A DEEP REINFORCEMENT LEARNING APPROACH TO FLOCKING AND NAVIGATION OF UAVS IN LARGE-SCALE COMPLEX ENVIRONMENTS

Chao Wang, Jian Wang, and Xudong Zhang

Department of Electronic Engineering, Tsinghua University, Beijing, China

ABSTRACT

This paper aims at enabling unmanned aerial vehicles (UAV) to flock and meanwhile perform navigation tasks in large-scale complex environments in a fully decentralized manner. By incorporating the insights of flocking control inspired by bird flocking in nature, the problem is structured as a Markov decision process and solved by deep reinforcement learning. In particular, coordination among agents is achieved by following a local interaction protocol that each agent only considers the relative position of the nearest two neighbors on its left side and right side. In addition, a flocking control-inspired reward scheme is designed for the emergence of flocking and navigation behaviors. Simulation results demonstrate that by training with three UAVs, the learned policy, shared across all agents, can enable a larger number of UAVs to perform navigation tasks as a group in large-scale complex environments.

***Index Terms*—** UAV flocking, UAV navigation, flocking control, deep reinforcement learning.

1. INTRODUCTION

The thriving of unmanned aerial vehicles (UAV) in recent years facilitates the development of many UAV-based applications such as remote sensing and good delivery [1, 2]. The goal of this paper is to develop a method that can enable UAVs to execute navigation tasks as a group in complex environments, e.g., urban areas full of high buildings.

A common approach is by following the flocking control law [3]: flock centering (agents try to stay close to nearby flockmates), collision avoidance (agents try to avoid collision with nearby agents), and velocity matching (agents try to match their velocity with nearby agents). La [4], for example, designs a multi-agent system for target tracking in noisy environments based on these basic rules. Similar works can also be found in [5–9]. However, these systems suffer from the limited adaptation ability of fixed rules to complex environments and systems. Alternatively, researchers turn to reinforcement learning (RL)-based or deep reinforcement learning (DRL)-based methods. RL (DRL) aims at deriving a control policy by maximizing a long-term cumulated reward of a Markov decision process (MDP), which characterizes a process that an agent at some state takes an action, transits to another state and obtains some reward. Morihiro [10], for instance, investigates grouping and anti-predator behaviors of multi-agent systems using RL. Specifically, the perceptual space of each agent is defined as its distances to nearby agents, and the choice of action includes attraction, parallel orientation and repulsion to nearby agents. Similar works can also be found in [11–13]. Some of these methods incorporate the idea of flocking control to construct the state spaces and action spaces of MDPs, but the incorporation is still at the elementary stage. Besides, most of them only focus on learning flocking and predator avoidance behaviors but neglect the navigation phase, and the environment is generally obstacle-free and small in size.

In this paper, by incorporating the insights of flocking control, we structure the problem of flocking and navigation of UAVs in large-scale complex environments as a MDP and develop a learning algorithm based on one of the state-of-the-art DRL algorithms, deep deterministic policy gradient algorithm (DDPG) [14]. The key of solving the problem lies in two aspects. Firstly, for the emergence of flocking and navigation behaviors, each agent needs to coordinate with other agents. Inspired by fish schooling and bird flocking, we design a local coordination protocol that each agent only observes the relative position of the nearest two neighbors on its left side and right side. Besides, a flocking control-inspired reward scheme is designed to facilitate agents to flock and in the meantime perform navigation. Secondly, all agents are identical and should behave in a fully decentralized manner. In this respect, based on DDPG, we develop a DRL algorithm that learns a shared policy in a centralized manner, executed in a fully decentralized manner.

The remainder of this manuscript is structured as follows. In Section 2 we define the flocking and navigation problem, followed by the details of the modeling process in the context of MDP and the solution method. In Section 3 we validate the feasibility and efficiency of our proposed method through simulation. Finally, conclusions are presented in Section 4.

2. PROBLEM FORMULATION AND LEARNING ALGORITHM

In this section we formulate flocking and navigation of UAVs in large-scale complex environments as a RL problem in the context of MDP, followed by the solution method.

2.1. Problem description

In our flocking and navigation scenario, we assume UAVs fly at a fixed altitude. As a result, they are restricted in the $x-y$ plane of a three-dimensional Cartesian coordinate. By denoting the position, the first perspective orientation and the speed of each UAV as (x, y, ω, v) and the steering control and the throttle control signals as (a^1, a^2), the dynamics of a UAV can be formulated as:

$$\begin{cases} \omega_{t+1} = \omega_t + a_t^1 \\ v_{t+1} = v_t + a_t^2 \\ x_{t+1} = x_t + v_{t+1}\cos(\omega_{t+1}) \\ y_{t+1} = y_t + v_{t+1}\sin(\omega_{t+1}). \end{cases} \quad (1)$$

The way each UAV interacts with others affects the flocking and navigation behavior as well as its generalization ability to systems with more agents. We propose an interaction protocol that each UAV only observes the nearest two neighbors on its left side and right side, as illustrated in Fig. 1 (b). For ease of reference, we denote it as Protocol I. By applying Protocol I, each UAV needs to coordinate its action with both the agents on its left side and right side so as to form a group and perform navigation tasks. We believe the designed protocol is already the simplest. For comparison purpose, we consider a simpler interaction protocol that each agent only observes the relative position of its nearest neighbor. For ease of reference, we denote it as Protocol II.

The objective of each UAV is thus to control its speed and orientation in order to flock with other UAVs and meanwhile navigate from arbitrary departure places to destinations in large-scale complex environments.

We solve the problem by modeling it as a MDP. A MDP consists 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 $r(s_t, a_t)$: $\mathcal{S}\times\mathcal{A} \to \mathcal{R}$ describing the environment feedbacks when executing action a_t at state s_t. Since UAVs' control signals are continuous, in this paper we focus on MDPs with continuous state and action spaces. The goal of RL is to learn an optimal policy $a_t = \mu(s_t)$ that projects states to actions by maximizing a long-term cumulated reward.

2.2. State representation and action space

States characterize each UAV's relation with the environment and other UAVs. In our flocking and navigation setting, the state of each UAV consists of four parts. As illustrated in Fig. 1, each UAV's internal state is described by (ω, v), where ω is the angle between each UAV's first-perspective orientation and the y-axis and v the speed. The relationship between each UAV and the target is described by (θ^1, d^1), where

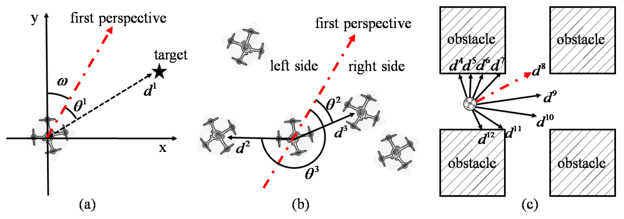

Fig. 1. Schematic illustration of a UAV's state space.

θ^1 represents the relative angle between each UAV's first-perspective direction and the direction of the target, and d^1 the distance between each UAV's current position and the target. The relationship between each UAV and other UAVs are described by $(\theta^2, \theta^3, d^2, d^3)$, where θ^2 and θ^3 denote the angles between each UAV's first-perspective direction and those of the nearest two neighbors on its left side and right side, respectively, and d^2 and d^3 the distances to them. The relationship between each UAV and the environment is characterized by $(d^4, \cdots, d^{12})$, where $d^4 \sim d^{12}$ denote the distances returned by nine virtual range finders. To summarize, the state space of each UAV is characterized by $s = (\omega, v, \theta^1, \theta^2, \theta^3, d^1, \cdots, d^{12})$, where ω and $\theta^1 \sim \theta^3$ range from 0 to 2π and v and $d^1 \sim d^{12}$ range from 0 to $+\infty$.

UAVs maneuver by implementing control signals. Since the movement of UAVs is restricted to a plane, the action space of each UAV is characterized by $a = (a^1, a^2)$, where a^1, the steering control signal, ranges from $-\pi/4$ to $\pi/4$ and a^2, the throttle control signal, ranges from -1.0 to 1.0.

2.3. Reward scheme

Reward determines the behavior of agents. For the purpose of flocking as well as navigation, in this paper, the designed reward is composed of four parts: namely transition reward, mutual reward, obstacle penalty and step penalty. At time step t, the transition reward is defined as:

$$r_{tr}(s_t, a_t) = \tanh(0.2(10 - v_{t+1}))(d_{t+1}^1 - d_t^1), \quad (2)$$

which incorporates the information that each UAV would get a reward proportional to the reduced distance to the target after one step transition. Besides, a speed-related weight is multiplied with the reduced distance in order to penalize large speed. The mutual reward is designed as:

$$r_m(s_t, a_t) = \begin{cases} -5, \text{ if } d_{t+1}^2 < 10 \text{ or } d_{t+1}^3 < 10, \\ 0, \text{ if } d_{t+1}^2 > 50 \text{ or } d_{t+1}^3 > 50, \\ 3e^{0.05(d_{t+1}^2 - 20)} + 3e^{0.05(d_{t+1}^3 - 20)}, \text{ else}, \end{cases} \quad (3)$$

encoding the information that a UAV would get positive rewards if its distances to the nearest two neighbors on its left side and right side are around 20 meters, and penalties if its distance to other agents is less than 10 meters. The obstacle penalty is defined as:

$$r_b(s_t, a_t) = \begin{cases} -5, \text{ if } \min(d_{t+1}^4, \cdots, d_{t+1}^{12}) < 10, \\ 0, \text{ else.} \end{cases} \quad (4)$$

which incorporates the information that a UAV would be penalized if its shortest distance to obstacles is less than 10 meters. Additionally, to encourage UAVs to head for destinations as soon as possible, after a transition each UAV would get a constant penalty $r_s(s_t, a_t) = -3$. The final reward is formulated as:

$$r(s_t, a_t) = r_{tr}(s_t, a_t) + r_m(s_t, a_t) + r_b(s_t, a_t) + r_s(s_t, a_t). \quad (5)$$

2.4. Learning objective and learning algorithm

Each agent aims at learning a shared policy $a = \mu(s)$ by maximizing:

$$\eta(\mu) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \quad (6)$$

where γ is a discount factor. In this paper, the optimization is done in the actor-critic architecture, which parameterizes the policy $a = \mu(s; \alpha)$ as a function and the parameter α is updated by the following gradient [14]:

$$\nabla\eta(\mu) = \mathop{\mathbb{E}}_{s\sim d_\mu(s)}\left[\nabla_\alpha \mu(s) \nabla_a \, Q_\mu(s,a)|_{a=\mu(s)}\right], \quad (7)$$

where $d_\mu(s)$ is the state distribution and $\mu(s)$ is renamed as actor. Besides, $Q_\mu(s, a)$ is referred as critic, characterizing the expected cumulated reward an agent can get when starting from state s, executing action a and then following the policy μ. Since all UAVs are identical, the critic is also shared across them. In practice, the critic is parameterized as a function $Q_\beta(s, a)$ with parameter β and estimated by minimizing [15]:

$$||r(s,a) + \gamma Q_\beta(s', a') - Q_\beta(s,a)||_2^2, \quad (8)$$

where s' refers to the next state after taking action a at state s and a' the action taken at s' by following the policy μ.

With the optimization methods of the actor and the critic defined above, DDPG learns a control policy by parameterizing the actor and the critic as two deep neural networks and using a target actor network, a target critic network and a replay memory to stabilize the learning procedure [14]. Since our problem involves in learning in multi-agent systems in a centralized manner with a shared actor, a shared critic and a shared replay memory, we develop a new learning algorithm based on DDPG. The designed algorithm, named Flocking-DDPG, is summarized in Table 1.

Table 1. Flocking-DDPG

```
Initialize globally shared actor $\mu_\alpha(s)$ and critic $Q_\beta(s,a)$
Initialize target actor $\mu_{\alpha'}(s)$ and target critic $Q_{\beta'}(s,a)$
with weights $\alpha' \leftarrow \alpha, \beta' \leftarrow \beta$
Initialize a globally shared replay memory $R$
for $episodes = 1, M$, do
  for $t = 1, T$ do
    for agent $j = 1, K$
      Receive state $s_t^j$
      Obtain action $a_t^j = \mu_\alpha(s_t^j) + N_t^j$
      Execute $a_t^j$, obtain reward $r_t^j$ and next state $s_{t+1}^j$
      Store transition $(s_t^j, a_t^j, r_t^j, s_{t+1}^j)$ into $R$
    end for
    Sample a batch of $L$ transitions $(s_i, a_i, r_i, s_i')$ from $R$
    Set $y_i = r_i + \gamma Q_{\beta'}(s_i', \mu_{\alpha'}(s_i'))$
    Update acctor using
      $\Delta\alpha = \frac{1}{L}\sum_i \frac{\partial Q_\beta(s_i,\mu_\alpha(s_i))}{\partial a}\frac{\partial \mu_\alpha(s_i)}{\partial \alpha}$
    Update critic using
      $\Delta\beta = \frac{1}{L}\sum_i (y_i - Q_\beta(s_i,a_i))\frac{\partial Q_\beta(s_i,a_i)}{\partial \beta}$
    Update the target networks ($\epsilon$ is the target update rate)
      $\alpha' \leftarrow \varepsilon\alpha + (1-\varepsilon)\alpha' \quad \beta' \leftarrow \varepsilon\beta + (1-\varepsilon)\beta'$
  end for
end for
```

3. SIMULATION RESULTS AND DISCUSSIONS

3.1. Experimental settings

We simulate a large-scale complex environment that covers two square kilometers and is full of buildings (abstracted as cylinders) with random height. A top view of a sample of the simulated environment is illustrated in Fig. 2 (a). Every time before agents start interacting with the environment, buildings with height following a uniform distribution $U(30.0m, 200.0m)$ are regenerated. The flight altitude is fixed to $100.0m$ and the maximum speed is restricted to $10m/s$. The states of each UAV are normalized to $[-1.0, 1.0]$ and the steering signal and throttle signal are normalized to $[-0.25, 0.25]$ and $[-1.0, 1.0]$, respectively.

The actor and the critic are approximated by two deep neural networks with one input layer, one output layer and two hidden layers. For the actor network, the size of the input layer equals to 17, the size of the two hidden layers are 300 and 400, and the size of the output layer equals to 2. The structure of the critic network is the same as the actor except that the second hidden layer is concatenated with the control vector (a^1, a^2) and the output layer outputs a scalar. Besides, an exploration noise having a Gaussian distribution, denoted by $N((0.0, 0.0)^T, (0.05, 0.25)^T)$, is used to explore the state spaces. The maximum time step T is set to 500. The maximum size of the globally shared replay memory R is set to 30000. The learning procedure begins if the number of transition tuples in R is larger than 5000 and terminates if M, the number of episodes, is larger than 1000. The rest of the hyper-parameters are set the same as those in DDPG [14].

Since each UAV only considers the relative position of at

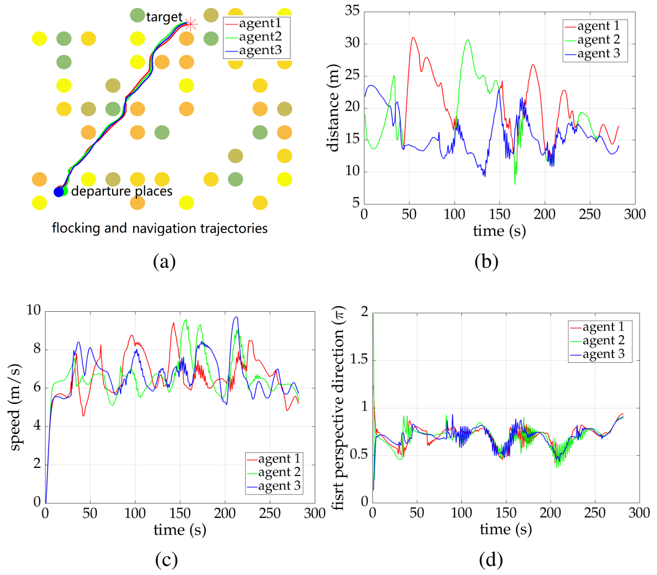
Fig. 2. Flocking and navigation results generated by the policy trained with Protocol I.

most two nearby neighbors, we implement Flocking-DDPG in a multi-agent system with only three UAVs, i.e., $K = 3$.

3.2. Flocking and navigation behaviors

Fig. 2 (a) shows the flocking and navigation trajectories generated by the policy trained with Protocol I, (b) the shortest distance of each UAV to its nearest neighbor in the whole flocking and navigation process. As we can see, each UAV keeps a proper distance to its nearest neighbor and they together fly from the departure places to the destination. Fig. 2 (c) and (d) show the speed and first-perspective orientation of each UAV, respectively. As can be seen, by implementing the learned policy, each agent successfully coordinates its speed and orientation with other agents.

We also make a quantitative survey over the success rate of navigation missions by executing the policies trained with different interaction protocols. As illustrated in the second column in Table 2, the success rates of the policy trained with Protocol I reaches up to 95%, while that of the policy trained with Protocol II is only 72%. The impressive performance of the policy trained with Protocol I demonstrates the feasibility and efficiency of our proposed method.

3.3. Generalization ability

Below we test the generalization ability of the learned policy to systems with more agents. We execute the policy trained with Protocol I in a multi-agent system with seven UAVs. As shown in Fig. 3, the policy successfully enables all the UAVs to perform navigation as a group. By contrast, if the policy

Table 2. Success rates under different conditions

Success rate / Agents / Protocols	3 UAVs	5 UAVs	7 UAVs
Protocol I	95.00%	95.00%	90.00%
Protocol II	72.00%	69.00%	51.00%

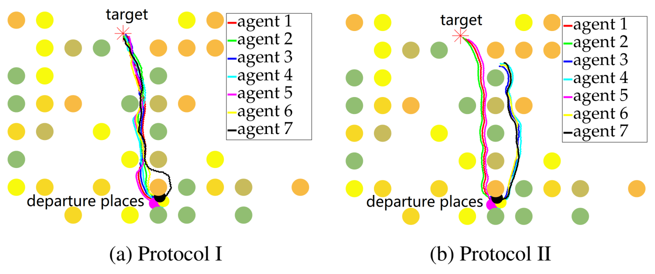
Fig. 3. Flocking and navigation trajectories of the policies trained with different coordination protocols.

trained with Protocol II is performed, UAVs are easily separated by obstacles and never reunite again. Table 2 illustrates the success rates of the two policies when applied to systems with different number of UAVs. As we see, the policy trained with Protocol I exhibits strong generalization ability, while the policy trained with Protocol II degenerates significantly.

The poor generalization ability results from the limited coordination ability of Protocol II. When each UAV only observes the relative position of its nearest neighbor, they tend to form cliques. The distances of each UAV within the clique to UAVs outside the clique are uniformly longer that those inside the clique. As a consequence, UAVs within each clique are completely separated from those outside the clique. In contrast, when Protocol I is applied, UAVs will never be separated, because for each UAV, the interaction protocol divides the rest of the UAVs into two groups and the UAV needs to coordinate its action with both the two groups simultaneously.

4. CONCLUSION

In this paper we combine the insights of RL and flocking control to solve the problem of autonomous navigation of UAVs as a group in large-scale complex environments. A local interaction protocol that each UAV only considers the relative position of the nearest two neighbors on its left side and right side is designed for the purpose of behavior coordination among agents. Additionally, a flocking control-inspired reward is designed for the emergence of flocking and navigation behaviors. By executing the designed Flocking-DDPG algorithm in a system with only three UAVs, the learned policy is efficient in enabling a larger number of UAVs to perform navigation as a group in large-scale complex environments.

5. REFERENCES

[1] C. Wang, J. Wang, X. Zhang, and X. Zhang, "Autonomous navigation of uav in large-scale unknown complex environment with deep reinforcement learning," in *Signal and Information Processing (GlobalSIP), 2017 IEEE Global Conference on*. IEEE, 2017, pp. 858–862.

[2] K. Anderson and K. J. Gaston, "Lightweight unmanned aerial vehicles will revolutionize spatial ecology," *Frontiers in Ecology and the Environment*, vol. 11, no. 3, pp. 138–146, 2013.

[3] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *ACM SIGGRAPH computer graphics*, vol. 21, no. 4. ACM, 1987, pp. 25–34.

[4] H. M. La and W. Sheng, "Flocking control of multiple agents in noisy environments," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4964–4969.

[5] Y. Jia, J. Du, W. Zhang, and L. Wang, "Three-dimensional leaderless flocking control of large-scale small unmanned aerial vehicles," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6208–6213, 2017.

[6] C. K. Hemelrijk and H. Hildenbrandt, "Schools of fish and flocks of birds: their shape and internal structure by self-organization," *Interface focus*, p. rsfs20120025, 2012.

[7] A. Shklarsh, A. Finkelshtein, G. Ariel, O. Kalisman, C. Ingham, and E. Ben-Jacob, "Collective navigation of cargo-carrying swarms," *Interface Focus*, vol. 2, no. 6, pp. 786–798, 2012.

[8] S. Hubbard, P. Babak, S. T. Sigurdsson, and K. G. Magnússon, "A model of the formation of fish schools and migrations of fish," *Ecological Modelling*, vol. 174, no. 4, pp. 359–374, 2004.

[9] H. Su, X. Wang, and Z. Lin, "Flocking of multi-agents with a virtual leader," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 293–307, 2009.

[10] K. Morihiro, H. Nishimura, T. Isokawa, and N. Matsui, "Learning grouping and anti-predator behaviors for multi-agent systems," in *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*. Springer, 2008, pp. 426–433.

[11] K. Morihiro, T. Isokawa, H. Nishimura, M. Tomimasu, N. Kamiura, and N. Matsui, "Reinforcement learning scheme for flocking behavior emergence." *JACIII*, vol. 11, no. 2, pp. 155–161, 2007.

[12] S.-M. Hung and S. N. Givigi, "A q-learning approach to flocking with uavs in a stochastic environment," *IEEE transactions on cybernetics*, vol. 47, no. 1, pp. 186–197, 2017.

[13] M. Hüttenrauch, A. Šošić, and G. Neumann, "Learning complex swarm behaviors by exploiting local communication protocols with deep reinforcement learning," *arXiv preprint arXiv:1709.07224*, 2017.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.