Crowd-Aware Robot Navigation for Pedestrians with Multiple Collision Avoidance Strategies via Map-based Deep Reinforcement Learning

Shunyi Yao[1,*], Guangda Chen[1,*], Quecheng Qiu[2], Jun Ma[2], Xiaoping Chen[1], and Jianmin Ji[1,†]

Abstract— It is challenging for a mobile robot to navigate through human crowds. Existing approaches usually assume that pedestrians follow a predefined collision avoidance strategy, like social force model (SFM) or optimal reciprocal collision avoidance (ORCA). However, their performances commonly need to be further improved for practical applications, where pedestrians follow multiple different collision avoidance strategies. In this paper, we propose a map-based deep reinforcement learning approach for crowd-aware robot navigation with various pedestrians. We use the sensor map to represent the environmental information around the robot, including its shape and observable appearances of obstacles. We also introduce the pedestrian map that specifies the movements of pedestrians around the robot. By applying both maps as inputs of the neural network, we show that a navigation policy can be trained to better interact with pedestrians following different collision avoidance strategies. We evaluate our approach under multiple scenarios both in the simulator and on an actual robot. The results show that our approach allows the robot to successfully interact with various pedestrians and outperforms compared methods in terms of the success rate.

I. INTRODUCTION

Many practical applications of mobile robots, like domestic service robots [1] and office delivery robots [2], require them to navigate through human crowds. It is a tough challenge, as pedestrians around the robot can not be simply considered as regular static or dynamic obstacles and the robot needs to interact with a crowd of people in a socially compliant manner. Existing crowd navigation approaches commonly make assumptions on the movements [3] or the collision avoidance strategies [4], [5] of pedestrians. However, the performance of these approaches is usually limited when pedestrians behave differently.

On the other hand, deep reinforcement learning (DRL) approaches have been applied for robot navigation with promising results [6]–[8]. Recently, DRL approaches are also applied for crowd navigation. Everett et al. [9] and Chen et al. [10] assume that pedestrians follow a certain collision avoidance strategy, i.e., optimal reciprocal collision

avoidance (ORCA) [5]. Moreover, both approaches require perfect sensing, as the inputs of their networks include exact positions and movements of surrounding obstacles. Guldenring et al. [11] assume pedestrians following social force model (SFM) [4] and use the sensor data directly. Notice that, above approaches do not distinguish the information between regular obstacles and pedestrians, which increases the difficulty for the trained model to handle pedestrians with multiple collision avoidance strategies.

In this paper, we propose a map-based DRL approach for crowd navigation with pedestrians following multiple different collision avoidance strategies. We find out that it is more effective for the trained model to handle various pedestrians by distinguishing the information between regular obstacles and pedestrians. In particular, following our previous work [12] for robot navigation among regular obstacles, we use the sensor map to represent the environmental information around the robot, including its shape and observable appearances of obstacles, which can be directly generated from robot's sensor information. We also introduce the pedestrian map that specifies the movements of pedestrians around the robot in the local grid map. Then we apply the proximal policy optimization (PPO) algorithm [13] to train a convolutional neural network that directly maps both maps and the robot's target pose into low-level robot control commands. Note that, the information of pedestrians is specified by the pedestrian map, which can ease the training of the network to interact with various pedestrians. We first train the neural network in a specified simulator with static obstacles and pedestrians following two strategies, i.e., SFM and ORCA, and then we deploy the trained model to an actual robot for its navigation.

We evaluate our approach under multiple scenarios both in the simulator and on an actual robot. Experimental results show that, by applying both the sensor map and the pedestrian map, the network allows the robot not only to avoid regular obstacles but also pedestrians with different collision avoidance strategies. We compare our approach to a dynamic obstacle avoidance method with perfect sensing and a DRL method only using the sensor map. The results show that our approach outperforms these methods in terms of the success rate. We deploy the trained model to a robot and evaluate its performance in real-world environments, like corridors and halls. The demonstration video is also available. Our main contributions are summarized as follows:

- We propose a DRL-based crowd navigation method that allows pedestrians to follow multiple different collision avoidance strategies. The experimental results show that

[1] School of Computer Science and Technology, University of Science and Technology of China (USTC), Hefei 230026, China
[2] School of Data Science, USTC, Hefei 230026, China
* These authors contributed equally to the work. {ustcysy, cgdsss}@mail.ustc.edu.cn
† Corresponding author. jianmin@ustc.edu.cn

the approach is effective and outperforms compared methods in terms of the success rate.

- We apply both the sensor map and the pedestrian map as the inputs of the neural network. We show that, by distinguishing the information between regular obstacles and pedestrians, the network allows the robot not only to avoid regular obstacles but also various pedestrians.
- We show that, the map-based approach allows the robot to generate the sensor map and the pedestrian map from multiple sensor data or sensor fusion results [12], which make it to be effective and easy to be deployed to an actual robot.

II. RELATED WORK

Existing dense crowd navigation methods can be generally divided into two categories, i.e., trajectory-based and rule-based. A trajectory-based method first predicts the future path of other agents in the scene, and then selects the optimal path based on the prediction [14], [15]. This type of methods normally require a high computational cost and their performances are significantly affected by the accuracy of the prediction.

On the other hand, a rule-based method commonly assumes that all agents in the environment follow the same collision avoidance strategy. Helbing and Molnar [4] propose social force model (SFM), which assumes that pedestrians are subject to three main forces in the environment: the influence of the driving force, the force between people, and the force between people and obstacles. The resultant force of these forces produces the acceleration of the pedestrian's movement. Reciprocal velocity obstacles (RVO) [3] and ORCA [5] assume that the agents can perfectly perceive the movements of others and cooperate with each other to generate collision-free movements respectively. Note that, the performance of these methods is usually limited when pedestrians behave differently.

Recently, imitation learning methods have been applied for crowd navigation. Kretzschmar et al. [16] propose an approach that allows a mobile robot to learn the behavior of interacting agents such as pedestrians from demonstration via Inverse Reinforcement Learning. Liu et al. [17] imitate the well-tuned planner, which considers the costmap[1] of the robot as the input and outputs robot's control commands, i.e., linear and angular velocities. Tai et al. [18] train the neural network in Gazebo simulator[2] with simulation pedestrians driven by SFM. However, these methods require a large number of labeled data for training.

DRL methods are also applied for crowd navigation. Fan et al. [19] consider the sensor data of the 2D laser scanner as the input of the policy network and trains the network in simulation environments with multiple robots driven by the same policy network, which allows the robots to properly interact with each other. Similarly, Chen et al. [20] consider the movement information of other agents as the input of the

policy network and trains the network with multiple robots driven by the same network in [21]. In our previous work [12], the egocentric sensor map is considered as the input of the policy network, which is also trained with multiple robots driven by the same network. Notice that, these methods do not distinguish the information between regular obstacles and pedestrians.

On the other hand, Güldenring et al. [11] use SFM to drive pedestrians in their training environments. Meanwhile, the shapes and movements of pedestrians' legs are carefully modeled in the simulator to distinguish pedestrians with other obstacles. Chen et al. [10] use ORCA to drive pedestrians in their training environments and considers the movement data of pedestrians as the input, where regular obstacles are considered as special agents that do not move. Note that, above methods assume that pedestrians only follow a certain collision avoidance strategy, which is often not appropriate in practice.

Liu et al. [22] use two strategies, i.e., ORCA and moving forward, to drive pedestrians respectively. However, it is hard to generate the approach for pedestrians with other collision avoidance strategies, like SFM. In this paper, we distinguish the information between regular obstacles and pedestrians and propose a map-based DRL approach for crowd navigation with pedestrians following multiple different strategies.

III. PRELIMINARIES

In DRL, the problem is specified as a Markov decision process (MDP). An MDP is a tuple $M = (S, A, P, R, \gamma)$, where S is the state space, A is the action space, P represents the transition probability between states, R is the reward function, and γ is the discount factor in $(0, 1)$. The purpose of DRL is to find a strategy π^* that maximizes the expected cumulative return of each trajectory τ.

We use proximal policy optimization (PPO) [13] to train our crowd navigation policy. PPO maintains two networks during the training, i.e., a policy network and a value network. It uses importance sampling when updating the parameters, so that each sampled data can be applied multiple times. PPO also introduces the clip operation when the policy is updated to reduce the gap between the behavior policy and the target policy. We use generalized advantage estimator (GAE) [23] to estimate the advantage function used in PPO.

IV. APPROACH

In this section, we first specify key components of DRL for crowd navigation. Then, we describe its training process.

A. Reinforcement Learning Components

1) Observation space: An observation is composed of three parts, i.e., the sensor map, the three-channel pedestrian map, and the robot's target pose.

Sensor map is specified by an egocentric local grid map $\mathbf{M}_{sens}$ of the robot. As illustrated in Fig. 1, the sensor map represents the environmental information around the robot, including its shape and observable appearances of obstacles. The sensor map is constructed from a costmap that can be

---

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

generated from various sensor data, like outputs of 2D lasers or depth cameras [12].

Pedestrian map is a local grid map $\mathbf{M}_{ped}$ with three channels that indicates the location and speed of pedestrians around the robot. As illustrated in Fig. 1, for a pedestrian map, the first channel specifies locations of surrounding pedestrians and the rest two channels specify the speeds of corresponding pedestrians for *x*-axis and *y*-axis, respectively.

Target pose $g^t = (x^t, y^t, \alpha^t)$ at time step t consists of the target position (x^t, y^t) and the target orientation α^t of the robot.

2) Action space: In this paper, we implement the approach on a differential drive robot that follows desired speed commands. Then an action a^t at time step t consists of a linear velocity v^t and an angular velocity ω^t, i.e., $a^t = (v^t, \omega^t)$. We implement both discrete and continuous actions for the robot. In specific, for discrete actions, we set a linear velocity $v^t \in \{0.0, 0.2, 0.4, 0.6\}$ and an angular velocity $\omega^t \in \{0.9, -0.6, -0.3, 0.0, 0.3, 0.6, 0.9\}$. For continuous actions, we set $v^t \in [0, 0.6]$ and $\omega^t \in [-0.9, 0.9]$. Both discrete and continuous actions can be directly performed by the differential robot in our experiments. Notice that, $v^t \geq 0$, i.e., moving backwards is not allowed, due to the lack of rear sensors.

3) Reward function: In crowd navigation, the objective of a navigation policy is to minimize the arriving time of the robot without collision. In our approach, we specify the reward r^t at time step t as

$$r^t = r^t_{goal} + r^t_{safe} + r^t_{step} + r^t_{shaping},$$

$$r^t_{goal} = \begin{cases} r_{arr} & \text{if target is reached,} \\ 0 & \text{otherwise,} \end{cases}$$

$$r^t_{safe} = \begin{cases} r_{col} & \text{if collision,} \\ -\varepsilon_1 \left(1 - d^t_{\min}\right) & \text{if } d^t_{\min} < 1, \\ 0 & \text{otherwise,} \end{cases}$$

$$r^t_{shaping} = \varepsilon_2 \left(\left\| \mathbf{p}^{t-1} - \mathbf{p}_g \right\| - \left\| \mathbf{p}^t - \mathbf{p}_g \right\| \right),$$

where $r_{arr} > 0$ and r^t_{goal} specifies the reward when the robot arrives its target, $r_{col} < 0$ denotes the penalty for the collision, $-\varepsilon_1 \left(1 - d^t_{\min}\right)$ denotes the penalty when the robot is close to a pedestrian, i.e., d^t_{min} denotes the minimal distance from the robot to its closest pedestrian, ε_1 is a hyper-parameter, $r^t_{step} < 0$ denotes a small penalty to encourage short paths, $\mathbf{p}^t$, $\mathbf{p}_g$ denote the positions of the robots and its target, ε_2 is a hyper-parameter, and the reward shaping item $r^t_{shaping}$ encourages the robot to move toward the target.

In our experiments, we set $r_{arr} = 500$, $r_{col} = -500$, $\varepsilon_1 = 50$, $\varepsilon_2 = 200$, and $r^t_{step} = -5$.

4) Network architecture: The architecture of the convolutional network for the crowd navigation policy in PPO is shown in Fig. 1. The input of the network consists of the sensor map, the pedestrian map, and the target pose. For discrete actions, the network outputs a 28-dimensional vector from a softmax layer to choose the pair of linear and angular velocities. For continuous actions, the network outputs the mean of the action sampled from a Gaussian distribution.
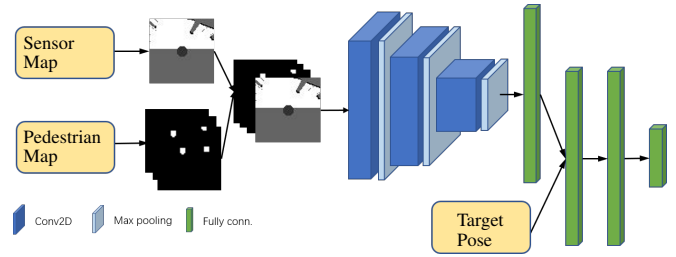

Fig. 1.   The architecture of the crowd navigation policy network.

In particular, the network first produces feature maps for the corresponding sensor map and the pedestrian map using three convolutional layers and three max pooling layers. Followed by a fully-connected layer with 512 units, these feature maps are converted to a 512 dimensional vector. The network also projects the local goal to a 3 dimensional vector. Then the network combines both vectors and feeds them to two fully-connected layers with 512 units. At last, for discrete actions, the network uses a softmax layer with 28 units to choose the pair of linear and angular velocities from corresponding values. For continuous actions, the network applies a fully-connect layer with 2 units without activation to produce the mean of the linear velocity v^t and the mean of the angular velocity ω^t. Then the continuous actions is sampled from the Gaussian distribution $\mathcal{N}(a^t_{mean}, a^t_{logstd})$, where a^t_{logstd} is the log standard deviation generated by a standalone network and $a^t_{mean} = (v^t, \omega^t)$. A clip function is also applied to ensure that the resulting actions are valid in the action space.

The value network has the same architecture as the policy network, except its last layer is modified to only output the value of the state.

B. Training with Multiple Strategies and Environments

We train the network in environments that are constructed by a customized simulator based on OpenCV[3]. We extend the simulator from the one used in [12] to add pedestrians driven by different collision avoidance strategies. In particular, we add the shapes of pedestrians' legs and corresponding walking movements as in [11] to the simulator. We also implement two collision avoidance strategies, i.e., ORCA[4] and SFM[5] to drive pedestrians in the simulator.

As illustrated in Fig. 2, we specify two scenarios for the training, i.e., random scenario and circular scenario. In particular, an environment in random scenario contains two robots, four pedestrians, and four static obstacles, which would randomly choose locations for obstacles, the starting and target positions of robots and pedestrians. Then the robots in the environment would be driven by the policy network and pedestrians would be driven either by ORCA or SFM. Note that, there are two robots sharing the same navigation policy in the environment and they also need

[3]https://opencv.org/.
[4]https://github.com/snape/RVO2.
[5]https://github.com/chgloor/pedsim.
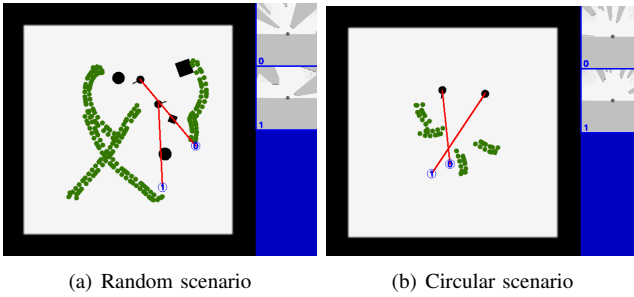
(a) Random scenario   (b) Circular scenario

Fig. 2. Two scenarios for the training, where the blue digital circles represent the target positions of robots with the corresponding number, red lines specify the straight paths from the starting position to the target for robots, green spots denote the legs' trajectories of pedestrians, black blocks denote static obstacles, and blue boxes on the right illustrate sensor maps of each robot.

to avoid collisions with each other. As discussed in [12], this helps the network to learn the ability of multi-robot obstacle avoidance, which can also help the robot to avoid pedestrians. Training in environments of random scenario enables the policy network to avoid static and dynamic obstacles, and interact with pedestrians. An environment in circular scenario contains two robots and four pedestrians, which would randomly place robots and pedestrians on a circle with a random radius. Training in environments of circular scenario enhances the policy network to interact with pedestrians.

In the training process, we extend PPO to collect experiences from four different environments in parallel at each iteration. In particular, we construct two environments of random scenario and drive pedestrians in both environments with ORCA and SFM, respectively. We also construct two environments of circular scenario with pedestrians driven by ORCA and SFM, respectively. Then we train the network based on experiences generated from these four environments.

V. EXPERIMENTS

In this section, we evaluate our PPO based crowd navigation approach in both the simulation and the real world. We first specify details of our implementation including the hyper-parameters, hardware, and software for the training. Then, we quantitatively evaluate the performance of our crowd navigation policy in various simulation scenarios and compare it with other approaches. We also deploy the trained model to a differential drive robot and test its navigation performance in the real world. Both qualitative and quantitative experiments show that our approach allows the robot to successfully interact with various pedestrians and is effective with a high success rate. The demonstration video is also available.

A. Reinforcement Learning Setup

We trained our crowd navigation policy for the differential drive robot following the PPO algorithm with the hyper-parameters listed in Table I.

Hyper-parameter	Value
learning rate for policy	5×10^{-5}
learning rate for value	1×10^{-3}
discount factor (γ)	0.99
replay buffer size	2048
image size	48×48
maximum episode length	200
robot radius	0.17
maximum pedestrian speed	0.5

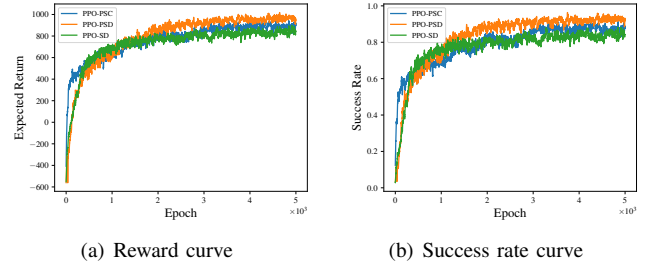(a) Reward curve   (b) Success rate curve

Fig. 3. The average reward curve and the success rate curve of approaches.

Both the policy network and the value network are implemented in TensorFlow[6] and trained with the Adam optimizer [24]. We also want to train network on MindSpore[7], which is a new deep learning computing framework. These problems are left for future work. The training hardware is a computer with an i9-9900k CPU and an NVIDIA Titan RTX GPU.

We use PPO-PSC to denote our PPO based approach with continuous actions, PPO-PSD to denote our PPO based approach with discrete actions. We compare our approaches to ORCA[8] with perfect sensing and PPO-SD, the modification of our approach PPO-PSD by only using the sensor map.

Fig. 3 shows the average reward curve and the success rate curve of the three PPO based approaches, i.e., PPO-PSC, PPO-PSD, and PPO-SD. It can be seen that the introduction of the pedestrian map can improve the performance of the PPO based crowd navigation.

B. Experiments on Simulation Scenarios

1) Performance metrics: We introduce three metrics to evaluate the performance of approaches for crowd navigation as the following:

- *Success rate ($\bar{\pi}$)*: the ratio of the episodes that end with robots reaching their targets without collision.
- *Extra time ($\bar{t}$)*: the time required for the robot to successfully reach its target without collision minus the time for the robot to drive straight to its target with the maximum speed.

---

[6]https://www.tensorflow.org/.
[7]https://www.mindspore.cn/.
[8]With a slight abuse of notation, we use ORCA here to denote the corresponding crowd navigation policy for the robot.

- *Average angular velocity change ($\nabla\omega$)*: the average changes of the angular velocity at each step, which reflects the smoothness of the trajectory.

2) Comparative experiments: Now we compare the performance of four crowd navigation approaches, i.e., ORCA, PPO-SD, PPO-PSC, and PPO-PSD, in simulation environments of random and circular scenarios.

Table II summarizes the average results of four approaches in 500 different environments of each scenario, where pedestrians are driven by ORCA and SFM. Note that the angular velocity change value of ORCA in the table is empty, because the classic ORCA algorithm only outputs the linear velocity in the x and y directions and not the angular velocity.

Note that, ORCA requires every moving agents to cooperate with each other in the environment. Then there will be collisions between pedestrians in environments, where some pedestrians are driven by ORCA and some are driven by SFM, and these environments are not applicable for testing the performance of crowd navigation approaches.

Environments	Methods	$\bar{\pi}$	$\bar{t}$	$\nabla\omega$
Random scenario with ORCA	ORCA	0.464	2.97	-
	PPO-SD	0.856	3.60	0.63
	PPO-PSC	0.862	5.64	0.74
	PPO-PSD	0.936	4.83	0.57
Circular scenario with ORCA	ORCA	0.862	7.54	-
	PPO-SD	0.916	4.25	0.82
	PPO-PSC	0.948	5.18	1.02
	PPO-PSD	0.994	4.87	0.57
Random scenario with SFM	ORCA	0.318	1.53	-
	PPO-SD	0.840	3.45	0.63
	PPO-PSC	0.873	4.90	0.74
	PPO-PSD	0.968	4.99	0.56
Circular scenario with SFM	ORCA	0.216	1.43	-
	PPO-SD	0.902	4.49	0.84
	PPO-PSC	0.924	6.22	0.97
	PPO-PSD	0.996	5.90	0.63

Table II shows that, PPO based approaches allow the robot to successfully interact with pedestrians driven either by ORCA or SFM. Moreover, the introduction of the pedestrian map can improve the performance. PPO-PSD outperforms others in terms of the success rate.

Fig. 4 illustrates robot's trajectories generated by different approaches, where the trajectory generated by ORCA leads the robot to be suck due to nearby obstacles and pedestrians, the trajectory generated by PPO-SD leads the robot to collide with a pedestrian as the pedestrian map is not considered in it, and both PPO-PSC and PPO-PSD successfully drive the robot to its target.

The experimental results show that our approach PPO-PSD is effective and outperforms others.

(a) Trajectories by ORCA   (b) Trajectories by PPO-SD

(c) Trajectories by PPO-PSC   (d) Trajectories by PPO-PSD
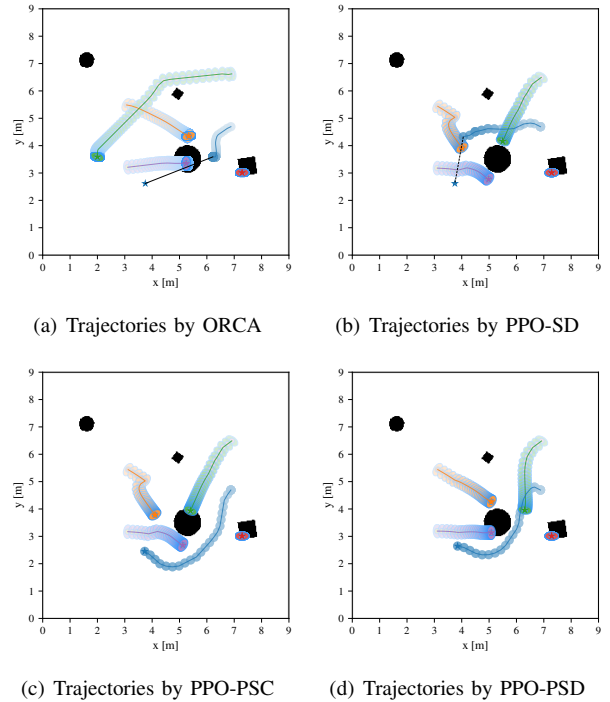
Fig. 4. Robot's trajectories generated by different approaches. The blue spots denote the trajectories of the robot and other spots denote the trajectories of pedestrians.

3) Pedestrian with multiple strategies: As specified in Section IV-B, we train the network on both environments with pedestrians driven by ORCA and SFM, respectively. Here we compare this training process with alternatives.

Most existing crowd navigation approaches assume that pedestrians follow a predefined collision avoidance strategy and test their performance under the same strategy. Here we share the configurations for the training environments, except the collision avoidance strategies for pedestrians. We introduce two variants of our PPO-PSD approach, i.e., PPO-ORCA and PPO-SFM. In particular, PPO-ORCA is modified from PPO-PSD by only considering pedestrians driven by ORCA, and PPO-SFM is modified from PPO-PSD by only considering SFM. With a slight abuse of notation, we also use PPO-multi to denote our approach PPO-PSD.

We evaluate the three approaches, i.e., PPO-multi, PPO-ORCA, and PPO-SFM, in environments of five scenarios. In specific, ORCA-random (resp. SFM-random) denotes the scenario from random scenario by only considering pedestrians driven by ORCA (resp. SFM), ORCA-circular (resp. SFM-circular) denotes the scenario from circular scenario by only considering pedestrians driven by ORCA (resp. SFM), and PPO-circular denotes the scenario that only contains five robots in its environment and randomly places the starting and target positions of the five robots on a circle with a random radius.

The performance of the three approaches on these five scenarios is shown in Fig. 5, i.e., the average success rate of each approach in 500 different environments of each scenario. The results show that PPO-SFM outperforms PPO-ORCA
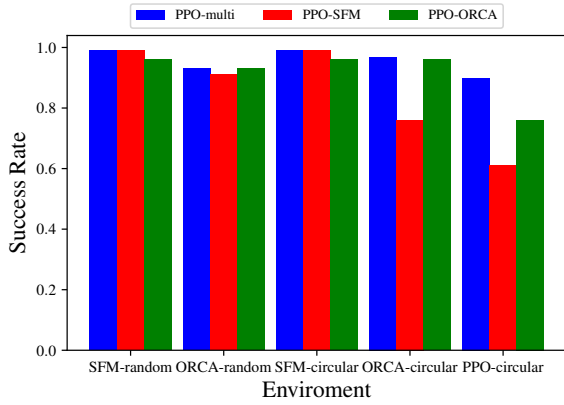
Fig. 5. Success rate of three approaches on five scenarios.

on SFM-random and SFM-circular, PPO-ORCA outperforms PPO-SFM on ORCA-random and ORCA-circular, and both of the approaches do not perform well on PPO-circular. PPO-multi (PPO-PSD) outperforms all other approaches in environments of all five scenarios, which implies that our training process is effective and allows the robot to successfully interact with pedestrians driven either by ORCA or SFM.

C. Deployment on Actual Robot

We deploy the trained model by PPO-PSD to an actual differential drive robot to perform crowd navigation in the real world.

As shown in Fig. 6, the robot is based on TurtleBot 2 with Kobuki base and uses a Hokuyo UTM-30LX scanning laser Rangefinder as the 2D laser sensor, a RealSense D455 depth camera as the pedestrian tracking sensor. The robot equips with two NVIDIA Jetson TX2 for the computation, where one is used to detect and track pedestrians and the other is used to run the trained network.
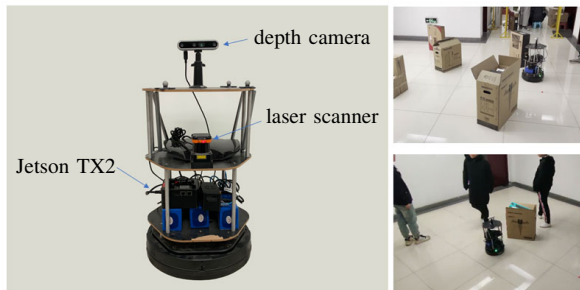

Fig. 6. The robot and its evaluation environments. Environments with paper boxes to serve as static obstacles (upper right) and walking pedestrians (bottom right).

We use SPENCER people tracking framework [25] in our implementation to track surrounding pedestrians of the robot and generate the pedestrian map for the model. We also replace the people detection module in SPENCER by YOLOv3 [26] for the better detection. We use the costmap generated by the laser sensor to construct the sensor map.

Both the sensor map and the pedestrian map have the fixed size 6.0×6.0m with the resolution 0.125m. We apply a particular filter based state estimator to generate robot's local targets from its global path in navigation. Then we deploy the trained model to be fed with these inputs and let the differential robot execute the outputs, i.e., desired linear and angular velocities, directly.

We evaluate the performance of crowd navigation of the robot in following real-world scenarios.

- *Static scenario*: environments that randomly place multiple paper boxes and a suitcase to block the robot from its starting position to its target.
- *Dynamic scenario*: environments that the robot needs to pass through a hall while some pedestrians walking around the robot and some paper boxes on its way.
- *Corridor scenario*: environments that the robot needs to pass through a corridor while some pedestrians walking towards the robot.

Notice that, the corridor in our experiments is quite narrow, then it is challenging for crowd navigation in environments of corridor scenario.

Our experiments show that the robot can interact with various pedestrians successfully and accomplish navigation tasks safely in most environments of these three scenarios. Fig. 7 and 8 illustrate the performance of the robot on the scenarios. More experiments can be found in our demonstration video. Note that, in few cases, the robot did not avoid the pedestrian in front of it. As discussed in [19], a safety strategy can be added to ensure the safety.

(a) Simple dynamic scenario
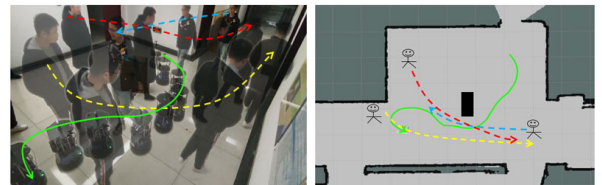
(b) Dense dynamic scenario

Fig. 7. Trajectories of the robot in environments of dynamic scenario. (a) illustrates a simple environment where the global path of the robot is a straight line to the target and the network drives the robot to avoid the obstacles and the walking pedestrian. (b) illustrates a dense environment with multiple pedestrians and static obstacles, where the network still needs to drive the robot to avoid pedestrians walking towards it.

VI. CONCLUSIONS

In this paper, we argue that pedestrian with multiple different collision avoidance strategies need to be handled by crowd navigation approaches. To address this challenge, we propose a PPO based crowd navigation approach that allows
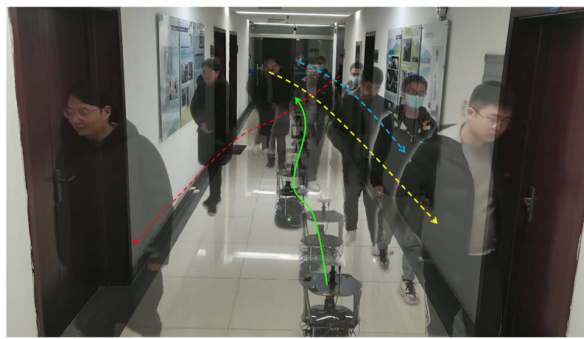
(a) Corridor scenario
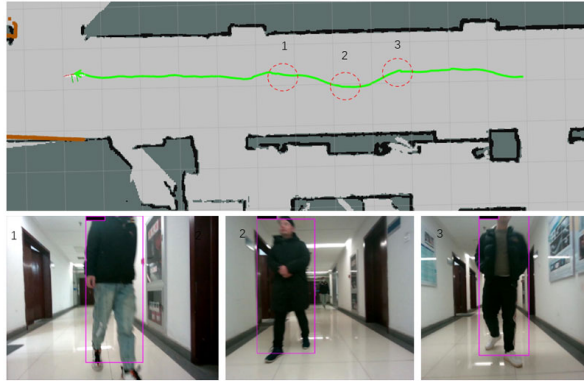
(b) Camera view

Fig. 8. Trajectories of robots in an environment of corridor scenario. (a) illustrates the trajectories of the robot and the surrounding walking pedestrians. (b) denotes the map of the environment and corresponding trajectory of the robot (upper image), and illustrates the camera views when the robot is at the corresponding dotted circles in the map (bottom image).

pedestrians driven by either OCRA or SFM. In particular, we apply both the sensor map and the pedestrian map as the inputs of the neural network. We show that distinguishing the information between regular obstacles and pedestrians improves the performance of crowd navigation. We also train the network in both simulation environments with pedestrians following OCRA and SFM respectively. We show that this training process allows the robot to successfully interact with various pedestrians. We also deploy the trained model to an actual robot and evaluate its performance in the real world. The experimental results show that our approach, PPO-PSD, is effective and outperforms compared methods in terms of the success rate.

Our work makes an improvement towards rapid and successful crowd navigation. Moving forward, we will consider more collision avoidance strategies for strategies and further investigate the generalization of the trained model.

REFERENCES

[1] J. Forlizzi and C. DiSalvo, "Service robots in the domestic environment: a study of the roomba vacuum in the home," in *Proceedings of HRI-2006*, 2006, pp. 258–265.

[2] R. Simmons, R. Goodwin, K. Z. Haigh, S. Koenig, and J. O'Sullivan, "A layered architecture for office delivery robots," in *Proceedings of ICAA-1997*, 1997, pp. 245–252.

[3] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *Proceedings of ICRA-2008*. IEEE, 2008, pp. 1928–1935.

[4] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

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

[6] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *Proceedings of IROS-2017*, 2017, pp. 31–36.

[7] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion control for mobile robot navigation using machine learning: a survey," *arXiv preprint arXiv:2011.13112*, 2020.

[8] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.

[9] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of IROS-2018*, 2018, pp. 3052–3059.

[10] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proceedings of ICRA-2019*, 2019, pp. 6015–6022.

[11] R. Güldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *Proceedings of IROS-2020*, 2020, pp. 6053–6060.

[12] G. Chen, S. Yao, J. Ma, L. Pan, Y. Chen, P. Xu, J. Ji, and X. Chen, "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, no. 17, p. 4836, 2020.

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

[14] Z. Chen, C. Song, Y. Yang, B. Zhao, Y. Hu, S. Liu, and J. Zhang, "Robot navigation based on human trajectory prediction and multiple travel modes," *Applied Sciences*, vol. 8, no. 11, p. 2205, 2018.

[15] B. D. Luders, G. S. Aoude, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe avoidance of dynamic obstacles with uncertain motion patterns," Tech. Rep., 2011.

[16] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[17] Y. Liu, A. Xu, and Z. Chen, "Map-based deep imitation learning for obstacle avoidance," in *Proceedings of IROS-2018*, 2018, pp. 8644–8649.

[18] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Proceedings of ICRA-2018*, 2018, pp. 1111–1117.

[19] T. Fan, P. Long, W. Liu, and J. Pan, "Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios," *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 856–892, 2020.

[20] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proceedings of IROS-2017*, 2017, pp. 1343–1350.

[21] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proceedings of the ICRA-2017*, 2017, pp. 285–292.

[22] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *Proceedings of IROS-2020*, 2020, pp. 5671–5677.

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

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

[25] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments," in *Proceedings of ICRA-2016*. IEEE, 2016, pp. 5512–5519.

[26] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.