Multi-Robot Collision Avoidance with Map-based Deep Reinforcement Learning

Shunyi Yao*, Guangda Chen*, Lifan Pan, Jun Ma, Jianmin Ji† and Xiaoping Chen
School of Computer Science and Technology, University of Science and Technology of China
Hefei, 230026, China
Email: {ustcysy, cgdsss, lifanpan, markjun}@mail.ustc.edu.cn, {jianmin, xpchen}@ustc.edu.cn.
*These authors contributed equally to the work, †the corresponding author.

***Abstract*—Multi-robot collision avoidance in a communication-free environment is one of the key issues for mobile robotics and autonomous driving. In this paper, we propose a map-based deep reinforcement learning (DRL) approach for collision avoidance of multiple robots, where robots do not communicate with each other and only sense other robots' positions and the obstacles around them. We use the egocentric grid map of a robot to represent the environmental information around it, which can be easily generated by using multiple sensors or sensor fusion. The learned policy generated from the DRL model directly maps 3 frames of egocentric grid maps and the robot's relative local goal positions into low-level robot control commands. Compared to other methods, the map-based approach is more robust to noisy sensor data and does not require the expensive movement data of other robots, like velocities, accelerations and paths. We first train a convolutional neural network for the navigation policy in a simulator of multiple mobile robots using proximal policy optimization (PPO), where curriculum learning strategy is used to accelerate the training process. Then we deploy the trained model to real robots to perform collision avoidance in their navigation. We evaluate the approach with various scenarios both in the simulator and on three differential-drive mobile robots in the real world. Both qualitative and quantitative experiments show that our approach is efficient with high success rate. The demonstration video can be found at https://youtu.be/jcLKlEXuFuk.**

***Index Terms*—multi-robots collision avoidance, reinforcement learning, egocentric grid map**

I. Introduction

Multiple robot systems are widely used in various robotics applications, such as autonomous warehouse, search and rescue operations, and autonomous driving. Multi-robot collision avoidance is a key issue [1] for these applications, which allows each robot to reach its target position from a starting place while avoiding collisions with other robots and obstacles. The dynamic interactions between diverse autonomous robots and the uncertainty in the environment make the problem highly challenging. To improve the reliability and performance of multiple robot systems, it is vital to develop an approach for efficient and safe collision avoidance of multiple robots.

Multi-robot collision avoidance methods can generally be classified into two categories: centralized methods and decentralized methods. A centralized method usually provides a center server to determine each robot's action using an optimization algorithm after collecting all the relevant information [2], [3]. However, such a system is fragile due to its requirements of a static central server and the reliable communication between the server and all robots. Moreover, it becomes more and more challenging for the system to overcome the increasing number of robots [3].

This work is partially supported by the 2030 National Key AI Program of China 2018AAA0100500 and Guangdong Province Science and Technology Plan Projects (No. 2017B010110011).

Different from the centralized method, a decentralized method allows each robot to perform collision avoidance replying on its local perception of the surrounding environment, which no longer requires a central server. Most existing decentralized methods require the movement data of other robots, like velocities, accelerations and paths. For instance, Velocity Obstacles (VO) [4] based approaches are widely used, which require velocities of other robots to induce the VO for the selection of collision-free motion. Optimal reciprocal collision avoidance (ORCA) [5] is one of the most successful VO based approaches, which guarantees collision-free trajectories except for densely packed conditions. ORCA has also been extended to nonholonomic robots (NH-ORCA) [1], [6], [7]. These approaches usually require perfect perception to extract information on shapes and speeds of other robots, which is not easy to achieve in most real-world applications. On the other hand, a large amount of human-engineered parameters are required for these approaches to achieve good performances in various scenarios, which limits their scalability in practice.

Recently, deep learning based approaches have received considerable attentions in robot navigation. In particular, imitation learning has been used to train an end-to-end network to predict the steering angle of the car with center/front image as input [8]. Liu et al. [9] train a neural network to produce the collision avoidance policy for a single robot that maps egocentric grid maps to robot control commands. ORCA datasets have also been used to train a multi-robot collision avoidance policy [10]. However, supervise learning approaches need large amounts of labeled data to achieve a good performance.

Inspired by ORCA, Chen et al. [11] train a collision avoidance policy using deep reinforcement learning (DRL) with the motion data of other robots as parts of states of the DRL model, which maps the motions states of the robot and others to its control commands. The approach requires perfect perception of the environment and high computational effort to generate these motion data, which limits its applications. Later the work is extend by using recurrent neural network to receive multi-robot information [12]. However, perfect perception and high computational effort are still required for the extended approach.

On the other hand, Fan et al. [13], [14] introduce an end-to-end DRL navigation approach by using raw sensor data, i.e., 2D laser scan data, the target position, and the velocity of the robot to predict collision-free motion commands. Notice that, expensive motion data of other robots are not required in the system. A one-dimensional convolutional neural network is used to process the 2D laser scan data. After training in several simulation scenarios, the approach produces a policy that can generate collision-free trajectories for the robot However, the approach is only restricted to 2D laser sensors.

Egocentric gird maps of a robot are used to represent the environmental information around it in our previous work [15], which are considered as inputs of the trained network in deep Q-learning to predict collision-free motion commands for a single robot in static environments. Egocentric gird maps, i.e., local costmaps[1], have been widely applied in robot navigation [9], [16], which can be easily generated by using multiple sensors or sensor fusion with strong noise resistance.

In this paper, we extend our work and propose an end-to-end decentralized DRL multi-robot collision avoidance approach in a communication-free environment. The learned policy directly maps 3 frames of egocentric grid maps and the robot's relative target position into low-level robot control commands. We first train a convolutional neural network of the navigation policy in a simulator of multiple mobile robots using proximal policy optimization (PPO) [17], where curriculum learning strategy [18] is used to accelerate the training process. Then we deploy the trained model to real robots to perform collision avoidance in their navigation. Notice that, egocentric grid maps can be easily generated by using multiple sensors or sensor fusion with strong noise resistance. Then it is easier to deploy the approach to a real robot. We evaluate the approach with various scenarios both in the simulator and on three differential-drive mobile robots in the real world. Both qualitative and quantitative experiments show that our approach is efficient with high success rate.

Our main contributions are summarized as follow:

- We propose a map-based DRL multi-robot collision avoidance approach in a communication-free environment. Egocentric grid maps are used to represent the environmental information around the robot.
- We train the neural network of the navigation policy in a simulator of multiple mobile robots using PPO, which can be deployed to real robots to perform collision avoidance in their navigation.
- We evaluate our approach with various scenarios both in the simulator and on three differential-drive mobile robots in the real world. Both qualitative and quantitative experiments show that our approach is efficient and safe.

[1]http://wiki.ros.org/costmap_2d

The rest of this paper is organized as follows. The problem is formulated in Section II. Section III describes our approach. Section IV provides experimental details and results both in the simulator and on robots in the real world, followed by conclusions in Section V.

II. Problem Formulation

Multi-robot collision avoidance requires a group of N mobile robots with nonholonomic or holonomic constraints to perform corresponding trajectories to their target places while avoiding collisions with each other and with obstacles in their environment.

To simplify the discussion, we assume that each robot is a circular nonholonomic differential drive mobile robot with the same radius R. At each time step t, each robot i $(1 \le i \le N)$ first receives its sensing data s_i^t, then chooses an action a_i^t to move towards its local goal g_i^t, where g_i^t contains robot's relative position and orientation (pose) in a path that is generated by its global planner to reach its target. In specific, we consider

$$\begin{aligned} \mathbf{M}_i^t &= f(s_i^t), \\ a_i^t &= \pi_\theta(\mathbf{M}_i^t, g_i^t), \end{aligned}$$

where $\mathbf{M}_i^t$ denotes the egocentric grid map generated from the current sensing data s_i^t by the function f to specify the 2D map of the obstacles that can be obtained by robot's sensors in a plane horizontal to the ground, and π_θ denotes the policy specified by parameters θ to choose an action a_i^t based on the current egocentric grid map $\mathbf{M}_i^t$ and the local goal g_i^t. In this paper, we denote action $a_i^t = (v_i^t, \omega_i^t)$, where v_i^t is the linear velocity and ω_i^t is the angular velocity that the robot i needs to perform until the next time step $t+1$.

In multi-robot collision avoidance, each robot i moves from the starting position $\mathbf{p}_i^0$ to the target position $\mathbf{p}_i^g$, while avoiding collisions with each other and with obstacles B_k $(1 \le k \le M)$ in the environment. We intend to minimize the expectation of the arrival time t_i^g for every robot $1 \le i \le N$ under the constraint that no collision occurs. In specific,

$$\begin{aligned} &\arg\min_\theta \ \mathbb{E}[\sum_{i=1}^{N} t_i^g \mid \pi_\theta], \\ &\text{s.t. for each } 1 \le i \le N,\ 1 \le j \le N,\ i \ne j, \text{ and } 1 \le k \le M, \\ &\qquad \mathbf{p}_i^{t+1} = \mathbf{p}_i^t + \Delta t \cdot \pi_\theta(\mathbf{M}_i^t, g_i^t) \text{ and } \mathbf{p}_i^{t_i^g} = \mathbf{p}_i^g, \\ &\qquad\qquad \left\|\mathbf{p}_i^t - \mathbf{p}_j^t\right\| > 2R \text{ and } \left\|\mathbf{p}_i^t - \mathbf{B}_k\right\| > R. \end{aligned}$$

Notice that, the second line of conditions requires each robot reaching its target and the third line requires no collision. In the following, we introduce our approach towards the intent.

III. Approach

We begin this section by describing key ingredients of the proximal policy optimization (PPO) reinforcement learning algorithm for multiple robots. Then, we describe the details on the network architecture and the training process for the multi-robot collision avoidance policy.

A. Proximal policy optimization with multiple robots

There are mainly two kinds of reinforcement learning algorithms: value function based and policy gradient methods. Value function based methods attempt to optimize the performance by training the value function $V(s)$ or the action-value function $Q(s,a)$ for a state s and an action a. Normally, these methods enjoy better sampling efficiency. However, multi-robot collision avoidance requires a stochastic policy for each robot, where value function based methods may be fragile. In contrast, policy gradient methods directly optimize the stochastic policy, which is more robust in this case.

Generally, policy gradient methods optimize a stochastic policy π_θ by maximizing the objective function $J(\pi_\theta)$ using stochastic gradient ascent. In specific,

$$\nabla_\theta J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{\mathrm{E}}\left[\sum_{t=0}^{T}\nabla_\theta \log \pi_\theta\left(a_t|s_t\right) A^{\pi_\theta}\left(s_t,a_t\right)\right],$$
$$\theta_{k+1} = \theta_k + \alpha\nabla_\theta J\left(\pi_{\theta_k}\right),$$

where τ is a trajectory and A^{π_θ} is the advantage function for the policy π_θ.

Vanilla Policy Gradient method [19] only works on-policy, i.e., the value of an action estimated by the critic must have been produced recently by the actor, otherwise the bias would increase dramatically. This prevents the use of an experience replay memory as in DQN [20] to stabilize learning. Then the method is unstable for long trajectories, which leads to poor sampling efficiency.

Proximal policy optimization (PPO) [17] is to introduce a surrogate objective which avoids performance collapse by guaranteeing monotonic policy improvement. It updates and maintains two networks in the training process, i.e., policy network π_θ, which is used to predict the next action based on the input state, and value network V_ϕ, which is used to estimate the state's expected return. There are two primary variants of PPO: PPO-penalty and PPO-clip. We apply PPO-clip in this paper. In specific, PPO-clip updates policies via

$$\theta_{k+1} = \arg\max_\theta \mathrm{E}_{s,a\sim\pi_{k_k}}\left[L(s,a,\theta_k,\theta)\right], \tag{1}$$

and L is defined as

$$L(s,a,\theta_k,\theta) = \min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), \quad g\left(\varepsilon, A^{\pi_{\theta_k}}(s,a)\right)\right),$$
$$g(\varepsilon,A) = \begin{cases}(1+\varepsilon)A & A\geq 0,\\ (1-\varepsilon)A & A<0,\end{cases}$$

where ε is the clip function ration and A is the advantage function that is computed by generalized advantage estimation (GAE) [21], i.e.,

$$A_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \cdots + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{2}$$

where $\delta_t = r_t + \gamma V_\phi(s_{t+1}) - V_\phi(s_t)$, the discount factor $0\leq\gamma<1$, and the parameter $0\leq\lambda\leq 1$.

In the following, we introduce details of our PPO based approach, including the observation space, the action space, the reward function, and the network architecture.

1) Observation space: We use the latest three frames of egocentric grid maps $\langle \mathbf{M}_i^{t-2}, \mathbf{M}_i^{t-1}, \mathbf{M}_i^t\rangle$ and relative local goals $\langle g_i^{t-2}, g_i^{t-1}, g_i^t\rangle$ to be our input $\mathbf{o}_i^t$. In specific, $\mathbf{M}_i^t$ is generated by a laser scan with 180 degrees horizontal Field of View (FOV), which encodes positions of obstacles and other robots. The local goal $g_i^t = (x_i^t, y_i^t, \alpha_i^t)$ is a tuple that includes the next position (x_i^t, y_i^t) and the orientation (pose) α_i^t in the path towards the target. Notice that, we can estimate motion information, i.e., velocities, accelerations and paths, of both the ego robot and other robots from these three frames of egocentric grid maps.

There is a value in each cell of an egocentric grid map to represent the environment. In particular, cells of value 25 denote the obstacles and other robots around the robot. Cells of value 200 denote free space and cells of value 125 denote the undetected place with the laser scan. In the center of each egocentric grid map, there is a circle which denotes the ego robot and the cells in the circle have the value 75. Moreover, we convert the egocentric grid map to a gray image to ease the set up of the network. Without loss of generality, we identity a grid map with its corresponding gray image in the paper.

2) Action space: The action space is a set of permissible velocities in continuous space. The action a_i^t of a differential robot consists of a linear velocity v_i^t and an angular velocity ω_i^t, i.e., $a_i^t = (v_i^t, \omega_i^t)$. In this paper, we set $v_i^t \in [0, 0.6]$ and $\omega_i^t \in [-0.9, 0.9]$, which can be directly performed by differential robots used in our experiment. Note that, $v_i^t \geq 0$, i.e., moving backwards is not allowed, due to lack of rear sensors.

3) Reward: The goal of the agent is to maximize its cumulative reward in reinforcement learning. In our problem, the objective is minimize the mean arrival time for each robot arriving its local goal under the collision-free constraint. Each robot has the same reward function in our setting. We use r^t to denote the reward received by the robot at time step t. We use the following reward function in this work,

$$r^t = r_g^t + r_c^t + r_s^t.$$

Note that r^t consists of three parts, r_g^t, r_c^t, and r_s^t.

In particular, r_g^t specifies the penalty when the robot goes far of its local goal. $r_{arr} > 0$ is the consistent reward when the robot arrives its local goal. We define r_g^t as:

$$r_g^t = \begin{cases} r_{arr} & \text{if } \left\|\mathbf{p}^t - \mathbf{p}_g\right\| < 0.3,\\ \varepsilon_1\left(\left\|\mathbf{p}^{t-1} - \mathbf{p}_g\right\| - \left\|\mathbf{p}^t - \mathbf{p}_g\right\|\right) & \text{otherwise,}\end{cases}$$

where $\mathbf{p}^t$ is the position of the robot at time t, $\mathbf{p}_g$ is the position of the local goal, and ε_1 is the hyper parameter that controls the penalty amount.

r_c^t specifies the penalty when the robot encounters a collision. Note that PPO considers stochastic policies, we add a penalty when the robot gets closer to obstacles or other robots. We define r_c^t as:

$$r_c^t = \begin{cases} r_{col} & \text{if collision,}\\ \varepsilon_2\left(d_{min}^t - d_{min}^{t-1}\right) & \text{if } d_{min}^t < 1,\\ 0 & \text{otherwise,}\end{cases}$$

where $r_{col} < 0$ is the consistent penalty for the collision, d^t_{min} denotes the distance between the robot and the closest obstacle or other robot boundary at time t, and ε_2 is the hyper parameter that controls this penalty amount.

At last, we apply a small negative penalty for each time step, i.e., $r^t_s < 0$, to encourage short paths.

In this work, we set r_{arr} = 500, $\varepsilon_1 = 100$, $\varepsilon_2 = 200$, $r_{col} = -500$, and $r^t_s = -5$ in the training procedure.

4) Network Architecture: The architecture of our policy network is shown in Fig. 1. The inputs of the network consist of two parts, i.e., three consecutive frames of egocentric grid maps with 48×48 gray pixels and corresponding local goals.

The network first produces feature maps for grid maps using four convolutional layers and four max pooling layers. Followed by a fully-connected layer with 512 units, these feature maps are converted to a 512 dimensional vector. The network also projects three frames of local goals to a 9 dimensional vector. Then the network combines both vectors and feeds them to two fully-connected layers with 512 units. At last, the network applies a fully-connect layer with 2 units without activations to produce the output, i.e., the mean of the linear velocity v^t and the mean of the angular velocity ω^t.

Actions of the robot are sample from the Gaussian distribution $\mathcal{N}(a^t_{mean}, a^t_{logstd})$, where a^t_{logstd} is the log standard deviation generated by a standalone network and $a^t_{mean} = (v^t, \omega^t)$. We also use a clip function to ensure that the resulting actions are valid in the action space. The value network has the same architecture as the policy network, except the last layer is modified to only output the value of the state.

B. Curriculum Learning

Curriculum learning [18] has recently been shown as an efficient strategy that can help find a better local minima and accelerate the training process. The main idea of curriculum learning is to decompose a hard learning task into several simple ones, start with the simplest task and level up the difficulty gradually.

In this work, we use Gazebo [22] to build environments with multiple robots and obstacles to train the robot for multi-robot collision avoidance. Two examples of our training scenarios are shown in Fig. 2. The first example illustrates environments that randomly choose locations for obstacles, the starting and target positions of robots, which would help the robot to be able to avoid obstacles, a.k.a., random scenario. The second example illustrate environments that randomly place robots on a circle with a random radius, a.k.a., circular scenario, which help the robot to be able to interact with other robots.

In the training process, we gradually increase the number of obstacles and robots, and increase the distance from the starting position to the target position to level up the complexity of multi-robot obstacle avoidance problem. We level up the difficulty only when robots have performed well in current environments. In particular, we divide the training process into two stages. In the first stage, the robot is trained in environments with 6 different robots and 4 different shape obstacles, and the distance from the starting position to the target position for each robot is gradually increased to level up the complexity. In the second stage, the robot is trained simultaneously in these two scenarios, which improve the robustness of the DRL policy in the real world.

Algorithm 1: Learning Procedure

```
1  Randomly initialize policy network $\pi_\theta$ and value
     network $V_\phi$;
2  Initialize PPO buffer $D_i$, $i = 1,2,\dots,N$ for each robot;
3  for epoch $e = 1,2,\dots,E$ do
4      for step $k = 1,2,\dots,K$ do
5          $a^t_i = \pi_\theta(o^t_i)$, $i = 1,2,\dots,N$
6          $(r^t_i, o^{t+1}_i) = env(a^t_i)$, $i = 1,2,\dots,N$
7          $(o^t_i, r^t_i, a^t_i) \to D_i$, $i = 1,2,\dots,N$
8          if all robots are arrived, encounter a collision,
             or get stuck then
9              reset env
10         if length $D_i > D_{max}/N$, for every $i = 1,2,\dots,N$
             then
11             compute advantages $A^t_i$ using GAE with
                 Equation (2)
12             $\hat{R}^t_i = \sum_{t'=t}^{T}\left(\gamma^{t'-t} r^{t'}_i + V_\phi(o^T_i)\right)$
13             update policy $\theta$ by Equation (1)
14             update $\phi$ by
                 $\phi_{k+1} = \arg\min_\phi \frac{1}{|D|}\sum_{o^t_i \in D}\left(V_{\phi_k}(o^t_i) - \hat{R}^t_i\right)^2$
15             clear $D$ and reset env
16             break
17     end
18 end
```

The training procedure at each stage is specified in Alg. 1. Parameters for the policy network π_θ and the value network V_ϕ are initialized randomly (Line 1). Each robot has the same size of the buffer (Line 2) and stores the collected experience in it (Line 5-7). When all buffers are full, the advantages A^t_i of each piece of experience (Line 11) and the cumulative discount rewards $\hat{R}^t_i$ are calculated (Line 12) to update θ via stochastic gradient ascent (Line 13) and update ϕ via gradient descent (Line 14). Finally, the environment is reseted after clearing the buffer (Line 15).

IV. Experiments

In this section, we evaluate our PPO based multi-robot obstacle avoidance approach in both the simulation and real world. We first specify details of our implementation including the hyper parameters, hardware and software for training the networks. Then, we quantitatively evaluate the performance of our map-based multi-robot navigation policy in various simulation scenarios and compare it with other existing approaches. We also deploy the trained model to a real robot for its multi-robot obstacle avoidance module in the navigation system and evaluate the navigation performance of the robot in real world. Both qualitative and quantitative experiments show that our approach is efficient with high success rate.

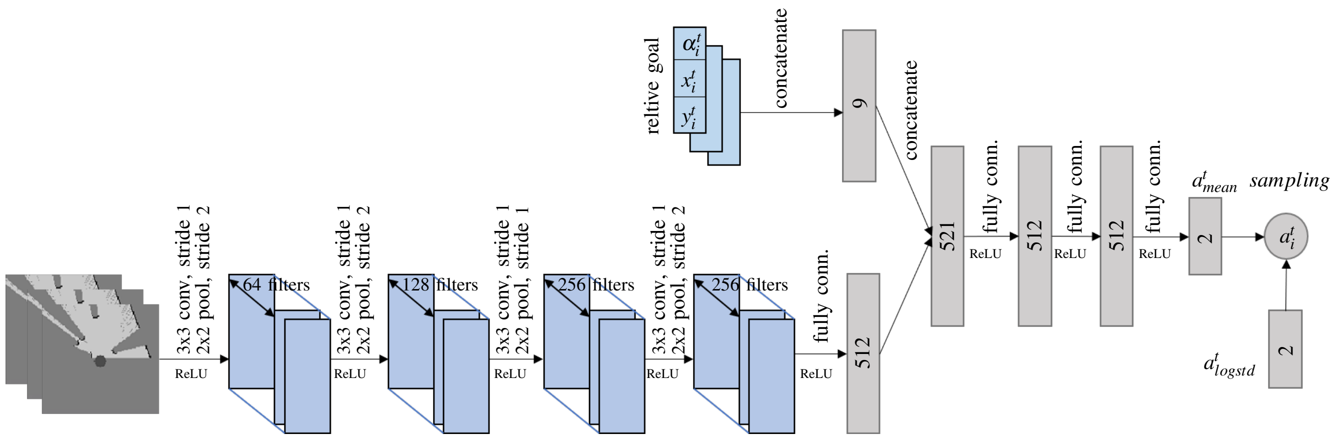
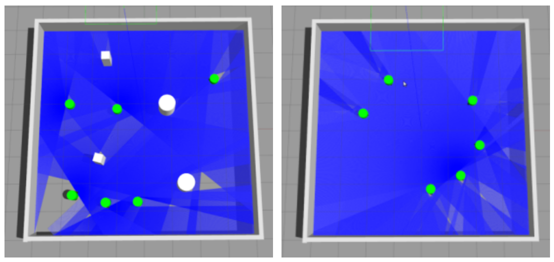
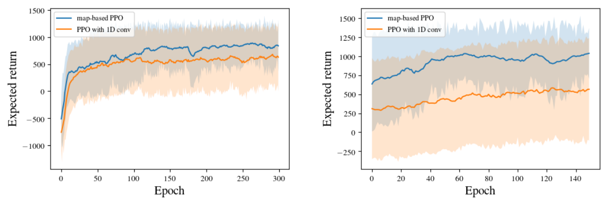


Fig. 1. The architecture of our PPO policy network. The network takes three frames of egocentric grid maps and three relative local goals as inputs, and outputs a linear velocity and an angular velocity.



(a) Random scenario (b) Circular scenario

Fig. 2. Environments of both training scenarios in our experiment. (a) The environment randomly places six robots and four obstacles with different shapes. (b) The environment randomly places six robots on a circle with a random radius. Green cylinders denote robots in the environment, white cuboids and cylinders denotes static obstacles.



(a) Reward curve at first stage (b) Reward curve at second stage

Fig. 3. The average reward curve of both PPO based approaches, i.e., our map-based PPO approach and the PPO approach with 1D convolution network, at two stages of the training procedure. (a) shows the average reward curve of both approaches at the first stage. When both approaches reach their convergences at the first stage, we start to train them for the second stage. (b) shows the average reward curve at the second stage.

A. Reinforcement Learning Setup

We implement the approach for a customized differential drive robot in a simulation environment using Gazebo. A laser scanner with 180 degrees horizontal FOV is mounted on the front of the robot as shown in Fig. 2. We trained our PPO based agent with the hyper parameters listed in Table I.

TABLE I
SYSTEM PARAMETERS

Parameter	Value
learning rate for policy	3×10^{-4}
learning rate for value	1×10^{-3}
discount factor (γ)	0.99
lambda in GAE (λ)	0.97
replay buffer size (D_{max})	2000
image size	48×48
max episode length	300
clip ratio (ε)	0.2
Robot radius (R)	0.3

We implement neural networks in TensorFlow and train them using the Adam optimizer [23]. A computer with an i9-9900k CPU and an NVIDIA Titan RTX GPU is used for the training. It takes around 36 hours to train the networks for a good performance in testing environments, where 90% of the time is used to collect experience.

Besides our map-based PPO approach, we also implemented another PPO based approach, i.e., PPO with one-dimensional convolution network, introduced in [14], and an ORCA based approach, NH-ORCA [1], [6], [7]. We trained both PPO based approaches at the same time and guaranteed them to reach convergence at each stage in the training procedure. In specific, Fig. 3 shows the comparison of the average reward curve of both PPO based approaches. It can be seen that the average reward obtained by our approach is higher than the other PPO based approach at both stages. Fig. 3(b) also shows that the converge speed of our approach is much faster than that of the PPO approach with one-dimensional convolution network.

B. Experiments on simulation scenarios

1) Performance metrics: We introduce three metrics to evaluate the performance of approaches for multi-robot obstacle avoidance as the following:

- *Success rate* $\bar{\pi}$: the ratio of the episodes that end with every robots reaching their targets without any collision.
- *Extra time* $\bar{t}$: the time required for every robots to successfully reach their targets without any collisions minus

the time for every robots to drive straight to their targets with the maximum speed.

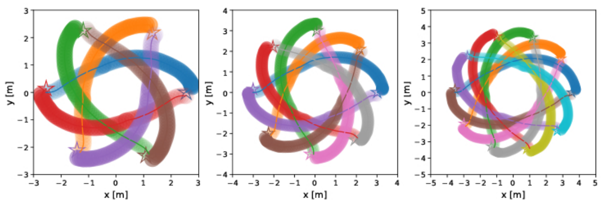
- *Average linear velocity* $\bar{v}$: the average linear velocity for every robots during the navigation.

TABLE II
METRICS FOR DIFFERENT APPROACHES IN CIRCULAR SCENARIO

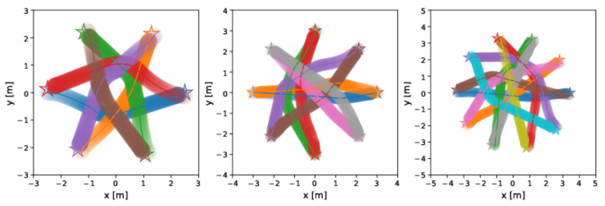
Enviroment	Method	$\bar{\pi}$	$\bar{t}$ (mean/std)	$\bar{v}$ (mean/std)
6 robots	PPO with 1D conv	1.00	3.65/0.92	0.55/0.09
	NH-ORCA	1.00	3.17/1.68	0.44/0.16
	Map-based PPO	**1.00**	**2.87/0.45**	**0.55/0.09**
8 robots	PPO with 1D conv	1.00	4.43/0.48	0.53/0.11
	NH-ORCA	0.97	4.50/2.43	0.52/0.11
	Map-based PPO	**1.00**	**3.01/0.43**	**0.56/0.06**
10 robots	PPO with 1D conv	0.98	4.89/0.76	0.54/0.10
	NH-ORCA	0.89	6.15/3.90	0.39/0.17
	Map-based PPO	**1.00**	**4.78/0.48**	**0.56/0.07**

2) Comparative experiments: Now we compare the performance of three different approaches for multi-robot obstacle avoidance, i.e., our map-based PPO approach, NH-ORCA, and the PPO approach with 1D convolution network. We first compare these approaches in environments of the circular scenario. The scenario is similar to the training scenario, except that the starting and target positions of these robots are uniformly along the circle. Table II shows the performance metrics of these approaches in the circular scenario, where metrics are calculated from the averaging results of 50 different environments in each case. Note that, our map-based PPO approach outperforms others in this scenario. In particular, both PPO based approaches perform better than NH-ORCA for success rate $\bar{\pi}$ and extra time $\bar{t}$. Our map-based PPO approach received a 100% success rate for all environments. In first two kinds of environments, i.e., 6 and 8 robots in environments, extra time $\bar{t}$ of our approach is much shorter than others. In dense-robot environments, i.e., more than 10 robots, extra time $\bar{t}$ of our approach is close to that of the PPO approach with 1D convolution network, as it requires more computational effort maintaining safe distances between each robots. Moreover, variances of these metrics for our approach is smaller than others, which shows that the generated polices of our approach are more stable and corresponding trajectories are smoother than that of others. Fig. 4 illustrates trajectories of robots in environments of the circular scenario generated by different approaches.

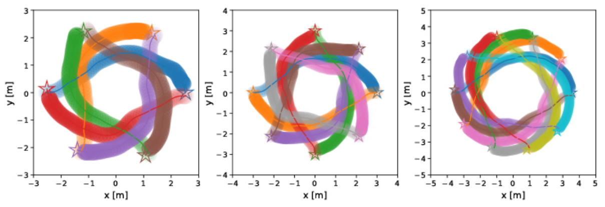
3) Robustness: Note that, NH-ORCA requires perfect perception to avoid other robots, which is difficult to achieve in practice. On the other hand, PPO based approaches are more robust to noisy sensor data. To expose this problem, we add Gaussian noise to the position and velocity information of other robots for each robot in environments with 10 robots of the circular scenario. Meanwhile, these Gaussian noise information is projected to corresponding laser data for both PPO based approaches.



(a) Trajectories by PPO with 1D convolution network



(b) Trajectories by NH-ORCA



(c) Trajectories by Map-based PPO

Fig. 4. Trajectories of robots in environments of the circular scenario with 6, 8, and 10 robots for different approaches.

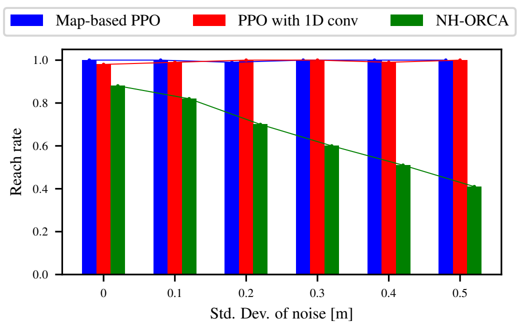
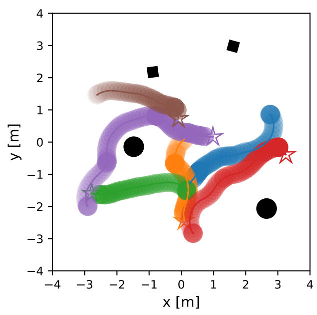


Fig. 5. Success rate (reach rate) of different approaches when the variance of Gaussian noise increases.

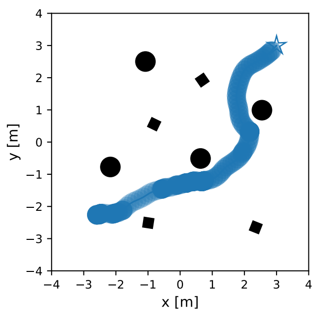
Fig. 5 shows the corresponding success rate for different approaches when the variance of Gaussian noise increases. Clearly, the performance of NH-ORCA declines dramatically.

4) Navigation in other simulation scenarios: We also test the performance of the obstacle avoidance policy generated by our approach on other simulation scenarios. In particular,

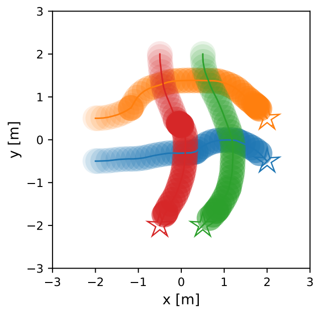
- *Random mixing scenario*: environments that randomly place six robots and four obstacles with different shapes, and set the target position of each robot 2 to 5 meters away from its starting position.
- *Random static scenario*: environments that randomly place one robot and eight obstacles with different shapes, and set the target position of the robot 2 meters away



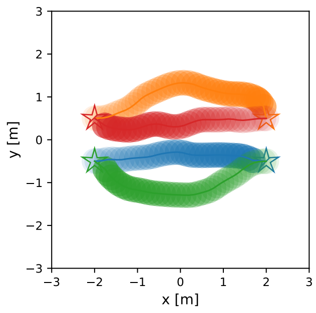
(a) Random mixing scenario



(b) Random static scenario



(c) Cross scenario



(d) Swap scenario

Fig. 6. Trajectories of robots from the obstacle avoidance policy generated by our map-based PPO approach in different scenarios.

from its starting position.

- *Cross scenario*: environments that require two groups of robots to move cross each other.
- *Swap scenario*: environments that require two groups of robots to move towards each other and swap their positions.

Notice that, we only use environments from the random scenario and the circular scenario to train the networks for the obstacle avoidance policy in our map-based PPO approach. We want to show that the generated policy can also perform well to other unseen scenarios. In specific, Table III shows the success rate of the generated policy in 100 different environments of the above four scenarios, which indicates that our approach can generalize well to new scenarios. In environments of both mixing and static scenario, we found few collisions when the path of a robot from its starting point to its target has to go through a really dense area. In some rare cases, a robot might rub the other in some environments of the swap scenario. Fig. 6 illustrates trajectories of robots from the obstacle avoidance policy generated by our map-based PPO approach in these scenarios.

C. Navigation in real world

We also deploy the trained model to real robots to perform multi-robot obstacle avoidance in real world. As shown in Fig. 7, the robot is based on TurtleBot 2 with Kobuki base and uses a Hokuyo UTM-30LX scanning laser Rangefinder as the 2D laser sensor. The robot applies an NVIDIA Jetson

TABLE III
SUCCESS RATE IN OTHER SIMULATION SCENARIOS

scenario	success rate
Random mixing scenario	96.7
Static random scenario	98.0
Cross scenario	100.0
Swap scenario	97.5

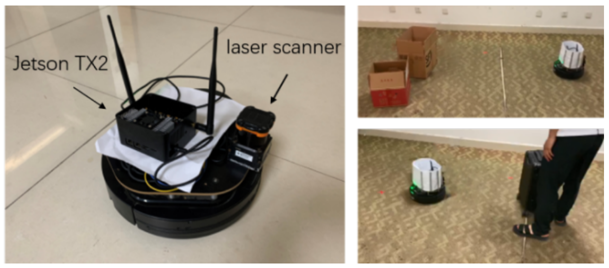


Fig. 7. The robot is based on TurtleBot 2 with Kobuki base using a Hokuyo UTM-30LX scanning laser Rangefinder and an NVIDIA Jetson TX2. Real world testing environments (right) are composed with paper boxes as static obstacles (upper right) and a moving suitcase (pushed by a person) as a dynamic obstacle (bottom right).

TX2 as its computing platform. In experiments, relative local goals of the robot are provided by a particle filter based on a state estimator. The egocentric grid map is constructed from the laser data, which has the fixed size 6.0×6.0m and the resolution 0.1m at each time step. We also use paper boxes and suitcases to act as static and dynamic obstacles in tests.

We implement three versions of such a robot and test the performance of our approach in following scenarios.

- *Static scenario*: environments that place paper boxes and the suitcase to block the robot's path from its starting position to its target.
- *Dynamic scenario*: environments in which a person pushes the suitcase across the path of the robot.
- *Multi-robot scenario*: environments that place three robots at the vertices of a triangle and set the target of each robot at the center of its opposite side.

The experiment shows that by deploying the trained model to their obstacle avoidance module in the navigation system, these robots can efficiently reach their targets with no collision in all three scenarios. Fig. 8 illustrates trajectories of three robots in an environment of the multi-robot scenario. The demonstration video for both simulation and real world experiments can be found at https://youtu.be/jcLKlEXuFuk.

V. CONCLUSIONS

In this paper, we extend our previous work for single robot and propose a map-based PPO approach for multi-robot collision avoidance, where robots do not communicate with each other and only sense other robots' positions and the obstacles around them. We consider three consecutive frames of egocentric grid maps and corresponding logical goal

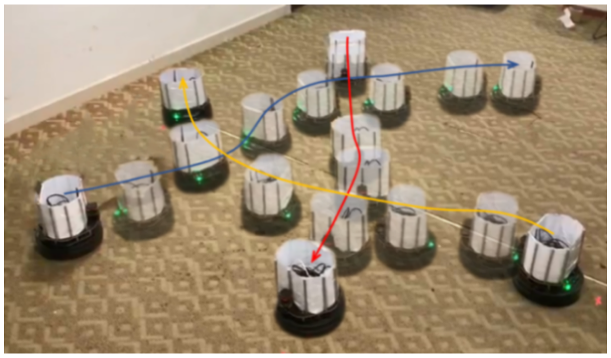


Fig. 8. Trajectories of robots in an environment of the multi-robot scenario. Every robot reaches its target smoothly and efficiently without any collision.

positions as inputs of the training networks in PPO, which outputs the required linear velocity and angular velocity of the robot. We apply a two stage training procedure to train networks using environments from the random scenario and the circular scenario. We also show that the trained model can generalize well to new scenarios. Both qualitative and quantitative experiments on simulation environments show that our approach is efficient and outperforms other existing approaches with higher success rate. We also show that the train model can be easily deployed to real robots to perform multi-robot collision avoidance in their navigation.

References

[1] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "Smooth and collision-free navigation for multiple robots under differential-drive constraints," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2010, pp. 4584–4589.

[2] Y. Chen, M. Cutler, and J. P. How, "Decoupled multiagent path planning via incremental sequential convex programming," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 5954–5961.

[3] D. Mellinger, A. Kushleyev, and V. Kumar, "Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams," in *2012 IEEE international conference on robotics and automation*. IEEE, 2012, pp. 477–483.

[4] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760–772, 1998.

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

[6] D. Claes, D. Hennes, K. Tuyls, and W. Meeussen, "Collision avoidance under bounded localization uncertainty," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1192–1198.

[7] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty." in *AAMAS*, 2012, pp. 147–154.

[8] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[9] Y. Liu, A. Xu, and Z. Chen, "Map-based deep imitation learning for obstacle avoidance," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 8644–8649.

[10] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[11] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.

[12] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052–3059.

[13] 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," *arXiv preprint arXiv:1808.03841*, 2018.

[14] 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.

[15] G. Chen, L. Pan, Y. Chen, P. Xu, Z. Wang, P. Wu, J. Ji, and X. Chen, "Robot navigation with map-based deep reinforcement learning," *arXiv preprint arXiv:2002.04349*, 2020.

[16] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 709–715.

[17] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[18] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*. ACM, 2009, pp. 41–48.

[19] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in neural information processing systems*, 2000, pp. 1057–1063.

[20] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[21] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[22] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.