2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

# Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning

Pinxin Long[1]*, Tingxiang Fan[1]*, Xinyi Liao[1], Wenxi Liu[2], Hao Zhang[1] and Jia Pan[3]

***Abstract*— Developing a safe and efficient collision avoidance policy for multiple robots is challenging in the decentralized scenarios where each robot generates its paths without observing other robots' states and intents. While other distributed multi-robot collision avoidance systems exist, they often require extracting agent-level features to plan a local collision-free action, which can be computationally prohibitive and not robust. More importantly, in practice the performance of these methods are much lower than their centralized counterparts.**

**We present a decentralized sensor-level collision avoidance policy for multi-robot systems, which directly maps raw sensor measurements to an agent's steering commands in terms of movement velocity. As a first step toward reducing the performance gap between decentralized and centralized methods, we present a multi-scenario multi-stage training framework to learn an optimal policy. The policy is trained over a large number of robots on rich, complex environments simultaneously using a policy gradient based reinforcement learning algorithm. We validate the learned sensor-level collision avoidance policy in a variety of simulated scenarios with thorough performance evaluations and show that the final learned policy is able to find time efficient, collision-free paths for a large-scale robot system. We also demonstrate that the learned policy can be well generalized to new scenarios that do not appear in the entire training period, including navigating a heterogeneous group of robots and a large-scale scenario with 100 robots. Videos are available at https://sites.google.com/view/drlmaca.**

## I. Introduction

Multi-robot navigation has recently gained much interest in robotics and artificial intelligence, and has many real-world applications including multi-robot search and rescue, navigation through human crowds, and autonomous warehouse. One of the major challenges for multi-robot navigation is to develop a safe and robust collision avoidance policy for each robot navigating from its starting position to its desired goal.

Some of prior works, known as *centralized methods*, assume that the comprehensive knowledge about all agents' intents (e.g. initial states and goals) and their workspace (e.g. a 2D grid map) is given for a central server to control the action of agents. These methods can generate the collision avoidance action by planning optimal paths for all robots simultaneously. However, these centralized methods are difficult to scale to large systems with many robots and their performance can be low when frequent task/goal reassignment is necessary. Besides, in practice, they heavily rely on the reliable communication network between robots and the central server. Therefore, once the central server and/or the communication network fails, the multi-robot systems will break down. Furthermore, these centralized methods are inapplicable when multiple robots are deployed in an unknown and unstructured environments.

Fig. 1: Robot trajectories in the *Circle* scenario using our learned policy. Note that the robots are square-shaped. It shows the good generalization capability of the learned policy since we directly test the policy trained with disc-shaped robots on this scenarios.

Compared with centralized methods, some existing works propose *agent-level decentralized collision avoidance* policies, where each agent independently makes decision taking into account the observable states (e.g. shapes, velocities and positions) of other agents as inputs. Most agent-level policies are based on the velocity obstacle (VO) [1]–[5], and they can compute local collision-free action efficiently for multiple agents in cluttered workspaces. However, several limitations greatly restrict their applications. First, the simulation based works [1], [6] assume that each agent has perfect sensing about the surrounding environment which does not hold in real world scenarios due to omnipresent sensing uncertainty. To moderate the limitation of perfect sensing, previous approaches use a global positioning system to track the positions and velocities of all robots [2], [5] or design an inter-agent communication protocol for sharing position and velocity information among nearby agents [3], [4], [7]. However, these approaches introduce external tools or communication protocols into the multi-robot systems, which may not be adequately robust. Second, VO based policies have many tunable parameters that are sensitive to scenario settings and thus the parameters must be carefully set offline to achieve satisfying performance. Finally, the performance of previous decentralized methods in terms of navigation speed and navigation time is significantly lower than their centralized counterparts.

Inspired by VO based approaches, Chen et al. [8] train an

* denotes equal contribution.

[1]Pinxin Long, Tingxiang Fan, Xinyi Liao and Hao Zhang are with Dorabot Inc., Shenzhen, China pinxinlong@gmail.com

[2]Wenxi Liu is with the Department of Computer Science, Fuzhou University, Fuzhou, China wenxi.liu@hotmail.com

[3]Jia Pan is with the Department of Mechanical and Biomedical Engineering, City University of Hong Kong, Hong Kong, China jiapan@cityu.edu.hk

This paper is partially supported by HKSAR General Research Fund (GRF) CityU 21203216, and NSFC/RGC Joint Research Scheme (CityU103/16-NSFC61631166002)

agent-level collision avoidance policy using deep reinforcement learning, which learns a two-agent value function that explicitly maps an agent's own state and its neighbors' states to collision-free action, whereas it still demands the perfect sensing. In their later work [9], multiple sensors are deployed to perform tasks of segmentation, recognition, and tracking in order to estimate the states of nearby agents and moving obstacles. However, this complex pipeline not only requires expensive online computation but makes the whole system less robust to the perception uncertainty.

In this paper, we focus on *sensor-level decentralized collision avoidance* policies that directly map the raw sensor data to desired, collision-free steering commands. Compared with agent-level policies, the perfect sensing for the neighboring agents and obstacles, and offline parameter-tuning for different scenarios are not required. Sensor-level collision avoidance policies are often modeled by deep neural networks (DNNs) [10], [11] and trained using supervised learning on a large dataset. However, there are several limitations for learning policies under supervision. First, it requires a large amount of training data that should cover different kinds of interaction situations for multiple robots. Second, the expert trajectories in datasets are not guaranteed to be optimal in the interaction scenarios, which makes training difficult to converge to a robust solution. Third, it is difficult to hand-design a proper loss function for training robust collision avoidance policies. To overcome these drawbacks, we propose a multi-scenario multi-stage deep reinforcement learning framework to learn the optimal collision avoidance policy using the policy gradient method.

**Main results:** In this paper, we address the collision avoidance of multiple robots in a fully decentralized framework, in which the input data is only collected from onboard sensors. To learn the optimal collision avoidance policy, we propose a novel multi-scenario multi-stage training framework which exploits a robust policy gradient based reinforcement learning algorithm trained in a large-scale robot system in a set of complex environments. We demonstrate that the collision avoidance policy learned from the proposed method is able to find time efficient, collision-free paths for a large-scale nonholonomic robot system, and it can be well generalized to unseen scenarios. Its performance is also much better than previous decentralized methods, and can serve as a first step toward reducing the gap between centralized and decentralized navigation policies.

## II. Related Work

Learning-based collision avoidance techniques have been extensively studied for one robot avoiding static obstacles. Many approaches adopt the supervised learning paradigm to train a collision avoidance policy by imitating a dataset of sensor input and motion commands. Muller et al. [12] trained a vision-based static obstacle avoidance system in supervised mode for a mobile robot by training a 6-layer convolutional network which maps raw input images to steering angles. Zhang et al. [13] exploited a successor-feature-based deep reinforcement learning algorithm to transfer depth information

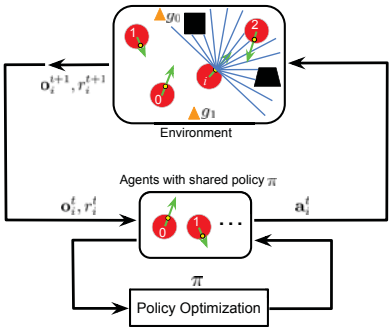



Fig. 2: An overview of our approach. At each timestep $t$, each robot receives its observation $\mathbf{o}_i^t$ and reward $r_i^t$ from the environment, and generates an action $a_i^t$ following the policy $\pi$. The policy $\pi$ is shared across all robots and updated by a policy gradient based reinforcement learning algorithm.

learned in previously mastered navigation tasks to new problem instances. Sergeant et al. [14] proposed a mobile robot control system based on multimodal deep autoencoders. Ross et al. [15] trained a discrete controller for a small quadrotor helicopter with imitation learning techniques. The quadrotor was able to successfully avoid collisions with static obstacles in the environment using only a single cheap camera. Yet only discrete movement (left/right) has to be learned and also the robot only trained within static obstacles. Note that the aforementioned approaches only take into account the static obstacles and require a human driver to collect training data in a wide variety of environments. Another data-driven end-to-end motion planner is presented by Pfeiffer et al. [11]. They trained a model maps laser range findings and target positions to motion commands using expert demonstrations generated by the ROS navigation package. This model can navigate the robot through a previously unseen environment and successfully react to sudden changes. Nonetheless, similar to the other supervised learning methods, the performance of the learned policy is seriously constrained by the quality of the labeled training sets. To overcome this limitation, Tai et al. [16] proposed a mapless motion planner trained through a deep reinforcement learning method. Kahn et al. [17] presented an uncertainty-aware model-based reinforcement learning algorithm to estimate the probability of collision in a priori unknown environment. However, the test environments are relative simple and structured, and the learned planner is hard to generalize to the scenarios with dynamic obstacles and other proactive agents.

Regarding *multi-agent* collision avoidance, the Optimal Reciprocal Collision Avoidance (ORCA) framework [1] has been popular in crowd simulation and multi-agent systems. ORCA provides a sufficient condition for multiple robots to avoid collisions with each other in a short time horizon, and can easily be scaled to handle large systems with many robots. ORCA and its extensions [2], [5] used heuristics or first principles to construct a complex model for the collision avoidance policy, which has many parameters that are tedious and difficult to be tuned properly. Besides, these methods are sensitive to the uncertainties ubiquitous in the real-world scenarios since they assume each robot to have perfect sensing about the surrounding agents' positions, velocities

and shapes. To alleviate the requirement of perfect sensing, communication protocols are introduce by [3], [4], [7] to share the state information including agents' positions and velocities among the group. Moreover, the original formulation of ORCA is based on holonomic robots which is less common than nonholonomic robots in the real world. To deploy ORCA on the most common differential drive robots, several methods have been proposed to deal with the difficulty of non-holonomic robot kinematics. ORCA-DD [18] enlarges the robot to twice the radius of the original size to ensure collision free and smooth paths for robots under differential constraints. However, this enlarged virtual size of the robot can result in problems in narrow passages or unstructured environments. NH-ORCA [19] makes a differential drive robot tracking a holonomic speed vector with a certain tracking error $\varepsilon$. It is preferred over ORCA-DD, since the virtual increase of the robots' radii is only by a size of $\varepsilon$ instead of doubling the radii.

In this paper, we focus on learning a collision-avoidance policy which can make multiple nonholonomic mobile robots navigate to their goal positions without collisions in rich and complex environments.

## III. Problem Formulation

The multi-robot collision avoidance problem is defined primarily in the context of a nonholonomic differential drive robot moving on the Euclidean plane with obstacles and other decision-making robots. During training, all of $N$ robots are modeled as discs with the same radius $R$, i.e., all robots are homogeneous.

At each timestep $t$, the $i$-th robot $(1 \leq i \leq N)$ has access to an observation $\mathbf{o}_i^t$ and computes a collision-free steering command $\mathbf{a}_i^t$ that drives it to approach the goal $\mathbf{g}_i$ from the current position $\mathbf{p}_i^t$. The observation $\mathbf{o}_i^t$ drawn from a probability distribution w.r.t. the underlying system state $\mathbf{s}_i^t$, $\mathbf{o}_i^t \sim \mathcal{O}(\mathbf{s}_i^t)$, only provides partial information about the state $\mathbf{s}_i^t$ since the $i$-th robot has no explicit knowledge about other robots' states and intents. Instead of the perfect sensing assumption applied in prior methods (e.g. [1], [3], [4], [6], [8], [9]), our formulation based on partial observation makes our approach more applicable and robust in real world applications. The observation vector of each robot can be divided into three parts: $\mathbf{o}^t = [\mathbf{o}_z^t, \mathbf{o}_g^t, \mathbf{o}_v^t]$ (here we ignore the robot ID $i$ for legibility), where $\mathbf{o}_z^t$ denotes the raw 2D laser measurements about its surrounding environment, $\mathbf{o}_g^t$ stands for its relative goal position (i.e. the coordinates of the goal in the robot's local polar coordinate frame), and $\mathbf{o}_v^t$ refers to its current velocity. Given the partial observation $\mathbf{o}^t$, each robot *independently* computes an action or a steering command, $\mathbf{a}^t$, sampled from a stochastic policy $\pi$ shared by all robots:

$$\mathbf{a}^t \sim \pi_\theta(\mathbf{a}^t \mid \mathbf{o}^t), \tag{1}$$

where $\theta$ denotes the policy parameters. The computed action $\mathbf{a}^t$ is actually a velocity $\mathbf{v}^t$ that guides the robot approaching its goal while avoiding collisions with other robots and obstacles $\mathbf{B}_k$ $(0 \leq k \leq M)$ within the time horizon $\Delta t$ until the next observation $\mathbf{o}^{t+1}$ is received.

Hence, the multi-robot collision avoidance problem can be formulated as a partially observable sequential decision making problem. The sequential decisions consisting of observations and actions (velocities) $(\mathbf{o}_i^t, \mathbf{v}_i^t)_{t=0:t_i^g}$ made by the robot $i$ can be considered as a trajectory $l_i$ from its start position $\mathbf{p}_i^{t=0}$ to its desired goal $\mathbf{p}_i^{t=t_i^g} \equiv \mathbf{g}_i$, where $t_i^g$ is the traveled time. To wrap up the above formulation, we define $\mathbb{L}$ as the set of trajectories for all robots, which are subject to the robot's kinematic (e.g. non-holonomic) constraints, i.e.:

$$\begin{aligned}\mathbb{L} = \{&l_i, i = 1, ..., N \mid \\ &\mathbf{v}_i^t \sim \pi_\theta(\mathbf{a}_i^t \mid \mathbf{o}_i^t), \\ &\mathbf{p}_i^t = \mathbf{p}_i^{t-1} + \Delta t \cdot \mathbf{v}_i^t, \\ &\forall j \in [1, N], j \neq i : \|\mathbf{p}_i^t - \mathbf{p}_j^t\| > 2R \\ &\wedge \forall k \in [1, M] : \|\mathbf{p}_i^t - \mathbf{B}_k\| > R \\ &\wedge \|\mathbf{v}_i^t\| \leq v_i^{\max}\}.\end{aligned} \tag{2}$$

To find an optimal policy shared by all robots, we adopt an objective by minimizing the expectation of the mean arrival time of all robots in the same scenario, which is defined as:

$$\underset{\pi_\theta}{\operatorname{argmin}} \quad \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} t_i^g | \pi_\theta], \tag{3}$$

where $t_i^g$ is the travel time of the trajectory $l_i$ in $\mathbb{L}$ controlled by the shared policy $\pi_\theta$.

The average arrival time will also be used as an important metric to evaluate the learned policy in Section V. We solve this optimization problem through a policy gradient based reinforcement learning method, which bounds the policy parameter updates to a trust region to ensure stability.

## IV. Approach

We begin this section by introducing the key ingredients of our reinforcement learning framework. Next, we describe the details about the architecture of the collision avoidance policy in terms of a deep neural network. Finally, we elaborate the training protocols used to optimize the policy.

### A. Reinforcement Learning Setup

The partially observable sequential decision making problem defined in Section III can be formulated as a Partially Observable Markov Decision Process (POMDP) solved with reinforcement learning. Formally, a POMDP can be described as a 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O})$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the state-transition model, $\mathcal{R}$ is the reward function, $\Omega$ is the observation space $(\mathbf{o} \in \Omega)$ and $\mathcal{O}$ is the observation probability distribution given the system state $(\mathbf{o} \sim \mathcal{O}(\mathbf{s}))$. In our formulation, each robot only has access to the observation sampled from the underlying system states. Furthermore, since each robot plans its motions in a fully decentralized manner, a multi-robot state-transition model $\mathcal{P}$ determined by the robots' kinematics and dynamics is not needed. Below we describe the details of the observation space, the action space, and the reward function.

*1) **Observation space**:* As mentioned in Section III, the observation $\mathbf{o}^t$ consists of the readings of the 2D laser range finder $\mathbf{o}_z^t$, the relative goal position $\mathbf{o}_g^t$ and robot's current velocity $\mathbf{o}_v^t$. Specifically, $\mathbf{o}_z^t$ includes the measurements of the last three consecutive frames from a 180-degree laser scanner which has a maximum range of 4 meters and provides 512 distance values per scanning (i.e. $\mathbf{o}_z^t \in \mathbb{R}^{3\times 512}$). In practice, the scanner is mounted on the forepart of the robot instead of the center (see the left image in Figure 1) to obtain a large unoccluded view. The relative goal position $\mathbf{o}_g^t$ is a 2D vector representing the goal in polar coordinate (distance and angle) with respect to the robot's current position. The observed velocity $\mathbf{o}_v^t$ includes the current translational and rotational velocity of the differential-driven robot. The observations are normalized by subtracting the mean and dividing by the standard deviation using the statistics aggregated over the course of the entire training.

*2) **Action space**:* The action space is a set of permissible velocities in continuous space. The action of differential robot includes the translational and rotational velocity, i.e. $\mathbf{a}^t = [v^t, w^t]$. In this work, considering the real robot's kinematics and the real world applications, we set the range of the translational velocity $v \in (0.0, 1.0)$ and the rotational velocity in $w \in (-1.0, 1.0)$. Note that backward moving (i.e. $v < 0.0$) is not allowed since the laser range finder can not cover the back area of the robot.

*3) **Reward design**:* Our objective is to avoid collisions during navigation and minimize the mean arrival time of all robots. A reward function is designed to guide a team of robots to achieve this objective:

$$r_i^t = ({}^g r)_i^t + ({}^c r)_i^t + ({}^w r)_i^t. \tag{4}$$

The reward $r$ received by robot $i$ at timestep $t$ is a sum of three terms, ${}^g r$, ${}^c r$, and ${}^w r$. In particular, the robot is awarded by $({}^g r)_i^t$ for reaching its goal:

$$({}^g r)_i^t = \begin{cases} r_{arrival} & \text{if } \|\mathbf{p}_i^t - \mathbf{g}_i\| < 0.1 \\ \omega_g(\|\mathbf{p}_i^{t-1} - \mathbf{g}_i\| - \|\mathbf{p}_i^t - \mathbf{g}_i\|) & \text{otherwise.} \end{cases} \tag{5}$$

When the robot collides with other robots or obstacles in the environment, it is penalized by $({}^c r)_i^t$:

$$({}^c r)_i^t = \begin{cases} r_{collision} & \text{if } \|\mathbf{p}_i^t - \mathbf{p}_j^t\| < 2R \\ & \text{or } \|\mathbf{p}_i^t - \mathbf{B}_k\| < R \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

To encourage the robot to move smoothly, a small penalty $({}^w r)_i^t$ is introduced to punish the large rotational velocities:

$$({}^w r)_i^t = \omega_w |w_i^t| \qquad \text{if } |w_i^t| > 0.7. \tag{7}$$

We set $r_{arrival} = 15$, $\omega_g = 2.5$, $r_{collision} = -15$ and $\omega_w = -0.1$ in the training procedure.

### B. Network architecture

Given the input (observation $\mathbf{o}_i^t$) and the output (action $\mathbf{v}_i^t$), now we elaborate the policy network mapping $\mathbf{o}_i^t$ to $\mathbf{v}_i^t$.

We design a 4-hidden-layer neural network as a non-linear function approximator to the policy $\pi_\theta$. Its architecture is shown in Figure 3. We employ the first three hidden layers to process the laser measurements $\mathbf{o}_z^t$ effectively. The first hidden layer convolves 32 one-dimensional filters with kernel size = 5, stride = 2 over the three input scans and applies ReLU nonlinearities [20]. The second hidden layer convolves 32 one-dimensional filters with kernel size = 3, stride = 2, again followed by ReLU nonlinearities. The third hidden layer is a fully-connected layer with 256 rectifier units. The output of the third layer is concatenated with the other two inputs ($\mathbf{o}_g^t$ and $\mathbf{o}_v^t$), and then are fed into the last hidden layer, a fully-connected layer with 128 rectifier units. The output layer is a fully-connected layer with two different activations: a sigmoid function is used to constrained the mean of translational velocity $v^t$ in $(0.0, 1.0)$ and the mean of rotational velocity $w^t$ in $(-1.0, 1.0)$ through a hyperbolic tangent function $(\tanh)$.

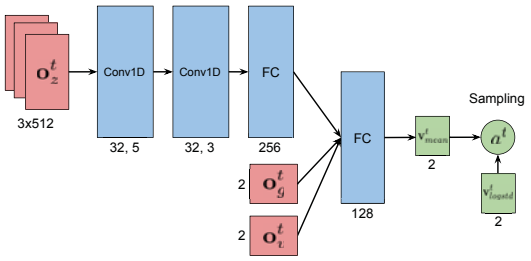



Fig. 3: The architecture of the collision avoidance neural network. The network has the scan measurements $\mathbf{o}_z^t$, relative goal position $\mathbf{o}_g^t$ and current velocity $\mathbf{o}_v^t$ as inputs, and outputs the mean of velocity $\mathbf{v}_{mean}^t$. The final action $\mathbf{a}^t$ is sampled from the Gaussian distribution constructed by $\mathbf{v}_{mean}^t$ with a separated log standard deviation vector $\mathbf{v}_{logstd}^t$.

Overall, the neural network maps the input observation vector $\mathbf{o}^t$ to a vector $\mathbf{v}_{mean}^t$. The final action $\mathbf{a}^t$ is sampled from a Gaussian distribution $\mathcal{N}(\mathbf{v}_{mean}^t, \mathbf{v}_{logstd}^t)$, where $\mathbf{v}_{mean}^t$ serves as the mean and $\mathbf{v}_{logstd}^t$ refers to a log standard deviation which will be updated solely during training.

### C. Multi-scenario multi-stage training

*1) **Training algorithm**:* Even if deep reinforcement learning algorithms have been successfully applied in mobile robot motion planning, they have mainly focused on a discrete action space [13], [21] or on small-scale problems [8], [9], [16], [17]. Here we focus on learning a collision avoidance policy which performs robustly and effectively with a large number of robots in complex scenarios with obstacles, such as corridors and mazes. We extend a recently proposed robust policy gradient algorithm, Proximal Policy Optimization (PPO) [22]–[24], to our multi-robot system. Our approach adapts the *centralized learning, decentralized execution* paradigm. In particular, each robot receives its own $\mathbf{o}_i^t$ at each time step and executes the action generated from the shared policy $\pi_\theta$; the policy is trained with experiences collected by all robots simultaneously.

As summarized in Algorithm 1 (adapted from [22], [23]), the training process alternates between sampling trajectories by executing the policy in parallel, and updating the policy with the sampled data. During data collection, each robot exploits the same policy to generate trajectories until they

collect a batch of data above $T_{max}$. Then the sampled trajectories are used to construct the surrogate loss $L^{PPO}(\theta)$, and this loss is optimized with the Adam optimizer [25] for $E_\pi$ epochs under the Kullback-Leiber (KL) divergence constraint. The state-value function $V_\phi(s_i^t)$, used as an baseline to estimate the advantage $\hat{A}_i^t$, is also approximated with a neural network with parameters $\phi$ on sampled trajectories. The network structure of $V_\phi$ is the same as that of the policy network $\pi_\theta$, except that it has only one unit in its last layer with a linear activation. We construct the squared-error loss $L^V(\phi)$ for $V_\phi$, and optimize it also with the Adam optimizer for $E_V$ epochs. We update $\pi_\theta$ and $V_\phi$ independently and their parameters are not shared since we have found that using two separated networks will lead to better results in practice.

This parallel PPO algorithm can be easily scaled to a large-scale multi-robot system with hundred robots in a decentralized fashion since each robot in the team is an independent worker collecting data. The decentralized execution not only dramatically reduce the time of sample collection, also make the algorithm suitable for training many robots in various scenarios.

*2) **Training scenarios**:* To expose our robots to diverse environments, we create different scenarios with a variety of obstacles using the Stage mobile robot simulator[1] (as shown in Figure 4) and move all robots concurrently. In scenario 1, 2, 3, 5, and 6 in Figure 4 (black solid lines are obstacles), we first select reasonable starting and arrival areas from the available workspace, then randomly sample the start and goal positions for each robot in the corresponding areas. Robots in scenario 4 are randomly initialized in a circle with a varied radius, and they aim to reach their antipodal positions by crossing the central area. As for scenario 7, we generate random positions for both robots and obstacles (shown in black) at the beginning of each episode; and the target positions of robots are also randomly selected. These rich, complex training scenarios enable robots to explore their high-dimensional observation space and are likely to improve the quality and robustness of the learned policy. Combining with the *centralized learning, decentralized execution* mechanism, the collision avoidance policy is effectively optimized at each iteration over a variety of environments.

*3) **Training stages**:* Although training on multiple environments simultaneously brings robust performance over different test cases (see Section V-C), it makes the training process harder. Inspired by the curriculum learning paradigm [27], we propose a two-stage training process, which accelerates the policy to converge to a satisfying solution, and gets higher rewards than the policy trained from scratch with the same number of epoch (as shown in Figure 5). In the first stage, we only train 20 robots on the random scenarios (scenario 7 in Figure 4) without any obstacles, this allows our robots learn fast on relatively simple collision avoidance tasks. Once the robots achieve reliable performance, we stop the Stage 1 and save the trained policy. The policy will continue to be updated in the Stage 2, where the number of robots increases to 58 and they are trained on richer and more complex scenarios shown in Figure 4.

[1] http://rtv.github.io/Stage/

**Algorithm 1** PPO with Multiple Robots

Initialize policy network $\pi_\theta$ and value function $V_\phi(s_t)$, and set hyperparameters as shown in Table I.
**for** iteration $= 1, 2, ...,$ **do**
  *// Collect data in parallel*
  **for** robot $i = 1, 2, ...N$ **do**
    Run policy $\pi_\theta$ for $T_i$ timesteps, collecting $\{\mathbf{o}_i^t, r_i^t, \mathbf{a}_i^t\}$, where $t \in [0, T_i]$
    Estimate advantages using GAE [26] $\hat{A}_i^t = \sum_{l=0}^{T_i} (\gamma\lambda)^l \delta_i^t$, where $\delta_i^t = r_i^t + \gamma V_\phi(s_i^{t+1}) - V_\phi(s_i^t)$
    **break**, if $\sum_{i=1}^{N} T_i > T_{max}$
  **end for**
  $\pi_{old} \leftarrow \pi_\theta$
  *// Update policy*
  **for** $j = 1, ..., E_\pi$ **do**
    $L^{PPO}(\theta) = \sum_{t=1}^{T_{max}} \frac{\pi_\theta(a_i^t|o_i^t)}{\pi_{old}(a_i^t|o_i^t)} \hat{A}_i^t - \beta \mathrm{KL}[\pi_{old} \mid \pi_\theta] + \xi \max(0, \mathrm{KL}[\pi_{old} \mid \pi_\theta] - 2\mathrm{KL}_{target})^2$
    **if** $\mathrm{KL}[\pi_{old} \mid \pi_\theta] > 4\mathrm{KL}_{target}$ **then**
      **break** and continue with next iteration $i + 1$
    **end if**
    Update $\theta$ with $lr_\theta$ by Adam [25] w.r.t $L^{PPO}(\theta)$
  **end for**
  *// Update value function*
  **for** $k = 1, ..., E_V$ **do**
    $L^V(\phi) = -\sum_{i=1}^{N} \sum_{t=1}^{T_i} (\sum_{t'>t} \gamma^{t'-t} r_i^{t'} - V_\phi(s_i^t))^2$
    Update $\phi$ with $lr_\phi$ by Adam w.r.t $L^V(\phi)$
  **end for**
  *// Adapt KL Penalty Coefficient*
  **if** $\mathrm{KL}[\pi_{old} \mid \pi_\theta] > \beta_{high}\mathrm{KL}_{target}$ **then**
    $\beta \leftarrow \alpha\beta$
  **else if** $\mathrm{KL}[\pi_{old} \mid \pi_\theta] < \beta_{low}\mathrm{KL}_{target}$ **then**
    $\beta \leftarrow \beta/\alpha$
  **end if**
**end for**

## V. Experiments and Results

In this section, we first describe the hyper-parameters and computational complexity of the training process. Then, we quantitatively compare our policy with other methods in various simulated scenarios. Lastly, we demonstrate the good generalization capability of the learned policy in several challenging and complex environments.

### A. Training configuration and computational complexity

The implementation of our algorithm is in TensorFlow and the large-scale robot group with laser scanners is simulated in the Stage simulator. We train the multi-robot collision avoidance policy on a computer with an i7-7700 CPU and a Nvidia GTX 1080 GPU. The offline training takes 12 hours (about 600 iterations in Algorithm 1) for the policy

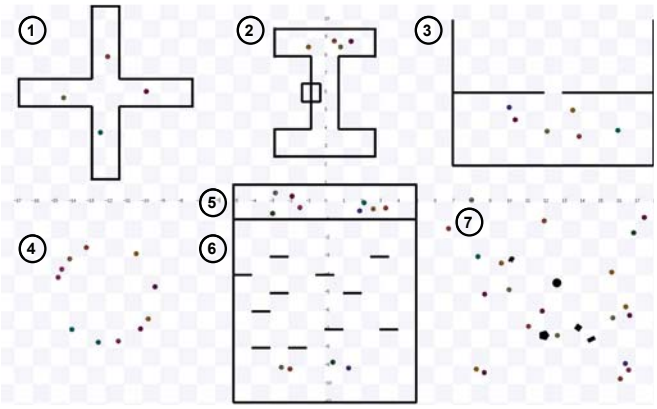



Fig. 4: Scenarios used to train the collision avoidance policy. All robots are modeled as a disc with the same radius. Obstacles are shown in black.

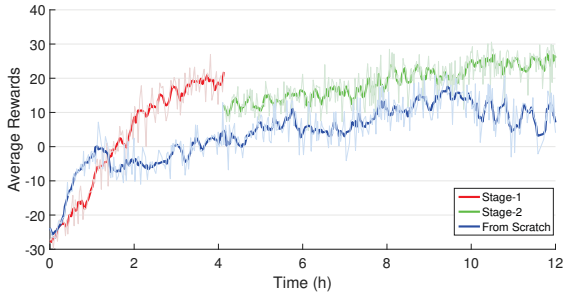



Fig. 5: Average rewards shown in wall time during the training process.

to converge to a robust performance in all scenarios. The hyper-parameters in Algorithm 1 is summarized in Table I. In particular, the learning rate $lr_\theta$ of policy network is set to $5e{-}5$ in the first stage, and is then reduced to $2e{-}5$ in the second training stage. As for running ten robots in the online decentralized control, it takes 3ms for the policy network to compute new actions on the CPU and about 1.3ms on the GPU.

### B. Quantitative comparison on various scenarios

*1)* ***Performance metrics****:* To compare the performance of our policy with other methods over various test cases, we use the following performance metrics. For each method, every test case is evaluated for 50 repeats.

- *Success rate* is the ratio of the number of robots reaching their goals within a certain time limit without any collisions over the total number of robots.
- *Extra time* $\bar{t}_e$ measures the difference between the travel time averaged over all robots and the lower bound of the travel time (i.e. the average cost time of going straight toward the goal for robots at max speed [7], [8]).
- *Extra distance* $\bar{d}_e$ measures the difference between the average traveled trajectory length of the robots and the lower bound of traveled distance of the robots (i.e. the average traveled distance for robots following the shortest paths toward the goals).
- *Average speed* $\bar{v}$ measures the average speed of the robot team during navigation.

Note that the extra time $\bar{t}_e$ and extra distance $\bar{d}_e$ metrics are measured over all robots during the evaluation which remove the effects due to the variance in the number of agents and the different distance to goals.

*2)* ***Circle scenarios****:* We first compare our multi-scenario multi-stage learned policy with the NH-ORCA policy [19], and the policy trained using supervised learning (SL-policy, a variation of [10], see below for details) on circle scenarios with different number of robots. The *circle* scenarios are similar to the scenario 4 shown in Figure 4, except we set robots uniformly on the circle. We use the open-sourced NH-ORCA implementation from [3], [4], and share the ground truth positions and velocities for all robots in the simulations. The policy learned in supervised mode, with the same architecture of our policy (described in Section IV-B), is trained on about 800,000 samples with the method adapted from [10], [11].

Compared to the NH-ORCA policy, our learned policy has significant improvement over it in terms of success rate, average extra time and travel speed. Although our learned policy has a slightly longer traveled paths than the NH-ORCA policy in scenarios with robot number above 15 (the third row in Table II), the larger speed (the fourth row in Table II) helps our robots reach their goals more quickly. Actually the slightly longer path is a byproduct of the higher speed since the robots need more space to decelerate before stopping at goals.

*3)* ***Random scenarios****:* Random scenarios are other frequently used scenes to evaluate the performance of multi-robot collision avoidance. To measure the performance of our method on random scenarios (as shown in the 7th scenario in Figure 4), we first create 5 different random scenarios with 25 robots in each case. For each random scenario, we repeat our evaluations 50 times. The results are shown in Figure 6, which compares our final policy with the policy only trained in the stage 1 (Section IV-C.1) and the NH-ORCA policy. We can observe that both policies trained using deep reinforcement learning have higher success rate than NH-ORCA policy (Figure 6a). It can also be seen that robots using the learned policies (in Stage 1 and 2) are able to reach their targets much faster than that of NH-ORCA (Figure 6b). Although the learned policies have longer trajectory length (Figure 6c), the higher average speed (Figure 6d) and success rate indicate that our policies have a

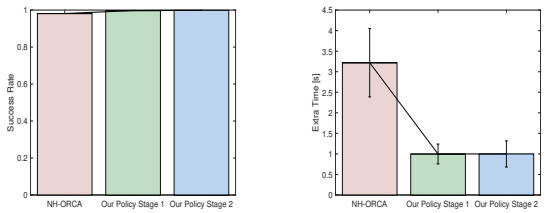
| Parameter | Value |
|---|---|
| $\lambda$ in line 6 | 0.95 |
| $\gamma$ in line 6 and 20 | 0.99 |
| $T_{max}$ in line 7 | 8000 |
| $E_\phi$ in line 11 | 20 |
| $\beta$ in line 12 | 1.0 |
| $\text{KL}_{target}$ in line 12 | $15e{-}4$ |
| $\xi$ in line 12 | 50.0 |
| $lr_\theta$ in line 16 | $5e{-}5$ (first stage), $2e{-}5$ (second stage) |
| $E_V$ in line 19 | 10 |
| $lr_\phi$ in line 21 | $1e{-}3$ |
| $\beta_{high}$ in line 24 | 2.0 |
| $\alpha$ in line 24 and 27 | 1.5 |
| $\beta_{low}$ in line 26 | 0.5 |

TABLE I: The hyper-parameters of our training algorithm described in Algorithm 1.

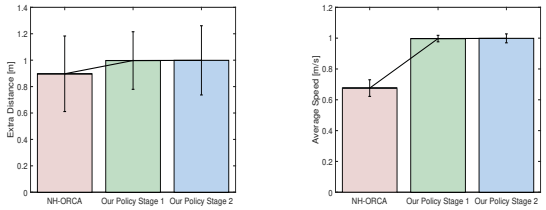
| Metrics | Method | 4 | 6 | 8 | 10 | 12 | 15 | 20 |
|---|---|---|---|---|---|---|---|---|
| Success Rate | SL-policy | 0.6 | 0.7167 | 0.6125 | 0.71 | 0.6333 | - | - |
| | NH-ORCA | **1.0** | 0.9667 | 0.9250 | 0.8900 | 0.9000 | 0.8067 | 0.7800 |
| | Our policy | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** | **1.0** | **0.9650** |
| Extra Time | SL-policy | 9.254 / 2.592 | 9.566 / 3.559 | 12.085 / 2.007 | 13.588 / 1.206 | 19.157 / 2.657 | - | - |
| | NH-ORCA | 0.622 / 0.080 | 0.773 / 0.207 | 1.067 / 0.215 | 0.877 /0.434 | 0.771 / 0.606 | 1.750 / 0.654 | 1.800 / 0.647 |
| | Our policy | **0.148 / 0.004** | **0.193 / 0.006** | **0.227 / 0.005** | **0.211 / 0.007** | **0.271 / 0.005** | **0.350 / 0.014** | **0.506 / 0.016** |
| Extra Distance | SL-policy | 0.358 / 0.205 | 0.181 / 0.146 | 0.138 / 0.079 | 0.127 / 0.047 | 0.141 / 0.027 | - | - |
| | NH-ORCA | **0.017 / 0.004** | **0.025 / 0.005** | 0.041 / 0.034 | **0.034 / 0.009** | 0.062 / 0.024 | **0.049 / 0.019** | **0.056 / 0.018** |
| | Our policy | **0.017 / 0.007** | 0.045 / 0.002 | **0.040 / 0.001** | 0.051 / 0.001 | **0.056 / 0.002** | 0.062 / 0.003 | 0.095 / 0.007 |
| Average Speed | SL-policy | 0.326 / 0.072 | 0.381 / 0.087 | 0.354 / 0.042 | 0.355 / 0.022 | 0.308 / 0.028 | - | - |
| | NH-ORCA | 0.859 / 0.012 | 0.867 / 0.026 | 0.839 / 0.032 | 0.876 / 0.045 | 0.875 / 0.054 | 0.820 / 0.052 | 0.831 / 0.042 |
| | Our policy | **0.956 / 0.006** | **0.929 / 0.002** | **0.932 / 0.001** | **0.928 / 0.002** | **0.921 / 0.002** | **0.912 / 0.003** | **0.880 / 0.006** |

TABLE II: Performance metrics (as mean/std) evaluated for different methods on the circle scenarios with different number of robots.

better ability to anticipate other robots' movements. Similarly to the circle scenarios above, the slightly longer path is due to robots' requirement to decelerate before stopping at goals.



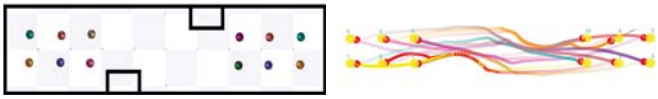

(a) Success rate (b) Extra time




(c) Extra distance (d) Average speed

Fig. 6: Performance metrics evaluated for our learned policies and the NH-ORCA policy on random scenarios.

*4) **Group scenarios**:* In order to evaluate the cooperation between robots, we would like to test our trained policy on more challenging scenarios, e.g. group swap, group crossing and group moving in the corridors. In the group swap scenarios, we navigate two groups of robots (each group has 6 robots) moving in opposite directions to swap their positions. As for group crossing scenarios, robots are separated in two groups, and their paths will intersect in the center of the scenarios. We compare our method with NH-ORCA on these two cases by measuring the average extra time $\bar{t}_e$ with 50 trials. It can be seen from Figure 8 that our policies performs much better than NH-ORCA on both cases. The shorter goal-reached time demonstrates that our policies have learned to produce more cooperative behaviors than reaction-based methods (NH-ORCA). We then evaluate three policies on the corridor scene, where two groups exchange their positions in a narrow corridor with two obstacles as shown in Figure 7a. However, only the Stage-2 policy can complete this challenging task (with paths as illustrated in Figure 7b).The failure of the Stage-1 policy shows that the co-training on a wide range of scenarios can lead to robust performance across different situations. NH-ORCA policy fails on this case because it relies on global planners to guide robots navigating in the complex environment. As mentioned in Section I, the agent-level collision avoidance policy (e.g. NH-ORCA) requires extra pipeline (e.g. a grid map indicating the obstacles) to explicitly identify and process the static obstacles while our method (sensor-level policy) implicitly infers the obstacles from raw sensor readings without any additional processing.



(a) Corridor scenario (b) Robot trajectories

Fig. 7: Two group of robots moving in a corridor with obstacles. (a) shows the corridor scenario. (b) shows trajectories generated by our Stage-2 policy.

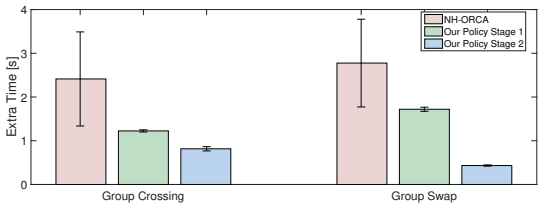
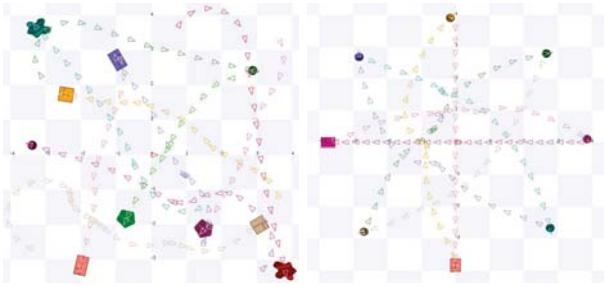



Fig. 8: Extra time $\bar{t}_e$ of our policies (Stage 1 and Stage 2) and the NH-ORCA policy on two group scenarios.

## C. Generalization

A notable feature benefited from multi-scenario training is the good generalization capability of the learned policy (Stage-2 policy). As mentioned in Section III, our policy is trained within a robot team where all robots share the same collision avoidance strategy. Non-cooperative robots are not introduced over the entire training process. Interestingly, the result shown in Figure 9b demonstrates that the learned policy can directly generalize well to avoid non-cooperative agents (i.e. the rectangle-shaped robots in Figure 9b which travel in straight lines with a fixed speed). Recall our policy is trained on robots with the same shape and a fixed radius. Figure 9a exhibits that the learned policy can also navigate efficiently a heterogeneous group of robots consisting of robots with different sizes and shapes to reach their goals without any collisions. To test the performance of our method on large-scale scenarios, we simulate 100 robots in a large circle moving to antipodal positions as shown in Figure 10. It shows that our learned policy can directly generalize to large-scale environments without any fine-tuning.



(a) Heterogeneous robots (b) Non-cooperative robots

Fig. 9: In the heterogeneous robot team (a), only the two disc-shaped robots are used in training. (b) shows 6 robots moving around two non-cooperative robots (rectangle-shaped), which traveled in straight lines with a fast speed.

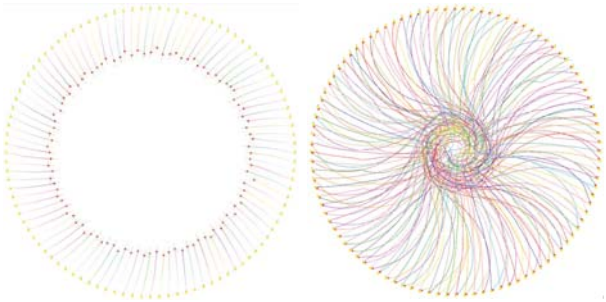


Fig. 10: Simulation of 100 robots trying to move through the center of a circle to antipodal positions.

## VI. Conclusion

In this paper, we present a multi-scenario multi-stage training framework to optimize a fully decentralized sensor-level collision avoidance policy with a robust policy gradient algorithm. The learned policy has demonstrated several advantages on an extensive evaluation of the state-of-the-art NH-ORCA policy in terms of success rate, collision avoidance performance, and generalization capability. Our work can serve as a first step towards reducing the navigation performance gap between the centralized and decentralized methods, though we are fully aware that the learned policy focusing on local collision avoidance cannot replace a global path planner when scheduling many robots to navigate through complex environments with dense obstacles.

## References


[1] 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.

[2] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[3] D. Hennes, D. Claes, W. Meeussen, and K. Tuyls, "Multi-robot collision avoidance with localization uncertainty," in *International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, 2012, pp. 147–154.

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

[5] D. Bareiss and J. van den Berg, "Generalized reciprocal collision avoidance," *The International Journal of Robotics Research*, vol. 34, no. 12, pp. 1501–1514, 2015.

[6] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *International Conference on Robotics and Automation*, 2008, pp. 1928–1935.

[7] J. Godoy, I. Karamouzas, S. J. Guy, and M. Gini, "Implicit coordination in crowded multi-agent navigation," in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

[9] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," *arXiv:1703.08862*, 2017.

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

[11] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, "From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots," in *International Conference on Robotics and Automation*, 2017, pp. 1527–1533.

[12] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, 2006, pp. 739–746.

[13] J. Zhang, J. T. Springenberg, J. Boedecker, and W. Burgard, "Deep reinforcement learning with successor features for navigation across similar environments," *arXiv preprint arXiv:1612.05533*, 2016.

[14] J. Sergeant, N. Sünderhauf, M. Milford, and B. Upcroft, "Multimodal deep autoencoders for control of a mobile robot."

[15] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *International Conference on Robotics and Automation*, 2013, pp. 1765–1772.

[16] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *International Conference on Intelligent Robots and Systems*, 2017.

[17] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *arXiv:1702.01182*, 2017.

[18] 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 *International Conference on Intelligent Robots and Systems*, 2010, pp. 4584–4589.

[19] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 203–216.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International conference on machine learning*, 2010, pp. 807–814.

[21] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *International Conference on Robotics and Automation*, 2017, pp. 3357–3364.

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

[23] N. Heess, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, A. Eslami, M. Riedmiller, *et al.*, "Emergence of locomotion behaviours in rich environments," *arXiv:1707.02286*, 2017.

[24] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, 2015, pp. 1889–1897.

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

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

[27] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *International conference on machine learning*, 2009, pp. 41–48.