As depicted in Fig. 1, we abstract real complex environment at a fixed altitude as five virtual environments, which include many traps and dense obstacles. Additionally, the virtual UAV's speed is set as 2 meters per time step.

Before learning, all the sensory inputs are normalized between [0,1] and the control signal is normalized between [-1,1], where '-1' means turning left for 180 degrees and '1' turning right for 180 degrees. As for hyper-parameters of the learning algorithm, we specify them exactly the same as those in [9] except the exploration noise. In our experiment, we use an exploration noise following a uniform distribution $U(-0.25, 0.25)$ in order to speed up the state space searching process. During learning, for each episode in Fast-RDPG, the five environments are chosen with equal chance and then a starting position and a target position are generated uniformly randomly in the chosen environment. Afterwards, the learning agent runs an episode, whose maximum length is T=300, and simultaneously updates its parameters.

Both DDPG and Fast-RDPG are employed to solve the large-scale navigation problem. To validate the efficiency of our proposed method, we randomly generate four pairs of starting positions and destinations in the most complex environment, the fifth virtual environment, and let both our Fast-RDPG agent and a RDPG agent "fly" from these starting positions to target positions. As depicted in Fig. 3, both of the two agents have learned to navigate without map



(a) (b)

Fig. 3. Illustration of trajectories, where circles represents starting positions and stars represent target positions. (a): trajectories generated by a DDPG agent (b): trajectories generated by a Fast-RDPG agent

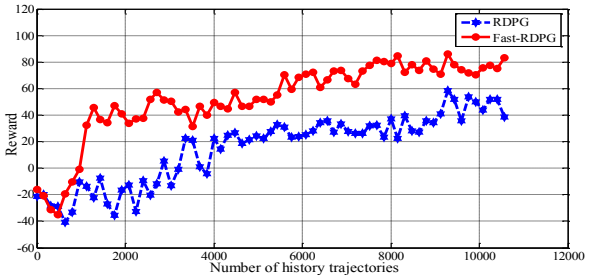


Fig. 4. Convergence curves of RDPG and Fast-RDPG

construction or path planning. Nevertheless, since our Fast-RDPG agent knows more local information than the RDPG agent, its learned navigation policy is less "greedy" than that of the RDPG agent and it possesses the ability to avoid or escape from traps while the DDPG agent is easily trapped.

To validate the efficiency of our proposed Fast-RDPG, we draw the convergence curves of Fast-RDPG and RDPG, depicted in Fig. 4. As expected, due to the break-up of high correlation present in observation sequences, Fast-RDPG converges much faster than DDPG.

4. CONCLUSION AND FUTURE WORK

It is challenging for UAVs to autonomously navigate in large-scale complex unknown environment. In our work, DRL is employed to solve the problem based on GPS signal and sensory information of the local environment. Though both the UAV and the environment are virtual, simulation results demonstrate the feasibility of our proposed approach. Furthermore, compared with original RDPG algorithm, our Fast-RDPG converges much faster.

However, there is still much work to do in the future. First of all, our proposed method needs to be tested in real environment, which includes more complex obstacles such as trees and electric wires. Second, it is critical to choose proper sensors. In our work we use five virtual range finders but in reality, cameras and radars seem to be more feasible. Third, the effect of sensor noise should be taken into consideration. Though we already regard the navigation problem as a POMDP, the effect of un-observability introduced by sensor noise is still unknown.

12. REFERENCES

[1] "Amazon Prime Air," [Online]. Available: https://www.amazon.com/b?node=8037720011. [Accessed 7 May 2017]

[2] Mohammed, F., Idries, A., Mohamed, N., Al-Jaroodi, J., & Jawhar, I. (2014, May). UAVs for smart cities: Opportunities and challenges. In Unmanned Aircraft Systems (ICUAS), 2014 International Conference on (pp. 267-273). IEEE.

[3] Cui, J. Q., Lai, S., Dong, X., & Chen, B. M. (2016). Autonomous navigation of UAV in foliage environment. Journal of Intelligent & Robotic Systems, 84(1-4), 259-276.

[4] Bachrach, A., Prentice, S., He, R., & Roy, N. (2011). RANGE-Robust autonomous navigation in GPS-denied environments. Journal of Field Robotics, 28(5), 644-666.

[5] Dissanayake, M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. IEEE Transactions on robotics and automation, 17(3), 229-241.

[6] Israelsen, J., Beall, M., Bareiss, D., Stuart, D., Keeney, E., & van den Berg, J. (2014, May). Automatic collision avoidance for manually tele-operated unmanned aerial vehicles. In Robotics and Automation (ICRA), 2014 IEEE International Conference on (pp. 6638-6643). IEEE.

[7] Zhang, A. M., & Kleeman, L. (2009). Robust appearance based visual route following for navigation in large-scale outdoor environments. The International Journal of Robotics Research, 28(3), 331-356.

[8] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., ... & Petersen, S. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540), 529-533.

[9] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., ... & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

[10] Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In Advances in neural information processing systems (pp. 1008-1014).

[11] Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. International Conference on Machine Learning, 2014:387-395.

[12] Heess, N., Hunt, J. J., Lillicrap, T. P., & Silver, D. (2015). Memory-based control with recurrent neural networks. arXiv preprint arXiv:1512.04455.

[13] Sutton, R. S., McAllester, D. A., Singh, S. P., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. In Advances in neural information processing systems (pp. 1057-1063).

[14] Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.