DRL-VO: Using Velocity Obstacles to Learn Safe and Fast Navigation

Zhanteng Xie and Philip Dames

***Abstract*— This paper presents a novel deep reinforcement learning-based control policy to enable a mobile robot to navigate safely and quickly through complex and human-filled environments. The policy uses a short history of lidar data, the kinematic data about nearby pedestrians, and a sub-goal point as its input, using an early fusion network to fuse these data. The reward function uses velocity obstacles to guide the robot to actively avoid pedestrians and move towards the goal. Through a series of simulated environments with 5-55 pedestrians, this policy is able to achieve a 31.6% higher success rate and 86.9% faster average speed than state-of-the-art model-based and learning-based policies. Hardware experiments demonstrate the ability of the policy to directly work in real-world environments.**

I. Introduction

One common application of autonomous mobile robots is replacing manual labor to provide indoor delivery services. For example, delivering sterile supplies and injection medicines to patients in hospitals, and delivering delicious food to customers in restaurants. Both tasks have time limits and require mobile robots to navigate safely and quickly to destinations through a partially known space filled with moving people and other robots, as shown in Fig. 1. The main challenges faced by these mobile robots are perceiving complex environments, especially unknown and dynamic pedestrians; extracting useful information; and generating a safe and fast navigation policy.

From traditional model-based approaches to supervised learning-based approaches to reinforcement learning-based approaches, there are many published studies that focus on robot navigation problems. Typical model-based approaches compute efficient paths and the parameters are easily interpretable, but they require manually adjusting model parameters for different scenarios [1]–[4]. Supervised learning-based approaches are purely data driven and easy to use, but require laboriously collecting a representative set of expert demonstrations to train the network [5]–[8]. Reinforcement learning-based approaches are experience-driven and similar to human learning, but require carefully designing a reward function [9]–[19]. In this paper, we use a novel deep reinforcement learning (DRL) approach due to the complexity of the environments we want the robot to operate in, where there are 10's of pedestrians moving in dense crowds. This makes it difficult to design clear rules within a model-based framework and to collect sufficient training data in a supervised learning setting.

*This work was funded by the Amazon Research Awards Program and NSF grant IIS-1830419.

Zhanteng Xie and Philip Dames are with the Department of Mechanical Engineering, Temple University, Philadelphia, PA, USA {zhanteng.xie, pdames}@temple.edu

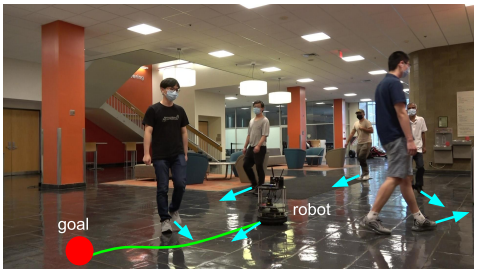


Fig. 1. A simple illustration of robot navigation problem. The green line denotes the nominal path to the goal, the red circle denotes the goal position, and the blue arrows denote the velocities.

With any of the different approaches to autonomous navigation, one key set of issues is which sensor data to use, how to process it, and which data representations to use. The dynamic window approach (DWA) [1] extracts the obstacle line field from lidars to construct dynamic local world model. The velocity obstacle (VO) based algorithms [2], [3] estimate the relative velocity of obstacles and map the dynamic environment into the robot velocity space. The model predictive control algorithm [4] incorporates the static obstacle information from an occupancy grid map and the dynamic obstacle motion from the Kalman filter [20] into robot constraints. End-to-End learning approaches focusing on static environments directly feed the raw sensor data, *e.g.,* lidars [5], [6] or cameras [7], and goal information into the convolution neural networks (CNN). While some DRL-based approaches [10], [11], [18] also feed raw lidar scans and its goal points into CNNs, all others prefer to extract and utilize pedestrian trajectory predictions [12], [13], robot-human interactions [9], [14]–[16], reconstructed lidar predictions [17], or obstacle cost matrices [19]. Most of these studies demonstrate that preprocessed data representations contain more useful information than the raw sensor data, and that information about pedestrian motion improves safety in dynamic environments. Based on these trends, we will also utilize preprocessed sensor data, namely a short history of lidar data and the current kinematics of pedestrians, which we previously showed was able to improve navigation safety in a supervised setting [8]. One advantage of this data is that it is much simpler than the complex pedestrian predictions or interactions used in other works.

One of the biggest challenges in using DRL is designing an appropriate reward function. Most approaches include two terms, one to reward progress towards the goal and one to penalize collisions [9]–[18]. Although this simple reward function design can work well, their sparse collision avoidance reward function does not give a direct and effective

reward signal to guide the robot to actively avoid obstacles. In fact, their passive collision avoidance reward function is a zero-order function, meaning it assumes the risk of collision depends only on the distance to the obstacle. We argue that the *relative motion* is more important since people often follow one another closely in dense crowds and give more space when walking in opposite directions. Towards this, Patel et al. [19] propose a square warning region-based reward function to encourage the robot to navigate in the direction opposite to the heading direction of obstacles. While this prioritizes safety, it leads to slower motion and can cause deadlock in dense crowds. We propose a novel first-order velocity obstacle-based heading direction reward function that is able to guide the robot to continuously tune its heading direction to actively avoid crowded pedestrians while moving towards to the goal.

The primary contribution of this paper is extending the concept of velocity obstacles to the DRL-based navigation policy in order to guide the robot to learn safe and fast navigation behaviors. Our proposed control policy only requires the preprocessed lidar history, the current pedestrian kinematics, and a sub-goal point. With the help of our novel VO-based reward function, the robot is able to tune its head direction to a desired direction that moves towards the goal while also avoiding collisions. We validate the navigation performance in our 3D simulation environments and demonstrate that the proposed control policy is safer, faster, and generalizes to crowd sizes better than state-of-art approaches, including model-based controller [1], supervised learning-based approach [8], and DRL-based approaches [11]. We also demonstrate the real-world applicability of our DRL-based policy through a set of hardware experiments.

II. Safe and Fast Navigation Policy

This section begins by formulating the safe and fast navigation problem through dynamic environments. It will then go on to our DRL-based approach, focusing on describing the preprocessed observation space, the DRL network architecture, and the novel VO-based reward function design.

A. Problem Formulation

Due to the limited field of view (FOV) of sensors (*e.g.,* lidars and cameras), there is no complete environmental perception for the robot in the real world. Extracting and processing useful information from sensors to obtain the partial observation of the environment $\mathbf{o^t}$ is the first step for the robot to safely and quickly navigate through complex and human-filled environments. The robot then feeds this partial observation $\mathbf{o^t}$ into a control policy π_θ to compute the suitable steering action $\mathbf{a^t}$, which is defined as

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

where θ are our control policy parameters. This complicate navigation decision-making process can be formulated by a large partially observable Markov decision process (POMDP), which can be denoted by a 6-tuple $\langle S, A, T, R, \Omega, O\rangle$ where S is the state space, A is the action space, T is the state-transition model, R is the reward function, Ω is the observation space, and O is the observation probability distribution. One of the most well-known tools for assessing such a large POMDP is the DRL-based approach.

B. Observation Space

The partial observation $\mathbf{o^t} = [\mathbf{r^t}, \mathbf{p^t}, \mathbf{g^t}]$ in our control policy is from our previous work [8], consisting of three components: lidar history ($\mathbf{r^t}$), pedestrian kinematics ($\mathbf{p^t}$), and the sub-goal position ($\mathbf{g^t}$). All data in the observation $\mathbf{o^t}$ are the preprocessed data and we express them in the local reference frame of the robot. Note: Our previous work [8] illuminates detailed preprocessing procedures. This subsection only gives them a brief description.

1) Pedestrian Kinematics: There are three steps to generate the pedestrian kinematic maps $\mathbf{p^t}$. To begin this process, we feed both the RGB image data and its corresponding 3-D point cloud data from a depth camera into the YOLOv3 [21] detector to detect pedestrians and extract their relative locations. Once these resulting instantaneous estimates are extracted, we use a multiple hypothesis tracker (MHT) [22] to track their *relative* positions and velocities. Finally, instead of using the raw output from the MHT, we encode the pedestrian kinematics into occupancy grid-style maps specifically designed for our DRL network, as Fig. 2(a) shows.

2) Lidar History: Prior to data preprocessing, we collect 0.5 s of lidar data (*i.e.,* 10 scans) as we found this to give more useful information than one single scan. Different from other works that use raw lidar scan data [5], [6], [10], [11], [18], we use a combination of minimum pooling and average pooling to downsample each individual lidar scan and create a feature map. To keep a uniformly-sized input to our early fusion network, we stack this downsampled historical data 4 times to create an 80×80 grid, as shown in Fig. 2(b).

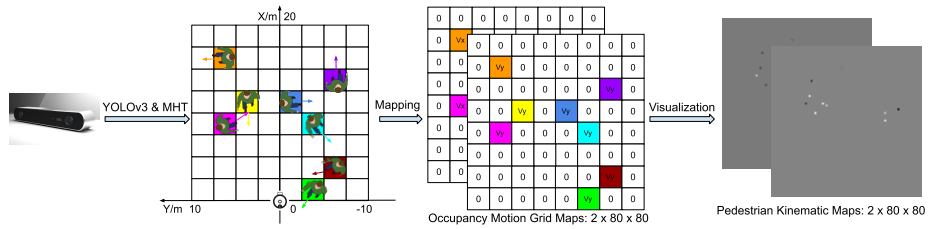
3) Goal Position: Lastly, the robot needs to know where the relative position of the goal. Instead of feeding the final goal point to our DRL network, we use the sub-goal point along a nominal path to the final goal as this allows the robot to traverse long paths through complex, non-convex environments. To select the sub-goal, we use the pure pursuit algorithm [23], which calculates the point along the full path that is 2 m ahead of the robot. By tuning this distance, the robot can look far ahead or right nearby.

C. Action Space

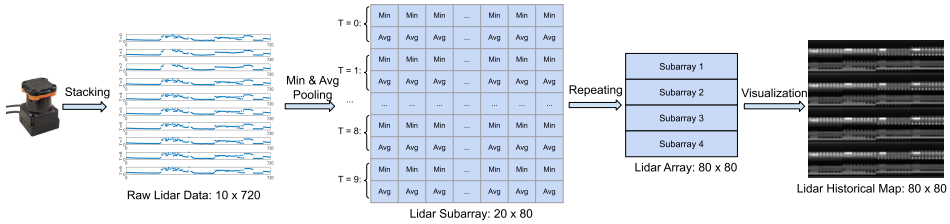
The action $\mathbf{a^t} = [v_x^t, w_z^t]$ of our DRL control policy are the steering velocities, where v_x^t is the translational velocity and w_z^t is the rotational velocity in the local robot frame. Note, we use a continuous action space as we believe this gives the robot a more accurate control. The range of the translational velocity v_x^t is set to $[0, 0.5]$ m/s, and the range of the rotational velocity w_z^t is set to $[-2, 2]$ rad/s.

D. Network Architecture

To represent the parametric control policy π_θ, we construct a deep reinforcement learning network, outlined in Fig. 3,



(a) Pedestrian kinematic maps



(b) Lidar historical map

Fig. 2. The illustration of pedestrian kinematic maps and lidar historical map. Note: We normalize the all data to $[-1, 1]$ before feeding them into our network. (a) The pedestrian kinematics are generated by the YOLOv3 detector and MHT tracker. We map these kinematic estimates in a total area of $20 \times 20\,\mathrm{m}$ into two 80×80 occupancy grid maps. The grid cell ID is determined by the pedestrian position and the grid cell value is given by its corresponding velocity. (b) We first obtain 10 raw lidar scans with 720 measurements per scan from the lidar sensor. Then, a compound pooling operation is used to downsample each individual lidar scan and get a 20×80 subarray. The final compatible lidar historical map (80×80) is generated by stacking this subarray four times.

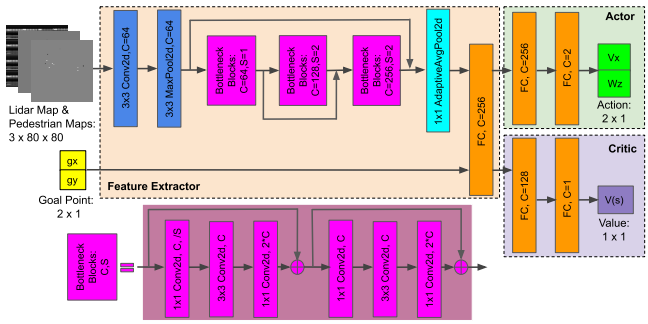


Fig. 3. The architecture of our DRL network. Three 80×80 low-level feature maps, one from lidar and two containing pedestrian information, along with the relative position of the sub-goal are fed to the feature extractor network, and the resulting high-level features are fed to the actor and critic networks separately. The critic network outputs the state value and the actor network outputs the linear and angular steering velocities. "3×3" or "1×1" denote the kernel size, and "C" and "S" denote the number of output channels and the stride length, respectively.

with three modules: feature extractor, actor, and critic. The feature extractor module is used to extract high-level features from the low-level partial observation $\mathbf{o^t}$, the critic module generates the state value, and the actor module generates steering action $\mathbf{a^t}$. The feature extractor network is identical to the backbone network of our previous work [8], which uses an early fusion architecture to fuse the lidar historical observation $\mathbf{r^t}$ and pedestrian kinematics observation $\mathbf{p^t}$.

We use the proximal policy optimization (PPO) algorithm [24] to train our DRL network, and use Adam optimizer [25], a stochastic gradient descent method, to find the optimal policy parameters θ^*.

E. Reward Function

An essential problem of reinforcement learning methods is how to design a good reward function to guide the agent to learn desired behaviours. Navigation task has two competing objectives, we want the robot to move as quickly as possible to reach the goal in minimum time but also safely to avoid colliding with any stationary objects or moving pedestrians. Thus, we design a multi-objective reward function:

$$r^t = r_g^t + r_c^t + r_w^t + r_d^t \tag{2}$$

where r_g^t is a reward for reaching the goal, r_c^t is a reward to penalize passively approaching or colliding with an obstacle, r_w^t is a reward to penalize rapid changes in direction, and r_d^t is a reward for tuning heading direction to actively avoid obstacles and move in the direction of the goal.

Note that most previous works [9]–[18] only use such a zero-order passive collision avoidance reward r_c^t, a position-based heuristics, to guide robots to passively avoid obstacles. A key contribution of our work is that we design an additional first-order active heading direction reward r_d^t, a relative velocity-based heuristics, to guide robots not only to actively avoid obstacles but also to move towards to the goal.

1) Reaching the Goal: The reward is given by

$$r_g^t = \begin{cases} r_{\text{goal}} & \text{if } d(p_r^t, p_g) < g_m \\ -r_{\text{goal}} & \text{else if } t \geq t_{\max} \\ r_{\text{path}}(d(p_r^{t-1}, p_g) - d(p_r^t, p_g)) & \text{otherwise} \end{cases} \tag{3}$$

where p_r^t is the position of the robot at time t, p_g is the position of the goal, and $d(\cdot, \cdot)$ is the distance between two points. We use $r_{\text{goal}} = 20$, $r_{\text{path}} = 3.2$, $g_m = 0.3\,\mathrm{m}$, and $t_{\max} = 25\,\mathrm{s}$.

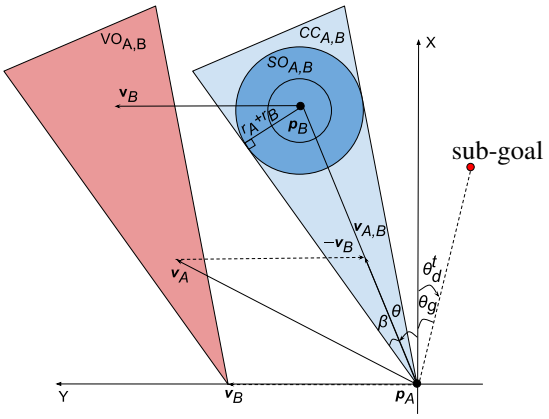


Fig. 4. The pictogram for velocity obstacle $VO_{A,B}$ (red cone), collision cone $CC_{A,B}$ (light blue cone), and special occupancy $SO_{A,B}$ (blue circle). A represents the robot with radius r_A, position $\mathbf{p}_A$, and velocities $\mathbf{v}_A$. B represents the moving pedestrian with radius r_B, position $\mathbf{p}_B$, and velocities $\mathbf{v}_B$. $\mathbf{v}_{A,B} = \mathbf{v}_A - \mathbf{v}_B$ represents the relative velocity of A with respect to B. Note that all quantities are in the robot's local frame.

2) Passive Collision Avoidance: The reward is given by

$$r_c^t = \begin{cases} r_{\text{collision}} & \text{if } d(p_r^t, p_o^t) \le d_r \\ r_{\text{obstacle}}(d_m - d(p_r^t, p_o^t)) & \text{else if } d(p_r^t, p_o^t) \le d_m \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where p_o^t is the position of the obstacle at time t. We use $r_{\text{collision}} = -20$, $r_{\text{obstacle}} = -0.2$, $d_r = 0.3\,\text{m}$, and $d_m = 1.2\,\text{m}$.

3) Path Smoothness: The reward is given by

$$r_w^t = \begin{cases} r_{\text{rotation}}|w_z^t| & \text{if } |w_z^t| > w_m \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $r_{\text{rotation}} = -0.1$ and $w_m = 1\,\text{rad/s}$.

4) Active Heading Direction: The reward is given by

$$r_d^t = r_{\text{angle}}(\theta_m - |\theta_d^t|) \tag{6}$$

where θ_d^t is the desired heading direction in the robot's local frame and θ_m is the maximum allowable deviation of the heading direction. We use $r_{\text{angle}} = 0.6$ and $\theta_m = \frac{\pi}{6}\,\text{rad}$.

The key to this reward term is to find the desired direction θ_d^t that moves towards the goal while also being collision free. To do this, we extend the concept of velocity obstacles [2]. The velocity obstacle $VO_{A,B}$ is the velocity space where the current velocity of robot A would cause a collision with an obstacle B at some future time, as Fig. 4 shows. To define the velocity obstacle $VO_{A,B}$, we first need to define the special occupancy $SO_{A,B}$ and collision cone $CC_{A,B}$. The special occupancy $SO_{A,B}$ is a open disc area with radius $r_A + r_B$ centered at $\mathbf{p}_B$, which is defined as

$$SO_{A,B} = \{\mathbf{p}_S \mid d(\mathbf{p}_S, \mathbf{p}_B) < r_A + r_B\}. \tag{7}$$

Then, the collision cone is defined as:

$$CC_{A,B} = \{\mathbf{v}_{A,B} \mid \exists t, \mathbf{p}_A + \mathbf{v}_{A,B}t \cap SO_{A,B} \neq \emptyset\} \tag{8}$$

Finally, the velocity obstacle $VO_{A,B}$ is defined as:

$$VO_{A,B} = CC_{A,B} \oplus \mathbf{v}_B \tag{9}$$

where $\oplus$ denotes the Minkowski vector sum operator.

The physical meaning of $CC_{A,B}$ is that any relative velocities $\mathbf{v}_{A,B} \in CC_{A,B}$ will cause a collision at future time. From the perspective of the direction angle of $\mathbf{v}_{A,B}$, the collision cone $CC_{A,B}$ can be defined as:

$$CC_{A,B} \in [\theta - \beta, \theta + \beta] \tag{10}$$

where θ and β can be easily calculated from Fig. 4 using simple geometric relationships:

$$\theta = \arctan 2\left(\frac{\mathbf{p}_{B_y} - \mathbf{p}_{A_y}}{\mathbf{p}_{B_x} - \mathbf{p}_{A_x}}\right) \tag{11}$$

$$\sin\beta = \frac{r_A + r_B}{d(\mathbf{p}_A, \mathbf{p}_B)} \tag{12}$$

Using the collision cone $CC_{A,B}$, the robot can know which heading direction angles will cause collisions with all of the moving pedestrians tracked by MHT. The robot then uses these collision cones and the direction of the sub-goal (θ_g) to find the desired heading direction angle θ_d^t using Algorithm 1. This sampling-based search algorithm is motivated by [3].

Algorithm 1: Search desired direction angle

Input: Sub-goal direction angle θ_g, pedestrians from MHT B_{peds}, robot linear velocity $\mathbf{v}_{A_x}$, number of samples N

Output: Optimal direction angle θ_d^t

initialize: $\theta_d^t \leftarrow \frac{\pi}{2}$
if $B_{\text{peds}} \neq \emptyset$ **then**
  $\theta_{\min} \leftarrow \infty$
  for $i = 1, 2, \ldots, N$ **do**
    $\theta_u \leftarrow$ sample from $[-\pi, \pi]$
    free $\leftarrow$ true
    for B *in* B_{peds} **do**
      $\theta_{v_{A,B}} \leftarrow \text{atan2}\left(\frac{\mathbf{v}_{A_x}\sin(\theta_u) - \mathbf{v}_{B_y}}{\mathbf{v}_{A_x}\cos(\theta_u) - \mathbf{v}_{B_x}}\right)$
      if $\theta_{v_{A,B}} \in [\theta - \beta, \theta + \beta]$ **then**
        free $\leftarrow$ false
        break
    if free **then**
      if $\|\theta_u - \theta_g\| < \theta_{\min}$ **then**
        $\theta_{\min} \leftarrow \|\theta_u - \theta_g\|$
        $\theta_d^t \leftarrow \theta_u$
else
  $\theta_d^t \leftarrow \theta_g$
return θ_d^t

III. Results

To demonstrate the efficacy and performance of our proposed control policy, we first use Gazebo simulator [26] and PEDSIM library [27] to conduct a set of simulated experiments, and then use a Turtlebot 2 robot to conduct the real-world experiments. This section describes the experimental setup, training procedure, and results.

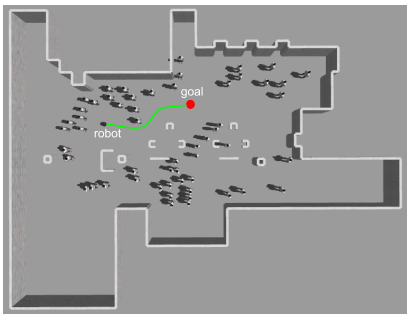


Fig. 5. The lobby simulation environment with 55 pedestrians.

A. Experimental Setup

1) Robot Configuration: For the simulated and hardware tests, we use a Turtlebot 2 robot with a Stereolabs ZED stereo camera and a Hokuyo UTM-30LX lidar. The maximum velocity of the our Turtlebot2 is limited to 0.5 m/s. The depth range of ZED camera is set to $[0.3, 20]$ m, and its FOV is $90°$. The measurement range of Hokuyo lidar is set to $[0.1, 30]$ m, its FOV is $270°$, and its angular resolution is $0.25°$. In the hardware tests, the Turtlebot is equipped with an NVIDIA Jetson TX2 embedded computer.

2) Simulation Configuration: Figure 5 shows the main Gazebo simulation environment, which is a replica of the lobby in our engineering building measuring approximately 25×10 m. This lobby environment simulates the Turtlebot 2 robot, static obstacles (*e.g.,* walls, pillars, chairs and tables), and dynamic pedestrians. The pedestrians in Gazebo are driven by the PEDSIM library, which uses the social force model [28] to simulate pedestrian motions.

We use an Nvidia DGX-1 server with 40 CPU cores, 8x Tesla V100 GPUs with 16 GB memory, and 512 GB of RAM to train our DRL networks. We use a desktop computer with an Intel i7-6800K CPU, a GeForce GTX 1080 GPU with 8 GB memory, and 16 GB of RAM to run all the simulations. Both training server and desktop computer run Ubuntu 20.04 system, ROS Noetic Ninjemys, and Gazebo 11.5.1.

B. Training Procedure

The stable-baselines3 framework [29] is used to implement and train our DRL networks. To train our control policies, we use the lobby environment with 34 pedestrians in it. The robot is then repeatedly assigned to reach a random goal from a random start position within the free space of the map. We used this procedure to train two different DRL-based control policies, one using the full reward (2) (DRL-VO) and one that does not use the heading direction reward r_d^t (DRL).

C. Simulation Results

We test these two DRL-based policies (*i.e.,* DRL and DRL-VO), along with other's DRL-based policies [11] (*i.e.,* A1-RD and A1-RC), the CNN-based policy [8], and the DWA planner [1], in the lobby environment with 5-55 (sampling interval 10) pedestrians in it. Four most commonly used metrics are used to evaluate their performance:

- **Success rate**: the fraction of collision-free trials.
- **Average time**: the average travel time of trials.
- **Average length**: the average trajectory length of trials.
- **Average speed**: the average speed during trials.

For each experimental configuration and control policy, we run total 4 trials with the same initial setting. Each trial consists of the robot navigating through a predefined series of 25 goal points around the lobby. Note, despite the same initial conditions, the results vary due to sensor noise and variations in the pedestrian motion from the social force model.

Table I presents the results obtained from these trials, where we observe three key phenomena. First, our DRL-VO policy has both a higher success rate and a higher average speed than our DRL policy in each crowd size. This shows that the proposed VO-based heading direction reward is beneficial and plays a key role in enabling the robot to learn safe and fast navigation behaviors. Second, compared with other control policies, our DRL-VO policy has the highest success rate and the fastest average speed in every situation, with these differences growing as the density of pedestrians increases. This indicates that our policy is able to better generalize than the other approaches, especially compared to the CNN-based policy [8], which could not even reach the goal in some situations. Third, the average speed of the robot using our DRL-VO policy remains nearly constant across all situations, regardless of crowd density, allowing it to reach the goal most quickly. Compared to DWA [1], we see that our planner takes a slightly longer route to the goal, indicating a preference for giving a wider berth to pedestrians.

In addition, what stands out in our simulated experiments is the difference of navigation behavior between the A1-RD policy [11] and our DRL-VO policy, which can can see from the accompanying video. We found that the high success rate of A1-RD policy in less crowd densities is from its "stop and wait" strategy, which also leads a low average speed. When there are obstacles detected by the robot, the robot deployed A1-RD policy will always turn to a free space, stop, and wait for obstacles to disappear. At that time, the A1-RD policy does not care about the number of obstacles and whether these obstacles are static or dynamic. Furthermore, the A1-RD policy can easily become frozen when it encounters a "C-shaped" obstacle because it is a typical passive collision avoidance policy based on distance and also only use raw lidar data as observation. In contrast to this, our DRL-VO policy can distinguish between the static obstacles and dynamic pedestrians using the combination of lidar data and pedestrian kinematics. Moreover, depending on the crowd densities and velocity obstacle space, our DRL-VO policy will always tune its heading direction to actively avoid pedestrians and move towards to the goal point.

Although our proposed DRL-VO control policy is robust to different crowd densities, there are still some failure cases. One is that it can become difficult to find a collision-free direction in very high crowds densities. The other is the unexpected sudden changes in pedestrian movement, such as a person suddenly stepping out of a crowd. Our analysis of these failures are caused by two reasons: 1) our simulated pedestrians sometimes bump into one another due to the

TABLE I

NAVIGATION RESULTS AT DIFFERENT CROWD DENSITIES

Environment	Method	Success Rate	Average Time (s)	Average Length (m)	Average Speed (m/s)
Lobby world with 5 pedestrians	DWA [1]	0.930	12.100	**5.079**	0.420
	CNN [8]	-	-	-	-
	A1-RD [11]	-	-	-	-
	A1-RC [11]	0.910	14.111	6.144	0.435
	DRL	0.840	13.215	6.082	0.460
	DRL-VO	**0.940**	**11.408**	5.282	**0.463**
Lobby world with 15 pedestrians	DWA [1]	0.940	12.205	**5.085**	0.417
	CNN [8]	-	-	-	-
	A1-RD [11]	0.940	15.692	5.784	0.369
	A1-RC [11]	0.940	14.363	6.401	0.446
	DRL	0.800	13.655	6.241	0.457
	DRL-VO	**0.950**	**11.339**	5.262	**0.464**
Lobby world with 25 pedestrians	DWA [1]	0.820	13.488	**5.123**	0.380
	CNN [8]	0.800	19.309	6.155	0.318
	A1-RD [11]	0.900	17.866	6.070	0.340
	A1-RC [11]	0.880	14.175	6.256	0.441
	DRL	0.810	13.733	6.241	0.454
	DRL-VO	**0.920**	**11.372**	5.287	**0.465**
Lobby world with 35 pedestrians	DWA [1]	0.820	14.179	**5.148**	0.363
	CNN [8]	0.810	14.296	5.397	0.378
	A1-RD [11]	0.860	17.823	6.059	0.340
	A1-RC [11]	0.770	16.812	6.893	0.410
	DRL	0.750	14.297	6.461	0.451
	DRL-VO	**0.880**	**11.422**	5.305	**0.464**
Lobby world with 45 pedestrians	DWA [1]	0.770	15.386	**5.163**	0.336
	CNN [8]	0.790	16.645	5.615	0.337
	A1-RD [11]	0.760	23.160	6.612	0.285
	A1-RC [11]	0.770	14.650	6.281	0.429
	DRL	0.690	13.956	6.412	0.459
	DRL-VO	**0.810**	**11.645**	5.373	**0.461**
Lobby world with 55 pedestrians	DWA [1]	0.670	16.052	**5.215**	0.325
	CNN [8]	0.700	19.127	5.937	0.310
	A1-RD [11]	0.670	26.902	6.579	0.245
	A1-RC [11]	0.660	16.361	6.732	0.411
	DRL	0.600	13.392	6.132	0.458
	DRL-VO	**0.790**	**11.758**	5.389	**0.458**

imperfect social force model, and 2) missing pedestrian tracks in the MHT due to occlusion and ambiguous data association. To alleviate these issues, future work will continue training the policies (learned in simulation) in real-world environments and aim to improve the MHT performance.

D. Hardware Results

Besides the simulated experiments, we also conduct real-world experiments to demonstrate the applicability of our DRL-VO policy. The robot drives around the same lobby environment through a series of 10 goal points with many students naturally walking around the robot, as Fig. 1 shows. The real Turtlebot 2 robot uses our DRL-VO policy trained from our Gazebo simulation platform without any modification. From the accompany video, we can see that our robot is able to safely and quickly navigate, traveling a total of 74.868 m at an average speed of 0.403 m/s without any collisions in different crowd sizes. These real-world experiments demonstrate that our DRL-VO policy can smoothly transfer from simulation to reality and be applied to real-world tasks.

IV. CONCLUSION

In this paper, we proposed a novel DRL-VO control policy for the mobile robot to safely and quickly navigate through crowded environments with both static obstacles and dynamic pedestrians. The key strengths of this work are its preprocessed data representation and its VO-based reward function design. Specifically, the robot first feeds one lidar historical map, two current pedestrian kinematic maps, and a sub-goal point into a early fusion network. Then, a novel first-order reward function based on velocity obstacles guides the robot to learn safe and fast navigation behaviors. We demonstrate that our DRL-VO policy generalizes better to different crowd sizes (*i.e.,* has both a significantly higher success rate and average speed) than other state-of-the-art policies (*e.g.,* the model-based planner, supervised learning-based policy, and DRL-based policies) through a series of simulated experiments. Furthermore, our proposed DRL-VO policy also shows its real-world applicability through hardware experiments.

ACKNOWLEDGMENT

This research includes calculations carried out on HPC resources supported in part by the National Science Foundation through major research instrumentation grant number 1625061 and by the US Army Research Laboratory under contract number W911NF-16-2-0189.

REFERENCES

[1] 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.

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

[3] D. Wilkie, J. Van Den Berg, and D. Manocha, "Generalized velocity obstacles," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 5573–5578.

[4] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[5] L. Tai, S. Li, and M. Liu, "A deep-network solution towards model-less obstacle avoidance," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 2759–2764.

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

[7] A. Loquercio, A. I. Maqueda, C. R. Del-Blanco, and D. Scaramuzza, "DroNet: Learning to fly by driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1088–1095, 2018.

[8] Z. Xie, P. Xin, and P. Dames, "Towards safe navigation through crowded dynamic environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Sep. 2021, accepted.

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

[10] 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.

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

[12] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 11 345–11 352.

[13] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 007–10 013.

[14] 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.

[15] 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 (IROS)*, 2020.

[16] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, accepted.

[17] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," *arXiv preprint arXiv:2012.04406*, 2020.

[18] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, accepted.

[19] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "Dwa-rl: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, accepted.

[20] G. Welch and G. Bishop, "An introduction to the Kalman filter," Department of Computer Science, University of North Carolina at Chapel Hill, Tech. Rep., 1995.

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

[22] K. Yoon, Y.-m. Song, and M. Jeon, "Multiple hypothesis tracking algorithm for multi-target multi-camera tracking with disjoint views," *IET Image Processing*, vol. 12, no. 7, pp. 1175–1184, 2018.

[23] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm," Carnegie-Mellon UNIV Pittsburgh PA Robotics INST, Tech. Rep., 1992.

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

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

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

[27] C. Gloor, "PEDSIM: Pedestrian crowd simulation," *URL http://pedsim.silmaril.org*, vol. 5, no. 1, 2016.

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

[29] A. Raffin, A. Hill, M. Ernestus, A. Gleave, A. Kanervisto, and N. Dormann, "Stable baselines3," https://github.com/DLR-RM/stable-baselines3, 2019.