Deep Reinforcement Learning Based Mobile Robot Navigation in Crowd Environments

Guang Yang and Yi Guo

Abstract— **Robots are becoming popular in assisting humans. The mobile robot navigation in human crowd environments has become more important. We propose a deep reinforcement learning-based mobile robot navigation method that takes the observation from the robot's onboard Lidar sensor as input and outputs the velocity control to the robot. A customized deep deterministic policy gradient (DDPG) method is developed that incorporates guiding points to guide the robot toward the global goal. We built a 3D simulation environment using an open dataset of real-world pedestrian trajectories that were collected in a large business center. The neural network models are trained and tested in such environments. We compare the performance of our proposed method with existing algorithms that include a classic motion planner, an existing DDPG method, and a generative adversarial imitation learning (GAIL) method. Using the measurement metrics of success rate, the number of times freezing, and normalized path length, extensive simulation results show that our method outperforms other state-of-the-art approaches in both trained and untrained environments. Our method has also better generalizability compared with the GAIL method.**

I. Introduction

Mobile robots have been used to assist people in various environments such as hotels [1] and warehouses [2]. In these applications, safety is the basic requirement and the robot needs to navigate collision-free in crowded environments. Traditional collision avoidance algorithms (e.g., [3] [4] [5]) find shortest paths through heuristic search with a cost function, or sample the velocity space and evaluate optimal motion trajectories (e.g., [6]). Considering the highly dynamic nature of human-centered environments, it is still a challenging problem for a mobile robot to autonomously navigate in a crowded environment and share the space with humans.

Learning-based algorithms have been developed for mobile robot navigation in human environments. A recent approach is to learn pedestrian behavior through demonstration. Inverse reinforcement learning (IRL) [7] or imitation learning (IL) [8] were applied to replace the classic mechanism of reward function selection or to learn directly from demonstration data. In such methods, the robot learns a synthetic reward to update motion policy by repeating the expert behavior, which helps the robot to generate the expert-like behavior. In [9], Tai et al. developed a generative adversarial imitation learning (GAIL) method, and the demonstration data was generated using a social force model (SFM) [10]. Although the SFM has long been used in simulating human crowd behaviors, it cannot replace pedestrian motion behaviors in the real world. Also, excessive reliance on demonstration data makes the robot difficult to generalize its motion behavior in different environments.

Deep Reinforcement Learning (DRL) based methods were developed for the robot navigation problem. Traditional DRL methods cannot be used for continuous actions. Deep Deterministic Policy Gradient (DDPG) [11] successfully solved the problem in continuous action generation. The approach presented in [12] used DDPG to generate continuous linear and angular velocities in a motion planner for stationary environments. They generalized the simulated model to real-world settings and demonstrated the performance of the method in a real-world robot navigation problem. The approaches reported in [13], [14] demonstrated that DRL solves the problem of collision avoidance in crowd-robot interaction for mobile robots. Compared to the imitation-learning approach, the DRL method requires longer training time, the algorithm has better generalizability, and the robot is able to navigate in a different environment without additional training. However, most existing DRL work avoids collision in a local region, its performance in a global crowd environment is yet to be demonstrated.

In this paper, we focus on solving the problem of mobile robot navigation in crowded environments. We propose a DRL-based algorithm using DDPG with global guiding points. The neural network model is combined with the guiding points that are generated from a global planner, and the robot can complete a global navigation task without freezing in crowded environments. We use a large human trajectory dataset from a real-world business center [15] to create the simulation environment, and the robot is trained in such an environment. In our simulation tests, the robot obtains a laser range scan from the onboard Lidar sensor and obtains the position of the odometry as input. The robot generates linear and angular velocities for robot motion control. We compare the performance of our proposed algorithm with other existing methods including DDPG [11] and imitation learning [8]. Simulation results show better navigation performances of our method in terms of less number of times freezing and a better success rate. We also demonstrate better generalizability of our method compared with the GAIL method.

This work was partially supported by the US National Science Foundation under Grants CMMI-1825709 and IIS-1838799.

The authors are with the Department of Electrical and Computer Engineering, Stevens Institute of Technology, Hoboken, NJ 07030, USA. Emails: {gyang11, yguo1}@stevens.edu

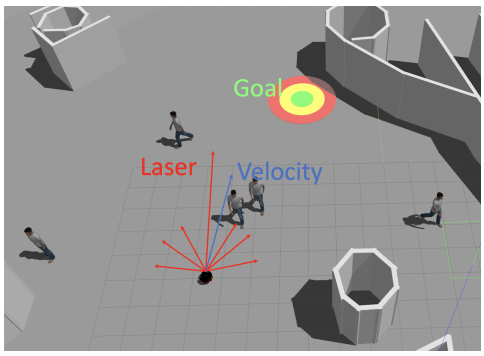


Fig. 1: The environment setup in ROS Gazebo.

The main contributions of this work include (1) a motion planner that takes the onboard Lidar sensor as input and outputs the robot's motion control; (2) neural network models trained end-to-end through DDPG with global guiding points; (3) the use of real-world pedestrian trajectory dataset in a mall-like environment; and (4) the trained model generalizable to different environments due to the benefit of the DDPG algorithms.

The rest of the paper is organized as follows: Section II formulates the robot navigation problem. Section III describes our proposed DRL approach. We present the simulation environment and evaluate the performance of our approach in Section IV. Discussions of methods and its limitations are provided in Section V. Finally, we conclude our work and discuss future work in Section VI.

II. The Robot Navigation Problem

In this paper, we consider the robot navigation task as a Markov Decision Process following prior work in [8]. The robot is placed in a crowded environment, and the task for the robot is to navigate to a given goal position autonomously without collisions with the stationary objects and the moving humans in the environment. At each time t, the robot takes the action $\boldsymbol{a}_t$ (defined as the robot's velocity $\boldsymbol{V}_t = (v_t, \omega_t)$ that represents the linear and angular velocities) according to the observation $\boldsymbol{o}_t$ obtained by the Lidar, the robot position $\boldsymbol{p}_r = (x_r, y_r)$ and heading ϕ_r, and the goal position $\boldsymbol{p}_g$. This decision-making process keeps generating new actions to control the robot until the robot reaches the goal position.

We build a 3D simulation environment in ROS Gazebo, see Fig. 1. The building structures and pedestrian trajectories are created from real-world data collection [15]. The robot is a Pioneer-3DX with an onboard Velodyne Lidar. The robot is placed at an initial position in the simulator and a goal position is given. The onboard Lidar sensor collects 360-degree scans along the horizontal plane as the observation $\boldsymbol{o}_t$. The existing SLAM algorithm is applied and the robot can localize itself in the world coordinate.

In the next section, we present our learning-based approach to generate the action $\boldsymbol{a}_t$ that controls the robot navigating to the goal position.

III. The Proposed Approach

To address the robot navigation problem, we propose a DRL method using the DDPG architecture. The overall diagram of our method is depicted in Fig. 2. The robot interacts with the environment to obtain the state $\boldsymbol{s}_t$ and the reward r_t. The state $\boldsymbol{s}_t$ contains the observation $\boldsymbol{o}_t$, the position of the robot $\boldsymbol{p}_r = (x_r, y_r)$, the heading of the robot ϕ_r, the distance vector between the robot and the goal position $\boldsymbol{d}_g$, that is, $\boldsymbol{s}_t = (\boldsymbol{o}_t, \boldsymbol{p}_r, \phi_r, \boldsymbol{d}_g)$. The state and the reward are stored in the experience replay buffer and randomly sampled to update the parameters of the neural networks. The neural networks contain an actor network and a critic network, whose structures are described in Section III-B.

A. Rewards Function and Global Guiding Point Selection

Existing DRL-based approaches, such as the one [12], guide robots to their goal by leveraging the orientation of the direct line between the robot and its goal. While effective in local areas, these methods struggle in crowded settings with obstacles and a global goal, leading to the robot making unneeded attempts to go through obstacles, failing to feasible path, or even freezing in place.

To mitigate the challenges inherent in traditional DRL approaches, we incorporate the notion of "global guiding points," enhancing the fundamental structure of existing methodologies by embedding the goal point directly within the reward functions. This integration improves the efficiency of the robot's global navigation tasks.

We first use an existing path planning method, such as D* [3], to generate a sequence of path points avoiding stationary obstacles. Given the variable trajectories of pedestrians, these path points act as flexible guiding markers instead of fixed waypoints, accommodating the dynamic nature of the environments. Then, we select the reward function r_t as follows:

$$r_t = \begin{cases} c_1 & \text{arrive;} \\ c_2 & \text{collision;} \\ c_3 * (d_{rg} + d_{gg}) & \text{others.} \end{cases} \quad (1)$$

where c_1 is a positive constant as a reward; c_2 is a negative constant as a penalty for collisions; c_3 is a constant parameter multiplied by the path cost, which is the sum of the distance between the robot and its current guiding point d_{rg} and the distance between the current guiding point and the goal d_{gg}. The selection of the guiding point is critical in our navigation approach, aimed at providing the robot with an efficient local direction to reach its destination. To determine the current guiding point, we consider the two closest path points to the robot's current position. For each of these candidate points, we calculate the sum of two distances: the distance from the robot to the guiding point (d_{rg}) and the distance from the guiding point to the goal (d_{gg}). The guiding point that minimizes this combined distance is selected as the current guiding point for the robot. This will guide the robot closer to the goal while optimizing its path with an effective local direction toward the target goal position. In our simulation

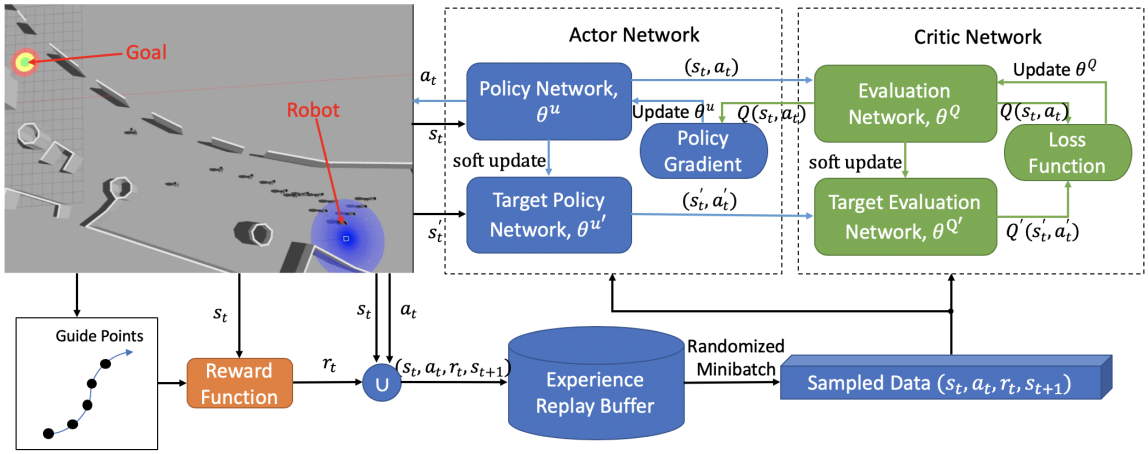


Fig. 2: The overall architecture.

study (described in Section IV), the parameters are chosen as $c_1 = 200, c_2 = -150, c_3 = -0.05$.

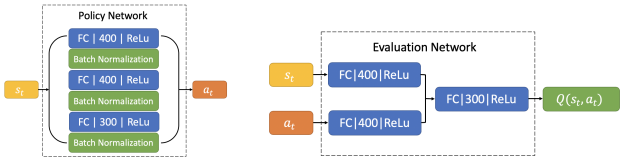
B. The Neural Network Structure

Our DRL structure adopts the DDPG framework, which is based on an Actor-Critic architecture and draws on the experience reply mechanism and the target network idea of the deep Q-learning [11], [16]. Each of the Actor and Critic networks consists of the current network and the target network. The Actor network takes the state $\boldsymbol{s}_t$ as input, and outputs action to the robot (i.e., the robot's linear and angular velocity controls), that is, $\boldsymbol{a}_t = \boldsymbol{u}_t(\boldsymbol{s}_t|\theta^u)$, where θ^u is the parameter of the actor-network and is updated using the gradient ascent method, that is,

$$\theta^u = \theta^u + \beta \cdot g, \tag{2}$$

$$g = \frac{\partial q(\boldsymbol{s}_t, \boldsymbol{u}_t(\boldsymbol{s}_t|\theta^u), \theta^Q)}{\partial \theta^u} \tag{3}$$

where g is the gradient and β is the learning rate. Fig. 3 (a) shows the structure of the policy network in the actor network.



(a) (b)

Fig. 3: The network structure of the Policy Network in (a) and the Evaluation Network in (b). Each layer is represented by its type, dimension, and activation.

The Critic network takes the state and action pairs as input to get the Q-value prediction, $q_t(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta^Q)$, where θ^Q is the parameter of critic network, and is updated using the loss gradient descent method:

$$\theta^Q = \theta^Q - \alpha \cdot \delta_t \cdot \frac{\partial q(\boldsymbol{s}_t, \boldsymbol{a}_t|\theta^Q)}{\partial \theta^Q}, \tag{4}$$

$$\delta_t = q_t - (r_t + \gamma \cdot q_{t+1}) \tag{5}$$

where q_{t+1} is the evaluation network makes prediction for time t+1, $\boldsymbol{a}'_{t+1}$ is the prediction of action, shown in equation (6), α is the learning rate. Fig. 3 (b) shows the structure of the evaluation network in the Critic network.

$$q_{t+1} = q(\boldsymbol{s}_{t+1}, \boldsymbol{a}'_{t+1}|\theta^Q), \& where \boldsymbol{a}'_{t+1} = \boldsymbol{u}_t(\boldsymbol{s}_{t+1}|\theta^u) \tag{6}$$

C. The Training Algorithm

Algorithm 1 illustrates the training algorithm of the DRL network. The input of Algorithm 1 is robot state $\boldsymbol{s}_t$ and the goal position $\boldsymbol{p}_g$. The output is the neural networks' parameter θ that includes the parameters of the policy network, evaluation network, target policy network, and target evaluation network, that is, $\theta^u, \theta^Q, \theta^{u'}$, and $\theta^{Q'}$.

In Algorithm 1, the neural network parameters θ and the experience reply buffer are initialized in lines 1-3. In the outer loop starting from line 4, the parameters θ are updated iteratively through M episodes. At the beginning of each epoch, the training environment is initialized, and the robot is put at its initial position in the environment with pedestrians. Then, guiding points are obtained using a global planner in each training epoch. In the inner loop starting from line 7, the environment evolves for N time steps. At each time step, the policy network generates an action $\boldsymbol{a}_t$ based on the current state $\boldsymbol{s}_t$. Then, the guiding point is obtained by comparing the path cost from the two closest guiding points of the robot. The shortest path cost is selected as the guiding point to be used for computing a reward, r_t. We then store the transition in the experience reply buffer and randomly sample a minibatch of transitions from the replay buffer for the update training of our models. The replay buffer allows the robot to learn from previous unrelated

Algorithm 1 The Training Algorithm

Input: The robot state $\boldsymbol{s}_t = (\boldsymbol{o}_t, \boldsymbol{p}_r, \phi_r, \boldsymbol{d}_g)$, the goal position $\boldsymbol{p}_g$
Output: Network parameters θ

1: Initialize policy network $u(s|\theta^u)$ and evaluation network $Q(s, a|\theta^Q)$ with parameter θ^u and θ^Q
2: Initialize target networks u' and Q' with parameter $\theta^{u'} \leftarrow \theta^u$, $\theta^{Q'} \leftarrow \theta^Q$
3: Initialize replay buffer D
4: **for** $episode \leftarrow 1$ to M **do**
5: Initialize the environment and obtain state $\boldsymbol{s}_1$
6: Obtain the global guide points from the global planner
7: **for** $t \leftarrow 1$ to N **do**
8: Propose and execute an action $\boldsymbol{a}_t$ and obtain state $\boldsymbol{s}_t$
9: Obtains guiding points from a global path planner
10: Calculate the rewards r_t using (1)
11: Store transition $[\boldsymbol{s}_t, \boldsymbol{a}_t, r_t, \boldsymbol{s}_{t+1}]$ into D
12: Randomly sample the minibatch from D for network training
13: Update the critic-network parameter from θ^Q_n to θ^Q_{n+1} using equation (4)
14: Update the actor-network parameter from θ^u_n to θ^u_{n+1} using equation (2)
15: Soft update the target networks
16: $\theta^{u'} \leftarrow \tau \cdot \theta^u + (1-\tau) \cdot \theta^{u'}$
17: $\theta^{Q'} \leftarrow \tau \cdot \theta^Q + (1-\tau) \cdot \theta^{Q'}$
18: **end for**
19: **end for**

Algorithm 2 Robot Motion Control

Input: State $\boldsymbol{s}_t$, goal position $\boldsymbol{p}_g$
Output: Robot velocity control $\boldsymbol{V}_t$

1: Load the network parameters θ trained by Algorithm 1
2: Initialize robot position $\boldsymbol{p}_r$, and set goal $\boldsymbol{p}_g$
3: **for** $timestep \leftarrow 1$ to N **do**
4: Obtain the robot state $\boldsymbol{s_t}$
5: Compute the action $\boldsymbol{a}_t$ using the policy network
6: Obtain $\boldsymbol{V}_t \leftarrow \boldsymbol{a}_t$
7: **end for**

experiences, which enables an effective and faster training process. Lines 13 and 14 use equations (2)-(5) to update the parameters of the policy network and the evaluation network. In lines 15 to 17, the target networks use the soft update to update the parameters, where τ is the constant coefficient to adjust the soft update factor. The weights of the target network are constrained to change slowly, greatly improving the stability of learning. After the training process, the policy neural network is deployed on the robot for use in motion control.

The training of our model was performed using a set of hyper-parameters, carefully chosen to ensure the convergence of the learning process. The Adam optimizer was utilized as an optimizer, and Gaussian noise was incorporated for exploration noise. The list of the selected hyper-parameters is presented in Table I.

TABLE I: Hyper-parameters used in training

Parameter	Value
Learning Rate (Actor)	0.001
Learning Rate (Critic)	0.002
Discount Factor (γ)	0.98
Batch Size	64
Max Episode	5000
Soft Update Coefficients (τ)	0.01

D. Robot Motion Control

After the training process of Algorithm 1, the robot uses the trained policy network to generate the control. In Algorithm 2, it takes the robot state $\boldsymbol{s}_t$ and the goal position $\boldsymbol{p}_g$ as input and outputs the velocity $\boldsymbol{V}_t$ that is applied to the mobile robot. In line 1 of Algorithm 2, the policy network parameters θ trained by Algorithm 1 are loaded. Line 2 initializes the robot position and the testing environment, and sets the goal position. In the loop from lines 3 to 7, the robot obtains the state $\boldsymbol{s}_t$ that is input into the trained policy network and outputs the action $\boldsymbol{a}_t$ that converts to velocity vector $\boldsymbol{V}_t$ to be deployed on the mobile robot.

IV. Simulations and Performance Evaluation

A. The Simulation Environment

We created a simulation environment in ROS Gazebo. A real-world pedestrian trajectory dataset [15] was used and integrated with our simulator. This dataset contains pedestrian motion trajectories at a Japanese business center, Asia Pacific Trade Center (ATC) (which is a mall-like environment), that were collected using 3D range sensors over 92 days. It recorded approximately 3.25 million pedestrians, and the total length of the trajectories in the dataset is 128,692 km. The measurement frequency of the recorded data is approximately 25 Hz.

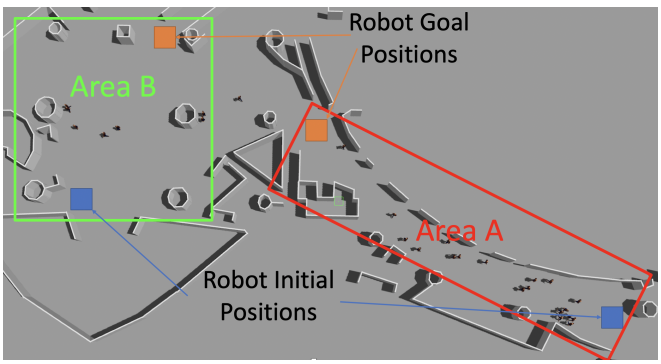


Fig. 4: The simulation environment created in ROS Gazebo using a real-world pedestrian trajectory dataset. The neural network was trained in Area A, and tested in both Area A and Area B. In each run, the robot started at a randomly selected position in the robot's initial position area (marked as the blue square), and the goal position was randomly selected in the robot's goal position area (marked as the orange square).

We selected the Pioneer 3-DX robot equipped with an onboard Lidar sensor in our simulation. The maximum linear velocity of the robot was 1.2m/s and the maximum angular velocity was 2rad/s. A distance of 0.2m was set as the safety distance between the robot and the object. The robot was considered to reach the goal if its distance to the goal is within 0.2m.

The pedestrian trajectories were loaded into the environment together with the robot. In this work, the motion goal of the robot is to navigate from its initial position to the goal position without interfering with humans' motion, thus pedestrians' motion is not affected by the robot. We marked two areas, Area A (of approximate size 57x15 m^2) and Area B (of approximate size 30x30 m^2), in the simulation as shown in Fig. 4. We trained the neural networks in Area A only and tested them in both Areas A and B for validation of generalizability.

During training, the robot was set at a random position in the initial position area of size 5x5 m^2 in Area A. The goal position was randomly generated in the goal position area of size 5x5 m^2 located in the other end of Area A, as marked in Fig. 4. Algorithm 1 was run and the training takes 5000 episodes.

During testing, we ran Algorithm 2 in both Areas A and B. The robot started from one point in the initial position area and navigated to a goal position that was randomly selected in the goal position area. For performance evaluation in environments of various complexity, we selected a "low density" scenario and a "high density" scenario, where a maximum of 15 pedestrians were allowed in the "low density" scenario and a maximum of 30 pedestrians were allowed in the "high density" scenario.

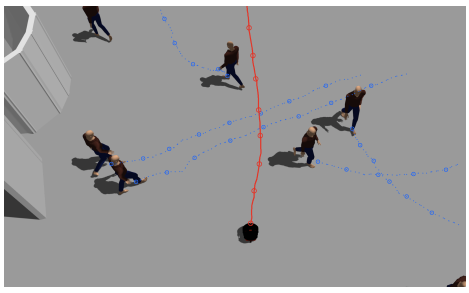


Fig. 5: Robot trajectory (in red) and pedestrian trajectory (in blue). The circles on the trajectory were drawn every second.

B. Performance Metrics

We use the following metrics to evaluate the performance:

- *Success rate (SR)* is defined as the ratio of the number of successful cases (where the robot reaches the goal) over the total number of tested cases. Collisions or timeout (defined as exceeding the maximum allowed time to complete the task) are counted as unsuccessful cases.
- *The number of times freezing* is defined as the number of times that the robot was freezing at its place without a decision of action for 5 seconds.
- *Normalized path length (NPL)* is defined as the ratio between the actual robot path length (PL) over the straight-line distance between the robot start and the goal positions.

C. Performance Evaluation

We performed simulation tests in Areas A and B, respectively. In each of the two areas, we tested with the "low density" condition and the "high density" condition, respectively, as defined in Section IV-A. We run 50 testing cases in each of the four environmental settings. We compare our DRL-based method with a few other existing methods:

- The "move_base (MB)" package in ROS serves as a baseline comparison, representing a classic navigation method. It offers an implementation of the widely used Dynamic Window Approach (DWA) [6] and uses a local planner and global planner to guide the robot to its goal. The local planner uses the DWA algorithm to compute feasible velocity commands that avoid obstacles, and the global planner plans a high-level path based on a costmap to drive the robot to reach the goal.
- The DDPG [11] method uses the DDPG architecture and Adam Optimizer for learning the parameters of the actor and critic networks. We used the same structure and parameters as in our proposed method except for the global guiding points and trained the model in Area A of our simulation environment.
- The GAIL [8] method learns human navigation behaviors using an imitation learning approach based on the generative adversarial imitation learning architecture. The GAIL model was trained in Area A only.

We report both robot trajectory performance and statistical results in the next.

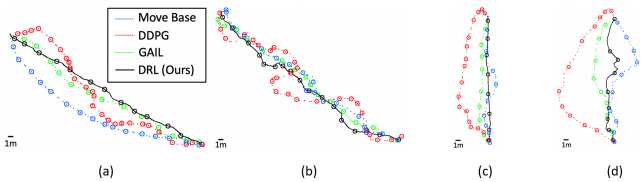


Fig. 6: Comparison of robot trajectories generated using different methods in the environmental setting: (a) Area A, low-density condition; (b) Area A, high-density condition; (c) Area B, low-density condition; (d) Area B, high-density condition. The circles on the trajectories were drawn every 4 seconds.

1) Robot Navigation Trajectories: Fig. 5 visualizes a scenario of a robot navigating among pedestrians. We can see that the robot avoids potential collisions with surrounding moving pedestrians and finds a path through them.

Fig. 6 compares the paths taken by robots using different navigation methods. Although all methods successfully guide the robot to the goal, there are significant differences in the trajectories. The sub-figures (a) and (b) show the testing results in Area A (the same environment used to train the neural network model) with low-density and high-density

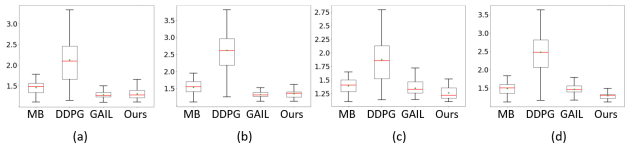


Fig. 7: Boxplot of the normalized path length of robot trajectories generated using different methods in the environmental setting: (a) Area A, low-density condition; (b) Area A, high-density condition; (c) Area B, low-density condition;(d) Area B, high-density condition. The central line shows the medians in each box, the edges of the boxes are the 25th and 75th percentiles, the whiskers extend to the maximum and minimum, and the dot in the box represents the mean in each boxplot.

conditions, respectively. The sub-figures (c) and (d) show the testing results in Area B, which were not seen by the robot as Area B was not used in training.

Comparing different methods, our proposed method can efficiently complete the navigation task in both trained and untrained environments. The planner of Move base tends to re-plan the path or detour a long way when the environment is crowded. The planner of DDPG can efficiently avoid potential collisions, but its planned trajectories are longer and it takes the robot longer to reach the goal. GAIL has good performance in Area A but does not perform well in Area B due to imitation learning that lacks generalizability.

2) Statistical Results: We run 50 testing cases in each of the two areas with two different crowd conditions. We use the metrics defined in IV-B to compare the performance of different methods.

Tables II and III show the statistical result of navigation time, NPL, success rate, and the number of times freezing of different methods with low and high density conditions in Areas A and B, respectively. In Area A, the mean navigation time (i.e., navigation time averaged over 50 runs) of GAIL is 52.52 and 1.30 mean NPL with 94% success rate in low-density crowds, and the mean navigation time of our method is 50.48 and 1.32 mean NPL with 92% success rate. The mean navigation time of GAIL is 59.13 and 1.33 mean NPL with 92% success rate in high-density crowds, and the mean navigation time of our method is 65.92 and 1.37 mean NPL with 84% success rate. We can see that GAIL has the best performance in the trained environment. Our method has the best performance in Area B, the untrained environment, with with 94% success rate in low-density crowds and 92% success rate in high-density crowds. The mean navigation time is 29.56 and NPL is 1.27 in low-density crowds and 32.44 and 1.31 in high-density crowds. Thus, our proposed method is quantitatively similar to the GAIL method in Area A and has the best performance in Area B. It demonstrates the generalizability of our DRL-based method.

The boxplot of the NPL is shown in Fig. 7 for each of the four testing scenarios (i.e., Areas A and B, low or high-density crowds). We can see that in Area A as shown in Fig. 7 (a) and (b), GAIL and our method have similar performances. In Area B as shown in Fig. 7 (c) and (d), our method has the best performance in terms of NPL. This is consistent with the other two metrics of success rate and the number of times freezing as shown above, and demonstrates the generalizability of our method.

V. Discussion and Limitation

The fundamental contribution of the paper is the development of a new DRL-based robot navigation algorithm. Compared with other recent work on robot navigation in crowded environments(e.g., [17]–[24]), we proposed an end-to-end approach that takes the robot Lidar scan as input and outputs the robot control commands. Also, we used an open dataset with real-world pedestrian trajectories to train the DRL algorithm. Existing works use random or model-based simulated pedestrian data for training and/or simulation validation, for example, PedSim simulator generated data were used in [22], the social force model-based pedestrian movement was used in [18], simulated pedestrian trajectories were generated by a model consistent with the fundamental diagram based on physiological and psychological factors in [21], ad-hoc or randomly spawned obstacle scenarios were used in [23], [24]. As it is well known, training data is important to reinforcement learning methods, and the performance largely depends on the quality and authenticity of the training data. Our algorithm was trained and validated using real-world pedestrian data, thus presenting more authentic pedestrian motion behaviors in real-world crowd environments.

It is worth noting that we do not consider complex human-robot interaction behaviors. Among existing works on robot navigation in crowded environments, some works resolved the freezing robot problem by accounting for human-robot cooperation, that is, the robot and the humans adjusted their trajectories cooperatively. Some DRL methods model cooperative behaviors between humans and robots implicitly or explicitly. However, for many applications, such as package delivery or helpers in hotels, the robot needs to navigate through dense crowd environments without collision, and at the same time, minimize the effect of its presence on the humans (i.e., be unobtrusive to the nearby pedestrians). This is the case studied in our paper. In other words, we aimed to validate the hypothesis that our DRL-based learning algorithm (trained using real-world pedestrian trajectories) can navigate in dense crowd environments without human-robot collaboration. A similar experiment setup was presented in [21], where the goal is to solve the freezing robot problem and to ensure the robot is unobtrusive to nearby pedestrians. Our method is complementary to the hybrid method proposed therein, and we demonstrated that a well-trained robot can move with the pedestrian flow smoothly without human-robot collaboration.

VI. Conclusion and Future Work

This paper presented an algorithm using DDPG combined with global guiding points to complete the robot navigation task in human crowd environments. We used an existing open human trajectory dataset and built a 3D simulation

TABLE II: Summary statistics for navigation time and NPL. The mean value and standard derivation (SD) are shown for 50 trajectories.

Method	Area A with Low Density		Area A with High Density		Area B with Low Density		Area B with High Density	
	Time(SD)[s]	NPL(SD)[m]	Time(SD)[s]	NPL(SD)[m]	Time(SD)[s]	NPL(SD)[m]	Time(SD)[s]	NPL(SD)[m]
Move_base	68.84 (6.73)	1.46 (0.10)	73.08 (8.1)	1.54 (0.13)	35.8 (3.61)	1.40 (0.06)	36.12 (3.08)	1.49 (0.08)
DDPG [11]	105.32 (18.45)	2.13 (0.43)	108.84 (11.34)	2.63 (0.55)	53.10 (7.04)	1.88 (0.28)	67.08 (9.11)	2.49 (0.50)
GAIL [8]	52.52 (5.64)	**1.30 (0.08)**	**59.13 (5.77)**	**1.33 (0.09)**	34.68 (3.75)	1.36 (0.06)	36.40 (3.21)	1.46 (0.04)
DRL (Ours)	**50.48 (4.54)**	1.32 (0.05)	65.92 (5.75)	1.37 (0.07)	**29.56 (2.81)**	**1.27 (0.04)**	**32.44 (3.80))**	**1.31 (0.04)**

TABLE III: Performance comparison in terms of success rate (SR) and the number of times freezing.

Method	Area A Low Density		Area A High Density		Area B Low Density		Area B High Density	
	SR[%]	# freezing	SR[%]	# freezing	SR[%]	# freezing	SR[%]	# freezing
Move_base	78	3	64	13	84	1	70	8
DDPG	32	0	24	0	40	0	28	0
GAIL	**94**	0	**92**	0	82	0	80	0
DRL (Ours)	92	0	84	0	**94**	0	**92**	0

environment in ROS Gazebo. We placed a mobile robot equipped with an onboard Lidar sensor into the simulation environment, and the robot was controlled by the trained neural network model to navigate to a given goal position in the crowd environments. The proposed method was evaluated and compared with other state-of-the-art algorithms quantitatively. Statistic results demonstrated the generalizability of our method. In the future, we plan to perform real robot experiments using our proposed method.

REFERENCES

[1] Y. Lee, S. Lee, and D.-Y. Kim, "Exploring hotel guests' perceptions of using robot assistants," *Tourism Management Perspectives*, vol. 37, p. 100781, 2021.

[2] R. Bogue, "Growth in e-commerce boosts innovation in the warehouse robot market," *Industrial Robot: An International Journal*, vol. 43, no. 6, pp. 583–587, 2016.

[3] A. Stentz *et al.*, "The focussed dˆ* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[4] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[5] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *IEEE International Conference on Robotics and Automation*, pp. 1986–1991, 2007.

[6] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[7] M. Fahad, Z. Chen, and Y. Guo, "Learning how pedestrians navigate: A deep inverse reinforcement learning approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 819–826, 2018.

[8] M. Fahad, G. Yang, and Y. Guo, "Learning human navigation behavior using measured human trajectories in crowded spaces," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 11154–11160, 2020.

[9] L. Tai, J. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *IEEE International Conference on Robotics and Automation*, pp. 1111–1117, 2018.

[10] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, May 1995.

[11] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *International Conference on Learning Representations*, pp. 1–14, 2016.

[12] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 31–36, 2017.

[13] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5671–5677, 2020.

[14] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *IEEE International Conference on Robotics and Automation*, pp. 6015–6022, 2019.

[15] D. Brščić, T. Kanda, T. Ikeda, and T. Miyashita, "Person tracking in large public spaces using 3-d range sensors," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 6, pp. 522–534, 2013.

[16] H. Gong, P. Wang, C. Ni, and N. Cheng, "Efficient path planning for mobile robot based on deep deterministic policy gradient," *Sensors*, vol. 22, no. 9, pp. 3579–3598, 2022.

[17] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1343–1350, IEEE, 2017.

[18] R. Guldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6053–6060, IEEE, 2020.

[19] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," in *IEEE International Conference on Robotics and Automation*, pp. 7829–7835, IEEE, 2021.

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

[21] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[22] L. Kästner, B. Fatloun, Z. Shen, D. Gawrisch, and J. Lambrecht, "Human-following and-guiding in crowded environments using semantic deep-reinforcement-learning for mobile service robots," in *International Conference on Robotics and Automation*, pp. 833–839, IEEE, 2022.

[23] L. Kästner, M. Meusel, T. Bhuiyan, and J. Lambrecht, "Holistic deep-reinforcement-learning-based training for autonomous navigation in crowded environments," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 1302–1308, IEEE, 2023.

[24] Q. Qiu, S. Yao, J. Wang, J. Ma, G. Chen, and J. Ji, "Learning to socially navigate in pedestrian-rich environments with interaction capacity," in *International Conference on Robotics and Automation*, pp. 279–285, IEEE, 2022.