A Two-Stage Reinforcement Learning Approach for Multi-UAV Collision Avoidance Under Imperfect Sensing

Dawei Wang, Tingxiang Fan, Tao Han, and Jia Pan

***Abstract*—Unlike autonomous ground vehicles (AGVs), unmanned aerial vehicles (UAVs) have a higher dimensional configuration space, which makes the motion planning of multi-UAVs a challenging task. In addition, uncertainties and noises are more significant in UAV scenarios, which increases the difficulty of autonomous navigation for multi-UAV. In this letter, we proposed a two-stage reinforcement learning (RL) based multi-UAV collision avoidance approach without explicitly modeling the uncertainty and noise in the environment. Our goal is to train a policy to plan a collision-free trajectory by leveraging local noisy observations. However, the reinforcement learned collision avoidance policies usually suffer from high variance and low reproducibility, because unlike supervised learning, RL does not have a fixed training set with ground-truth labels. To address these issues, we introduced a two-stage training method for RL based collision avoidance. For the first stage, we optimize the policy using a supervised training method with a loss function that encourages the agent to follow the well-known reciprocal collision avoidance strategy. For the second stage, we use policy gradient to refine the policy. We validate our policy in a variety of simulated scenarios, and the extensive numerical simulations demonstrate that our policy can generate time-efficient and collision-free paths under imperfect sensing, and can well handle noisy local observations with unknown noise levels.**

***Index Terms*—Collision avoidance, deep learning in robotics and automation.**

I. Introduction

With THE widely increasing application of unmanned aerial vehicles in tracking [1], [2], disaster rescue [3] and environment exploration [4], collision avoidance algorithm for aerial robots becomes more and more important. However, the complicated and diversified workspace for aerial robots in real-world has posted a great challenge for the robustness of the multi-UAV systems since any minor error in aerial robots may cause great damage or loss for persons or other properties. Some researchers proposed solutions to multi-UAV collision avoidance systems based on centralized algorithms, which rely on a central server to communicate with each agent, and to generate global control commands according to the global observations for all robots [5]–[7]. However, the centralized multi-UAV system's reliance on communication makes it difficult to be deployed in practice, because it is usually difficult or even infeasible to maintain stable communication in large-scale and complicated scenarios due to issues such as radio interference and the radio shelter regions.

Concerning these difficulties, many methods for decentralized control are introduced for the multi-agent systems without communication [8], [9]. Many of these methods use a single sensor on-board for collision avoidance, e.g., one RGB-D camera is used in [10]. However, UAVs have six degrees of freedom and thus a single sensor cannot capture sufficient information for reliable 3D collision avoidance of a UAV. Some other methods, with ORCA3D [11] as a typical example, leverage the information of local neighbors for collision avoidance decision making. However, these methods assume that each agent can obtain a perfect estimation about the location of itself and its neighbors, as shown in Fig. 1(a), which unfortunately is difficult to achieve in real-world applications due to the imperfect sensing, as shown in Fig. 1(b). This limits many previous works in simulation, without being able to be deployed to multi-UAV systems in real world.

In this letter, we proposed a decentralized collision avoidance policy by reinforcement learning, which leverages local neighbors' information to accomplish robust multi-UAV motion planning without the perfect sensing assumption. UAV motion planning has a higher DOF than 2D AGV planning, and our attempts to apply our state-of-the-art 2D collision avoidance approaches from our prior work [12]–[14] showed the 3D problem to be harder to solve and more expensive to train. Therefore, we propose two-stage reinforcement learning approach. Different from supervised learning, reinforcement learning (RL) does not have a fixed training set with ground truth labels, which makes RL methods suffer from high variance and low reproducibility [15]. To make our approach fast converge to the global minimum and reduce variance, we propose two-stage training method. The first stage is supervised training method with a loss function that encourages the agent to follow the well-known reciprocal collision avoidance strategy, which can optimize deep RL network parameters into a theoretically optimal zone [16]. The second stage is using traditional RL training method (Policy Gradient [17]) to maximize reward function and refine the policy. We demonstrate

Manuscript received September 10, 2019; accepted January 19, 2020. Date of publication February 18, 2020; date of current version March 4, 2020. This letter was recommended for publication by Associate Editor Dr. A. Faust and Editor Prof. N. Amato upon evaluation of the reviewers' comments. This work was partially supported by the HKSAR General Research Fund under Grant HKU 11202119 and 11207818. *(Corresponding author: Jia Pan.)*

Dawei Wang, Tingxiang Fan, and Jia Pan are with the Department of Computer Science, The University of Hong Kong, Hong Kong, China (e-mail: dawei@hku.hk; tingxiangfan@gmail.com; panjia1983@gmail.com).

Tao Han is with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong, China (e-mail: thancn@gmail.com).

Digital Object Identifier 10.1109/LRA.2020.2974648

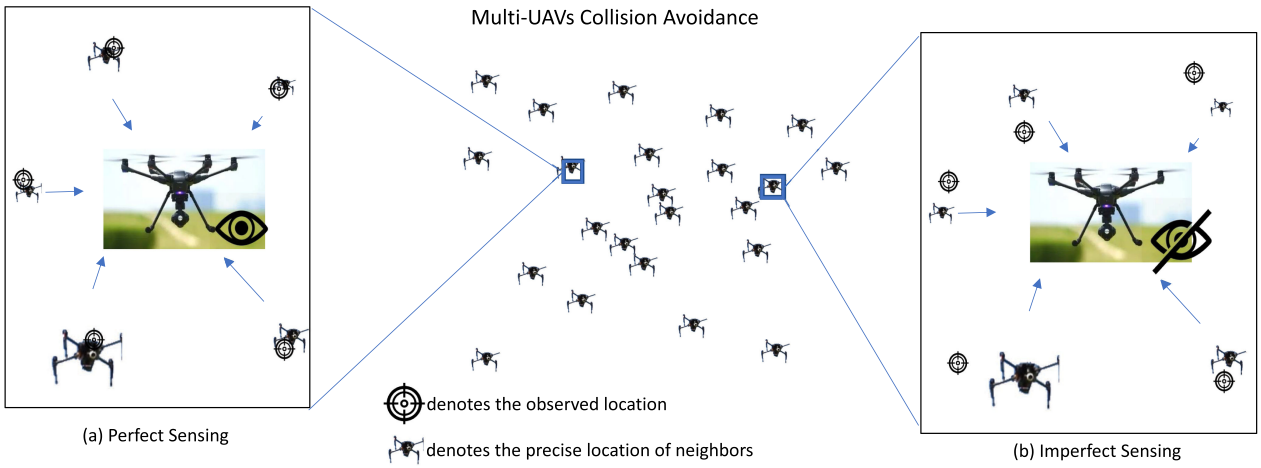

Fig. 1. Most existing collision avoidance methods assume that the environment sensing is perfect and noise-less, as shown in (a). But in the real-world environment, the sensed information is always noised and delayed (b). How to deal with such complex situation in the real-world environment is a well-known challenge for collision avoidance. In this work we propose a two-stage reinforcement learning approach to solve this challenge and demonstrate its performance in simulated scenarios where additive Gaussian noises are added to simulate imperfect sensing.

that our approach behaved robust and stable compared to the RL model without two-stage training procedure. We also validate the performance of our algorithm on unseen large-scale scenarios, which shows a good generalization of our proposed approach. The main contributions can be summarized as:

- We propose a decentralized reinforcement learning based policy for multi-UAV collision avoidance leveraging imperfect local observation, which provides robust performance in large-scale unseen testing scenarios with more than 200 agents in 3D workspace.
- We present a two-stage training method to reduce variance and make our approach fast converge to the global minimum.
- Our policy can be easily extended to handle general noise level tasks. We have trained a single policy for blind environment noise without manually adjusting parameters, which outperforms a state-of-the-art baseline.

II. Related Work

Long *et al.* [13] categorized the decentralized collision avoidance method into two types: sensor-level methods, which only feed raw on-board sensor data to the policy network, and agent-level methods, which leverage the observable states of other agents instead of raw sensor data.

A. Sensor-Level Navigation

In 2016, Bojarski *et al.* [18] achieved simple real-world autonomous driving by only using the raw camera data, which validated the feasibility to use deep learning for sensor-level navigation. Tai *et al.* [19] trained the navigation policy in the simulation via deep reinforcement learning, and then transferred the policy to the real robot. These works obtained good performance in AGVs collision avoidance problem, but they cannot be successfully applied to the collision avoidance of UAVs, which have a dimensional configuration space and also face more significant environment noises. For example, the differential wheeled robots only has two degrees of freedom to control, i.e. moving along x-axis or rotating by z-axis, and its observation can often be simplified as a few 2D points generated by a 2D LiDAR. Whereas aerial robots have six degrees of freedom of motion and thus a much larger action space, which makes decision making optimization more expensive. In addition, a single on-board sensor could only cover a very small part of the surrounding 3D environments and thus increases the observation uncertainty of the UAVs.

Some researchers presented sensor-level UAV motion planning [20] and navigation [21] algorithms. Gao *et al.* [22] used a stereo camera and IMU to build a teach-repeat-replan UAV system. Campos *et al.* [10] used a single RGB-D camera to establish an autonomous navigation framework for reaching a goal in unknown 3D cluttered environments. However, these methods could only work well in static scenarios, and UAVs' action space is also restricted for simplifying the problem. Thus, they cannot be applied to high-speed dynamic collision avoidance tasks. In addition, due to the large configuration space of a UAV, it is necessary to use more than one sensors to make a complete observation about its surrounding world. However, due to the limited battery life and carrying capability of nowadays UAVs, a successful multi-UAV collision avoidance system must survive the high observation uncertainty due to limited sensor view angles.

B. Agent-Level Navigation

Early traditional methods [23], [24] have been successfully used for crowd navigation and multi-robot collision avoidance, but they cannot robustly adapt to different scenarios due to their hand-crafted parameters. Thus, Chen *et al.* [25] introduced the value network of deep reinforcement learning to model the human-robot cooperative behaviors in dynamic environments. However, these approaches are all based on perfect sensing assumption and thus cannot work well in real-world applications with imperfect sensing. Recently, Tolstay *et al.* [26] applied

Graph Neural Network (GNN) in agent-level navigation, which requires local communications during training and testing. This GNN based approach cannot be used in large-scale scenarios, because wireless communication in crowd scenarios is unreliable and any delay or unstable connection may cause collision and damage in the air. Therefore, decentralized UAVs collision avoidance, leveraging imperfect sensed agent-level information without communications between each agent, is more desirable than existing agent-level approaches.

III. Approach

In this section, we introduce our multi-UAV reinforcement learning framework firstly. Next, we describe the network structure of the control policy for the multi-UAV system. Finally, we introduce the two-stage training method for our reinforcement learning model.

A. Problem Formulation

The agent-level multi-UAV decision model can be formulated as a Partially Observable Markov Decision Process (POMDP) and can be solved using a reinforcement learning framework [12], [14]. Formally, we describe a POMDP as a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O})$, where $\mathcal{S}$ is a set of states ($\mathbf{s} \in \mathcal{S}$), $\mathcal{A}$ is a set of actions ($\mathbf{a} \in \mathcal{A}$), $\mathcal{T}$ is the transition probabilities between states $\mathcal{T}(\mathbf{s}' \mid \mathbf{s}, \mathbf{a})$, $\mathcal{R}$ is the reward function ($\mathcal{S} \times \mathcal{A} \to \mathbb{R}$), Ω is a set of observations $\mathbf{o} \in \Omega$, and $\mathcal{O}$ is the observation distribution given the state $\mathbf{o} \sim \mathcal{O}(\mathbf{s})$. In our formulation, the state $\mathbf{s}^t$ consists of the agents' position $\mathbf{p}$, velocity $\mathbf{v}$, and the goal $\mathbf{g}$ at t time step, i.e. $\mathbf{s}^t = \langle \mathbf{p}^t, \mathbf{v}^t, \mathbf{g} \rangle$. The action $\mathbf{a}^t$ is the steering command of a different robot in terms of linear velocities. The observation at time t is $\mathbf{o}^t$, which includes the position ${}^i o_p^t$ and velocity ${}^i o_v^t$ of the neighbors that are observable to the robot. $\mathbf{o}^t$ also includes robot's preferred velocity o_{prefv}^t, which is the direction from its current location to its goal with the maximum speed as the magnitude. In other words, $\mathbf{o}^t = \sum_i^N \langle {}^i o_p^t, {}^i o_v^t \rangle + o_{prefv}^t$, where N is the number of the observable neighbors. The optimal policy π^* is defined according to the Bellman equation:

$$\begin{aligned} \pi^* = & \arg\max_{\mathbf{a}^*} R(\mathbf{s}^t, \mathbf{a}^t) \\ & + \gamma \int_{\mathbf{s}^{t+1}} \mathcal{T}(\mathbf{s}^{t+1} | \mathbf{s}^t, \mathbf{a}^t) V^*(\mathbf{s}^{t+1}) \mathrm{d}\mathbf{s}^{t+1} \\ V^*(\mathbf{s}^t) = & \sum_{t'=t}^{T} \gamma^{t'-t} R(\mathbf{s}^{t'}, \mathbf{a}^{t'}), \end{aligned} \tag{1}$$

where $R(\mathbf{s}^t, \mathbf{a}^t)$ is the reward given $\mathbf{s}^t$ and $\mathbf{a}^t$ at time t and γ is the discount factor in reinforcement learning. In this letter, the reward function is proposed as:

$$r^t = \begin{cases} 20 & \text{if } \|\mathbf{p}^t - \mathbf{g}\| < 0.1 \\ -20 & \text{else if } \textbf{collision} \\ 2.5 \cdot (\|\mathbf{p}^{t-1} - \mathbf{g}\| - \|\mathbf{p}^t - \mathbf{g}\|) & \text{otherwise.} \end{cases} \tag{2}$$

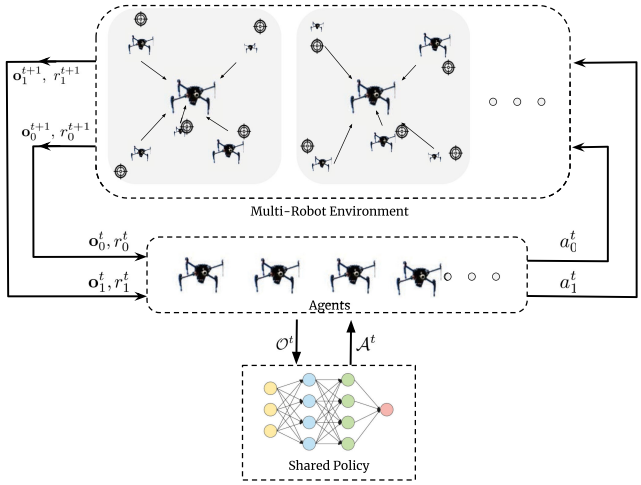

Fig. 2. Illustration of the framework of our method. Each robot obtains its neighbors' information by directly communicating with them and then leverages such information to determine control commands.

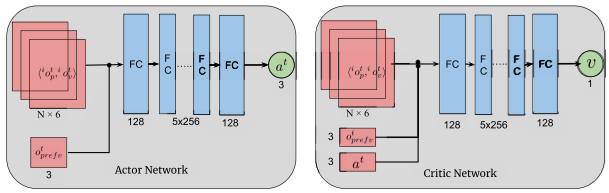

Fig. 3. The architecture of the neural network.

B. Multi-UAV Reinforcement Learning Framework

We construct a multi-robot environment with noisy sensor data and uncertainly state estimation in simulation as the data source of reinforcement learning algorithm (as shown in Fig. 2). In this environment, all agents shared the common single control policy. Overall, the data stream of the multi-UAV reinforcement learning framework is simple. The UAVs interact with the noisy environment to get all observations $\mathbf{o}_i^t$ and all reward r_i^t at time t, and these data is fed into the shared policy to compute all control commands of different robots a_i^t. Finally, these commands are executed in the multi-robot environment simultaneously, as shown in Fig. 2. All the data generated by the multi-robot environment are used for the reinforcement learning training.

C. Network Structure

To achieve the best performance for aerial robots, we deploy the state-of-the-art off-policy actor-critic [27] based reinforcement learning algorithm, the deep deterministic policy gradient (DDPG) [28], [29]. The architecture of our neural network consists of two networks, i.e., the actor network and the critic network, as shown in Fig. 3.

In the actor network, the input layer is a fully connected (FC) layer containing 256 hidden neurons, followed by the input layer, which includes five FC layers with 128 hidden neurons. Each layer uses a hyperbolic tangent ($\tanh$) activation function. At the end, there is an output layer which also uses a fully connected

structure and maps the output of the previous layer to a three-dimensional vector $\mathbf{a} = [v_x, v_y, v_z]$, which serves as the velocity control demand for UAVs.

In the critic network, the input layer is one FC layer containing 128 hidden neurons. After the input layer, there are five FC layers with 128 hidden neurons. The output of the critic network is the value of the value function v, generated by the output layer with a fully connected structure.

D. Two-Stage Training Method

The traditional reinforcement learning algorithm is hard to train and reproduce, which has been reported, e.g., in [15]. This is because reinforcement learning does not have fixed training dataset and ground truth targets. To address such difficulty, we propose a two-stage training method for RL policy in multi-UAV collision avoidance system, where the first stage is a pre-training stage supervised by optimal reciprocal collision avoidance principle, and the second stage is a unsupervised training stage using deep deterministic policy gradient algorithm.

To enable cooperative collision avoidance among multiple robots in a manner without communication, we adopt the same assumption as used in previous decentralized collision avoidance works such as ORCA [30]. In particular, we assume that all agents in the swarm will follow the same collision avoidance policy. Under this assumption, [30] proved that, if the velocity action taken by every agent falls outside the velocity obstacle zone that is constructed according to the current velocities of this agent and its neighbors, then all robots are theoretically collision-free in a given time horizon, if a perfect sensing is available.

In an imperfect sensing scenario, we believe that the prior knowledge encoded in ORCA's velocity obstacle would also provide a strong guidance to a robust multi-agent collision avoidance policy. Our idea is to design a loss function (called "ORCA loss") based on the ORCA velocity obstacle, which is used to project the actor network's output into a reciprocal collision avoidance zone. In particular, ORCA velocity obstacle is made by a set of half-planes in the velocity space associated with each neighbor of an agent. The agent needs to optimize its navigation velocity subject to the constraint that the velocity shall fall on the negative side of all half-planes, by using linear programming. For an agent's velocity output from an actor network, its ORCA loss is computed as the distance to the closest ORCA plane of this agent's velocity obstacle. Because it has been proved that the ORCA zone is theoretically collision-free, the actor network can then converge to an optimal situation where most outputs are in the collision-free zone after several epochs training.

We implemented the ORCA loss guided pre-training in the first training stage, where the algorithm collects observations, rewards and other states information from the simulator, and computes ORCA half-planes for each agent, then stores these training data into replay buffers. The critic network will be trained as usual, while the actor network will be trained by minimizing ORCA loss, which is formally defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{p\in P}(\max(0, -||\mathbf{a}_i, p||)) + (\mathbf{a}_i - \mathbf{v}_i^{prefer}), \quad (3)$$

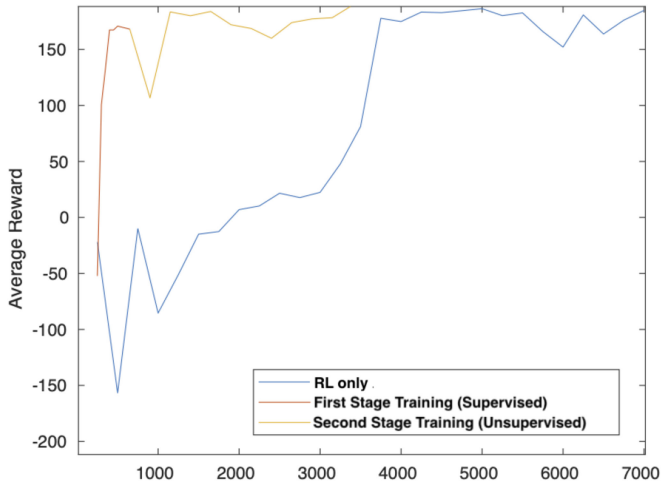

Fig. 4. Average rewards for each step during the training process.

where N is the number of agents in scenarios, P denotes the ORCA plane for each agent, $\mathbf{a}_i$ is the predicted action of agent i from actor network, $||\mathbf{a}_i, p||$ is the distance between $\mathbf{a}_i$ and ORCA half-plane p, $\mathbf{v}^{prefer}$ is the preferred velocity from agent's current position to its goal.

When the average rewards of each epoch is stable during the first training stage, the actor network and critic network both converge. Although the actor network at this moment can generate paths that are roughly collision-free, the performance of this policy is restricted by ORCA. To further increase the performance, we need to get rid of ORCA and optimize the policy by maximizing the reward function directly. Hence, we switch to the second unsupervised training stage, where we use the deep deterministic policy gradient algorithm (DDPG) [28] for training.

Our two stage training method adapts the centralized training, decentralized inference paradigm. Specifically, all training data collected by multiple robots in simulation is used to train the control policy shared by all agents in training process. For the inference process, the control policy can be deployed to every single robot individually. This two stage training method is summarized in Algorithm 1.

As we can observe in Fig. 4, the average rewards increase rapidly during the first stage training period, while the traditional RL without two-stage training (RL-only) converges very slowly. Because the change of optimization goal, the average reward curve drops slightly when we switch to the second training stage, but the curve increases to a higher point very soon. Our two stage training process converges fast after about $1.5K$ steps of training, while the traditional RL takes twice the time to converge with more than $3K$ training steps.

IV. Experiments and Results

In this section, we first describe the hyper-parameters and setups in our simulator and training process. Then we illustrate the performance metrics and simulation scenarios we used for evaluation. Based on that, we compare our method with other approaches and demonstrate both the quantitative and qualitative results. Besides, we conduct additional experiments to test the

Algorithm 1: Two Stage Training of Distributed DDPG Algorithm for Multi-UAV System.

Randomly initialize the critic network $\mathbf{Q}(\mathbf{s}, \mathbf{a}|\theta^{\mathbf{Q}})$ and actor network $\mu(\mathbf{s}|\theta^{\mu})$ with weight θ^{Q} and θ^{μ}.
Initialize target network Q' and μ' with critic and actor network initialization weights $\theta^{Q'} \leftarrow \theta^{Q}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$.
Initialize reply buffer R
for episode $= 1, 2, \ldots M$ **do**
  Receive initial observation states $s_{1,i}$
  for $t = 1, 2, \ldots T$ **do**
    for robot $i = 1, 2, \ldots ...N$ **do**
      Select actions $a_{t,i}$ according to the current policy.
    end for
    Execute actions $a_{t,i}$ in the simulator, observe rewards $r_{t,i}$ and obtain new observations $s_{t+1,i}$
    Store transitions $(s_{t,i}, a_{t,i}, r_{t,i}, s_{t+1,i})$
    // sample training data from replay buffer, note that t under this line denotes the t of sampled data
    Sample a random batch of transitions (s_t, a_t, r_t, s_{t+1}) from R
    Set $y_t = r_t + \gamma Q'(s_{t+1}, \mu'(s_{t+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic network by minimizing the loss: $\mathcal{L} = \frac{1}{N}\sum_t (y_t - Q((s_t, a_t|\theta^{Q}))^2$
    if episode $< E$ **then**
      // First Stage: Supervised by ORCA principle
      Update actor policy by minimizing the ORCA plane loss:
      $\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}\sum_{p\in P}(\max(0, -||\mathbf{a}_i, p||)) + (\mathbf{a}_i - \mathbf{v}_i^{prefer})$
    else
      // Second Stage: Policy gradient training
      Update the actor policy using the sampled policy gradient:
      $\nabla_{\theta\mu}\mathcal{J} \approx \frac{1}{N}\sum_t \nabla_a Q(s, a|\theta^{Q})|_{s=s_t, a=\mu(s_t)} \nabla_{\theta\mu}\mu(s|\theta^{\mu})|_{s_t}$
    end if
    if episode$\%\tau == 0$ **then**
      Update target networks:
      $\theta^{Q'} \leftarrow \theta^{Q}$; $\theta^{\mu'} \leftarrow \theta^{\mu}$
    end if
  end for
end for

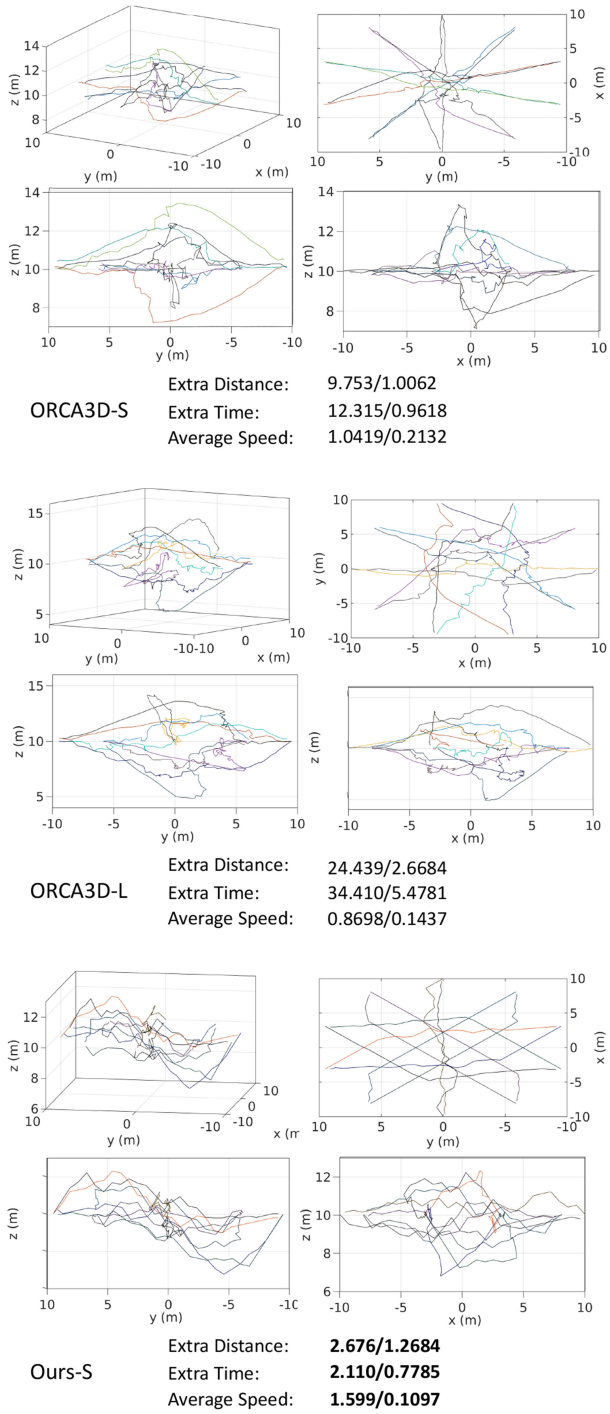

Fig. 5. Illustration of the trajectories of ORCA3D-S, ORCA3D-L and our policy under circle scenario (noise level = 0.5, number of agents = 10) in three-view drawing and perspective drawing. The performance metrics are shown as mean/std among all test cases. Our policy produce more cooperative behavior and economical trajectory comparing to ORCA3D.

robustness and generalization capability of our method in large-scale crowd scenarios.

A. Simulation Environment and Training Setup

To build simulation environment for multi-robot training, we implement our algorithm with ROS kinetic and Gazebo 9.0 [31]. We build our policy model based on TensorFlow [32] and train it on a PC with Intel i7-8700 k and NVIDIA GTX 1080ti. During the training process, the learning rate is fixed as $1e{-}3$. We set the parameters in our Algorithm 1 as $\gamma = 0.99$, $\tau = 400$ and $E = 3$. The simulation environment is set as a ball region whose radius is 50 m. The neighbor distance of ORCA [30] and our algorithm are both set as 10 m. Thus the agent will ignore the neighbors beyond such distance in its observation. We also limit the agent's maximum neighbor number to be 5. In other words, only the nearest 5 neighbors information will be leveraged for navigation and collision avoidance by the agent.

In order to simulate imperfect sensing environment, we add Gaussian noise on the agent's observation, i.e., its neighbors'

TABLE I
PERFORMANCE METRICS (SHOWN AS MEAN/STD) EVALUATED FOR DIFFERENT METHODS ON DIFFERENT SCENARIOS WITH GAUSSIAN NOISE LEVEL = 1 AND NUMBER OF AGENTS = 10. THE BEST RESULTS IN EACH CATEGORY ARE IN **BOLD**. SEE SEC. IV-B FOR MORE INFORMATION ABOUT EVALUATION METRICS

$\sigma = 1$, Agent Number=10						
Scenarios	Metric	ORCA3D-S	ORCA3D-L	RL-only	Ours-S	Ours-B
Circle	Success Rate	0.96/0.894	**0.985/0.051**	0.9/0.106	**0.985/0.082**	0.98/0.185
	Extra Time	21.590/2.210	57.019/5.941	2.269/0.754	**2.136/0.975**	3.578/1.586
	Extra Distance	19.690/1.888	57.152/4.674	**3.077/0.180**	3.976/0.435	4.852/1.643
	Average Speed	1.189/0.008	0.981/0.003	1.620/0.129	**1.691/0.113**	1.607/0.136
Ball	Success Rate	0.98/0.044	**1/0**	0.90/0.091	**1/0**	**1/0**
	Extra Time	29.901/5.649	104.088/19.953	**2.141/1.065**	2.569/0.958	2.585/0.998
	Extra Distance	23.778/2.332	60.544/15.684	**3.096/0.754**	3.625/1.862	3.893/1.268
	Average Speed	1.189/0.007	0.828/0.150	1.645/0.107	**1.672/0.128**	1.618/0.115
Random	Success Rate	0.983/0.040	**1/0**	**1/0**	**1/0**	**1/0**
	Extra Time	15.682/3.001	150.299/7.352	2.754/0.536	**1.690/0.383**	2.272/1.538
	Extra Distance	16.965/3.827	70.128/20.333	**1.698/0.835**	1.765/0.431	1.765/0.431
	Average Speed	1.217/0.221	0.551/0.091	1.532/0.028	**1.706/0.139**	1.651/0.103

positions and velocities: $\mathbf{o}^{\Phi} = \mathbf{o} + \Phi, \Phi \sim \mathcal{N}(0, \sigma^2)$, where ob is the original observation of each agent, Φ represents the Gaussian noise follow the standard normal distribution, and $\mathbf{o}^{\Phi}$ denotes the actual noisy observation received by the agent.

During the experiments, we simulate two different cases of the observation noise to train our policy model. In the first case, the Gaussian noise Φ is generated based on a known parameter σ. In particular, we consider two noise levels, including $\sigma = 0.5$ and $\sigma = 1$, and train different policy models for different noise levels respectively. We refer to our models trained with known noise levels as Ours-S. In the second case, we generate the Gaussian noise Φ based on a unknown parameter σ. To achieve this, we randomly sample the noise parameter σ from the range [0,1] and generate each noisy observation independently based on different σ samples. Then we train a single policy model based on the observations containing a wide variety of unknown noise levels. We refer to such unified model trained under blind noise as Ours-B. For performance evaluation, we test all our models and other baseline approaches under the same noise level condition as in training Ours-S.

For ORCA policy, we found that its performance is influenced by a series of hyper-parameters. The most important one is the "Safety Space" parameter, which denotes the minimum distance of each agent keeping away from its neighbors. When we increase the "Safety Space" value in ORCA, the agent will prefer trajectories farther away from other agents and thus the probability of collision will be lower. However, the travel distance and time cost will then increase significantly, and vice versa. Thus, one needs to balance the safety and efficiency of the algorithm when determining the "Safety Space" parameter. In our experiments, we set "Safety Space" parameter based on the noise level in testing scenarios. Specifically, we set up two ORCA policies with different "Safety Space" values, including the ORCA3D-small-safety-space (ORCA3D-S) with safety space $= 0.5\sigma$, and the ORCA3D-large-safety-space (ORCA3D-L) with safety space $= 2\sigma$.

B. Performance Metrics and Experiment Scenarios

For performance comparison between our approach and other methods, we present the following metrics for quantitative evaluation:

1) Success Rate: the percentage of agents that successfully reach their own goals in the time limit without any collisions.
2) Extra Time: the average extra travel time of agents spending on their planned trajectories compared with going straight toward their goals.
3) Extra Distance: the average extra travel distance of agents spending on their planned trajectories compared with going straight toward their goals.
4) Average Speed: the average speed of all agents during testing.

We employ three types of testing scenarios in our experiment:

1) Circle Scenario: All UAVs are located uniformly on a circle at a specific altitude. Their goals will be set at the opposite side on the circle, as shown in Fig. 5.
2) Random Scenario: The start and goal positions of all UAVs are initialized randomly in the simulation environment.
3) Ball scenario: All UAVs are placed randomly on the surface of a ball, and their goals are randomly placed on a concentric ball surface with a larger radius.

C. Comparison on Various Scenarios

We compare our policy with the state-of-the-art agent-level collision avoidance algorithm, ORCA policy [30] (ORCA3D-S and ORCA3D-L), and traditional RL policy without two stage training (RL-only), whose network structure is same as our proposed two-stage RL.

Fig. 5 illustrates the trajectory of ORCA3D and our learned policy in the circle scenario. We observe that ORCA3D-L with a larger safety space behaves less economically and is more messy than ORCA3D-S with a small safety space. It can be also seen that UAVs using our learned policy are able to move toward their goals faster than ORCA3D. Meanwhile the trajectory of our policy is uniform and symmetric, implying that our policy has learned to produce more cooperative behaviors and economical trajectories than ORCA3D.

Tables I and II show the performance evaluated for different methods on different scenarios with different levels of Gaussian noises and the best results are highlighted in **bold**. It can be seen that the proposed policy (Ours-S) yields the best performance in most cases. For RL policy without two-stage training

TABLE II
PERFORMANCE METRICS (SHOWN AS MEAN/STD) EVALUATED FOR DIFFERENT METHODS ON DIFFERENT SCENARIOS WITH GAUSSIAN NOISE LEVEL = 0.5 AND NUMBER OF AGENTS = 10. UNDER NOISE LEVEL = 0.5 TESTING SETUP, ALL THE METHODS IN EVERY SCENARIOS ACHIEVE 100% SUCCESS RATE, SO THE SUCCESS RATE METRIC IN THIS TABLE IS NOT LISTED. THE BEST RESULTS IN EACH CATEGORY ARE IN **BOLD**

$\sigma = 0.5$, Agent Number =10						
Scenarios	Metric	ORCA3D-S	ORCA3D-L	RL-only	Ours-S	Ours-B
Circle	Extra Time	12.315/0.961	34.410/5.478	2.159/0.483	**2.110/0.778**	2.282/0.595
	Extra Distance	9.753/1.006	24.439/2.668	2.777/0.094	**2.676/1.268**	3.249/1.008
	Average Speed	1.087/0.0206	0.869/0.1438	**1.607/0.119**	1.599/0.109	1.601/0.092
Ball	Extra Time	13.833/0.779	27.797/2.718	1.668/0.357	**1.582/0.437**	1.983/0.517
	Extra Distance	11.226/0.869	15.301/2.811	**1.726/0.301**	1.896/0.827	2.581/0.882
	Average Speed	1.080/0.007	0.897/0.068	1.594/0.117	**1.624/0.087**	1.600/0.080
Random	Extra Time	5.257/0.951	25.439/9.122	**1.034/0.257**	1.567/0.465	1.472/0.583
	Extra Distance	2.692/0.984	11.192/3.026	**1.173/0.582**	2.108/0.843	1.965/0.738
	Average Speed	1.042/0.021	0.709/0.209	1.600/0.105	**1.601/0.110**	1.598/0.086

TABLE III
PERFORMANCE METRICS ON THE CIRCLE SCENARIO WITH DIFFERENT TYPE OF NOISE, WITH 10 UAVS, NOISE LEVEL $\sigma = 1$. THE BEST RESULTS ARE IN **BOLD**

Method	ORCA3D-S	ORCA3D-L	Ours-S
Success Rate	0.96/0.90	**1/0**	0.98/0.05
Extra Distance	29.87/3.68	78.21/7.06	**19.38/2.59**
Extra Time	26.55/1.09	62.30/5.85	**8.78/1.46**
Average Speed	1.21/0.21	1.14/0.13	**1.43/0.08**

(RL-only), it outperforms our two-stage training RL policy in some metrics. This is because the first supervised training stage in Ours-S adopts ORCA3D's prior knowledge, In this way, our policy not only maximizes the reward function, but also tries to follow the reciprocal collision avoidance principle, which may be sub-optimal in some situations and thus may result in a longer trajectory than RL-only policy. However, our policy outperforms RL-only in success rate over all scenarios, implying that the prior knowledge from reciprocal collision avoidance principle not only helps the RL training converge fast, but also makes the policy safe and robust in unseen scenarios.

The performance of the single policy for unknown noise levels (Ours-B) is also shown in Tables I and II. As we can observe, Ours-B also achieves better performance than ORCA3D variants. In addition, unlike ORCA3D that needs to manually adjust hyper-parameters to survive in different noise levels, Ours-B can easily adapt to different unknown noise levels without parameter tuning in advance.

To check the policy's generalization capability to scenarios with different noises, we also evaluated the performance of different approaches in the Circle scenario under the uniform distribution noise $\frac{1}{4\sigma}[-2\sigma, 2\sigma]$ with $\sigma = 1$ and the result is shown in Table III. We can observe that the performance of all methods becomes worse than in Table I, because the noise type is very different between training and testing. However, our proposed method still outperforms baselines and survive in this new scenario with unseen noises.

D. Large-Scale Scenario Experiments

To test the performance of our proposed method on large-scale crowded scenarios, we simulate 200 UAVs in Gazebo simulator, as shown in Fig. 6. The result is shown in Table IV, which

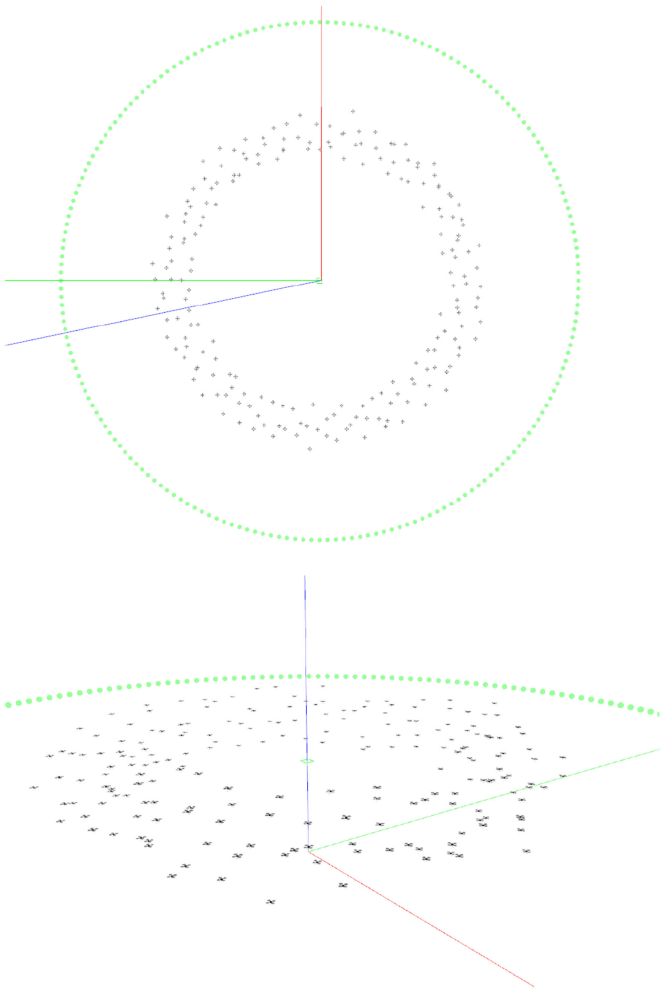

Fig. 6. Large scale scenarios with 200 UAVs in Gazebo Simulator. UAVs (black) are moving forward to their targets (green).

TABLE IV
PERFORMANCE METIRCS (SHOWN AS MEAN/STD) ON LARGE-SCALE TESTING SCENARIOS, WITH 200 UAVS, NOISE LEVEL = 1. THE BEST RESULTS ARE IN **BOLD**

Methods	ORCA3D-S	ORCA3D-L	Ours-S
Success Rate	0.86/0.0259	0.93/0.017	**0.95/0.059**
Extra Distance	31.876/9.753	70.913/22.674	**20.393/5.180**
Extra Time	28.774/5.801	170.399/15.001	**18.974/3.821**
Average Speed	1.303/0.310	0.971/0.024	**1.622/0.113**

demonstrates that our learned policy can be directly applied to large-scale crowded unseen scenarios and generates better trajectory comparing to baseline methods.

V. Conclusion

In this letter, we presented a novel decentralized agent-level collision avoidance policy using reinforcement learning for multi-UAV system without perfect sensing assumption. We also introduced a two-stage training method to make our proposed policy more robust and converge fast. The proposed policy with two-stage training has been demonstrated several advantages over ORCA3D policy and RL policy without two-stage training in terms of success rate, trajectory length and cost of time. Benefited from strong generalization capacity of the deep neural network, our policy can generalize to large-scale unseen scenarios without fine-tuning or retraining. In contrast to benchmark methods which need to manual adjust hyper-parameters for certain observation noise level, our single policy also has the capacity to handle unknown noise level.

Our proposed two-stage RL policy currently only takes into account the observation at the current time and thus may have oscillation when the scenario changes abruptly due to moving obstacles. Some recent work such as [33] has demonstrated that it is important to use recurrent neural network structure to encode historical information during the navigation for reducing oscillations, and we would love to explore different recurrent network structures to solve the oscillation problem.

References

[1] H. Liu, Y. Tian, F. L. Lewis, Y. Wan, and K. P. Valavanis, "Robust formation tracking control for multiple quadrotors under aggressive maneuvers," *Automatica*, vol. 105, pp. 179–185, 2019.

[2] H. Liu, W. Zhao, S. Hong, F. L. Lewis, and Y. Yu, "Robust backstepping-based trajectory tracking control for quadrotors with time delays," *IET Control Theory Appl.*, vol. 13, no. 12, pp. 1945–1954, 2019.

[3] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Proc. Auton. Robots Agents*, 2007, pp. 9–16.

[4] S. Thrun and Y. Liu, "Multi-robot slam with sparse extended information filers," in *Proc. Int. Soundex Reunion Registry*, 2005, pp. 254–266.

[5] J. S. Bellingham, M. Tillerson, M. Alighanbari, and J. P. How, "Co-operative path planning for multiple uavs in dynamic and uncertain environments," in *Proc. 41st IEEE Conf. Decis. Control*, 2002, vol. 3, pp. 2816–2822.

[6] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2012, pp. 477–483.

[7] K. H. Movric and F. L. Lewis, "Cooperative optimal control for multi-agent systems on directed graph topologies," *IEEE Trans. Autom. Control*, vol. 59, no. 3, pp. 769–774, Mar. 2014.

[8] W. Liu, W. Gu, W. Sheng, X. Meng, Z. Wu, and W. Chen, "Decentralized multi-agent system-based cooperative frequency control for autonomous microgrids with communication constraints," *IEEE Trans. Sustain. Energy*, vol. 5, no. 2, pp. 446–456, Apr. 2014.

[9] M. Abouheaf, W. Gueaieb, and F. Lewis, "Online model-free reinforcement learning for the automatic control of a flexible wing aircraft," *IET Control Theory Appl.*, vol. 14, no. 1, pp. 73–84, 2020.

[10] L. C.-Macías, R. A.-López, R. de la Guardia, J. I. P.-Vilchis, and D. G.-Gutiérrez, "Autonomous navigation of mavs in unknown cluttered environments," 2019, *arXiv:1906.08839*.

[11] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, *International Symposium on Robotics Research*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Reciprocal n-Body Collision Avoidance, pp. 3–19.

[12] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *Robot. Autom. Lett.*, vol. 2, no. 2, pp. 656–663, 2017.

[13] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 6252–6259.

[14] T. Fan, P. Long, W. Liu, and J. Pan, "Fully distributed multi-robot collision avoidance via deep reinforcement learning for safe and efficient navigation in complex scenarios," 2018, *arXiv:1808.03841*.

[15] A. Irpan, "Deep reinforcement learning doesn't work yet," https://www.alexirpan.com/2018/02/14/rl-hard.html, 2018.

[16] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Proc. Artif. Intell. Statist.*, 2009, pp. 153–160.

[17] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Conf. Neural Inf. Process. Syst.*, 2000, pp. 1057–1063.

[18] M. Bojarski *et al.*, "End to end learning for self-driving cars," 2016, *arXiv:1604.07316*.

[19] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proc. Int. Conf. Intell. Robots Syst.*, 2017, pp. 31–36.

[20] Y. Gui *et al.*, "Airborne vision-based navigation method for uav accuracy landing using infrared lamps," *J. Intell. Robot. Syst.*, vol. 72, no. 2, pp. 197–218, 2013.

[21] C.-S. Y. C.-S. Yoo and I.-K. A. I.-K. Ahn, "Low cost GPS/INS sensor fusion system for UAV navigation," in *Proc. Digit. Avionics Syst. Conf.*, 2003, vol. 2, pp. 8–A.

[22] F. Gao, L. Wang, B. Zhou, L. Han, J. Pan, and S. Shen, "Teach-repeat-replan: A complete and robust system for aggressive flight in complex environments," 2019, *arXiv:1907.00520*.

[23] D. Helbing, I. Farkas, and T. Viscek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, 2000.

[24] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2008, pp. 1928–1935.

[25] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 285–292.

[26] E. Tolstaya, F. Gama, J. Paulos, G. Pappas, V. Kumar, and A. Ribeiro, "Learning decentralized controllers for robot swarms with graph neural networks," 2019, *arXiv:1903.10527*.

[27] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Conf. Neural Inf. Process. Syst.*, 2000, pp. 1008–1014.

[28] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.

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

[30] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Berlin, Germany: Springer, 2011, pp. 3–19.

[31] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2149–2154.

[32] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, *arXiv:1603.04467*.

[33] L. Wang, W. Zhang, X. He, and H. Zha, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 2447–2456.