

Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios

The International Journal of
Robotics Research
2020, Vol. 39(7) 856–892
© The Author(s) 2020
Article reuse guidelines:
sagepub.com/journals-permissions
DOI: 10.1177/0278364920916531
journals.sagepub.com/home/ijr



Tingxiang Fan^{1*}, Pinxin Long^{2*} , Wenxi Liu³ and Jia Pan¹ 

Abstract

Developing a safe and efficient collision-avoidance policy for multiple robots is challenging in the decentralized scenarios where each robot generates its paths with limited observation of other robots' states and intentions. Prior distributed multi-robot collision-avoidance systems often require frequent inter-robot communication or agent-level features to plan a local collision-free action, which is not robust and computationally prohibitive. In addition, the performance of these methods is not comparable with their centralized counterparts in practice. In this article, we present a decentralized sensor-level collision-avoidance policy for multi-robot systems, which shows promising results in practical applications. In particular, our policy directly maps raw sensor measurements to an agent's steering commands in terms of the 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 in rich, complex environments simultaneously using a policy-gradient-based reinforcement-learning algorithm. The learning algorithm is also integrated into a hybrid control framework to further improve the policy's robustness and effectiveness. We validate the learned sensor-level collision-avoidance policy in a variety of simulated and real-world scenarios with thorough performance evaluations for large-scale multi-robot systems. The generalization of the learned policy is verified in a set of unseen scenarios including the navigation of a group of heterogeneous robots and a large-scale scenario with 100 robots. Although the policy is trained using simulation data only, we have successfully deployed it on physical robots with shapes and dynamics characteristics that are different from the simulated agents, in order to demonstrate the controller's robustness against the simulation-to-real modeling error. Finally, we show that the collision-avoidance policy learned from multi-robot navigation tasks provides an excellent solution for safe and effective autonomous navigation for a single robot working in a dense real human crowd. Our learned policy enables a robot to make effective progress in a crowd without getting stuck. More importantly, the policy has been successfully deployed on different types of physical robot platforms without tedious parameter tuning. Videos are available at <https://sites.google.com/view/hybridmrca>.

Keywords

Distributed collision avoidance, multi-robot systems, multi-scenario multi-stage reinforcement learning, hybrid control

1. Introduction

Multi-robot navigation has recently received much interest in robotics and artificial intelligence and has many applications including multi-robot search and rescue, navigation through human crowds, and autonomous warehouses. One of the major challenges for multi-robot navigation is to develop a safe and robust collision-avoidance policy such that each robot can navigate from its starting position to its desired goal safely and efficiently. More importantly, a sophisticated collision-avoidance skill is also a prerequisite for robots to accomplish many complex tasks, including

¹Department of Computer Science, University of Hong Kong, Hong Kong, China

²Baidu Research, Baidu, Inc., Beijing, China

³College of Mathematics and Computer Science, Fuzhou University, China

*These authors contributed equally.

Corresponding author:

Jia Pan, Department of Computer Science, University of Hong Kong, Chow Yei Ching Building, Room 410, Pokfulam, Hong Kong, China.
Email: jpan@cs.hku.hk

multi-robot collaboration, tracking, and navigation through a dense crowd.

Some of the prior works, known as *centralized methods*, such as those by Schwartz and Sharir (1983), Yu and LaValle (2016), and Tang et al. (2018), assume that the actions of all agents are determined by a central server aware of comprehensive knowledge about all agents' intents (e.g., initial states and goals) and their workspace (e.g., a 2D grid map). These centralized approaches generate collision-avoidance actions by planning optimal paths for all robots simultaneously. They can guarantee safety, completeness, and approximate optimality but are difficult to scale to large systems with many robots owing to several main limitations, including high computational cost for multi-robot scheduling, heavy reliance on reliable synchronized communication, low tolerance to failures or disturbances, and inapplicability to unknown and unstructured environments with human co-workers.

Compared with centralized methods, some existing works propose *agent-level decentralized collision-avoidance* policies, where each agent independently makes its decision by taking into account the observable states (e.g., shapes, velocities, and positions) of other agents. Most agent-level policies are based on the velocity obstacle (VO) framework (Bareiss and van den Berg, 2015; Claes et al., 2012; Hennes et al., 2012; Snape et al., 2011; van den Berg et al., 2011a), which infers local collision-free motion efficiently for multiple agents in cluttered workspaces. However, these methods have several severe limitations which prevent them from being widely used in real scenarios. First, the simulation-based works (van den Berg et al., 2011a, 2008) assume that each agent has perfect sensing about the surrounding environment, but this assumption does not hold in real-world scenarios owing to the omnipresent sensing uncertainty. To moderate the limitation of perfect sensing, some previous approaches use a global positioning system to track the positions and velocities of all robots (Bareiss and van den Berg, 2015; Snape et al., 2011). However, such external infrastructure is too expensive to scale to large multi-robot systems. Some other methods design inter-agent communication protocols for sharing position and velocity information among nearby agents (Claes et al., 2012; Godoy et al., 2016a; Hennes et al., 2012). However, the communication systems introduce additional difficulties such as delay or blocking of communication signals owing to obstacle occlusions. The second limitation is that the VO-based policies are controlled by multiple tunable parameters that are sensitive to scenario settings and thus must be carefully set to accomplish satisfactory multi-robot motion. Finally, the timing performance of previous decentralized methods for navigation is significantly inferior to their centralized counterparts in practical applications.

Inspired by VO-based approaches, Chen et al. (2017b) trained an agent-level collision-avoidance policy using deep reinforcement learning (DRL), which learns a two-agent value function that explicitly maps an agent's own state and

its neighbors' states to a collision-free action, whereas it still demands the perfect sensing. In their later work (Chen et al., 2017a), multiple sensors were 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 also makes the whole system less robust to the perception uncertainty.

In this article, 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, our approach requires neither perfect sensing for neighboring agents and obstacles nor tedious offline parameter tuning for adapting to different scenarios. We have successfully deployed the learned policy to a large number of heterogeneous robots to achieve high-quality, large-scale multi-robot collaboration in both simulation and physical settings. Our method also endows a robot with high mobility to navigate safely and efficiently through a dense crowd with moving pedestrians.

Sensor-level collision-avoidance policies are often modeled using deep neural networks (DNNs) and trained using supervised learning on a large dataset (Long et al., 2017b; Pfeiffer et al., 2017). However, there are several limitations to learning policies under supervision. First, it demands a large amount of training data that should cover different interaction situations for multiple robots. Second, the expert trajectories in datasets are not guaranteed to be optimal in interaction scenarios, which makes training difficult to converge to a robust solution. Third, it is difficult to design a proper loss function by hand for training robust collision-avoidance policies. To overcome the above drawbacks of supervised learning, we propose a multi-scenario multi-stage DRL framework to learn the optimal collision-avoidance policy using the policy gradient method. The policy is trained in simulation for a large-scale multi-robot system in a set of complex scenarios. The learned policy outperforms the existing agent- and sensor-level approaches in terms of navigation speed and safety. It can be deployed to physical robots directly and smoothly without tedious parameter tuning.

Our collision-avoidance policy is trained in simulation rather than in the real world because it is difficult to generate training data safely and effectively for physical robots while the performance of robot learning is heavily dependent on the quality and quantity of the collected training data. In particular, to learn a sophisticated collision-avoidance policy, robots need to undergo billions of collisions with static environments and moving pedestrians in order to fully explore different reactions and interaction behaviors. Such data collection and policy update procedures would be expensive, time-consuming, and dangerous if implemented in the real world. As a result, we follow the recent trend of simulation-to-real methods (Peng et al., 2017; Rusu et al., 2016; Sadeghi and Levine, 2017; Tobin et al., 2017), by first training a control policy in virtual simulation and then deploying the learned model to the real robot directly.

However, it is non-trivial to transfer learned policies from simulation to the real world and may suffer a significant loss in performance, because the real world is messy and contains an infinite number of novel scenarios that may not be encountered during training. Fortunately, our policy uses an input space that is robust to the simulation-to-real gap, and therefore, we are able to use simulation to greatly accelerate our robot learning process. In particular, the environment state is measured by the LiDAR sensor, which is easier to transfer between simulation and the real world than the camera sensor used in most previous works. This is due to two reasons. First, the LiDAR sensor can be efficiently simulated with high fidelity, whereas simulating a camera sensor with high fidelity is not only computationally expensive owing to the sophisticated rendering pipeline required, but sometimes inapplicable owing to the inherent complexity of the real-world scenario in terms of different illumination conditions and materials. Thus, the behavior difference between the simulation and the real-world data of the LiDAR sensor is much smaller than that of the camera sensor, which is in favor of a better simulation-to-real transfer. Second, the LiDAR sensor has a measurement dimension of 512, which is significantly lower than the dimension of the camera captured data, e.g., a typical RGB image with $640 \times 480 \times 3$. A smaller dimension of the input feature is known to be beneficial for the simulation-to-real transfer in practice.

Another challenge for learning-based collision avoidance is the lack of guarantee for the safety and completeness of the learned policy, which is considered as one of the main concerns when using deep learning in robotics. Inspired by the hybrid control framework (Egerstedt and Hu, 2002), we aim to alleviate this difficulty by combining the learning-based policy with traditional control. In particular, the traditional control takes charge of relatively simple scenarios or emergent situations, while the learned policies will handle more-complex scenarios by generating more sophisticated behaviors bounded in limited action space. In this way, we can further improve the performance of the collision-avoidance policy in terms of safety without sacrificing efficiency.

Contributions. Our main contributions in this paper are as follows.

- We develop a fully decentralized multi-robot collision-avoidance framework, where each robot makes navigation decisions independently without any communication with others. The navigation policy has the ranging data collected from a robot's onboard sensors as the input and outputs the velocity command. The policy is optimized offline via a novel multi-scenario multi-stage reinforcement-learning algorithm and trained with a robust policy gradient method. The learned policy can be executed online on each robot using a relatively cheap on-board computer, e.g., a NUC or a Movidius neural compute stick.

- We further combine the deep-learned policy with traditional control approaches to achieve a robust hybrid navigation policy. The hybrid policy is able to find time-efficient and collision-free paths for a large-scale non-holonomic robot system and it can be safely generalized to unseen simulated and real-world 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.
- Our decentralized policy is trained in simulation but does not suffer from the simulation-to-real gap and can be directly deployed to physical robots without tedious fine-tuning.
- We deploy the policy in a warehouse scenario to control multiple robots in a decentralized manner. Compared with previous solutions to the multi-robot warehouse, our system requires neither a pre-constructed bar-code infrastructure nor a map. It also alleviates the demands for high-band and low-latency synchronized communication that is considered as the main bottleneck when scaling a multi-robot system. In particular, we use an ultra-wide-band (UWB) system to provide rough guidance to the robots about their goals, and all robots navigate toward their goals independently by using our hybrid policy to resolve the collisions during the navigation. Hence, our system can be easily extended to hundreds or more robots.
- We deploy the policy directly on a physical robot to achieve smooth and effective navigation through a dense crowd with moving pedestrians. Moreover, the policy can be easily deployed to different robotic platforms with different shapes and dynamics, and still can provide satisfactory navigation performance.

The rest of this article is organized as follows. Section 2 provides a brief review of related works. In Section 3, we introduce notation and formulate our problem. In Section 4, we discuss the details about how to use reinforcement learning to train a high-quality multi-robot collision-avoidance policy, and how to combine the learned policy with traditional optimal control schemes to achieve a more robust hybrid control policy. Finally, we highlight our experimental results in both the simulation scenarios (Section 5) and the real-world scenarios (Section 6).

This article is an extension of our previous conference version (Long et al., 2017a), which presents a multi-robot navigation policy that demonstrates the state-of-the-art performance in simulated scenarios. However, we find that this policy cannot provide satisfactory behavior for physical robots owing to the simulation-to-real gap. To overcome this difficulty left in the previous work, we propose a hybrid control framework that combines the reinforcement-learned policy and traditional control schemes. Thus, we can obtain a heuristic but powerful controller that can be conveniently transplanted on different robotic platforms and provide robust performance for a wide variety of scenarios. In addition, the hybrid control framework introduces a convenient

parameter tuning scheme for particular hyper-parameters, which can be used to balance the navigation efficiency and safety for the specific robotic platform in a user-friendly manner. In addition, to demonstrate the excellent performance of the proposed hybrid scheme, we evaluate our framework in extensive experiments, including detailed quantitative analysis of the navigation behavior in simulation, the navigation experiment on multiple physical robots, and the single-robot navigation in dense human crowds, by comparing with other state-of-the-art approaches.

2. Related works

In this section, we first briefly survey the related works on multi-robot navigation, including the centralized and decentralized approaches. Next, we summarize recent studies about how to solve the multi-robot navigation problem in a data-driven manner with machine learning techniques. We further discuss the simulation-to-real gap of the learning-based navigation control, and then summarize the recent progress in resolving this issue, including a short introduction to the hybrid control. Finally, we briefly survey the DRL and its application to reactive control.

2.1. Centralized multi-robot navigation

Over the past decade, the centralized multi-robot motion planning methods have been widely applied in many applications, such as task allocation (Chen and Sun, 2011; Stone and Veloso, 1999), formation control (Balch and Arkin, 1998; Chen et al., 2010; De La Cruz and Carelli, 2008; Sun et al., 2009), and object transportation (Alonso-Mora et al., 2017; Hu and Sun, 2011; Michael et al., 2011; Turpin et al., 2014). The centralized methods assume that each robot can have access to the complete information (e.g., velocity and desired goal) of other robots via a global communication system for motion planning. In this way, they can guarantee a safe, optimal, and complete solution for navigation (Luna and Bekris, 2011; Sharon et al., 2015; Tang et al., 2018; Yu and LaValle, 2016). However, these centralized approaches are difficult to deploy in large-scale real-world systems owing to several reasons. First, the centralized control and scheduling become computationally prohibitive as the number of robots increases. Second, these methods require a reliable synchronized communication between the central server and all robots, which is either uneconomical or not feasible for large-scale systems. Third, the centralized system is vulnerable to failures or disturbances of the central server, the communication between robots, or motors and sensors of any individual robot. Furthermore, these centralized methods are inapplicable when multiple robots are deployed in an unknown and unstructured environment, e.g., in a warehouse with human co-workers. Owing to these limitations, in this article, we present a decentralized method to accomplish large-scale multi-robot navigation.

2.2. Decentralized multi-robot navigation

Compared with the centralized methods, the decentralized methods have been studied extensively mainly for the collision-avoidance task in multi-agent systems.

The early works solve the decentralized multi-robot navigation problem by extending the classic single-robot collision-avoidance algorithms into the multi-robot scenario, such as potential fields (Haddad et al., 1998), social force (Helbing and Molnar, 1995), dynamic window (Fox et al., 1997), and model predictive control (Keviczky et al., 2004). These methods are simple in theory and computationally efficient. However, it is challenging for them to model the coordination among heterogeneous agents, so the effectiveness of these methods is far from being satisfactory. In addition, these methods usually rely on cost functions to regulate the behavior of agents, but the parameters of the cost functions may be sensitive to different configurations of scenarios and, thus, it often requires a non-trivial empirical manual parameter tuning process to produce good performance.

Another representative approach, the Optimal Reciprocal Collision Avoidance (ORCA) framework (van den Berg et al., 2011a), has been widely used in crowd simulation and multi-agent systems. In particular, ORCA provides a sufficient condition for multiple agents to avoid collisions with others in a short time horizon, and can easily be scaled to large systems with many agents. In addition, ORCA and its variants (Bareiss and van den Berg, 2015; Snape et al., 2011) used heuristics to construct a parametric collision-avoidance model, which are tedious to tune for satisfactory performance when a scenario contains heterogeneous robots with a wide variety of behaviors. In addition, these methods are difficult to adapt to real-world scenarios with ubiquitous uncertainties, because they assume that each robot has perfect sensing about the surrounding agents' positions, velocities, and shapes. To alleviate the demand for perfect sensing, communication protocols have been introduced by Hennes et al. (2012), Claes et al. (2012), and Godoy et al. (2016a) to share agents' state information, including positions and velocities among the group. However, introducing the communication network hurts the robustness and flexibility of the multi-robot system. To alleviate this problem, Zhou et al. (2017) planned the robots' paths by computing each robot's buffered Voronoi cell. Unlike previous work that needs both the position and velocity knowledge about adjacent robots, this method only assumes each robot knowing the position of surrounding agents, although it still uses a motion-capture system as the global localization infrastructure.

In addition, the original formulation of ORCA is based on holonomic robots, while the robots in the real-world scenarios are often non-holonomic. In order to deploy ORCA on the most common differential drive robots, several methods have been proposed to deal with the kinematics of non-holonomic robots. ORCA-DD (Snape et al., 2010) doubles the effective radius of each agent to ensure collision-free and smooth paths for robots under

differential constraints. However, it causes problems for the navigation in narrow passages and unstructured environments. NH-ORCA (Alonso-Mora et al., 2018, 2013) enabled a differential drive robot to follow the desired velocity command within an error of ϵ . It only needs to slightly increase the effective radius of a robot according to the value of ϵ and, thus, outperforms ORCA-DD in collision-avoidance performance.

2.3. Learning-based collision-avoidance policy

The decentralized policies discussed above are based on some first-principle rules for collision avoidance in general situations, and thus cannot achieve the optimal performance for a specific task or scenario, such as navigation in the crowded pedestrian scenario. To solve this problem, learning-based collision-avoidance techniques try to optimize a parameterized policy using the data collected from a specific task. Most recent works focus on the task of single-robot navigation in environments with static obstacles. Many approaches adopted the supervised learning paradigm to train a collision-avoidance policy by mapping sensor input to motion commands. Muller et al. (2006) presented a vision-based static obstacle-avoidance system. They trained a six-layer convolutional network that maps raw input images to steering angles. Zhang et al. (2016a) exploited a DRL algorithm based on the successor features (Barreto et al., 2017) to transfer the depth information learned in previously mastered navigation tasks to new problem instances. Sergeant et al. (2015) proposed a mobile robot control system based on multimodal deep autoencoders. Ross et al. (2013) trained a discrete controller for a small quadrotor helicopter with imitation learning techniques to accomplish the task of collision avoidance using a single monocular camera. In their formulation, only discrete movements (left/right) need to be learned. Note that all of the aforementioned approaches only take into account the static obstacles and require manually collecting training data in a wide variety of environments.

To deal with unstructured environments, one data-driven motion planner is presented by Pfeiffer et al. (2017). They trained a model that maps laser range findings and target positions to motion commands using expert demonstrations generated by the ROS navigation package. This model can navigate a robot through an unseen environment and successfully react to sudden changes. Nonetheless, similar to other supervised learning methods, the performance of the learned policy is severely constrained by the quality of the training sets. To overcome this limitation, Tai et al. (2017) proposed a map-less motion planner trained through a DRL method and Kahn et al. (2017) presented an uncertainty-aware model-based reinforcement-learning algorithm to estimate the probability of collision in an unknown environment. However, these test environments are relatively simple and structured, so they cannot well represent the real-world environments. The learned planner is also difficult to

generalize to complex scenarios with dynamic obstacles and other proactive agents.

To address highly dynamic unstructured environments, some decentralized multi-robot navigation approaches are proposed recently. Godoy et al. (2016b) introduced a Bayesian inference approach to predict surrounding dynamic obstacles and to compute collision-free command through the ORCA framework (van den Berg et al., 2011b). Chen et al. (2017b), Chen et al. (2017a), and Everett et al. (2018) proposed multi-robot collision-avoidance policies by DRL, which requires deploying multiple sensors to estimate the states of nearby agents and moving obstacles. However, to decentralize the perceived agent-level information in the surrounding environment, these methods require a complicated pipeline including visual detection, tracking, and depth estimation, which degrades the robustness and generalization capability in real-world scenarios. In addition, these methods assume that they have perfect sensing/understanding about the surrounding environment, which is not feasible in real-world scenarios owing to the limited onboard computation resources and the unknown uncertainty of the perception.

2.4. Simulation-to-real gap and hybrid control

In order to learn a sophisticated control policy with reinforcement learning, robots need to interact with the environment for a long period to accumulate knowledge about the consequences of different actions. Collecting such interaction data in the real world is expensive, time-consuming, and sometimes infeasible owing to safety issues. As a result, most reinforcement-learning studies are conducted in simulation. However, a policy learned in the simulation may be problematic when directly adapting to real-world applications, known as the simulation-to-real gap. Many recent works attempt to address this problem. Rusu et al. (2016) proposed a progressive nets architecture that assembles a set of neural networks trained for different tasks to build a new network that is able to bridge the simulation and the real world. Sadeghi and Levine (2017) leveraged diversified rendering parameters in simulation to accomplish collision avoidance in the real world without the input of any real images. Tobin et al. (2017) introduced domain randomization to train a robust and transferable policy for manipulation tasks.

In this article, we utilize the hybrid control framework to alleviate the simulation-to-real problem. Hybrid control architecture has been widely used for robot navigation problems in the past two decades (Adouane, 2009; Egerstedt and Hu, 2002), which designs a high-level control strategy to manage a set of low-level control rules. Hybrid control has also been heavily leveraged in multi-robot systems (Alur et al., 1999; Quotrup et al., 2004). For distributed control, Shucker et al. (2007) proposed switching rules to deal with challenging cases where collisions or noises cannot be handled appropriately. Gillula et al. (2011) introduced a combination of hybrid decomposition and reachability analysis to design a decentralized collision-

avoidance algorithm for multiple aerial vehicles. Different from prior methods, we use hybrid control to improve the transferability and robustness of the learned control policies. In particular, we use traditional control policies to deal with situations with the large simulation-to-real gap (i.e., the situations cannot be handled well by learned policy alone), and use a high-level heuristic to switch between the learned policy and traditional control policies.

2.5. DRL and reactive control

Deep learning methods have been successfully used in a wide variety of applications, including computer vision (He et al., 2016; Krizhevsky et al., 2012) and nature language processing (Amodei et al., 2016; Graves et al., 2013). These successes indicate that DNNs are good at extracting hierarchical features from complex and high-dimensional data and making high-quality decisions. These advantages make DNNs a powerful tool for robotic applications, which also need to deal with complex raw input data captured from onboard sensors. Recently, there is an increasing amount of research that appeals for DNNs to solve robotic perception and control problems. Muller et al. (2006) developed a vision-based obstacle-avoidance system for a mobile robot by training a six-layer convolutional network that maps raw input images to steering angles. Levine et al. (2016) presented an end-to-end framework that learns control policies mapping raw image observations to torques at the robot's motors. DNNs have also been integrated with model predictive control to control robotic systems (Lenz et al., 2015; Zhang et al., 2016b). Ondruska et al. (2016); Ondruska and Posner (2016) combined recurrent neural networks with convolutional operations to learn a direct mapping from raw 2D laser data to the unoccluded state of the entire scene; the mapping is then used for object tracking. In this article, we trained a multi-layer neural network to produce the navigation velocity for each robot according to observations.

3. Problem formulation

The multi-robot collision-avoidance problem is formulated primarily in the context of a non-holonomic differential drive robot moving on the Euclidean plane and avoiding collision with obstacles and other fellow robots.

To tackle this problem, we first assume all robots in the environment are homogeneous. Specifically, all N robots are modeled as disks with the same radius R , and the multi-robot collision avoidance is formulated as a partially observable decision-making problem. At each timestep t , the i th robot ($1 \leq i \leq N$) has access to an observation \mathbf{o}_i^t , which only provides partial knowledge about the world and then computes a collision-free steering command \mathbf{a}_i^t that drives itself toward the goal \mathbf{g}_i from the current position \mathbf{p}_i^t .

In our formulation, the observation \mathbf{o}_i^t is drawn from a probability distribution with respect to the latent system state \mathbf{s}_i^t , i.e., $\mathbf{o}_i^t \sim \mathcal{O}(\mathbf{s}_i^t)$ and it only provides partial information about the state \mathbf{s}_i^t . In particular, the i th robot cannot

access other robots' states and intents, which is in accord with the real-world situation. Compared with prior methods assuming a fully observable environment (e.g., Chen et al., 2017a,b; Claes et al., 2012; Hennes et al., 2012; van den Berg et al., 2011a, 2008), the partial observation assumption of our proposed approach is more applicable and more robust in real-world applications, because it is difficult for physical robots to accurately perceive the position and velocity of nearby moving obstacles, not to speak of the complete information about the entire scenario. The observation vector of each robot is 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 denotes its relative goal position (i.e., the coordinate of the goal in the robot's local polar coordinate frame), and \mathbf{o}_v^t refers to its current velocity. Note that our sensor-level collision-avoidance formulation does not need to distinguish between the static obstacles, moving obstacles, and active moving agents and the noisy raw sensor input, and thus it does not depend on the unreliable decision-making components based on object segmentation and recognition that may introduce additional error and ambiguity into the navigation system. In this way, the sensor-level collision avoidance is more generalized and robust than the agent-level collision-avoidance formulation in Chen et al. (2017a,b), van den Berg et al. (2008), and van den Berg et al. (2011a). In addition, the sensor-level collision avoidance also favors map-less navigation in an unstructured environment, as shown in the real-world experiments in Section 6.

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 π shared by all robots:

$$\mathbf{a}^t \sim \pi_\theta(\mathbf{a}^t | \mathbf{o}^t) \quad (1)$$

where θ denotes the policy parameters. The computed action \mathbf{a}^t is a vector representing the velocity \mathbf{v}^t driving the robot to approach its goal while avoiding collisions with other robots and obstacles \mathbf{B}_k ($0 \leq k \leq M$) within the time horizon Δt until the next observation \mathbf{o}^{t+1} is received. Hence, each robot sequentially makes a decision until it reaches the goal. Given the sequential decisions consisting of observations and actions (velocities) $(\mathbf{o}_i^t, \mathbf{v}_i^t)_{t=0:t_i}$, the trajectory of the i th robot, l_i , can be computed, starting from the position $\mathbf{p}_i^{t=0}$ to its desired goal $\mathbf{p}_i^{t=t_i} \equiv \mathbf{g}_i$, where t_i 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.,

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

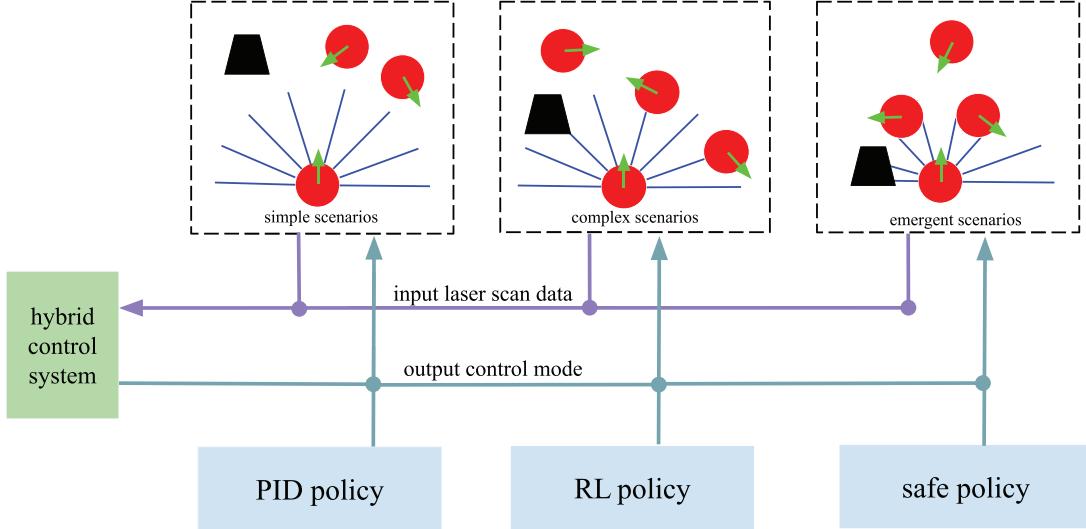


Fig. 1. An overview of the hybrid control architecture. According to the sensor measurement, a robot will classify its surrounding environment into three categories: the simple scenario, the complex scenario, and the emergent scenario. The robot will choose different controllers for different scenarios, in particular, the PID policy for simple scenarios, the reinforcement-learned policy for complex scenarios, and the safe policy for emergent scenarios. In this way, the robot can make a good balance between finishing its tasks efficiently and avoiding obstacles safely. For more details, please refer to Section 4.4.

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

$$\arg \min_{\pi_\theta} \mathbb{E} \left[\frac{1}{N} \sum_{i=1}^N t_i | \pi_\theta \right] \quad (3)$$

where t_i is the traveled time of the trajectory l_i in \mathbb{L} controlled by the shared policy π_θ .

The average arrival time will also be used as an important metric to evaluate the learned policy in Section 5. 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.

4. 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 based on a DNN. Then we discuss the training protocols used to optimize the policy and to bridge the simulation-to-real gap. Finally, we introduce the hybrid control framework (as shown in Figure 1), which combines the learning-based policy with traditional rule-based control methods to improve the controller's safety, effectiveness, and transferability.

4.1. Reinforcement-learning setup

The partially observable sequential decision-making problem defined in Section 3 can be formulated as a partially

observable Markov decision process (POMDP) solved with reinforcement learning. Formally, a POMDP can be described as a six-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, Ω 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, because 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. In the following, we describe the details of the observation space, the action space, and the reward function.

4.1.1. Observation space. As mentioned in Section 3, the observation \mathbf{o}^t consists of three parts: the readings of the 2D laser range finder \mathbf{o}_z^t , the relative goal position \mathbf{o}_g^t , and the 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° laser scanner that provides 512 distance values per scanning (i.e., $\mathbf{o}_z^t \in \mathbb{R}^{3 \times 512}$) with a maximum range of 4 m. In practice, the scanner is mounted on the forepart of the robot instead of in the center (as shown in the left image in Figure 2) 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.



Fig. 2. Direct deployment of our policy trained in simulation to physical robots without parameter fine-tuning. The physical robots (left) are square-shaped and twice the size of the disk-shaped robots for which we trained the policy in the simulation. The physical robot trajectories (right) in the *Circle* scenario generated from our learned navigation policy are smooth and safe. Here we use different colors to distinguish trajectories for different robots and use the color transparency to indicate the timing along a trajectory sequence. This experiment verifies the excellent generalization of our learned policy.

4.1.2. 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 $v \in (0.0, 1.0)$ m/s and the rotational velocity in $w \in (-1.0, 1.0)$ rad/s. Note that backward moving (i.e., $v < 0.0$ m/s) is not allowed because the laser range finder cannot cover the back region of the robot.

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

$$r_i^t = (^g r)_i^t + (^c r)_i^t + (^w r)_i^t \quad (4)$$

The reward r received by robot i at time step 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:

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

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

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

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

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

The reward functions defined above provide both the sparse and the dense feedback for agents. The sparse feedback includes r_{arrival} for rewarding the agents arriving at the goal and $r_{\text{collision}}$ for penalizing the potential collisions. These two types of feedback regulate the behavior of the robots to satisfy the physical hard constraints. On the other hand, the dense feedback encourages the robots either to reach the goal as fast as possible (i.e., ${}^g r$) or to move smoothly without large oscillations (i.e., ${}^w r$), which adjusts the behavior of the robots to satisfy task-related soft constraints. For a robotic application, the hard constraints are often more crucial than the soft constraints, so we assign large weights for hard constraints with $r_{\text{arrival}} = 15$ and $r_{\text{collision}} = -15$, and small weights for soft constraints with $\omega_g = 2.5$ and $\omega_w = -0.1$ in the training stage.

4.2. Network architecture

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

We design a four-hidden-layer neural network as a nonlinear function approximation to the policy π_θ . Its architecture is shown in Figure 3. We employ the first three hidden convolutional layers to effectively extract the high-level features from the laser measurements \mathbf{o}_z^t with local correlations (Mnih et al., 2015). The first layer convolves 32 one-dimensional filters (Conv1D) with kernel size = 5, stride = 2 over the three input scans and applies ReLU nonlinearities (Nair and Hinton, 2010). The second layer convolves 32 one-dimensional filters with kernel size = 3, stride = 2, again followed by ReLU nonlinearities. The third layer is a fully connected (FC) 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 FC layer with 128 rectifier units. The output layer is a FC layer with two different activations: a sigmoid

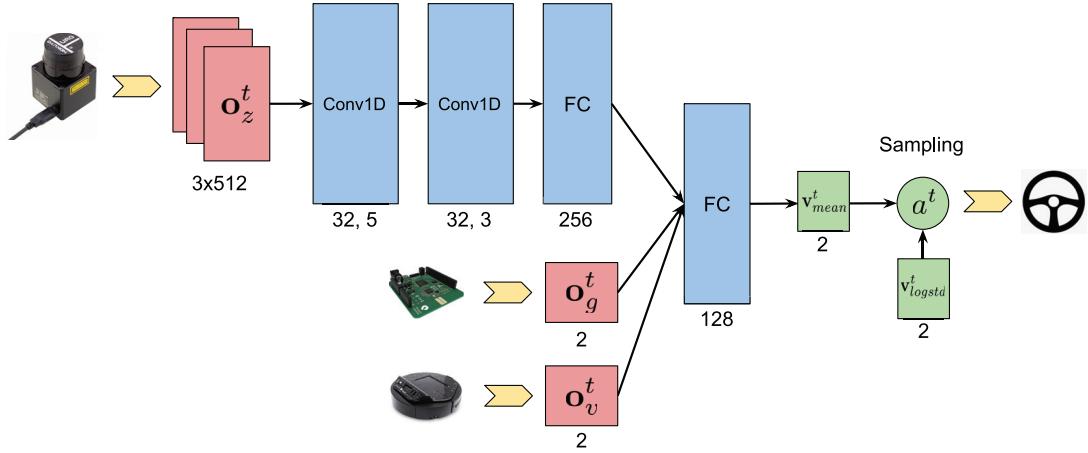


Fig. 3. The architecture of the policy network. The input of the network includes the scan measurements \mathbf{o}_z^t , the relative position \mathbf{o}_g^t of the goal with respect to the robot's current position and the current velocity \mathbf{o}_v^t . The network computes the mean of velocity $\mathbf{v}_{\text{mean}}^t$ by one-dimensional convolutional (Conv1D) layers and fully connected (FC) layers. The final action \mathbf{a}^t is sampled from a Gaussian distribution with a mean $\mathbf{v}_{\text{mean}}^t$ and a separated log standard deviation vector $\mathbf{v}_{\text{logstd}}^t$. The $\mathbf{v}_{\text{logstd}}^t$ vector is a standalone parameter that is optimized in the same way as in many policy gradient methods such as those in Schulman et al. (2015a) and Schulman et al. (2017a).

function, which is used to constrain the mean of translational velocity v^t within $(0.0, 1.0)$, and a hyperbolic tangent function (\tanh), which constrains the mean of rotational velocity w^t within $(-1.0, 1.0)$.

As a summary, the neural network maps the input observation vector \mathbf{o}^t to a vector $\mathbf{v}_{\text{mean}}^t$. The final action \mathbf{a}^t is sampled from a Gaussian distribution $\mathcal{N}(\mathbf{v}_{\text{mean}}^t, \mathbf{v}_{\text{logstd}}^t)$ that is used to model the stochastic policy formulated in (1), where $\mathbf{v}_{\text{logstd}}^t$ is a separate set of parameters referring to a log standard deviation, which will be updated only during the training process. Note that, although the action selection is stochastic during the training process, the actual policy executed by the robot is deterministic, i.e., the robot will choose the mean of the Gaussian distribution as the action to be taken. In other words, the robot will execute the maximally likely action that minimizes the probability of colliding with moving obstacles by taking into account the uncertainty in the environment. The entire input–output architecture is as shown in Figure 3.

4.3. Multi-scenario multi-stage training

In order to learn a robust policy, we present a multi-stage training scheme in varied scenarios.

4.3.1. Training algorithm. Although DRL algorithms have been successfully applied to mobile robot motion planning, they have mainly focused on a discrete action space (Zhang et al., 2016a; Zhu et al., 2017) or on small-scale problems (Chen et al., 2017a,b; Kahn et al., 2017; Tai et al., 2017). Large-scale, distributed policy optimization over multiple robots and the corresponding algorithms have been less studied for mobile robot applications.

Here we focus on learning a robust collision-avoidance policy for navigating a large number of robots in complex scenarios (e.g., mazes) with static obstacles. To accomplish

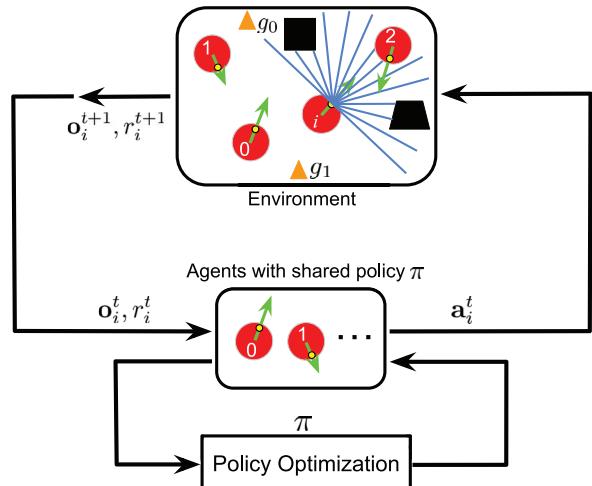


Fig. 4. An overview of our approach. At each time step t , each robot receives its observation \mathbf{o}_i^t and reward r_i^t from the environment, and generates an action a_i^t following the policy π . The policy π is shared across all robots and updated by a policy-gradient-based reinforcement-learning algorithm.

this, we deploy and extend a recently proposed robust on-policy policy gradient algorithm, the proximal policy optimization (PPO) (Heess et al., 2017; Schulman et al., 2015a, 2017b) in our multi-robot system. Our approach adapts the *centralized learning, decentralized execution* paradigm. In particular, the policy is learned from the centralized experiences collected by all robots simultaneously during training. In contrast, the testing is completely decentralized, i.e., each robot receives its own \mathbf{o}_i^t at each time step t and executes the action generated from the shared policy π_θ .

As summarized in Algorithm 1, which is adapted from Schulman et al. (2017b) and Heess et al. (2017), the

Algorithm 1. PPO for multiple robots

```

1: Initialize policy network  $\pi_\theta$  and value function  $V_\phi(s_t)$ , and set hyper-parameters as shown in Table 1.
2: for iteration = 1, 2, ..., do
3:   // Collect data in parallel
4:   for robot  $i = 1, 2, \dots, N$  do
5:     Run policy  $\pi_\theta$  for  $T_i$  time-steps, collecting  $\{\mathbf{o}_i^t, r_i^t, \mathbf{a}_i^t\}$ , where  $t \in [0, T_i]$ 
6:     Estimate advantages using GAE (Schulman et al., 2015b)  $\hat{A}_i^t = \sum_{l=0}^{T_i} (\gamma \lambda)^l \delta_i^l$ , where  $\delta_i^t = r_i^t + \gamma V_\phi(s_i^{t+1}) - V_\phi(s_i^t)$ 
7:     break, if  $\sum_{i=1}^N T_i > T_{\max}$ 
8:   end for
9:    $\pi_{\text{old}} \leftarrow \pi_\theta$ 
10:  // Update policy
11:  for  $j = 1, \dots, E_\pi$  do
12:     $L^{\text{PPO}}(\theta) = \sum_{t=1}^{T_{\max}} \frac{\pi_\theta(a_i^t | o_i^t)}{\pi_{\text{old}}(a_i^t | o_i^t)} \hat{A}_i^t - \beta \text{KL}[\pi_{\text{old}} | \pi_\theta] + \xi \max(0, \text{KL}[\pi_{\text{old}} | \pi_\theta] - 2 \text{KL}_{\text{target}})^2$ 
13:    if  $\text{KL}[\pi_{\text{old}} | \pi_\theta] > 4 \text{KL}_{\text{target}}$  then
14:      break and continue with next iteration  $i + 1$ 
15:    end if
16:    Update  $\theta$  with  $lr_\theta$  by Adam (Kingma and Ba, 2014) with respect to  $L^{\text{PPO}}(\theta)$ 
17:  end for
18:  // Update value function
19:  for  $k = 1, \dots, E_V$  do
20:     $L^V(\phi) = - \sum_{i=1}^N \sum_{t=1}^{T_i} (\sum_{t' > t} \gamma^{t'-t} r_i^{t'} - V_\phi(s_i^t))^2$ 
21:    Update  $\phi$  with  $lr_\phi$  by Adam with respect to  $L^V(\phi)$ 
22:  end for
23:  // Adapt KL penalty coefficient
24:  if  $\text{KL}[\pi_{\text{old}} | \pi_\theta] > \beta_{\text{high}} \text{KL}_{\text{target}}$  then
25:     $\beta \leftarrow \alpha \beta$ 
26:  else if  $\text{KL}[\pi_{\text{old}} | \pi_\theta] < \beta_{\text{low}} \text{KL}_{\text{target}}$  then
27:     $\beta \leftarrow \beta / \alpha$ 
28:  end if
29: end for

```

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 shared policy to generate trajectories until the maximum time period T_{\max} is reached and a batch of trajectory data is sampled. Then the sampled trajectories are used to construct the surrogate loss $L^{\text{PPO}}(\theta)$, which is optimized with the Adam optimizer (Kingma and Ba, 2014) for E_π epochs under the Kullback–Leibler (KL) divergence constraint. As a baseline for estimating the advantage \hat{A}_i^t , the state-value function $V_\phi(s_i^t)$ is approximated with a neural network with parameters ϕ using the sampled trajectories. The network structure of V_ϕ is the same as that of the policy network π_θ , except that it has only one linear activation unit in its last layer. In addition, we adopt L_2 -Loss $L^V(\phi)$ for V_ϕ , and optimize it with the Adam optimizer for E_V epochs as well. We update π_θ and V_ϕ independently and their parameters are not shared because we have found that using two separate networks could provide better performance in practice.

This parallel PPO algorithm can be easily scaled to a large-scale multi-robot system with hundreds of robots in a decentralized fashion since each robot in the team serves as an independent worker collecting data. The decentralized execution not only dramatically reduces the sampling time cost, but also makes the algorithm suitable for training a large number of robots in various scenarios. In this way,

our network can quickly converge to a solution with good generalization performance.

4.3.2. Training scenarios. To achieve a generalized model that can handle different complex real-world scenarios, we empirically build up multiple scenarios with a variety of obstacles using the Stage mobile robot simulator (as shown in Figure 5) and move all robots concurrently. For scenarios 1, 2, 3, 5, and 6 shown in Figure 5 (where the 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. In scenario 4, robots are randomly initialized in a circle with varying radii. As each robot only has a limited-range observation about its surrounding environment, these robots cannot make a farsighted decision to avoid congestion. As for scenario 7, we generate random positions for both robots and obstacles (as marked by the black areas) at the beginning of each episode; and the goals of robots are also selected randomly. In general, these rich, complex training scenarios enable robots to explore their high-dimensional observation space and can improve the quality, robustness, and generalization of the learned policy.

4.3.3. Training stages. Although training on multiple environments simultaneously brings robust performance

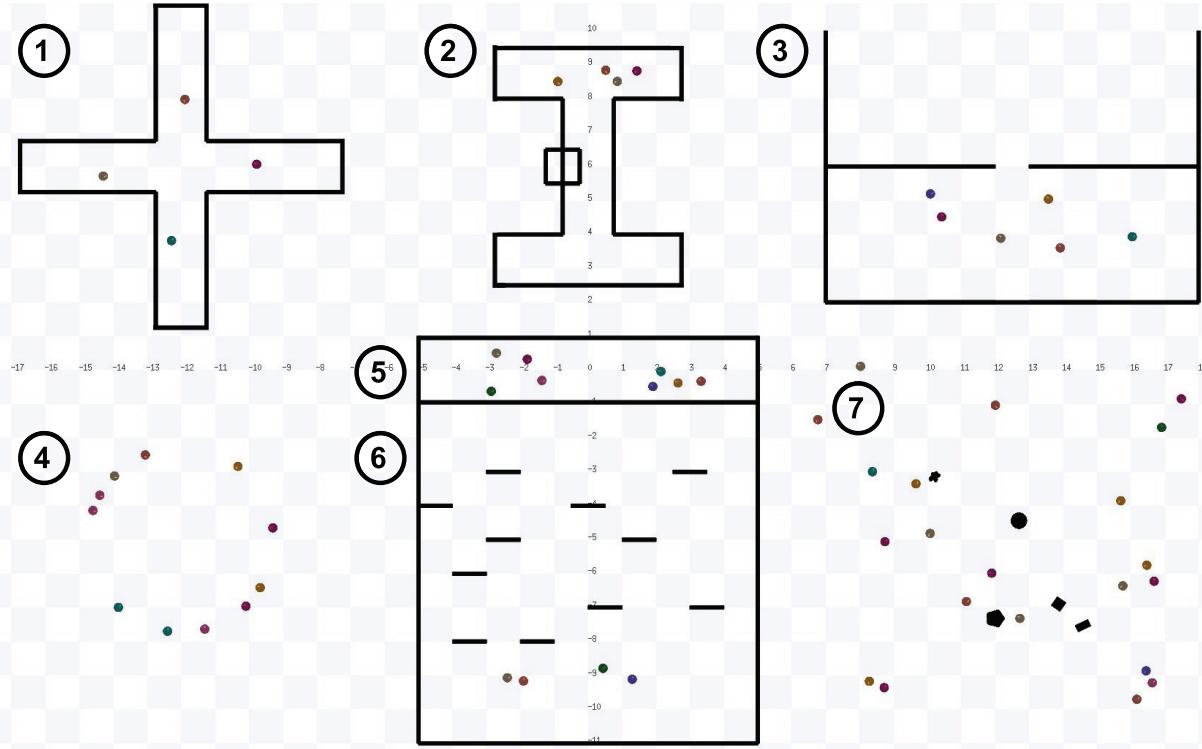


Fig. 5. Scenarios used to train the collision-avoidance policy. All robots are modeled as a disk with the same radius. Obstacles are shown in black.

over different test cases (as discussed in Section 5.2), it makes the training process harder. Inspired by the curriculum learning paradigm (Bengio et al., 2009), we propose a two-stage training process, which accelerates the policy to converge to a satisfactory solution, and gets higher rewards than the policy trained from scratch with the same number of epoch (as shown in Figure 6). In the first stage, we only train 20 robots on the random scenarios (i.e., scenario 7 in Figure 5) without any obstacles. This allows finding a reasonable solution quickly on relatively simple collision-avoidance tasks. Once the robots achieve stable performance, we stop the first stage and save the trained policy. The policy will continue to be updated in the second stage, where the number of robots increases to 58. The policy network is trained on the richer and more-complex scenarios shown in Figure 5. In our experiments discussed in Section 5, we call the policy generated after the first stage as the *RL Stage-1 Policy*, and the policy generated after the second stage as the *RL Stage-2 Policy*.

4.4. Hybrid control architecture

Our reinforcement-learned policy generalizes well in complex scenarios in both simulation and the real world. Unfortunately, it still cannot produce perfect behaviors all the time. There exist some trivial cases that the learned policy cannot provide high-quality collision-avoidance decisions. For instance, as a robot runs towards its goal through a wide-open space without other agents, the robot may

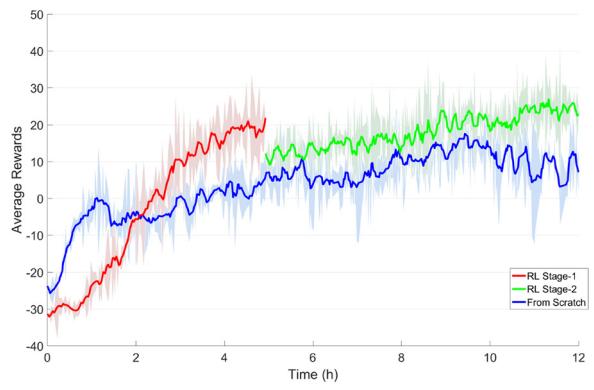


Fig. 6. Comparison of the average rewards shown in wall time for training in two stages (red line for the *RL Stage-1 policy* and green line for the *RL Stage-2 policy*) and for training from scratch (blue line). Multi-stage training scheme can achieve satisfactory performance within a shorter period of time.

approach the goal in a curved trajectory rather than in a straight line. We have also observed that a robot may wander around its goal rather than directly moving toward the goal, even though the robot is already in the close proximity of the target. In addition, the learned collision-avoidance policy is still more vulnerable in the real world than in simulation in terms of a higher collision rate.

These imperfect behaviors are due to several properties of the learning scheme. First, it is difficult to collect

Algorithm 2. Hybrid controller for multi-robot collision avoidance

Input: the 2D laser measurements \mathbf{o}_z^t and the distance between the robot's position to its goal \mathbf{o}_g^t

Output: the robot steering command \mathbf{a}^t

```

1: Configure three sub-policies: the PID control policy  $\pi_{\text{PID}}$ ,  

   the RL trained policy  $\pi_{\text{RL}}$  and the conservative safety  

   policy  $\pi_{\text{safe}}$ ; set the safety radius  $r_{\text{safe}}$  and the emergent  

   radius  $r_{\text{risk}}$   

2: if  $\min\{\mathbf{o}_z^t\} > r_{\text{safe}}$  or  $\min\{\mathbf{o}_z^t\} > \mathbf{o}_g^t$  then  

3:    $\mathbf{a}^t \leftarrow \pi_{\text{PID}}$   

4: else if  $\min\{\mathbf{o}_z^t\} \leq r_{\text{risk}}$  then  

5:    $\mathbf{a}^t \leftarrow \pi_{\text{safe}}$   

6: else  

7:    $\mathbf{a}^t \leftarrow \pi_{\text{RL}}$   

8: end if  

9: return  $\mathbf{a}^t$ 

```

training data from some special situations, e.g., the chance to sample a straight-line trajectory in our training scenarios is close to zero because the policy is modeled as a conditional probability distribution. Second, in high-dimensional continuous action space, the probability to learn low-dimensional optimal action subspaces, e.g., turnaround in place or moving in a straight line, is also extremely low. Third, the noises added in the action space during the training procedure also increase the difficulty of learning a deterministic optimal control. Finally, the simulation-to-real gap may fail the collision-avoidance policy in marginal cases. For instance, a robot that learned its collision-avoidance capability with simulated crowds may difficult to avoid real-world pedestrians. Such a simulation-to-real gap is caused by the difference between the simulation and the real world, including a robot's shape, size, dynamics, and the physical properties of the workspace.

However, the situations that are challenging for the learned policy could be simple for traditional control. For example, we can use a PID controller to move a robot in a straight line or to quickly push a robot toward its goal in a wide-open space without any obstacles. Hence, we present a hybrid control framework (as shown in Figure 1) that combines the learned policy and several traditional control policies to generate an effective composite policy. In particular, we first classify the scenario faced by a robot into three categories, and then design separate control sub-policies for each category. During the execution (Algorithm 2), the robot heuristically switches among different sub-policies. This simple switching process, even though heuristic, is user-friendly for tuning our model in the real-world scenarios.

4.4.1. Scenario classification. According to a robot's sensor measurement about the surrounding environment, we classify the scenarios faced by the robot into three categories: the simple scenarios, the complex scenarios, and the

Algorithm 3. Conservative safety policy

Input: the observation \mathbf{o}^t stated in Section 4.1.1.

Output: the robot steering command \mathbf{a}^t

```

1: Set the scale parameter  $\mathbf{p}_{\text{scale}}$  and the maximal moving  

   velocity  $\mathbf{v}_{\text{max}}$   

2: if  $\mathbf{o}_v^t > \mathbf{v}_{\text{max}}$  then  

3:    $\mathbf{a}^t \leftarrow (0, 0)$   

4: else  

5:    $\hat{\mathbf{o}}_z^t = \mathbf{o}_z^t / \mathbf{p}_{\text{scale}}$   

6:    $\mathbf{a}^t \leftarrow \pi_{\text{DRL}}(\hat{\mathbf{o}}_z^t, \mathbf{o}_g^t, \mathbf{o}_v^t)$   

7:    $\mathbf{a}^t \leftarrow \text{clip}(\mathbf{a}^t, -\mathbf{v}_{\text{max}}, \mathbf{v}_{\text{max}})$   

8: end if  

9: return  $\mathbf{a}^t$ 

```

emergent scenarios, by using the safe radius r_{safe} and the risk radius r_{risk} as classification parameters. The classification rule is as follows. When the measured distances from the robot to all nearby obstacles are larger than r_{safe} (i.e., $\min\{\mathbf{o}_z^t\} > r_{\text{safe}}$ and thus the robot is still far away from obstacles) or when the distance between the robot and target position is smaller than the distance to the nearest obstacle (i.e., $\mathbf{o}_g^t < \min\{\mathbf{o}_z^t\}$ and, thus, the robot may go straight toward the target), we classify this scenario as a simple scenario where the robot can focus on approaching the goal without worrying too much about collision avoidance. When the robot's distance to an obstacle is smaller than r_{risk} , we consider the situation as an emergent scenario because the robot is too close to the obstacle and, thus, needs to be cautious and conservative when making movement decisions. All other scenarios are classified as complex scenarios where the robot's distance to the surrounding obstacles is neither too small nor too large and the robot needs to make a good balance between approaching the goal and avoiding collisions.

In this work, r_{safe} and r_{risk} are two hyper-parameters and, thus, the switching rule among different scenarios is designed manually. Ideally, we hope to use DRL to learn these switch parameters automatically from data, similar to the meta-learning scheme (Frans et al., 2017). However, in practice, we find that it is difficult to guarantee DRL to recognize the emergent scenarios reliably. As a result, in this work, we only use the r_{safe} and r_{risk} as hyper-parameters to robustly distinguish different scenarios, and leave the meta-learning as the future work.

4.4.2. Sub-policies. Each of the three scenarios discussed above will be handled by one separate sub-policy. For the simple scenario, we will apply the PID control law π_{PID} , which provides an almost optimal solution to move the robot toward its target and guarantees stability. For the emergent scenario, we apply the conservative safe policy π_{safe} as shown in Algorithm 3 to generate a conservative movement by leveraging the DRL trained policy but scaling the neural network's input and restricting the output of the robot. For the complex scenario, we use the reinforcement-

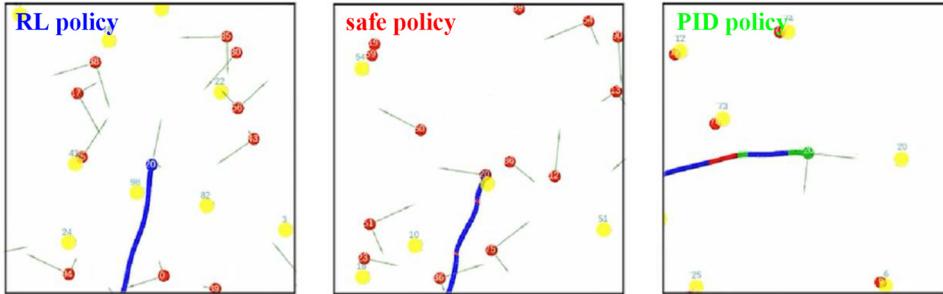


Fig. 7. Illustration of the three sub-policies in the hybrid control framework and their corresponding situations. We use different colors to indicate the different sub-policy taken at a given point on the trajectory. The blue color means that the robot is taking the RL policy π_{RL} to balance the navigation efficiency and safety in a situation that is neither too crowded nor too open. The red color indicates that the robot is taking the safe policy π_{safe} to deal with the obstacles that are very close. The green color means that the robot is in an open space and is taking PID control π_{PID} to approach its target quickly. Here the red points are the agents' current positions and the yellow points are the agents' goals.

learned policy π_{RL} directly, which uses the multi-scenario multi-stage reinforcement-learning framework to achieve state-of-the-art navigation performance. An intuitive illustration of the three sub-policies and their corresponding situations are provided in Figure 7.

There are two primary motivations for introducing the PID policy in the hybrid control framework. First, for many applications, especially in industrial scenarios, it is desirable to have a flexible controller whose parameters can be easily and intuitively tuned by engineers to adapt to different mobile platforms. The original neural network based policy, unfortunately, is difficult to tune due to its lack of interpretability. To overcome this limitation, we incorporate it with the PID controller that is well known for its ease of tuning, to improve the adaptivity of the navigation system. Second, it is believed that a DNN is easier to learn the residual of a complex function with reference to an identity function than directly learning the complex function itself, according to the insight of the widely used deep model ResNet (Xie et al., 2017). Similarly, here we believe that it is beneficial to let the PID controller deal with the collision avoidance in easy cases, so that deep-learned policy can focus on the challenging residual situations.

The safety policy is designed with two parameters for conveniently adjusting the navigation behavior in emergent cases. First, robots with large sizes should be more conservative in an emergency because their large volume results in relatively shorter response time. Thus, we use a scaling factor p_{scale} to make the reinforcement-learning policy more cautious for large robots in emergency (line 5 in Algorithm 3). Second, in an emergency we expect a robot to move slowly to increase the response time. Thus, we use the parameter v_{max} to restrict the movement speed of the robot (line 7 in Algorithm 3). To validate the above two changes of the reinforcement-learning policy in emergent cases, the speed of the obstacles should be slower than v_{max} as well to allow the reciprocal collision avoidance. When the obstacles move

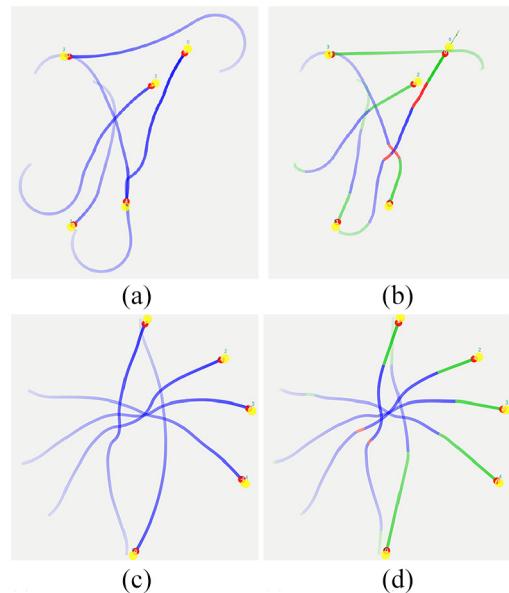


Fig. 8. Comparison of trajectories generated by RL policy and Hybrid-RL policy for a group of 5 agents in the random scene (top row) and in the circle scene (bottom row). In the random scene, the agents are starting from random positions and moving toward random goals. (a) Trajectories generated by the RL policy and (b) trajectories generated by the Hybrid-RL policy. In the circle scene, the agents are starting from positions on the circle and moving toward antipodal positions. (c) Trajectories generated by the RL policy and (d) trajectories generated by the Hybrid-RL policy. In (b) and (d), we use three colors to distinguish among different sub-policies chosen by each robot at run-time: red for the safe policy π_{safe} , green for the PID policy π_{PID} , and blue for the learned policy π_{RL} . In all figures, the color transparency is used to indicate the timing along a trajectory.

faster than v_{max} , the robot will be halted to guarantee safety (lines 2 and 3 in Algorithm 3).

4.4.3. Hybrid control behaviors. In Figure 8, we compare different behaviors of the reinforcement-learning policies with or without using the hybrid control, which are denoted as RL policy and Hybrid-RL policy, respectively. In both benchmarks, we can observe that Hybrid-RL will switch to π_{PID} in the open space, which helps the robot to go toward the goal in the most efficient way, rather than taking the curved paths as the RL policy. In cases when the collision risk is high, the agent will switch to π_{safe} to guarantee safety. For other cases, the agents will enjoy the flexibility of π_{RL} . The switch among sub-policies allows robots to fully leverage the available space by taking trajectories with short lengths and small curvatures. More experimental results verifying the effectiveness of the hybrid control architecture are also available latter in Section 5.

5. Simulation experiments

In this section, we first describe the setup details for the policy training, including the simulation setup, the hyper-parameters for the DRL algorithm, and the training time. Next, we provide a thorough evaluation of the performance of our hybrid multi-scenario multi-stage deep reinforcement-learned policy by comparing it with other approaches.

In particular, we introduce and compare four approaches in the experiments.

- The *NH-ORCA policy* is the state-of-the-art rule-based approach proposed by Alonso-Mora et al. (2013), Hennes et al. (2012), and Claes et al. (2012).
- The *SL policy* is trained with the supervised learning approach proposed by Long et al. (2017b)
- The *RL policy* is the original multi-scenario multi-stage reinforcement-learned policy without the hybrid architecture, as proposed by Long et al. (2017a).
- The *Hybrid-RL policy* is our proposed method in this article, which augments the RL policy with the hybrid control.

For the NH-ORCA policy, the *extended radius* is an important parameter that significantly influences its navigation behavior. In particular, a larger radius leads to a safer and more conservative collision-avoidance policy and vice versa. To present a fair comparison between the NH-ORCA policy and our approach, we consider three different radius settings in our experiment (i.e., 0.12, 0.15, and 0.18 m), which represent an aggressive NH-ORCA policy (NH-ORCA-A), a normal NH-ORCA policy (NH-ORCA-N), and a conservative NH-ORCA policy (NH-ORCA-C), respectively. The details for the implementation of the NH-ORCA are available in Appendix B.

We build the SL policy upon an identical neural network architecture to that in the Hybrid-RL policy (see Section 4.2), and follow the strategy of Long et al. (2017b) and Pfeiffer et al. (2017) to train the network in a supervised manner based on about 800,000 samples.

To further understand the behavior produced by the RL policy, we involve both the RL Stage-1 model and the RL Stage-2 model generated from the two-stage training process in our experiment. In addition, we also study the performance of our proposed multi-scenario multi-stage framework embedded with different reinforcement-learning algorithms. In particular, we replace the on-policy PPO algorithm used in our approach by the state-of-the-art off-policy algorithm Ape-X (Horgan et al., 2018; Lillicrap et al., 2015; Schaul et al., 2015) and investigate its corresponding performance in experiments. The implementation details of the Ape-X in our framework are available in Appendix A.

For the *Hybrid-RL policy*, we further compare its performance with the safe sub-policy π_{safe} , to evaluate the effectiveness of the individual sub-policy in the Hybrid-RL framework.

The comparison of these policies is performed from different perspectives, including the generalization to unseen scenarios and tasks, the navigation efficiency, and the robustness to the agent density and to the robots' inaccurate shape and dynamics. These experiments demonstrate the superiority of our proposed methods over previous methods.

5.1. Training setup

The training of our policy is implemented in TensorFlow and the large-scale multi-robot system with the 2D laser scanner measurements is simulated in the Stage simulator. The simulation time step is set as 0.1 s and the robot's control frequency is 10 Hz. Note that we do not add any noise into the simulation environment during the training process, in order to optimize the policy with low variance. Moreover, since the simulation-to-real gap of the 2D LiDAR data is not as significant as other sensor data such as images, it is not necessary to add noise to the observation to simulate the real-world situation. We train the multi-robot collision-avoidance policy on a computer with one i7-7700 CPU and one Nvidia GTX 1080 GPU. The offline training takes 12 hours to run about 600 iterations in Algorithm 1 so that the policy converges in all the training scenarios. The hyper-parameters in Algorithm 1 are summarized in Table 1. Specifically, the learning rate lr_θ of the policy network is set to 5×10^{-5} in the first training stage, and is then reduced to 2×10^{-5} in the second training stage. The hyper-parameters for the hybrid control in all experiments are summarized in Table 2. The training stops when the average cumulative reward converges (to 25 in our experiments).

The online execution of the learned policy for both simulation and the real-world robots runs in real-time. In simulation, it takes 3 ms on CPU or 1.3 ms on GPU for the policy network to compute new actions for 10 robots. Deployed on a physical robot with one Nvidia Jetson TX1, the policy network takes about 5 ms to generate online robot control commands.

Table 1. Hyper-parameters of our training algorithm described in Algorithm 1

Parameter	Value
λ in line 6	0.95
γ in line 6 and 20	0.99
T_{\max} in line 7	8,000
E_{ϕ} in line 11	20
β in line 12	1.0
KL_{target} in line 12	15×10^{-4}
ξ in line 12	50.0
lr_{θ} in line 16	5×10^{-5} (1st stage), 2×10^{-5} (2nd stage)
E_V in line 19	10
lr_{ϕ} in line 21	1×10^{-3}
β_{high} in line 24	2.0
α in line 24 and 27	1.5
β_{low} in line 26	0.5

Table 2. Hyper-parameters of our hybrid control system described in Algorithms 2 and 3.

Parameter	Value
r_{safe}	0.1
r_{risk}	0.8
$\mathbf{p}_{\text{scale}}$	1.25
v_{\max}	0.5

5.2. Generalization capability

A notable benefit of the multi-scenario training is the excellent generalization capability of our learned policy after the two-stage training. Thus, we first demonstrate the generalization of the RL policy with a series of experiments.

5.2.1. Non-cooperative robots. As mentioned in Section 3, our policy is trained on a team of robots, in which all robots share the same collision-avoidance strategy. Non-cooperative robots are not introduced in the entire training process. However, as shown in Figure 9, our learned policy can directly generalize to avoid non-cooperative agents, i.e., the rectangle-shaped robots which travel in straight lines with a fixed speed.

5.2.2. Heterogeneous robots. Our policy is trained on robots with the same disk shape and a fixed radius. Figure 10 demonstrates that the learned policy can efficiently navigate a group of heterogeneous robots with different sizes and shapes to reach their goals without any collisions.

5.2.3. Large-scale scenarios. To evaluate 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 11(b). This simulation experiment serves as a pressure test for our learned collision-avoidance policy, because the robots may run into congestion easily owing to

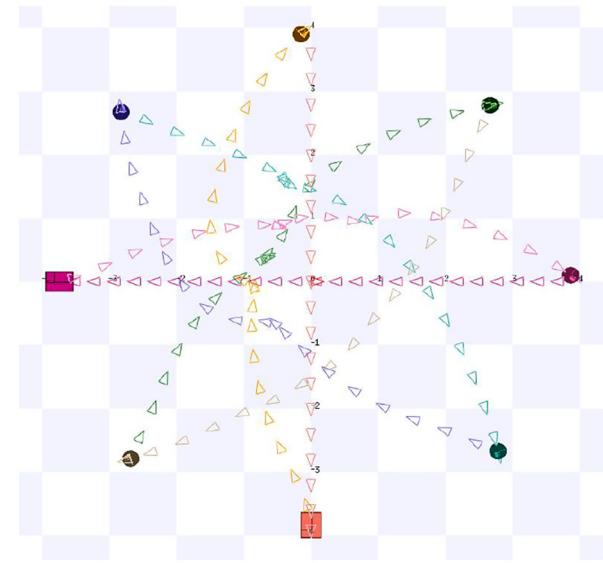


Fig. 9. Six disk-shaped robots controlled by the learned policy interact with two non-cooperative robots with the rectangle shape. The non-cooperative robots are traveling in straight lines with fixed high speed.

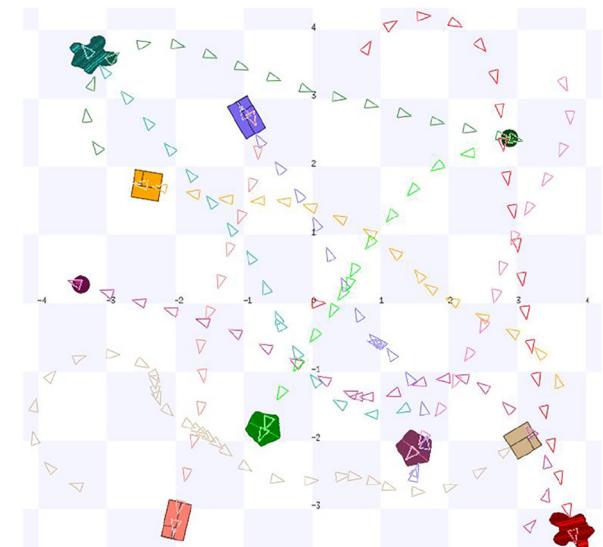


Fig. 10. Illustration of the collision-free and smooth trajectories of ten heterogeneous robots with different shapes and sizes. All robots adopt the same navigation policy which is trained by the prototype disk-shaped robots.

their limited sensing about the environment. The result shows that our learned policy generalizes well to large-scale environments without any fine-tuning. In addition, we also simulate 100 robots in the random scenario as shown in Figure 12(b). All these robots are able to arrive at their destinations without any collisions.

Note that the good generalization capability of the RL policy is also preserved in the Hybrid-RL policy, because all decisions in the complex and emergent situations are

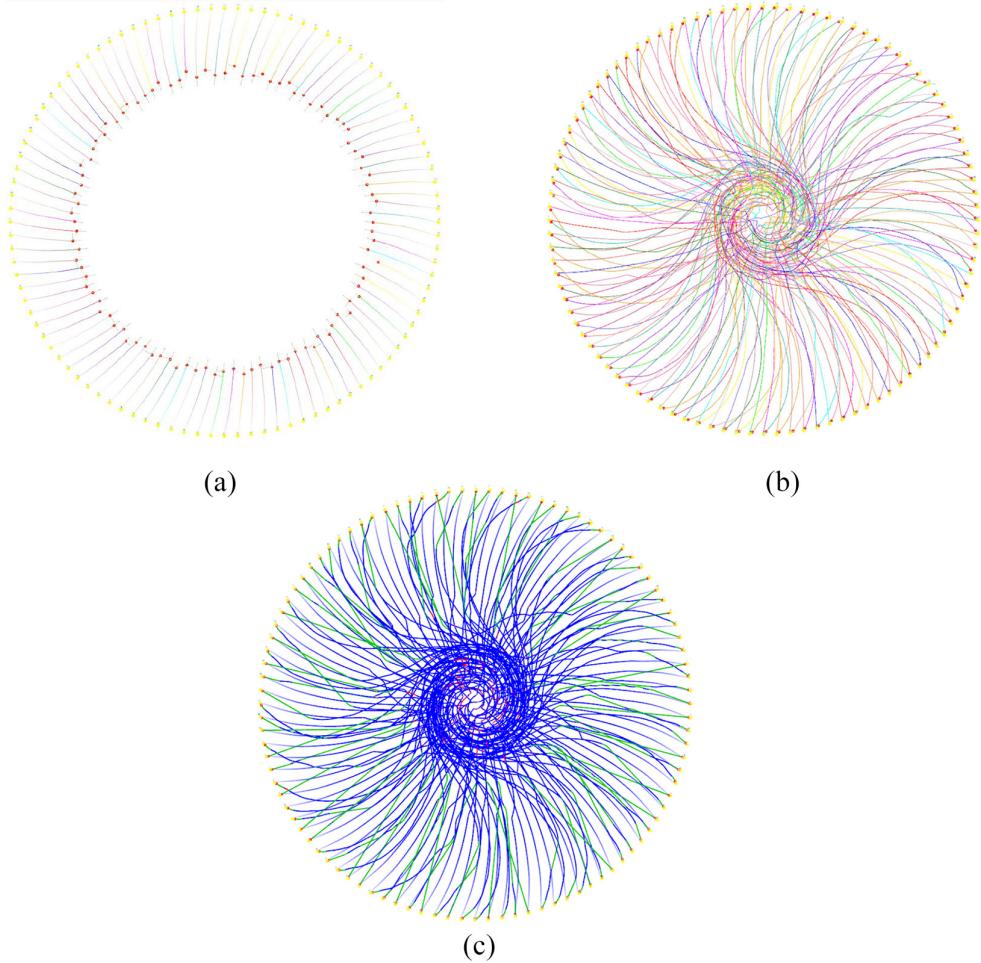


Fig. 11. (a) Simulation of 100 robots that are uniformly placed around a circle in the beginning and are trying to move through the center of a circle toward the antipodal positions. The red points are the robots' current positions. The yellow points are the robots' goals on the circle. In (b), we use different colors to distinguish among trajectories of different robots and use the color transparency to indicate the timing along a trajectory sequence. In (c), we use three colors to distinguish the different sub-polices chosen by each robot in run-time: red for the safe policy π_{safe} , green for the PID policy π_{PID} , and blue for the learned policy π_{RL} . It is worth noting that there is no explicit coordination among agents. Instead, the neural network policy learns how to accomplish coordinated behavior in a large-scale scenario unseen in the training stage. Please also refer to the video for a detailed comparison between the trajectories of RL policy and Hybrid-RL policy.

made by the RL policy. The generalization of the Hybrid-RL policy is also verified by the pressure test of 100 robots, as shown in Figures 11(c) and 12(c). In both scenarios, most robots adopt the π_{PID} policy near the start positions and goals (shown as green lines in Figures 11(c) and 12(c)) owing to the considerable distance to its neighbors. They then switch to the π_{RL} or π_{safe} policy when they encounter other robots (shown as blue or red lines in Figures 11(c) and 12(c)).

5.3. Efficiency evaluation

In this section, we evaluate the performance of the proposed hybrid collision-avoidance policy in terms of the navigation efficiency. In particular, we first present multiple

performance metrics for evaluating the navigation efficiency and then compare different policies.

5.3.1. Performance metrics. To compare the performance of our approach with other methods, we use the following metrics.

- *Success rate* is the ratio of the number of robots reaching their goals in a certain time limit without any collisions to the total number of robots.
- *Extra time* \bar{t}_e measures the difference between the average travel time of all robots and the lower bound of robots' travel time, which is computed as the average travel time of going straight toward goals at the

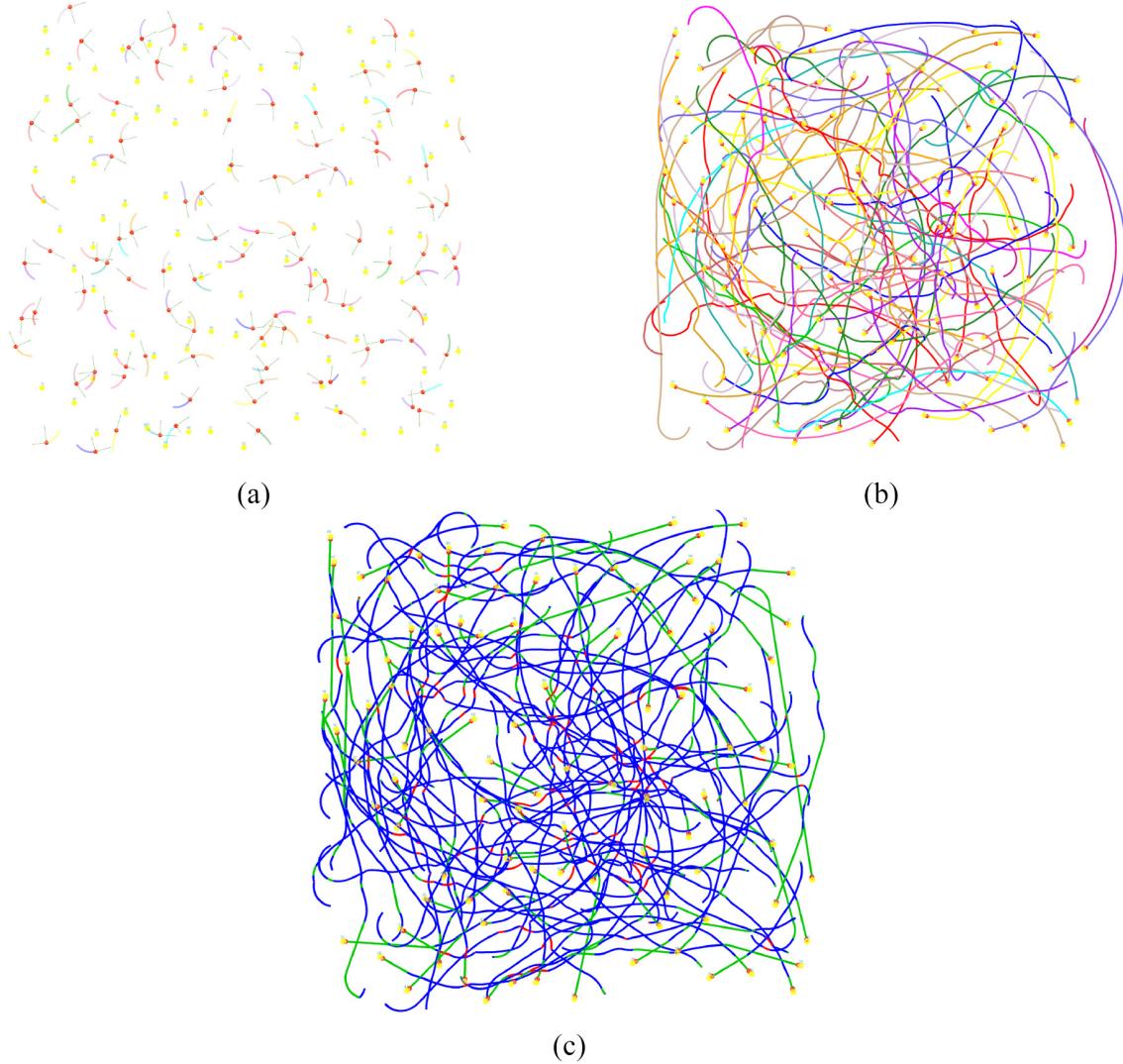


Fig. 12. (a) Simulation of 100 robots that move from their random start positions to their own random goals. The red points are the robots' current positions and the yellow points are the robots' destinations. In (b), we use different colors to distinguish the trajectories of different robots and use the color transparency to indicate the propagation of each trajectory. In (c), we use three colors to distinguish the different sub-polices chosen by each robot in run-time: red for the safe policy π_{safe} , green for the PID policy π_{PID} , and blue for the learned policy π_{RL} . Please also refer to the video for the detailed comparison between RL policy and Hybrid-RL policy.

maximum speed without collision avoidance (Chen et al., 2017b; Godoy et al., 2016a).

- *Extra distance* \bar{d}_e measures the difference between the average traveled trajectory length of all robots and the lower bound of robots' traveled distance, which is computed as the average traveled distance for robots following the shortest paths toward their goals.
- *Average speed* \bar{v} measures the average speed of all robots during the navigation.

Note that in the definitions of the *extra time* \bar{t}_e and *extra distance* \bar{d}_e , we use the lower-bound baseline to alleviate the biases brought by the different number of agents and

different distances to goals. Hence, we can provide a fair comparison among different methods, and this is important especially for the random scenarios.

As all four policies being compared (the NH-ORCA policy, the SL policy, the RL policy, and the Hybrid-RL policy) contain some randomness, we run each method 50 times for each test scenario, and then report the mean and variance for each metric.

5.3.2. Circle scenarios. We first compare our Hybrid-RL policy with other methods on several circle scenarios with different number of robots. These scenarios are similar to scenario 4 in Figure 5, except that the initial positions of

Table 3. Performance metrics (as mean/SD) evaluated for different methods on the circle scenarios with varied scene sizes and different number of robots.

Metrics	Method	#agents (radius of the scene)						
		4 (2.5 m)	6 (3.0 m)	8 (3.5 m)	10 (4.0 m)	12 (4.5 m)	15 (5.0 m)	20 (6.0 m)
Success rate	SL	0.6000	0.7167	0.6125	0.71	0.6333	—	—
	NH-ORCA-A	1.0	0.9667	0.9250	0.8900	0.9000	0.8067	0.7800
	NH-ORCA-N	0.9900	0.9933	0.9975	0.9860	0.9767	0.992	0.988
	RL (Ape-X)	0.9050	0.9567	0.9975	0.9920	0.9517	0.9613	0.9430
	RL (PPO)	1.0	1.0	1.0	1.0	1.0	1.0	0.9827
	Hybrid-RL	1.0	1.0	1.0	1.0	1.0	1.0	1.0
Extra time	SL	9.254/2.592	9.566/3.559	12.085/2.007	13.588/1.206	19.157/2.657	—	—
	NH-ORCA-A	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
	NH-ORCA-N	2.276/1.138	3.746/2.251	5.862/2.013	6.298/1.531	7.774/2.335	9.110/2.281	13.152/3.810
	RL (Ape-X)	0.316/0.149	1.120/0.443	0.529/0.166	0.567/0.134	1.458/0.386	1.681/0.417	3.11/0.526
	RL (PPO)	0.323/0.025	0.408/0.009	0.510/0.005	0.631/0.011	0.619/0.020	0.490/0.046	0.778/0.032
	Hybrid-RL	0.251/0.007	0.408/0.008	0.494/0.006	0.629/0.009	0.518/0.005	0.332/0.007	0.702/0.013
Extra distance	SL	0.358/0.205	0.181/0.146	0.138/0.079	0.127/0.047	0.141/0.027	—	—
	NH-ORCA-A	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
	NH-ORCA-N	0.380/0.022	0.452/0.055	0.494/0.049	0.518/0.041	0.542/0.036	0.571/0.036	0.608/0.042
	RL (Ape-X)	0.004/0.002	0.056/0.015	0.022/0.007	0.029/0.007	0.041/0.009	0.036/0.006	0.066/0.006
	RL (PPO)	0.028/0.006	0.028/0.001	0.033/0.001	0.036/0.001	0.038/0.002	0.049/0.005	0.065/0.002
	Hybrid-RL	0.013/0.001	0.028/0.001	0.031/0.001	0.036/0.001	0.039/0.001	0.033/0.001	0.058/0.001
Average speed	SL	0.326/0.072	0.381/0.087	0.354/0.042	0.355/0.022	0.308/0.028	—	—
	NH-ORCA-A	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
	NH-ORCA-N	0.702/0.095	0.645/0.132	0.558/0.092	0.566/0.065	0.547/0.079	0.530/0.058	0.487/0.070
	RL (Ape-X)	0.945/0.061	0.846/0.053	0.930/0.021	0.934/0.015	0.862/0.033	0.857/0.030	0.795/0.028
	RL (PPO)	0.939/0.004	0.936/0.001	0.932/0.001	0.927/0.001	0.936/0.002	0.953/0.004	0.939/0.002
	Hybrid-RL	0.952/0.001	0.936/0.001	0.934/0.001	0.927/0.001	0.946/0.001	0.968/0.001	0.945/0.001

these robots are uniformly along the circle. We test in seven circle scenarios with 4, 6, 8, 10, 12, 15, and 20 agents each. Accordingly, we adjust the circle radius in each scenario to make sure that its average agent density (i.e., the number of agents divided by the area of the circle) is similar (around 0.2agent/m²).

The results of the comparison are summarized in Table 3. The Hybrid-RL is superior to other approaches in terms of success rate, extra time, and average speed. The RL-based policies, including the RL (PPO) policy and the RL (Ape-X) policy, outperform their NH-ORCA-based competitors in both extra time and average speed while achieving similar success rates. Moreover, the RL (PPO) policy, when compared with its counterpart the RL (Ape-X) policy, has achieved slight improvements in all aspects due to the advantage of the on-policy PPO in learning the reciprocal behavior. Among the NH-ORCA-based policies, we observe that the success rate of the NH-ORCA-A policy decreases when increasing the number of agents, while the NH-ORCA-N policy always maintains a good success rate (at least 95%) and consistent performance. On the other hand, the NH-ORCA-A policy has a higher speed and shorter extra time than NH-ORCA-N policy. Note that we do not report the experimental results of the NH-ORCA-C policy because its performance is dominated by

the NH-ORCA-N policy with worse performance on extra time and average speed and a similar success rate. The SL policy has the lowest success rate and cannot achieve satisfactory performance in most scenes.

Furthermore, as indicated by the *extra distance*, the RL (Ape-X) policy provides a comparable or even shorter traveled path than other approaches since its off-policy reinforcement-learning algorithm tends to learn a more greedy behavior other than a reciprocal behavior, which leads to a more straightforward trajectory toward the goal. In contrast, our on-policy reinforcement-learned policies produce more consistent behavior for robots to resolve the congestion situation efficiently. Thus, the robots avoid unnecessary velocity reduction and obtain a higher average speed (as demonstrated in Figures 15 and 16). For the NH-ORCA-based policies, the NH-ORCA-A policy generates shorter trajectories than the NH-ORCA-N policy owing to its aggressive behavior that arises from the smaller extended radius (as shown in Figures 13 and 14).

We also observe that the on-policy RL policies (i.e., RL (PPO) and Hybrid-RL) produce slightly longer paths in some scenarios since the robots need more space to decelerate from high speed before stopping at goals or to rotate at a larger radius of curvature caused by the higher angular velocity. For example, in Figures 13, 14, 15, and 16, we

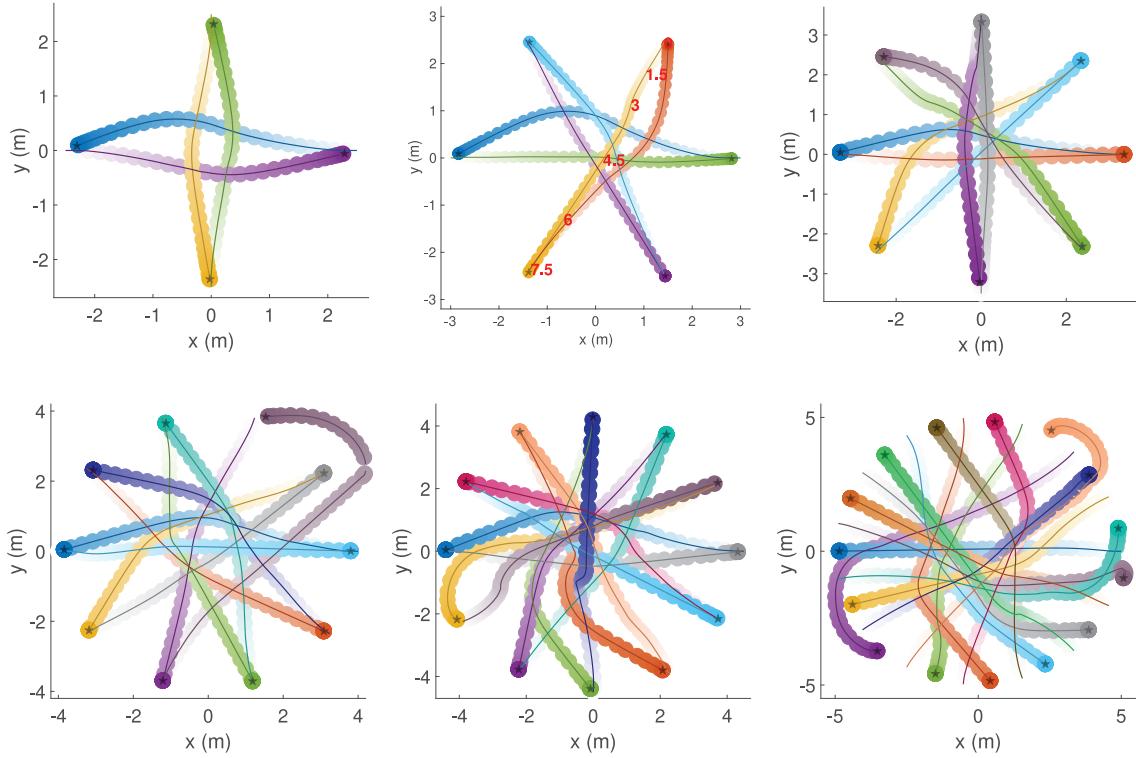


Fig. 13. Trajectories of agents executing the NH-ORC-A policy in circle scenarios with 4, 6, 8, 10, 12 and 15 agents, respectively. We use different colors to represent trajectories of different robots and use the color transparency to indicate the temporal state along each trajectory. For the scenario with 6 robots, we show the time stamps (in red) for a few waypoints on the trajectory of one robot.

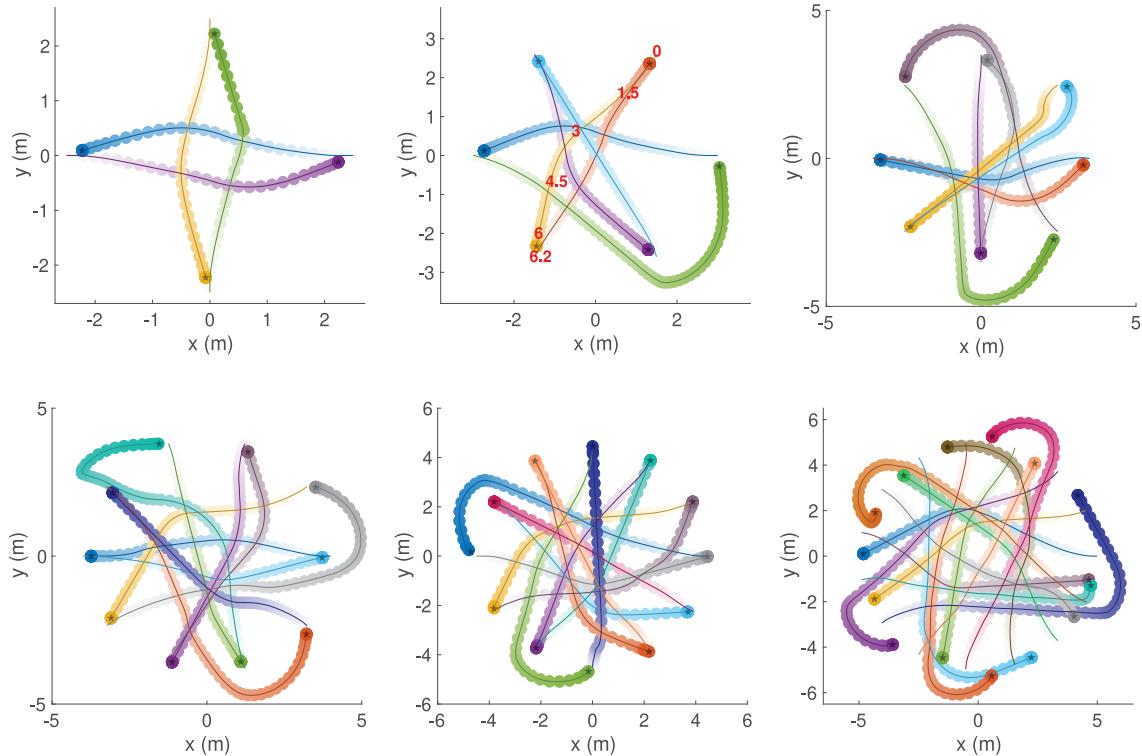


Fig. 14. Trajectories of agents executing the NH-ORCA-C in circle scenarios with 4, 6, 8, 10, 12, and 15 agents, respectively. We use different colors to represent trajectories of different robots and use the color transparency to indicate the temporal state along each trajectory. For the scenario with 6 robots, we show the time stamps (in red) for a few waypoints on the trajectory of one robot.

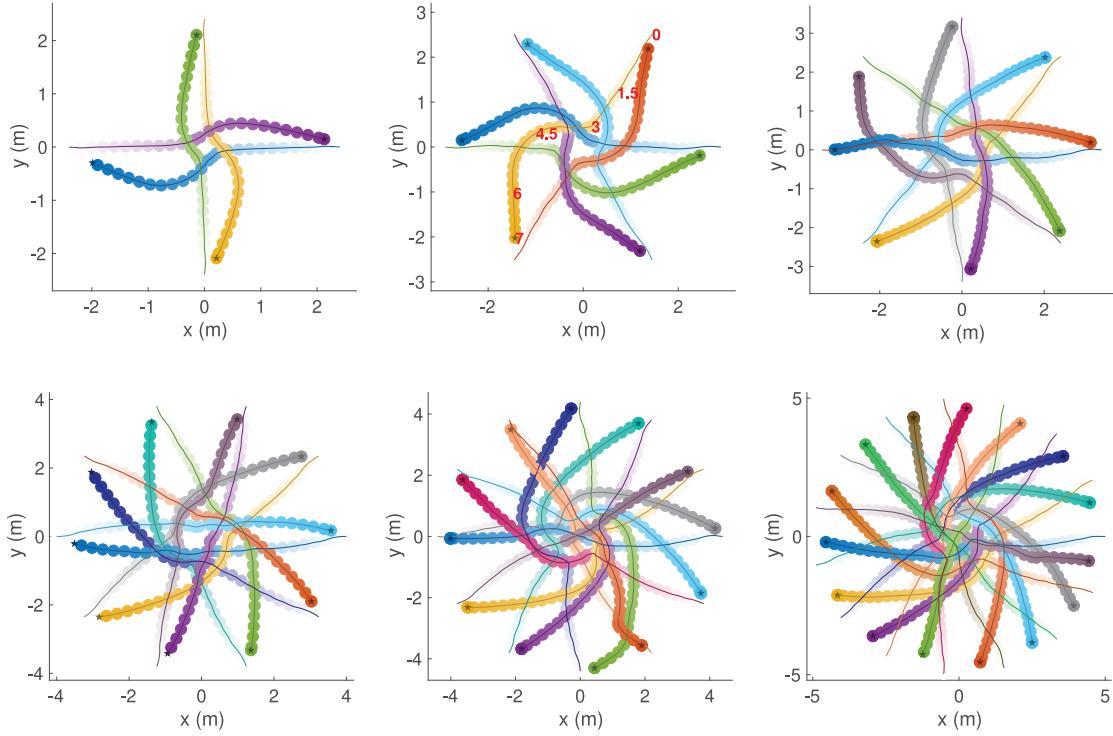


Fig. 15. Trajectories of agents executing the Ape-X in circle scenarios with 4, 6, 8, 10, 12, and 15 agents, respectively. We use different colors to represent trajectories of different robots and use the color transparency to indicate the temporal state along each trajectory. For the scenario with 6 robots, we show the time stamps (in red) for a few waypoints on the trajectory of one robot.

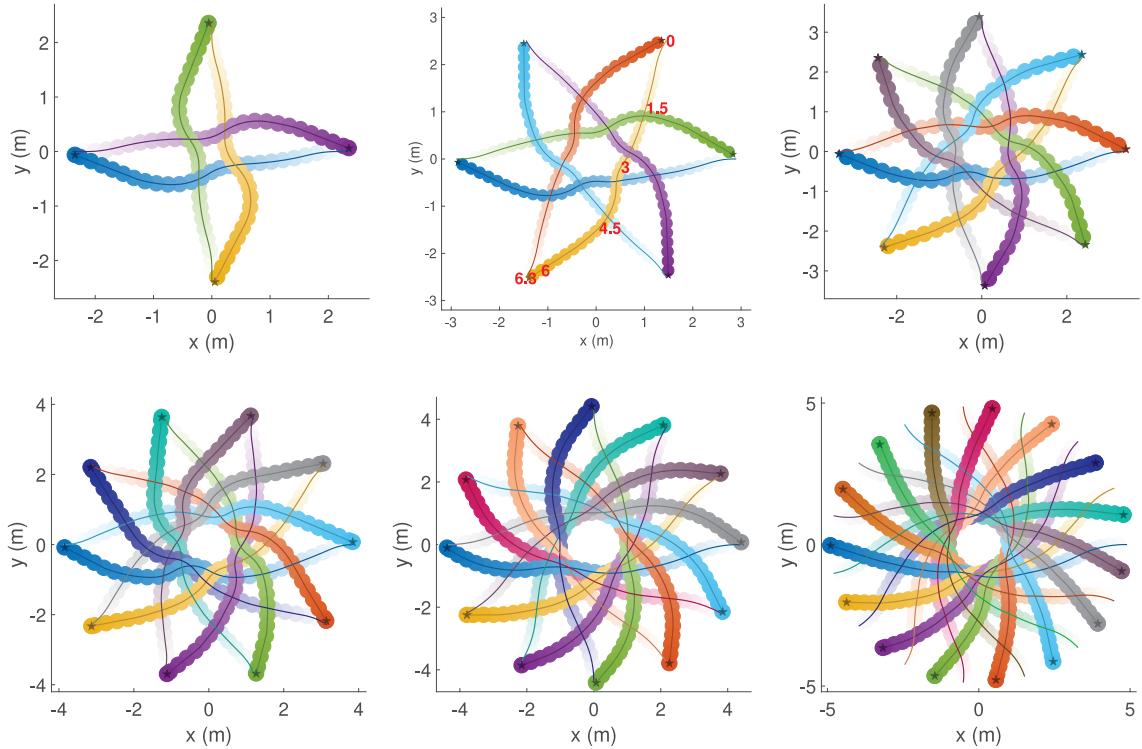


Fig. 16. Trajectories of agents executing our Hybrid-RL policy in circle scenarios with 4, 6, 8, 10, 12, and 15 agents, respectively. We use different colors to represent trajectories for different robots and use the color transparency to indicate the temporal state along each robot's trajectory sequence. For the scenario with 6 robots, we show the time stamps (in red) for a few waypoints on the trajectory of one robot.

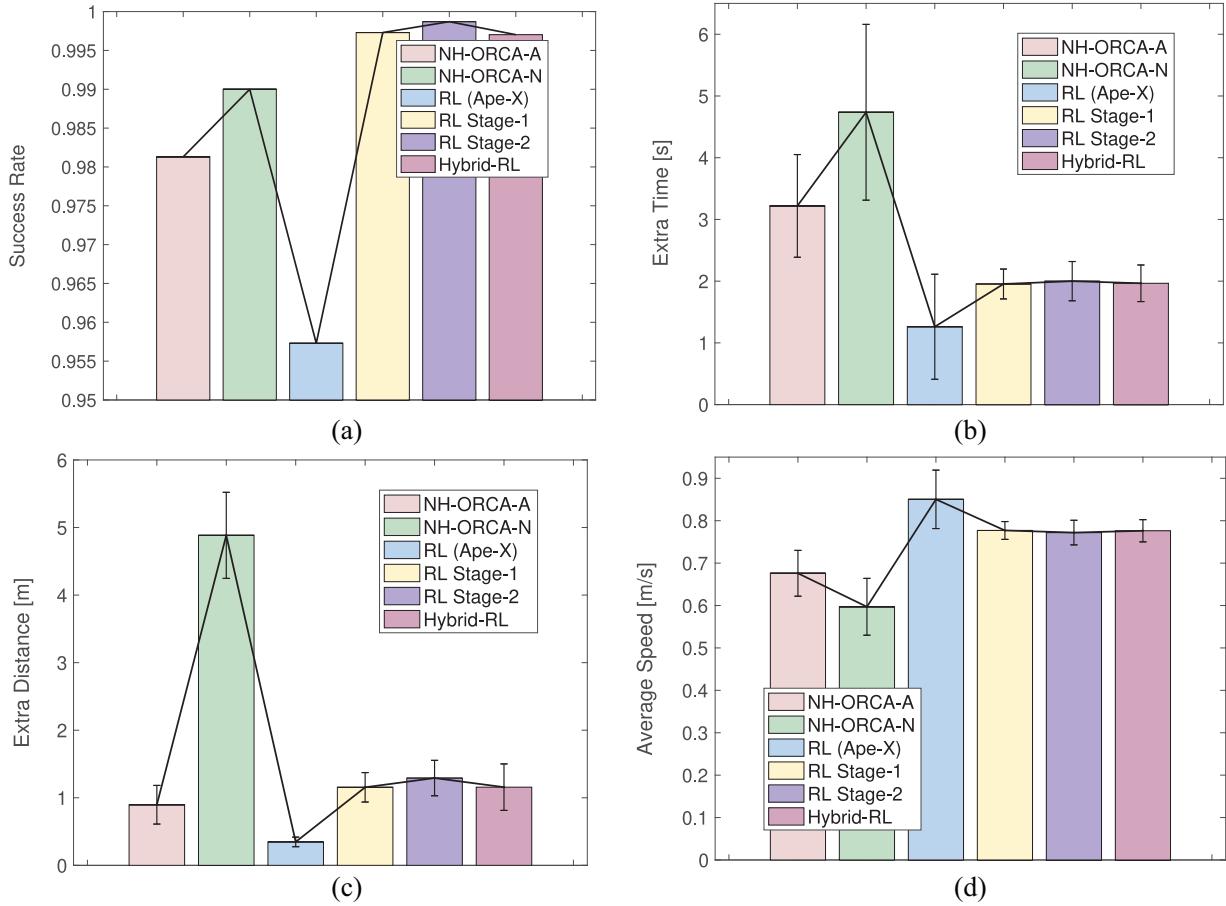


Fig. 17. Performance metrics evaluated for reinforcement-learning-based policies (RL (Ape-X), RL Stage-1 policy, RL Stage-2 policy, and Hybrid-RL policy) and the NH-ORCA policy on random scenarios: (a) success rate; (b) extra time; (c) extra distance; (d) average speed.

show the time stamps for the trajectory of the same robot in a scenario with six agents but using different navigation policies. As we can see, compared with the trajectory of the NH-ORCA-A, the Hybrid-RL indeed provides higher speed, though it has a longer path.

5.3.3. Random scenarios. Random scenarios are another type of scenario frequently used to evaluate the performance of multi-robot collision avoidance. One example of the random scenarios is shown in scenario 7 in Figure 5, where each robot is assigned a random start point and a random goal position. To measure the performance of our method on random scenarios, we create 5 different random 1scenarios with 15 robots in each. For each random scenario, we repeat our evaluations 50 times and the evaluation results are summarized in Figure 17. We compare the Hybrid-RL policy with the RL (Ape-X) policy, the RL Stage-1 policy, the RL Stage-2 policy, the NH-ORCA-A policy, and the NH-ORCA-N policy. Note that the difficulty of the random scenarios is lower compared with the

circle scenarios, because the random distribution of agents makes the traffic congestion unlikely to happen.

As shown in Figure 17(a), we observe that all policies trained using on-policy DRL achieve success rates close to 100%, which means they are safer than other methods. For the RL (Ape-X), it has a relatively lower success rate than others but is dominant on the other three metrics. Similar to the circle scenarios mentioned previously, the greedy behavior learned by off-policy reinforcement learning makes the whole trajectory shorter, and it can achieve better speed and shorter extra time in random scenes at the cost of a higher collision probability. Regarding the NH-ORCA-based policies, NH-ORCA-A has a slightly lower success rate but is much better than NH-ORCA-N on the other three metrics.

In addition, as shown in Figure 17(b), robots using the on-policy reinforcement-learned policies reach their goals much faster than those using the NH-ORCA policies. Moreover, the three on-policy reinforcement-learning-based policies have similar overall performance. The RL Stage-1 policy's performance is a bit higher than the RL Stage-2 policy (e.g., according to the extra distance metric). This is probably because the RL Stage-1 policy is trained in similar

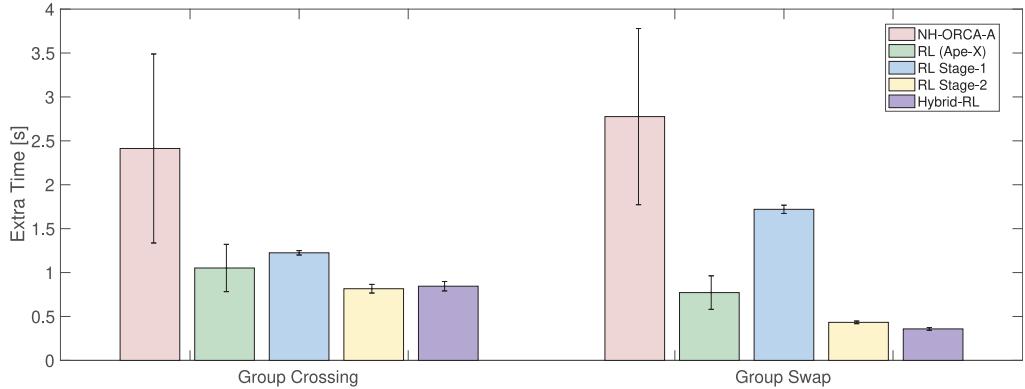


Fig. 18. Extra time \bar{t}_e of our policies (RL (Ape-X), RL Stage-1 policy, and RL Stage-2 policy) and the NH-ORCA policy on two group scenarios.

scenarios as the test scenarios and, thus, it may be overfitted, whereas the RL Stage-2 policy is trained in miscellaneous scenarios for better generalization. Our Hybrid-RL policy further improves the RL Stage-2 policy and achieves similar overall performance as the RL Stage-1 policy, because the traditional control sub-policies improve the trajectory’s optimality, and thus help to make a better balance between the optimality and generalization.

5.3.4. Group scenarios. In order to evaluate the cooperation between robots, we test our learned policy on more challenging group scenarios, including group swap, group crossing, and group moving in the corridors. In the group swap scenario, two groups of robots, each with five robots, are moving in opposite directions to swap their positions. As for the group crossing scenario, robots are separated into two groups with four robots each, and their paths intersect in the center of the scenario.

We compare our method with the NH-ORCA-A policy on these two cases by measuring the average extra time \bar{t}_e with 50 trials. As summarized in Figure 18, the reinforcement-learned policies perform much better than the NH-ORCA policy on both benchmarks. The robots take shorter time to reach goals when using the learned policies, which indicates that the learned policies produce more cooperative behaviors than the NH-ORCA policy. Among the four learned policies, the Hybrid-RL policy provides the best performance. Compared with the RL (Ape-X), the on-policy reinforcement-learned policies generated from the complete stage-2 training can learn more reciprocal behavior and it takes shorter extra time. In addition, we also illustrate the trajectories of different methods in both scenarios in Figures 19 and 20. From these trajectories, we observe that the NH-ORCA-A policy generates unnecessary circuitous trajectory. The RL (Ape-X) and the RL Stage-1 policy tend to generate stopping behaviors when one agent is blocked by another, whereas the RL Stage-2 policy and Hybrid-RL policy provide smooth and collision-free trajectories.

5.3.5. Corridor scenario. We further evaluate the four policies in a corridor scene, where two groups exchange their positions via a narrow corridor connecting two open regions, as shown in Figure 21(a).

Note that this is a benchmark with static obstacles, and we have to use different pipelines when using the NH-ORCA policy and the learned policies. In particular, the NH-ORCA requires prior knowledge about the global environment (usually a map from SLAM) to identify the static obstacles and then depends on global planners to guide robots navigating in the complex environment. The global planners do not consider the other moving agents, and thus such guidance is sub-optimal and significantly reduces the efficiency and safety of the navigation. For the learned policies, we are using them in a map-less manner, i.e., each robot only knows its goal but without any knowledge about the global map. The goal information can be provided by a global localization system such as the GPS or UWB in practice. A robot will use the vector pointing from its current position to the goal position as guidance toward the target. Such guidance may fail in buildings with complex topologies but works well for common scenarios with moving obstacles.

Navigation in the corridor scenario is a challenging task, and only the Hybrid-RL policy and the RL Stage-2 policy can complete it. The trajectories generated by the Hybrid-RL policy are illustrated in Figure 21(b). The failure of the RL Stage-1 policy indicates that the co-training on a variety of scenarios is beneficial for the robustness across different situations. The NH-ORCA policy fails with many collisions.

5.3.6. Summary. In this section, we have validated the superiority of our learned policies in multiple scenarios by comparing it with the state-of-the-art rule-based reaction controller NH-ORCA. In addition, we demonstrate that the on-policy reinforcement learning can learn a better reciprocal behavior than the off-policy reinforcement learning. Moreover, we also show that the hybrid control framework can effectively improve the learned policy’s performance in many scenarios.

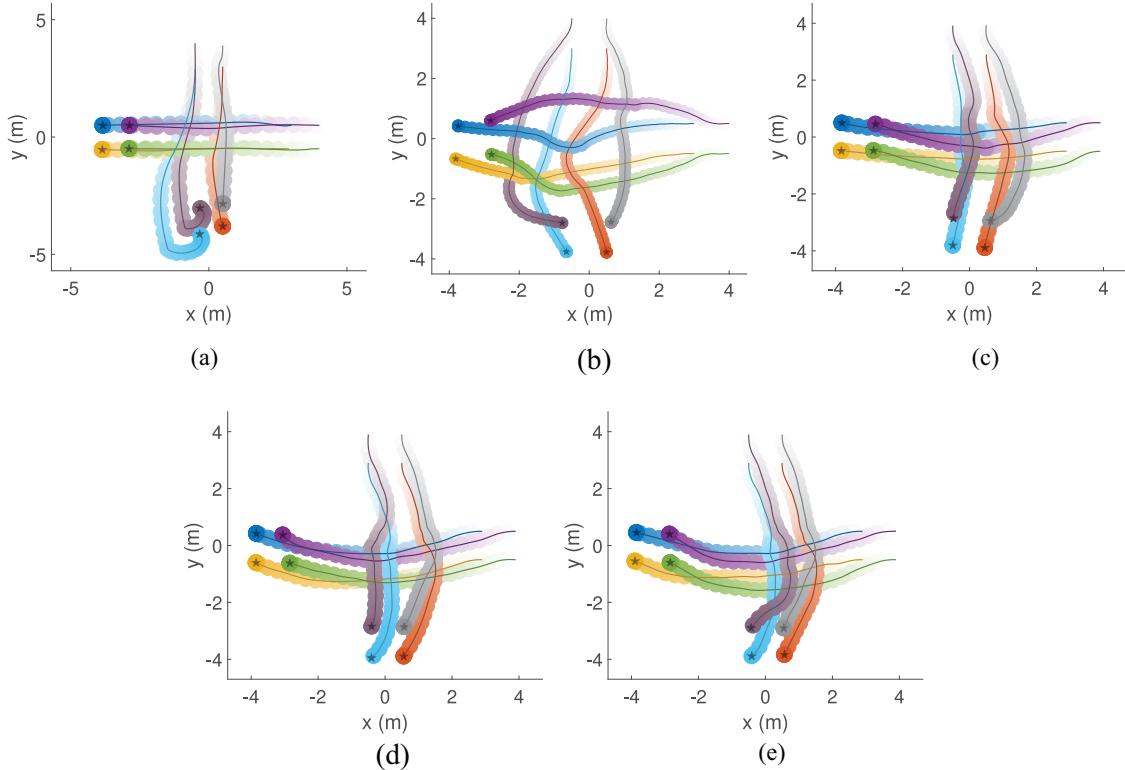


Fig. 19. Comparison of trajectories generated by different policies in group crossing scenarios: (a) NH-ORCA-A policy; (b) RL (Ape-X) policy; (c) RL Stage-1 policy; (d) RL Stage-2 policy; (e) Hybrid-RL policy. We use different colors to distinguish among trajectories of different agents and use the color transparency to indicate the timing of a trajectory sequence.

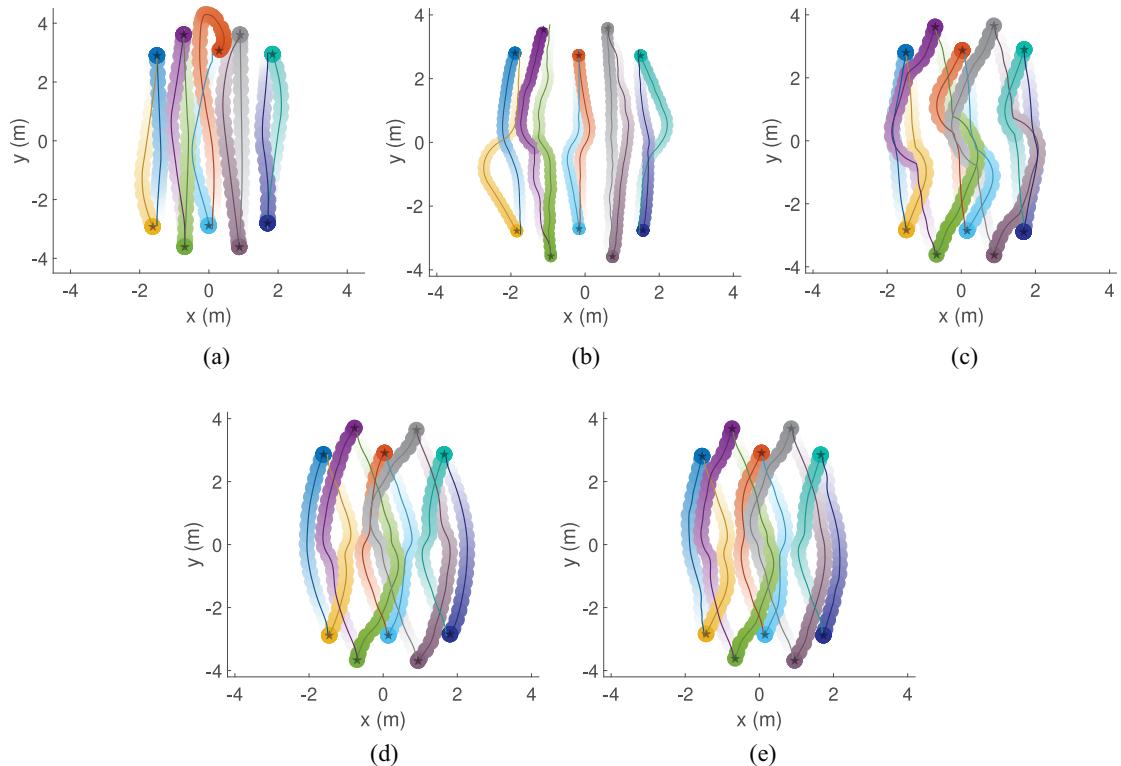


Fig. 20. Comparison of trajectories generated by different policies in group swap scenarios: (a) NH-ORCA-A policy; (b) RL (Ape-X) policy; (c) RL Stage-1 policy; (d) RL Stage-2 policy; (e) Hybrid-RL policy. We use different colors to distinguish among trajectories of different agents and use the color transparency to indicate the timing of a trajectory sequence.

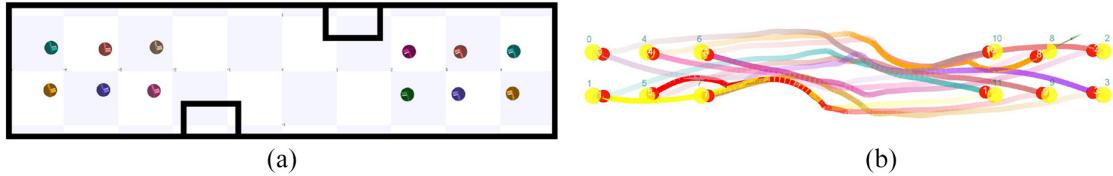


Fig. 21. Two groups of robots moving in a corridor with obstacles. (a) Corridor scenario and robots' initial positions. (b) Trajectories generated by the Hybrid-RL policy, where the initial points are in red and the goal points are in yellow. We use different colors to distinguish among trajectories of different agents and use the color transparency to indicate the timing of a trajectory sequence. Please also refer to the video for the comparison between the RL policy and the NH-ORCA policy in this scenario.

5.4. Robustness evaluation

In addition to the generalization and the quality of the policy, another main concern about DRL is whether the learned policy is stable and robust to model uncertainty and input noises. In this section, we design multiple experiments to verify the robustness of our learned policy.

5.4.1. Performance metrics. In order to quantify the robustness of our policy with respect to the workspace and robot setup, we design the following performance metrics.

- *Failure rate* is the ratio of robots that cannot reach their goals within a certain time limit.
- *Collision rate* is the ratio of robots that cannot reach their goals due to colliding with other robots or static obstacles during the navigation.
- *Stuck rate* is the ratio of robots that cannot reach their goals because they stuck in some position but without any collisions.

Note that the failure rate is equal to the collision rate plus the stuck rate. Similar to the experiments in Section 5.3, when computing each metric for a given method, we repeat the experiment 50 times and report the mean value.

5.4.2. Different agent densities. We first evaluate the performance of different policies in scenarios with varied agent densities. In particular, we design five scenarios which involve a $7 \times 7 \text{ m}^2$ region and allocate 20, 30, 40, 50, and 60 agents, respectively. The agent density for these scenarios are about 0.4, 0.6, 0.8, 1, and 1.2 robots per square meter, respectively. Note that the agent densities of these five scenarios are higher than the densities of the training and test scenarios in previous experiments. For instance, the agent density in circle scenarios in Table 3 is only 0.2 agents/ m^2 . As a result, this experiment is used to challenge the learned policy's robustness when handling the high agent density.

From the experimental results shown in Figure 22(a), we observe that the failure rates of all methods generally increase when the agent density increases. This is because in a crowded environment, each robot has limited space and

time windows to accomplish the collision-free movement and it has to deal with many neighboring agents. Thus, the robots are more likely running into collisions and getting stuck in congestion. Compared with other approaches except for the RL-Safe, the Hybrid-RL policy achieves the lowest failure rate and collision rate, as shown in Figure 22(a) and (b). Owing to its more conservative behavior, the RL-Safe policy is not sensitive to density and always maintains a very low collision rate, but such conservative behavior may easily cause it to get stuck when the density increases. For the NH-ORCA-based policies, although their conservative behavior can effectively avoid collisions, it also brings higher stuck rates, especially in high-density scenes. In addition, the collision rates and overall failure rates of all the reinforcement-learning-based policies are significantly lower than that of the NH-ORCA-based policies.

However, as we observe in Figure 22(c), the RL policy's stuck rate is slightly higher than that of the NH-ORCA policy in low-density situations. This is due to the limitation of the RL policy that we discussed in Section 4.4 when introducing the hybrid control framework: the robot controlled by the RL policy may wander around its goal when it is in close proximity to the goal. Such a navigation artifact of the RL policy takes place in the high-density scenarios, which leads to higher stuck rates. Resolving this difficulty by adopting proper sub-policies, our Hybrid-RL policy enables robots to reach goals efficiently and, thus, it significantly outperforms the RL policy and the NH-ORCA policy, as shown in Figure 22(c).

We also observe two interesting phenomena in Figure 22. First, the RL policy achieves the lowest stuck rate at a density of 0.8 robots/ m^2 . We believe that this is because the average density in the training set is close to 0.8 robots / m^2 and, thus, the RL policy slightly overfits in this case. Second, we observe that the stuck rate of the Hybrid-RL policy is a bit higher than the RL policy at a density of 1 robot/ m^2 . This is because in emergent cases, the RL policy has a higher probability to directly run into the obstacles and collide while the Hybrid-RL policy will use π_{safe} to keep safe but may get stuck. This results in a slightly lower stuck rate, but a much higher collision rate of the RL policy compared with the Hybrid-RL policy at a density 1 robot/ m^2 .

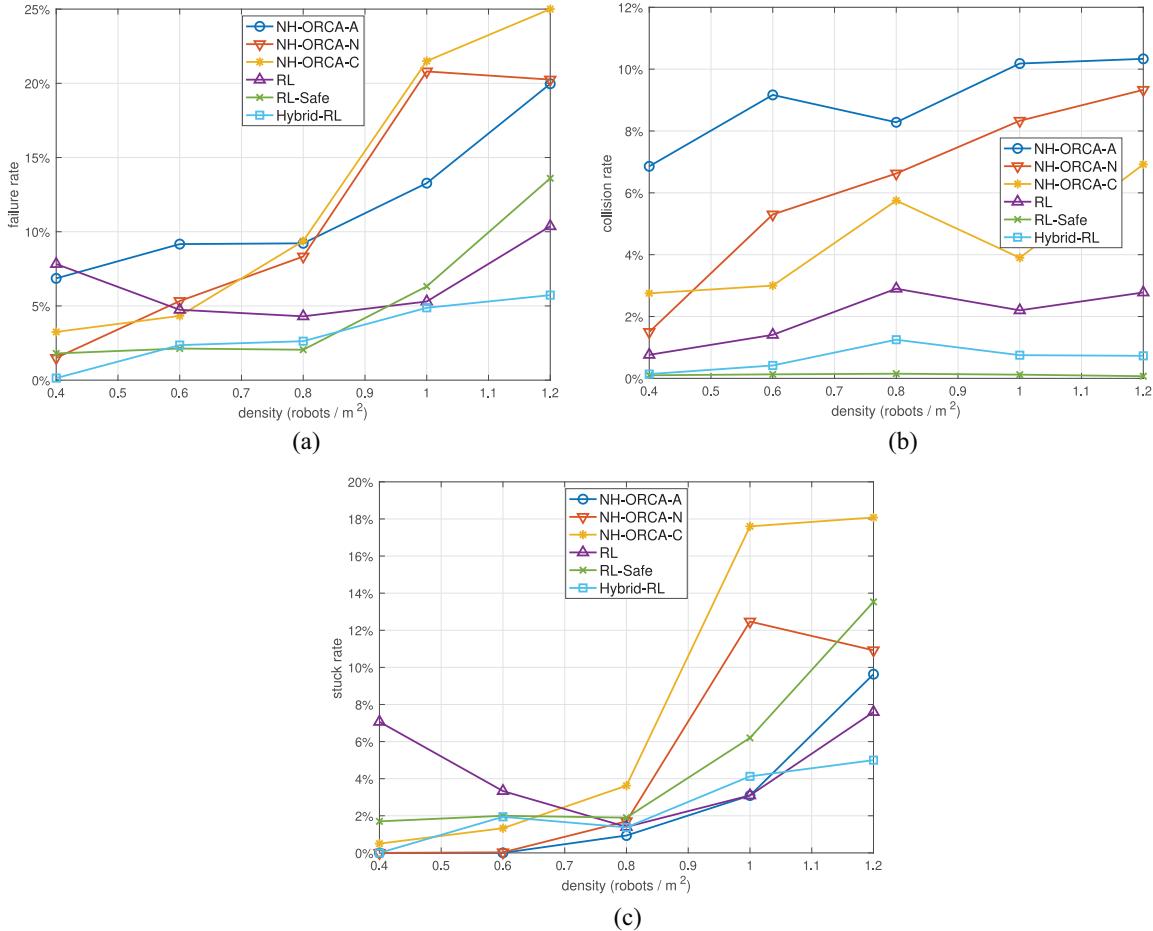


Fig. 22. Comparison of the NH-ORCA policy, the RL policy, and the Hybrid-RL policy in scenarios with different agent densities: (a) failure rate with respect to agent density; (b) collision rate with respect to agent density; (c) stuck rate with respect to agent density.

5.4.3. Different agent sizes. In our training scenarios, all robots are of the same disk shape with a radius of 0.12 m. When transferring the learned policy to unseen simulated or real-world scenarios, the robots may have different shapes and different sizes.

In Section 5.2, we directly deploy our learned policy to agents with similar sizes but different shapes to the prototype agents used in training, and we demonstrate that our method still provides satisfactory performance. In this section, we focus on evaluating the robustness of our policy when the robot's size is significantly different from the setup in training. In particular, we generate five testing scenarios, each of which is a 7×7 m² region with 10 round shape robots with the same radius, where the radius is 0.12, 0.18, 0.24, 0.30, and 0.36 m, respectively. In other words, the smallest agent has the same size as that of the prototype agent, and the largest agent has a radius three times larger than that of the prototype agent.

From the results shown in Figure 23, we observe that our Hybrid-RL policy can be directly deployed to robots that are twice as large in radius (and four times greater in area) than the prototype agent used for training, and the

robust collision-avoidance behavior can still be achieved, with a failure rate of about 3%. When the size disparity between the training and test becomes larger, the collision rate of all these policies increases. Surprisingly, the collision rate of the RL-Safe policy is significantly higher than that of the Hybrid-RL policy, which we think arises from the longer traveled time caused by the clipped maximum velocity in line 7 of Algorithm 3. The Hybrid-RL policy, on the other hand, adopts the appropriate policy according to the specific scenario and applies the RL-Safe policy only in the emergent situations, which contributes to an excellent balance between the navigation efficiency and safety.

5.4.4. Different maximum velocities. In our proposed policy, the output linear and angular velocities of the policy network are truncated to a certain range ([0, 1] m/s for the linear velocity and [0, 1] rad/s for the angular velocity in our training setup) to reduce the exploration space of reinforcement learning and, thus, to speed up the training process. However, because the robots in a variety of applications may have different dynamics and different ranges for actions, we here focus on studying the

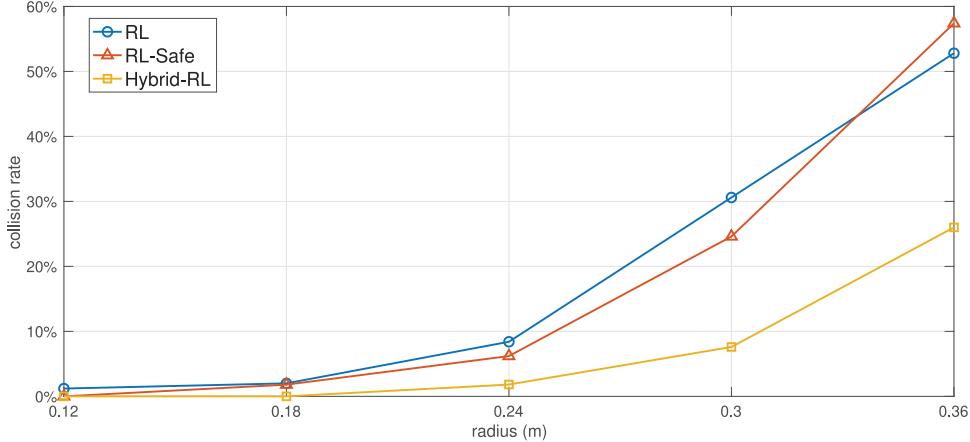


Fig. 23. Comparison of the collision rates of the RL policy, the RL-Safe policy, and the NH-ORCA policy as the robot radius increases.

robustness of the learned policy for the testing robots with different maximum linear and angular velocities.

In particular, we allocate 10 robots with the same range for the linear and angular velocities in a $7 \times 7 \text{ m}^2$ region. We create five testing scenarios, whose maximum setups for linear and angular velocities are (1 m/s, 1 rad/s), (1.5 m/s, 1.5 rad/s), (2 m/s, 2 rad/s), (2.5 m/s, 2.5 rad/s), and (3 m/s, 3 rad/s), respectively. Note that since the network output is constrained in training, there is no need to evaluate the velocities lower than (1 m/s, 1 rad/s). In this section, the RL-safe policy will not be evaluated, because its velocity is always clipped to be lower than 0.5 m/s.

The experimental result in Figure 24 shows that the RL policy is very sensitive to the changes of the network output range, i.e., as the maximum speed increases, its failure rate increases quickly to an unacceptable level, whereas our Hybrid-RL policy's collision rate is still below 10% and, thus, is still relatively safe even when the maximum speed is twice the value in training. In addition, the Hybrid-RL policy's stuck rate remains close to zero when the maximum speed is three times of the training setup, whereas the RL policy's stuck rate increases to about 20%, as shown in Figure 24(c). Thanks to the special sub-policy for the emergent situation (as discussed in Section 4.4), our Hybrid-RL policy outperforms the RL policy significantly.

5.4.5. Different control frequencies. In our training setup, we set the control frequency of executing our collision-avoidance policy to 10 Hz. Nevertheless, the actual control frequency depends on many factors, including the available computation resources of the on-board computer, the sensor's measurement frequency, and the localization system's frequency. As a result, it is difficult to maintain the control frequency at 10 Hz for long-term execution. In this section, we evaluate whether our collision-avoidance policy are able to avoid obstacles when the robot's control frequency varies.

We deploy 10 robots in a $7 \times 7 \text{ m}^2$ region, and run five experiments with the control frequencies to be 10, 5, 3, 2,

and 1 Hz, respectively. In other words, the robots' control period is 0.1, 0.2, 0.3, 0.5, and 1 s, respectively. From the experimental results in Figure 25, we observe that the collision rate of our policy increases dramatically when the control frequency is below 3 Hz. This is consistent with human experience: the navigation is difficult when blinking eyes frequently. The Hybrid-RL policy slightly outperforms the RL policy and the RL-Safe policy outperforms the Hybrid-RL policy when the control frequency varies, which may be caused by the failure of the policy switch in the hybrid control architecture.

5.4.6. Summary. After the experimental evaluations above, we demonstrate that our Hybrid-RL policy can be well generalized to unseen scenarios and is robust to the variations of the scenario configuration including densities, sizes, maximum velocities, and control frequencies of the robots. In this way, we validate that the Hybrid-RL policy is able to bridge the simulation-to-real gap and has the potential to be directly deployed on physical robots with different setups.

6. Real-world experiments

In this section, we demonstrate that our reinforcement-learning-based policy can be deployed to physical robots. In addition to the generalization capability and uncertainty modeling that we discussed in Section 5, there still exist other challenges of transferring our learned policy from simulation to the real world. First, in real situations, noises in observation are ubiquitous due to imperfect sensing. Second, the clock of each individual robot is not synchronized with each other and such an asynchronous distributed system is challenging for control. Finally, it is not guaranteed that all robots can provide consistent behavior given the same control command, because many real-world factors such as mechanics details, motor properties, and frictions cannot be accurately modeled in a simulation environment.

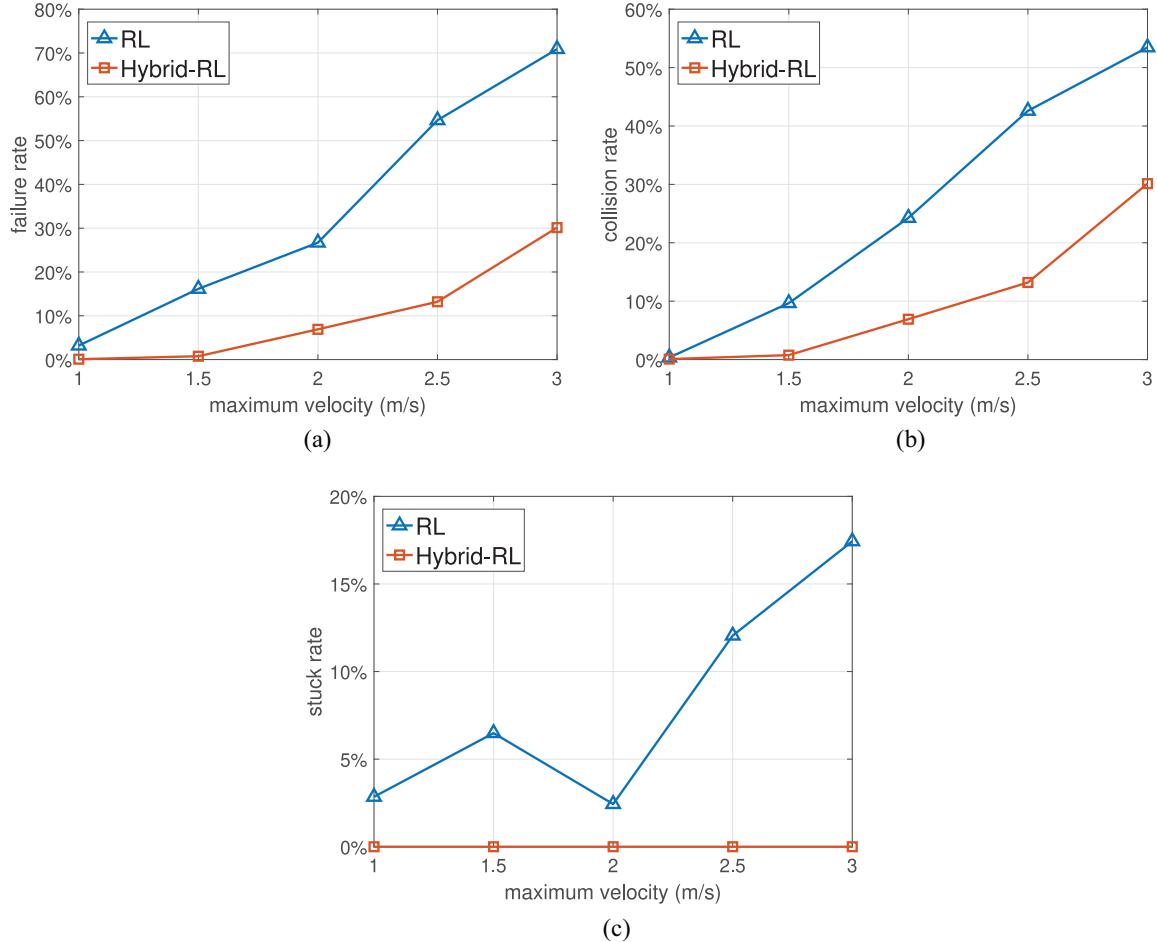


Fig. 24. Comparison of the RL policy and the Hybrid-RL policy for scenarios with different maximum velocity setup for robots: (a) failure rate with respect to maximum velocity; (b) collision rate with respect to maximum velocity; (c) stuck rate with respect to maximum velocity.

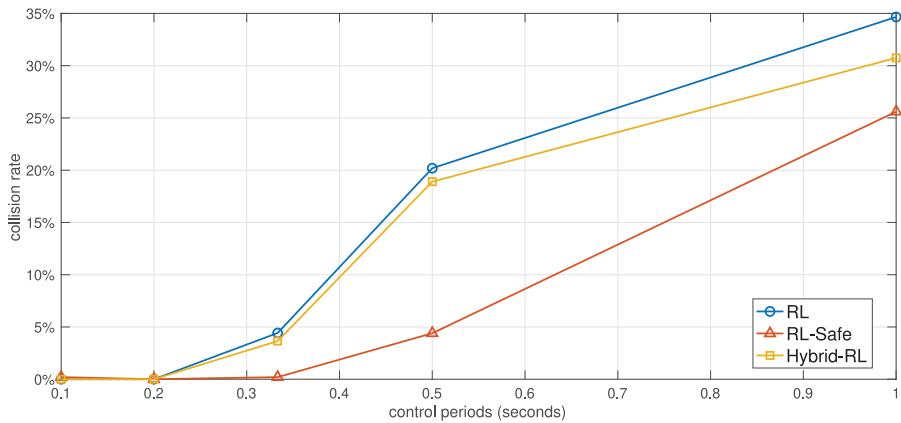


Fig. 25. Comparison of collision rates between the RL policy and the Hybrid-RL policy for different control periods.

In the following subsections, we first briefly introduce the hardware setup of our robots. Then, we present the multi-robot scenarios for evaluation. Lastly, we demonstrate the collision-avoidance behavior of the robot

controlled by our learned policy when interacting with the real human crowd.

To optimize the performance of the physical robot for collision-free behavior in the real-world experiments, we

tweak the scale parameter $\mathbf{p}_{\text{scale}}$ by taking into account the different sizes of the real-world robotic platform. On the other hand, in the simulation experiments, we use the default value in Table 2 to provide a fair and consistent comparison among different algorithms.

6.1. Hardware setup

To validate the transferability of our Hybrid-RL policy, we use a self-developed mobile robot platform for the real-world experiments. The mobile platform has a squared shape with a side length of 46 cm, as shown in Figure 26. Its shape, size, and dynamic characteristics are completely different from the prototype agents used in the training scenarios. Hence, the experiments based on such a physical platform provide a convincing evaluation of whether our method can bridge the simulation-to-real gap.

In our distributed multi-robot systems, each robot is mounted a sensor for measuring the distance to the surrounding obstacles, a localization system for measuring the distance from the robot to its goal, and an onboard computer for computing the navigation velocity in real-time. We use the Hokuyo URG-04LX-UG01 2D LiDAR as our laser scanner, the Pozyx localization system based on the UWB technology to localize our ground vehicles' 2D positions, and the Nvidia Jetson TX1 as our on-board computing device. The details of each component of our sensor kit are as follows.

- *LiDAR*: Its measurement distance is from 20 to 4,000 mm. Its precision is ± 30 mm when the measuring distance is between 60 and 1,000 mm, and the precision is 3% when the measuring distance is between 1,000 to 4,000 mm. Its angle range is $[-120^\circ, 120^\circ]$ (but we only use the measurements from -90° to 90°), and the angle resolution is 0.36° . The sensor's update frequency is 10 Hz.
- *UWB localization system*: Its measurement accuracy is around ± 150 mm in our experiment. Note that unlike other centralized collision-avoidance systems, our solution does not need the localization for collision avoidance and, thus, the ± 150 mm precision is sufficient.
- *On-board computer*: Nvidia Jetson TX1 is an embedded computing platform consisting of a Quad-core ARM A57 CPU and a 256-core Maxwell graphics processor.

6.2. Multi-robot scenarios

In this section, we design a series of real-world multi-robot navigation experiments to test the performance of our Hybrid-RL policy on the physical robots. First, we test the performance of the robots moving in a basic scenario without any obstacles. Next, we validate the policy on robotic platforms with more-complex scenarios by adding static and moving obstacles (e.g., pedestrians). In addition, we increase the number of robots in the same workspace (i.e.,

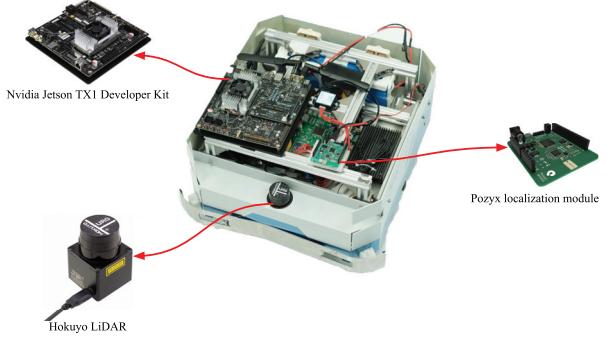


Fig. 26. The self-developed mobile platform used to verify the transferability of our reinforcement-learning-based policy deployed on physical robots.

increasing the density of robots) to further challenge the navigation algorithm's capability. Finally, we design a scenario that emulates the autonomous warehouse application to provide a comprehensive evaluation of our distributed collision-avoidance policy.

As the first step, we use a swap scenario (Figure 27(a)) and a crossing scenario (Figure 27(b)) to demonstrate that two robots can reliably avoid each other based on our Hybrid-RL policy. Although the size, shape, and dynamic characteristics of these physical robots are different from the agents trained in the simulation, our DRL-based collision-avoidance policy still performs well in these two scenarios.

Next, we add static obstacles in the swap benchmark and test whether the two robots can still successfully navigate in this complex scenario in a map-less manner. From the trajectories in Figure 27(c), we observe that both robots smoothly avoid static obstacles that are randomly placed in the environment. Then we allow two pedestrians to interfere with the robots in the same environment. Taking the trajectories shown in Figure 27(d) for example, the two robots adaptively slow down and wait for the pedestrians and then start again towards the goal after the pedestrian passes. It is important to note that there is no pedestrian data introduced in the training process, and the pedestrians' shape and dynamic characteristics are quite different from other robots and static obstacles. Moreover, we do not leverage any pedestrian tracking or prediction methods in the robot's navigation pipeline. As illustrated in the qualitative results, our learned policy demonstrates an excellent generalization capability in collision avoidance not only for the static obstacles but also for the dynamic obstacles. Hence, based on our learned policy without fine-tuning, robots can easily adapt to new and more-challenging tasks.

We further add another two robots in the two-robot benchmarks and design a four-robot benchmark as shown in Figure 28. From the resulting trajectories, we observe that all robots reach their goals successfully without any collisions. However, we also note that the quality of the robots' trajectories is lower than that in the simpler benchmarks demonstrated in Figure 27. This is mainly due to the

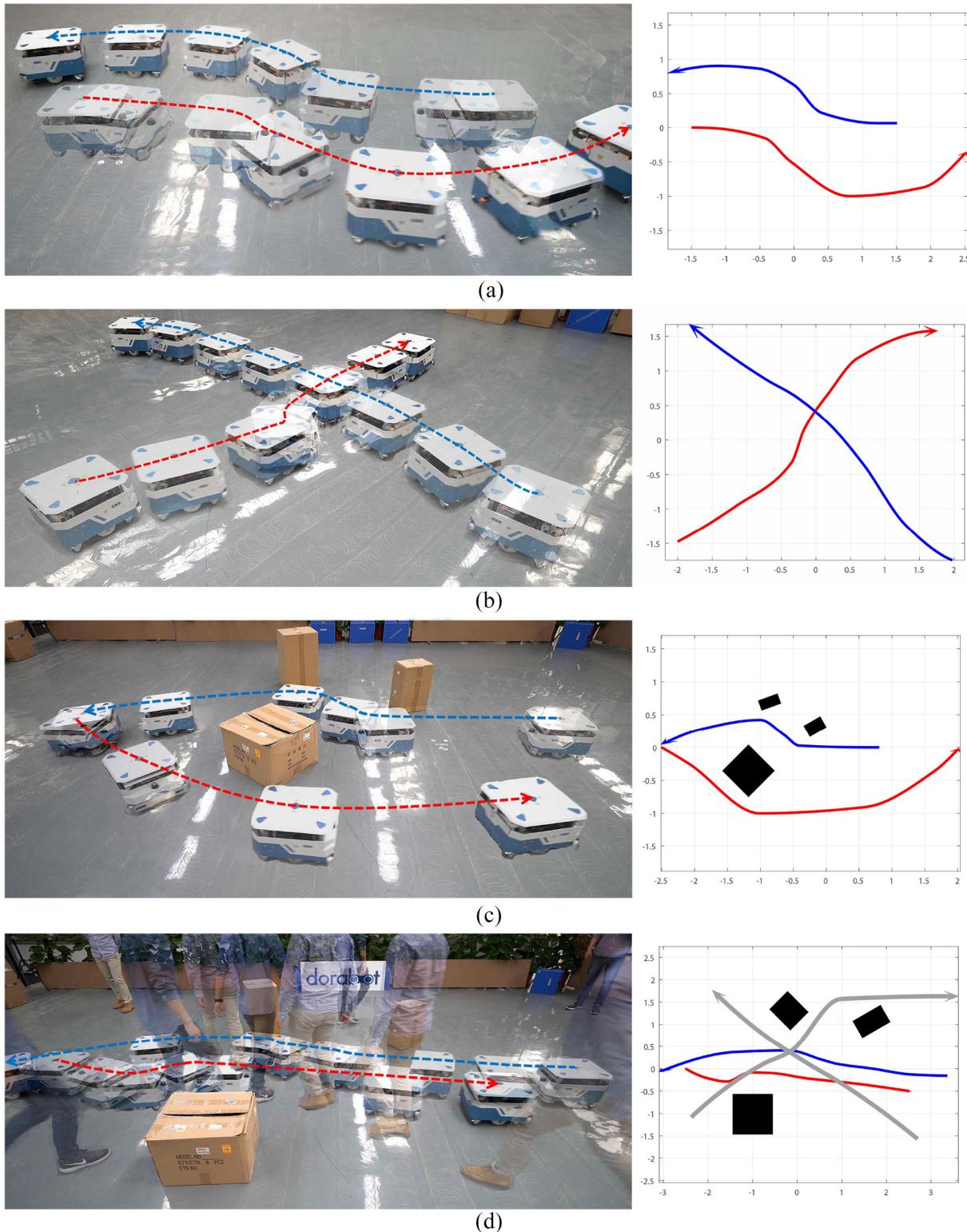


Fig. 27. (a) Swap scenario: two robots moving in the opposite directions swap their positions. (b) Crossing scenario: the trajectories of two robots intersect. (c) Static obstacles are placed in a swap scenario. (d) Both static obstacles and moving obstacles (i.e., pedestrians) are placed in the swap scenario. The figures on the left show the trajectories of the physical robots in the real scenario, and the figures on the right show the trajectories of the physical robots in the 2D coordinate system, which correspond to the figures on the left. In the figures on the right, the black boxes refer to the static obstacles, the gray trajectories refer to the pedestrians' paths, and other colored trajectories are the robots' paths. Please refer to the video for more details about the physical robots' behavior in these scenarios.

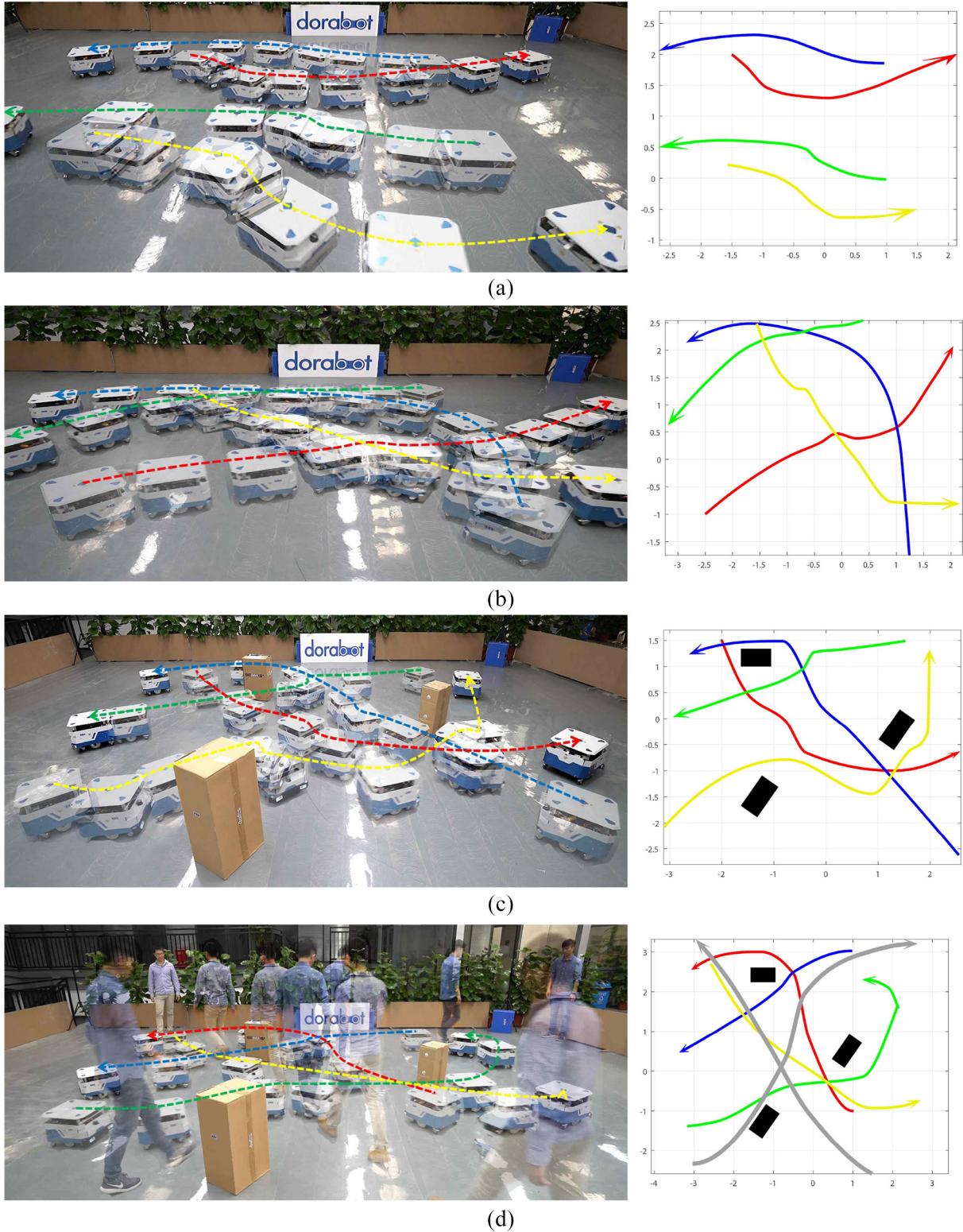


Fig. 28. (a) Swap scenario: two groups of robots moving in the opposite directions swap their positions. (b) Crossing scenario: the path of four robots will intersect. (c) Static obstacles are placed in a crossing scenario. (d) Both static obstacles and moving obstacles (i.e., pedestrians) are placed in the crossing scenario. The left column shows the trajectories of the physical robots in the real scenario, and the right column shows the trajectories of the physical robots in a 2D coordinate system. In the right column, the black boxes are the static obstacles, the gray trajectories are the pedestrians' paths, and other colored trajectories are the robots' paths. Please also refer to the video for more details about the robots' behavior in this scenario.

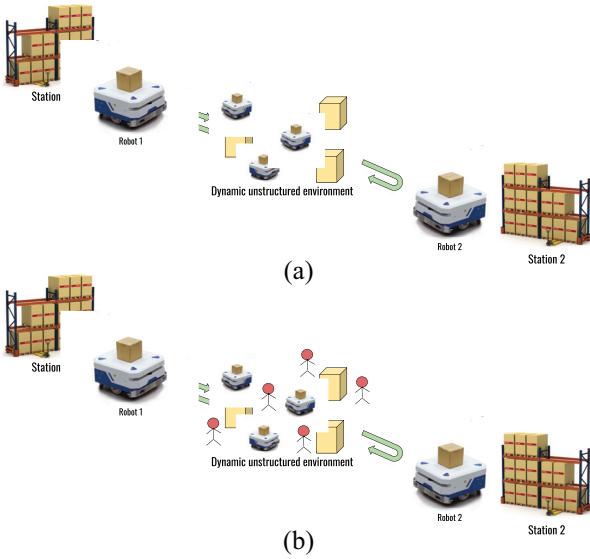


Fig. 29. Two types of autonomous warehouse scenarios for evaluating the performance of our multi-robot collision-avoidance policy: (a) without human co-workers; (b) with human co-workers.

positioning error of the UWB localization system and the disparity in robot hardware, both of which are amplified as the number of robots increases.

Finally, we design two scenarios that emulate the autonomous warehouse application, where multiple robots work together to achieve efficient and flexible transportation of commodities in an unstructured warehouse environment for a long time. In the first scenario as illustrated in Figure 29(a), there are two transportation stations in the warehouse and two robots are responsible for transporting objects between these two stations. Next, we add random moving pedestrians as the co-workers in the warehouse and build the second scenario as

illustrated in Figures 29(b) and 30. The second scenario is more complex, because the robots and the humans share the workspace, especially that the workers may block the robot, which challenges the safety of our multi-robot collision-avoidance policy. As illustrated from the results and the attached video, our policy enables multiple robots to accomplish efficient and safe transportation in these two scenarios. In our test, the entire system can safely run for more than 30 minutes without any faults. Please refer to our attached videos for more details.

6.3. Robotic collision avoidance in dense human crowds

There are many previous works about navigating a robot toward its goals in a dense human crowd without any collisions (Ess et al., 2010). The typical pipeline is first predicting pedestrian movements (Alahi et al., 2016; Gupta et al., 2018; Kim et al., 2015; Yi et al., 2016) and then using reactive control (similar to Flacco et al. (2012)) or local planning for collision avoidance. As it is difficult to predict the human motion accurately, such a pipeline is not reliable for practical applications.

In this section, we use our reinforcement-learning-based collision-avoidance policy to achieve high-quality navigation in a dense human crowd. Our method is applied for navigation tasks involving multiple cooperative robots in dynamic unstructured environments, but it also provides a high-quality collision-avoidance policy for a single robot task in a dynamic unstructured environment with non-cooperative agents (e.g., pedestrians).

Our experimental setup is as follows. We attach one UWB localization tag to a person so that the mobile robot will follow this person according to the UWB signal. Hence, the person controls the robot's desired motion direction by providing a time-varying goal. For evaluation, this

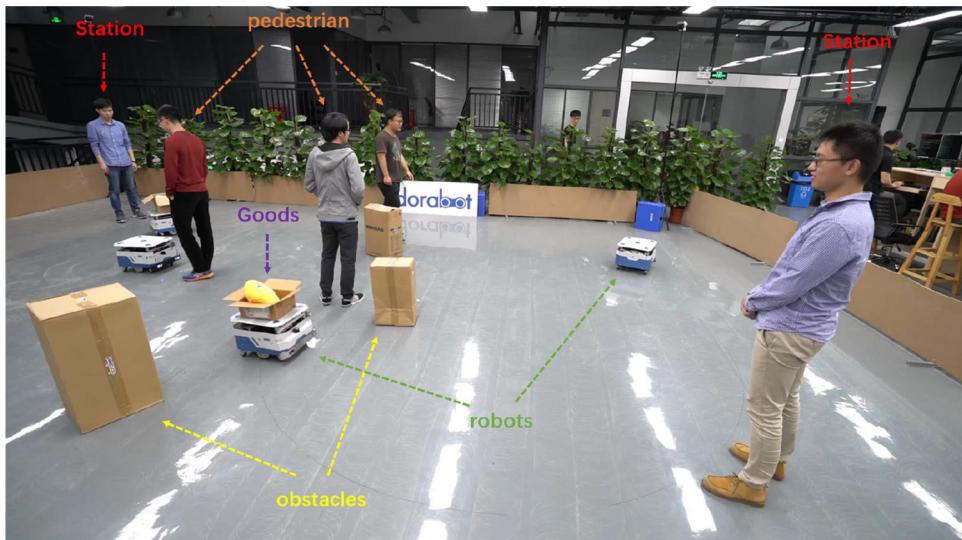


Fig. 30. The experimental scenario for the autonomous warehouse benchmark with human co-workers. Please refer to the video for more details about the robots' behavior in this scenario.

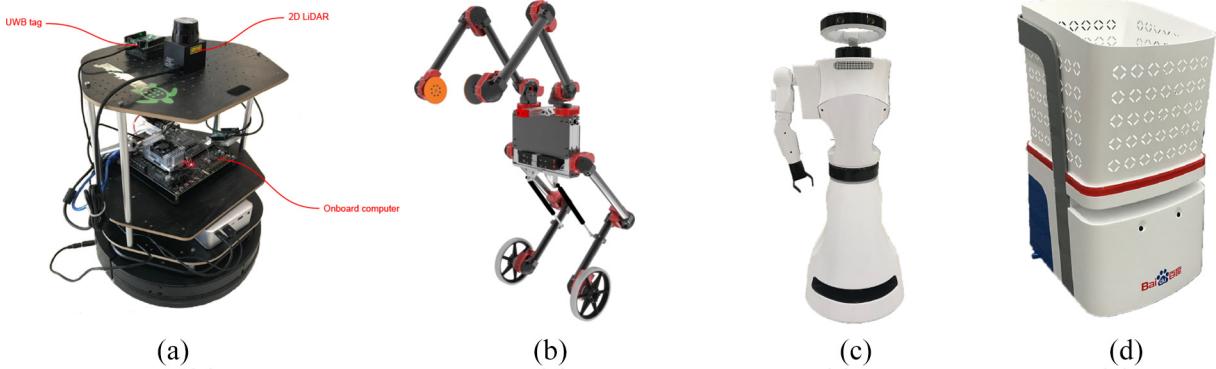


Fig. 31. Four mobile platforms are tested in our experiments for the collision avoidance in the dense human crowds: (a) the Turtlebot, (b) the Igor robot from Hebi robotics, (c) the Baidu bear robot, and (d) the Baidu shopping cart from Baidu, Inc.

person leads the robot into the dense human stream, which provides a severe challenge to the navigation policy. We test the performance of our method in a wide variety of scenarios using different types of robots.

For the testing mobile robot platform, we use four platforms, the Turtlebot platform, the Igor robot from Hebi robotics, the Baidu Bear robot, and the Baidu shopping cart from Baidu, Inc. These four different robots have their own characteristics. The Turtlebot (Figure 31(a)) is larger than the 12-cm-radius prototype agent used in the training process; the Igor robot (Figure 31(b)) is a self-balancing wheeled R/C robot; the Baidu Bear robot is a human-like service robot (Figure 31(c)); and the Baidu shopping cart uses differential wheels as the mobile base, whose weight is quite different to the Turtlebot. The maximum linear velocities for these robots are 0.7, 0.5, 0.4, and 0.4 m/s, respectively.

Our testing scenarios cover the real-world environments that a mobile robot may encounter, including the canteen, farmer's market, outdoor street, and indoor conference room, as shown in Figure 32. The Turtlebot running our Hybrid-RL policy safely avoids pedestrians and static obstacles with different shapes, even in some extremely challenging situations, e.g., when curious people suddenly block the robot from time to time. However, the Turtlebot's behavior is not perfect. In particular, the robot tends to accelerate and decelerate abruptly within dense obstacles. Such behavior is reasonable according to the objective function of the policy optimization algorithm (in (3)), because the robot will move to its goal as fast as possible in this way and its movement is more efficient. A refined objective function for optimizing a smooth motion in dense crowds would be left as our future work. In addition, we also observe that the human–robot interaction in the real-world experiments differs much from the multi-robot collision-avoidance problem. In particular, pedestrians may actively block or over-conservatively avoid the robot because a robot in a crowd may abruptly change its original behaviors during the interaction. How to make the robot navigation more socially compliant is, thus,

another promising direction. Please refer to Appendix C for more experimental results.

6.4. Summary

We demonstrate that our multi-robot collision-avoidance policy can be well deployed to different types of real robots, though their shape, size, and dynamic characteristics are quite different from the prototype agents used in the training simulation scenarios. We validate the learned policy on various real-world scenarios and show that our Hybrid-RL policy can run robustly for a long time without collisions and the robots' movement easily adapt with the pedestrians in the warehouse scenario. We also verify the possibility of using the Hybrid-RL policy to enable a single robot to navigate through a dense human crowd efficiently and safely.

7. Conclusion

In this article, we have presented a multi-scenario multi-stage training framework to optimize a fully decentralized sensor-level collision-avoidance policy with a robust policy gradient algorithm. We have evaluated the performance of our method using a series of comprehensive experiments and demonstrated that the learned policy significantly outperforms the state-of-the-art approaches in terms of the success rate, the navigation efficiency, and the generalization capability. Our learned policy can also be used to navigate a single robot through a dense pedestrian crowd, and have illustrated the excellent performance in a wide variety of scenarios and on different robot platforms.

Our work serves as the first step towards reducing the navigation performance gap between the centralized and decentralized methods, though we are fully aware of that, as a local collision-avoidance method, our approach cannot completely replace a global path planner when scheduling many robots to navigate through complex environments with dense obstacles or with dense pedestrians. Our future work would be how to incorporate our approach with classical mapping methods (e.g., SLAM) and global path planners



Fig. 32. Map-less navigation in complex and highly dynamic environments using different mobile platforms: (a) canteen; (b) farmer's market; (c) outdoor street; (d) indoor conference venue. Please refer to the video for more details about the robots' behavior in these challenging scenarios.

(e.g., RRT and A*) to achieve satisfactory performance for planning a safe trajectory through a dynamic environment.

Acknowledgements

The authors would like to thank Hao Zhang from Dorobot, Inc. and Ruigang Yang from Baidu, Inc. for their support to us when preparing the physical robot experiments. The authors would like to thank Dinesh Manocha from the University of Maryland for his constructive discussion about the method and the experiments.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work was partially supported by HKSAR General Research Fund (GRF; HKU grant numbers 11202119 and 11207818) and the NSFC/RGC Joint Research Scheme (grant number HKU103/16-NSFC61631166002).

Notes

- See <http://rtv.github.io/Stage/>

ORCID iDs

Pinxin Long <https://orcid.org/0000-0001-8440-3218>
Jia Pan <https://orcid.org/0000-0001-9003-2054>

References

- Adouane L (2009) Hybrid and safe control architecture for mobile robot navigation. In: *Proceedings of the Conference on Autonomous Robot Systems and Competitions*.
- Alahi A, Goel K, Ramanathan V, Robicquet A, Fei-Fei L and Savarese S (2016) Social LSTM: Human trajectory prediction in crowded spaces. In: *IEEE Conference on Computer Vision and Pattern Recognition*.
- Alonso-Mora J, Baker S and Rus D (2017) Multi-robot formation control and object transport in dynamic environments via constrained optimization. *The International Journal of Robotics Research* 36(9): 1000–1021.
- Alonso-Mora J, Beardsley P and Siegwart R (2018) Cooperative collision avoidance for nonholonomic robots. *IEEE Transactions on Robotics* 34(2): 404–420.
- Alonso-Mora J, Breitenmoser A, Rufli M, Beardsley P and Siegwart R (2013) Optimal reciprocal collision avoidance for multiple non-holonomic robots. In: *Distributed Autonomous Robotic Systems*. Berlin: Springer, pp. 203–216.
- Alur R, Esposito J, Kim M, Kumar V and Lee I (1999) Formal modeling and analysis of hybrid systems: A case study in multi-robot coordination. In: *Proceedings of International Symposium on Formal Methods*. Berlin: Springer, pp. 212–232.
- Amodei D, Anubhai R, Battenberg E, et al. (2016) Deep speech 2: End-to-end speech recognition in English and Mandarin. In: *International Conference on International Conference on Machine Learning*, pp. 173–182.
- Balch T and Arkin RC (1998) Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation* 14(6): 926–939.
- Bareiss D and van den Berg J (2015) Generalized reciprocal collision avoidance. *The International Journal of Robotics Research* 34(12): 1501–1514.
- Barreto A, Munos R, Schaul T and Silver D (2017) Successor features for transfer in reinforcement learning. In: *Proceedings of Neural Information Processing Systems*.

- Bengio Y, Louradour J, Collobert R and Weston J (2009) Curriculum learning. In: *Proceedings of International Conference on Machine Learning*, pp. 41–48.
- Chen J and Sun D (2011) Resource constrained multirobot task allocation based on leader–follower coalition methodology. *The International Journal of Robotics Research* 30(12): 1423–1434.
- Chen J, Sun D, Yang J and Chen H (2010) Leader–follower formation control of multiple non-holonomic mobile robots incorporating a receding-horizon scheme. *The International Journal of Robotics Research* 29(6): 727–747.
- Chen YF, Everett M, Liu M and How JP (2017a) Socially aware motion planning with deep reinforcement learning. *arXiv preprint arXiv:1703.08862*.
- Chen YF, Liu M, Everett M and How JP (2017b) Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 285–292.
- Claes D, Hennes D, Tuyls K and Meeussen W (2012) Collision avoidance under bounded localization uncertainty. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 1192–1198.
- De La Cruz C and Carelli R (2008) Dynamic model based formation control and obstacle avoidance of multi-robot systems. *Robotica* 26(3): 345–356.
- Egerstedt M and Hu X (2002) A hybrid control approach to action coordination for mobile robots. *Automatica* 38(1): 125–130.
- Ess A, Schindler K, Leibe B and Van Gool L (2010) Object detection and tracking for autonomous navigation in dynamic environments. *The International Journal of Robotics Research* 29(14): 1707–1725.
- Everett M, Chen YF and How JP (2018) Motion planning among dynamic, decision-making agents with deep reinforcement learning. *arXiv preprint arXiv:1805.01956*.
- Flacco F, Kröger T, Luca AD and Khatib O (2012) A depth space approach to human–robot collision avoidance. In: *IEEE International Conference on Robotics and Automation*. pp. 338–345.
- Fox D, Burgard W and Thrun S (1997) The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine* 4(1): 23–33.
- Frans K, Ho J, Chen X, Abbeel P and Schulman J (2017) Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*.
- Gillula JH, Hoffmann GM, Huang H, Vitus MP and Tomlin CJ (2011) Applications of hybrid reachability analysis to robotic aerial vehicles. *The International Journal of Robotics Research* 30(3): 335–354.
- Godoy J, Karamouzas I, Guy SJ and Gini M (2016a) Implicit coordination in crowded multi-agent navigation. In: *Proceedings of AAAI Conference on Artificial Intelligence*.
- Godoy J, Karamouzas I, Guy SJ and Gini ML (2016b) Moving in a crowd: Safe and efficient navigation among heterogeneous agents. In: *IJCAI*, pp. 294–300.
- Graves A, Mohamed AR and Hinton G (2013) Speech recognition with deep recurrent neural networks. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649.
- Gupta A, Johnson J, Fei-Fei L, Savarese S and Alahi A (2018) Social GAN: Socially acceptable trajectories with generative adversarial networks. In: *IEEE Conference on Computer Vision and Pattern Recognition*.
- Haddad H, Khatib M, Lacroix S and Chatila R (1998) Reactive navigation in outdoor environments using potential fields. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 2, pp. 1232–1237.
- He K, Zhang X, Ren S and Sun J (2016) Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition*.
- Heess N, Sriram S, Lemmon J, et al. (2017) Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*.
- Helbing D and Molnar P (1995) Social force model for pedestrian dynamics. *Physical Review E* 51(5): 4282.
- Hennes D, Claes D, Meeussen W and Tuyls K (2012) Multi-robot collision avoidance with localization uncertainty. In: *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pp. 147–154.
- Horgan D, Quan J, Budden D, et al. (2018) Distributed prioritized experience replay. *arXiv preprint arXiv:1803.00933*.
- Hu S and Sun D (2011) Automatic transportation of biological cells with a robot-tweezer manipulation system. *The International Journal of Robotics Research* 30(14): 1681–1694.
- Kahn G, Villaflor A, Pong V, Abbeel P and Levine S (2017) Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*.
- Keviczky T, Borrelli F and Balas GJ (2004) A study on decentralized receding horizon control for decoupled systems. In: *Proceedings of American Control Conference*, Vol. 6. IEEE, pp. 4921–4926.
- Kim S, Guy SJ, Liu W, Wilkie D, Lau RWH, Lin MC and Manocha D (2015) BRVO: Predicting pedestrian trajectories using velocity-space reasoning. *The International Journal of Robotics Research* 34(2): 201–217.
- Kingma D and Ba J (2014) Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Krizhevsky A, Sutskever I and Hinton GE (2012) Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.
- Lenz I, Knepper R and Saxena A (2015) DeepMPC: Learning deep latent features for model predictive control. In: *Proceedings of the Robotics: Science and System*.
- Levine S, Finn C, Darrell T and Abbeel P (2016) End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(39): 1–40.
- Lillicrap TP, Hunt JJ, Pritzel A, et al. (2015) Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.
- Long P, Fan T, Liao X, Liu W, Zhang H and Pan J (2017a) Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In: *Proceedings of the IEEE International Conference on Robotics and Automation*.
- Long P, Liu W and Pan J (2017b) Deep-learned collision avoidance policy for distributed multiagent navigation. *Robotics and Automation Letters* 2(2): 656–663.
- Luna R and Bekris KE (2011) Efficient and complete centralized multi-robot path planning. In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3268–3275.
- Michael N, Fink J and Kumar V (2011) Cooperative manipulation and transportation with aerial robots. *Autonomous Robots* 30(1): 73–86.
- Mnih V, Kavukcuoglu K, Silver D, et al. (2015) Human-level control through deep reinforcement learning. *Nature* 518(7540): 529.

- Muller U, Ben J, Cosatto E, Flepp B and Cun YL (2006) Off-road obstacle avoidance through end-to-end learning. In: *Proceedings of Advances in Neural Information Processing Systems*, pp. 739–746.
- Nair V and Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. In: *Proceedings of International Conference on Machine Learning*, pp. 807–814.
- Ondruska P, Dequaire J, Zeng Wang D and Posner I (2016) End-to-end tracking and semantic segmentation using recurrent neural networks. In: *Proceedings of Robotics: Science and Systems, Workshop on Limits and Potentials of Deep Learning in Robotics*.
- Ondruska P and Posner I (2016) Deep tracking: Seeing beyond seeing using recurrent neural networks. In: *Proceedings of AAAI Conference on Artificial Intelligence*. Phoenix, AZ.
- Peng XB, Andrychowicz M, Zaremba W and Abbeel P (2017) Sim-to-real transfer of robotic control with dynamics randomization. *arXiv preprint arXiv:1710.06537*.
- Pfeiffer M, Schaeuble M, Nieto J, Siegwart R and Cadena C (2017) From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1527–1533.
- Quotrup MM, Bak T and Zamanabadi R (2004) Multi-robot planning: A timed automata approach. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 5, pp. 4417–4422.
- Ross S, Melik-Barkhudarov N, Shankar KS, et al. Wendel A, Dey D, Bagnell JA and Hebert M (2013) Learning monocular reactive UAV control in cluttered natural environments. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1765–1772.
- Rusu AA, Vecerik M, Rothörl T, Heess N, Pascanu R and Hadsell R (2016) Sim-to-real robot learning from pixels with progressive nets. In: *Conference on Robot Learning*.
- Sadeghi F and Levine S (2017) CAD2RL: Real single-image flight without a single real image. In: *Robotics: Science and Systems*.
- Schaul T, Quan J, Antonoglou I and Silver D (2015) Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.
- Schulman J, Levine S, Abbeel P, Jordan M and Moritz P (2015a) Trust region policy optimization. In: *Proceedings of the International Conference on Machine Learning*, pp. 1889–1897.
- Schulman J, Moritz P, Levine S, Jordan M and Abbeel P (2015b) High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
- Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O (2017a) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Schulman J, Wolski F, Dhariwal P, Radford A and Klimov O (2017b) Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Schwartz JT and Sharir M (1983) On the piano movers' problem: III. Coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research* 2(3): 46–75.
- Sergeant J, Sünderhauf N, Milford M and Upcroft B (2015) Multimodal deep autoencoders for control of a mobile robot. In: *Proceedings of Australasian Conference on Robotics and Automation*.
- Sharon G, Stern R, Felner A and Sturtevant NR (2015) Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219: 40–66.
- Shucker B, Murphrey T and Bennett JK (2007) Switching rules for decentralized control with simple control laws. In: *American Control Conference, 2007 (ACC'07)*. IEEE, pp. 1485–1492.
- Snape J, Van Den Berg J, Guy SJ and Manocha D (2010) Smooth and collision-free navigation for multiple robots under differential-drive constraints. In: *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4584–4589.
- Snape J, van den Berg J, Guy SJ and Manocha D (2011) The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics* 27(4): 696–706.
- Stone P and Veloso M (1999) Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence* 110(2): 241–273.
- Sun D, Wang C, Shang W and Feng G (2009) A synchronization approach to trajectory tracking of multiple mobile robots while maintaining time-varying formations. *IEEE Transactions on Robotics* 25(5): 1074–1086.
- Tai L, Paolo G and Liu M (2017) Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*.
- Tang S, Thomas J and Kumar V (2018) Hold Or take Optimal Plan (HOOP): A quadratic programming approach to multi-robot trajectory generation. *The International Journal of Robotics Research* 37(9): 1062–1084.
- Tobin J, Fong R, Ray A, Schneider J, Zaremba W and Abbeel P (2017) Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 23–30.
- Turpin M, Michael N and Kumar V (2014) CAPT: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research* 33(1): 98–112.
- van den Berg J, Guy SJ, Lin M and Manocha D (2011a) Reciprocal n-body collision avoidance. In: *International Symposium on Robotics Research*. Berlin: Springer, pp. 3–19.
- van den Berg J, Guy SJ, Lin M and Manocha D (2011b) Reciprocal n-body collision avoidance. In: *Robotics Research*. Springer, pp. 3–19.
- van den Berg J, Lin M and Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1928–1935.
- Xie S, Girshick R, Dollar P, Tu Z and He K (2017) Aggregated residual transformations for deep neural networks. In: *International Conference on Computer Vision and Pattern Recognition*, pp. 1492–1500.
- Yi S, Li H and Wang X (2016) Pedestrian behavior understanding and prediction with deep neural networks. In: *European Conference on Computer Vision*, pp. 263–279.
- Yu J and LaValle SM (2016) Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics* 32(5): 1163–1177.
- Zhang J, Springenberg JT, Boedecker J and Burgard W (2016a) Deep reinforcement learning with successor features for navigation across similar environments. *arXiv preprint arXiv:1612.05533*.
- Zhang T, Kahn G, Levine S and Abbeel P (2016b) Learning deep control policies for autonomous aerial vehicles with MPC-guided policy search. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 528–535.

Zhou D, Wang Z, Bandyopadhyay S and Schwager M (2017) Fast, on-line collision avoidance for dynamic vehicles using buffered Voronoi cells. *IEEE Robotics and Automation Letters* 2(2): 1047–1054.

Zhu Y, Mottaghi R, Kolve E, et al. (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3357–3364.

Appendix A. Multi-robot collision avoidance policy trained by off-policy reinforcement-learning algorithm

Distributed prioritized experience replay (Ape-X) is a state-of-the-art off-policy distributed reinforcement-learning algorithm which successfully extends the prioritized experience replay and deep determined policy gradient (DDPG) to the distributed setting. In practice, this algorithm uses multiple actors with the same network parameters to generate the training data stored in the prioritized experience replay buffer. Then, it samples a prioritized batch of the training data from the replay buffer to update the network parameters according to the learning rule of DDPG. Therefore, Ape-X is an ideal method for solving the formulated multi-robot collision-avoidance problem. The implementation details are summarized in Algorithm 4.

Appendix B. NH-ORCA

We used the open-source NH-ORCA implementation from Hennes et al. (2012) and Claes et al. (2012) for the evaluation. The details of this implementation can be found in

Algorithm 4. Ape-X for a multi-robot system

```

1: Initialize policy network  $\pi_{\theta_t}$ , Q-value function  $Q_{\phi_t}(s_t, a_t)$ 
   and prioritized replay buffer  $\mathbb{B}$ ; set up a multi-robot
   simulator  $\mathbb{E}$ .
2: for iteration = 1, 2, ..., do
3:   // Collect data
4:   for timestep  $t = 1$  to  $T$  do
5:     for robot  $i = 1$  to  $N$  do
6:       // Get an action by  $\pi_{\theta_t}$ 
7:        $\mathbf{a}_i^t = \pi_{\theta_t}(\mathbf{o}_i^t)$ 
8:       // Execute action in the environment
9:        $(r_i^t, \mathbf{o}_i^{t+1}) \leftarrow \mathbb{E}.step(\mathbf{a}_i^t)$ 
10:      // Add data to the replay buffer
11:       $\mathbb{B}.add(\mathbf{o}_i^t, \mathbf{a}_i^t, r_i^t, \mathbf{o}_i^{t+1})$ 
12:    end for
13:  end for
14:  // Update network
15:  if  $\mathbb{B}.size() > B$  then
16:    // Sample a prioritized training data item
17:     $\tau \leftarrow \mathbb{B}.sample()$ 
18:    // Apply DDPG learning rule to compute the loss
19:     $l_t = \text{ComputeLoss}(\tau; \theta_t, \phi_t)$ 
20:    // Update network parameters
21:     $\theta_{t+1}, \phi_{t+1} = \text{UpdateParameters}(l_t; \theta_t, \phi_t)$ 
22:  end if
23: end for
```

http://wiki.ros.org/multi_robot_collision_avoidance. Since the agents executing the NH-ORCA policy cannot make decisions based on the raw sensor measurements, in the simulation we share the ground truth positions and velocities between robots in the simulation, while such information is not available to the learning-based approaches.

The hyper-parameters used in the comparison experiments are defined in Table 4.

Appendix C. Demonstrations on different robotic platforms

We have successfully deployed our hybrid RL policy on different robotic platforms and according to our experiments

Table 4. Hyper-parameters of the NH-ORCA algorithm used in our comparison experiments. For the details of definition of each parameter, please refer to Claes et al. (2012).

Parameter	Value
Time horizon	10.0
Time to holonomic	0.4
Minimum tracking error	0.02
Maximum tracking error	0.10
Epsilon	0.1
VO type	HRVO
Use ORCA	True
Use clearpath	True

tal results, our method can work in a wide variety of highly dynamic scenarios as shown in Figures 33–35.



Fig. 33. Turtlebot works in highly dynamic unstructured scenarios. Please refer to the video for more details about the robots' behavior in these scenarios.



Fig. 34. Igor reacts quickly to the bottles, human legs, and bags that suddenly appear in its field of view. Please also refer to the video for more details about the robots' behavior in these scenarios.



Fig. 35. Our collision-avoidance policy is deployed on Baidu Bear robot and the Baidu shopping cart.