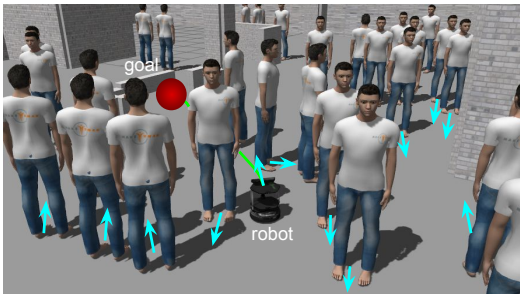
DRL-VO: Learning to Navigate Through Crowded Dynamic Scenes Using Velocity Obstacles

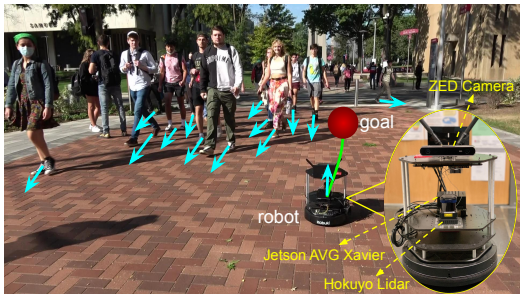
Zhanteng Xie, *Student Member, IEEE*, and Philip Dames, *Member, IEEE*

***Abstract*—This paper proposes a novel learning-based control policy with strong generalizability to new environments that enables a mobile robot to navigate autonomously through spaces filled with both static obstacles and dense crowds of pedestrians. The policy uses a unique combination of input data to generate the desired steering angle and forward velocity: a short history of lidar data, kinematic data about nearby pedestrians, and a sub-goal point. The policy is trained in a reinforcement learning setting using a reward function that contains a novel term based on velocity obstacles to guide the robot to actively avoid pedestrians and move towards the goal. Through a series of 3D simulated experiments with up to 55 pedestrians, this control policy is able to achieve a better balance between collision avoidance and speed (*i.e.*, higher success rate and faster average speed) than state-of-the-art model-based and learning-based policies, and it also generalizes better to different crowd sizes and unseen environments. An extensive series of hardware experiments demonstrate the ability of this policy to directly work in different real-world environments with different crowd sizes with zero retraining. Furthermore, a series of simulated and hardware experiments show that the control policy also works in highly constrained static environments on a different robot platform without any additional training. Lastly, we summarize several important lessons that can be applied to other robot learning systems. Multimedia demonstrations are available at https://www.youtube.com/watch?v=KneELRT8GzU&list=PLouWbAcP4zIvPgaARrV223lf2eiSR-eSS.**

***Index Terms*—Collision Avoidance, Deep Learning in Robotics and Automation, Field Robotics, Reactive and Sensor-Based Planning**



(a) Simulated indoor environment



(b) Real-world outdoor environment

Fig. 1. Our robot navigates through crowds in a simulated indoor environment and a real-world outdoor environment. The robot is following a nominal path (green line) to the goal (red disk) while avoiding pedestrians (blue arrows show current velocity).

I. Introduction

ONE common application of autonomous mobile robots is replacing manual labor to provide last-mile delivery services. For example, delivering sterile supplies and injection medicines to patients in hospitals, delivering materials to various packaging workstations in warehouses, and delivering delicious food or groceries to customers in restaurants and grocery stores [1]–[3]. All these tasks have time limits and require mobile robots to navigate autonomously and quickly to destinations through a partially known space filled with moving people and other static obstacles, 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 policy that yields autonomous navigation.

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

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

In this paper, we use a novel deep reinforcement learning-based (DRL) approach due to the complexity of the environments we want the robot to operate in, with 10's of pedestrians moving in dense crowds, which makes it difficult to design clear rules within a traditional model-based framework or to collect sufficiently representative training data to apply supervised learning techniques. While the use of DRL to generate control policies is becoming commonplace, this paper presents eight primary contributions: 1) We create a novel combination of data representations for the input to our neural network-based control policy, using a short history of pooled lidar data, the current kinematics of nearby pedestrians, and a sub-goal point. We design these representations to be robust to localization errors by putting all data into the robot's local coordinate frame and to improve generalizability to new environments by handcrafting higher-level representations from the raw sensor data. 2) We apply the theory of velocity obstacles to create a new reward function for the DRL framework that encourages the robot to proactively avoid collisions. 3) We validate the navigation performance in multiple simulated 3D environments and demonstrate that

the proposed control policy achieves a better trade-off between collision avoidance and speed, and generalizes better to different crowd sizes and new unseen environments than state-of-art approaches, including a model-based controller [4], a supervised learning-based approach [5], and two DRL-based approaches [6]. 4) We demonstrate that our policy, trained only within a single simulated environment with a single crowd size, can be directly used on a real-world robot in unseen environments, including both indoor and outdoor locations on Temple's campus, and with different crowd densities without any retraining. 5) We demonstrate that our proposed control policy works effectively in highly constrained static environments and on different robot platforms without any retraining. To do this, we competed in the Benchmark Autonomous Robot Navigation (BARN) Challenge at the 2022 IEEE International Conference on Robotics and Automation (ICRA), where we placed 1st in the simulation competition and 3rd in the hardware competition [7]. 6) We make our code and our 3D human-robot interaction simulator available open source at: https://github.com/TempleRAIL/drl_vo_nav. 7) We conduct a detailed literature review for the most closely related works ranging from model-based to learning-based approaches. 8) We summarize several important design principles from our work that can be applied to other robot learning systems.

II. Related Works

In this section, we provide a detailed description of prior work on robot navigation problems, which we divide into two categories: model-based and learning-based. Learning-based approaches can be further divided into supervised learning-based and reinforcement learning-based.

A. Model-based Approaches

A typical model-based approach is the ROS [8] navigation stack, which uses costmaps to store obstacle information and uses the dynamic window approach (DWA) planner [4] to do local planning. Based on the ROS navigation stack, [9] recently add human safety and visibility costmaps into both global and local planners to handle human-aware navigation problems. Another class of model-based approaches are velocity obstacle-based algorithms [10]–[12], which map the dynamic environment into the robot velocity space and generate safe control velocities from these velocity constraints. Similarly, Arul et al. [13] combine reciprocal velocity obstacles with buffered Voronoi cells to solve multi-agent navigation problems. A third class of model-based approaches uses model predictive control [14], [15], which can integrate collision avoidance and obstacle dynamics into robot constraints. Other model-based approaches attempt to directly model how humans and robots interact, using social forces [16], [17], Gaussian mixture models (GMM) [18], or a combination of potential functions and limit cycles [19]. All those model-based methods require knowledge of pedestrian kinematics, utilize multi-stage procedures to process sensor data, and often require practitioners to carefully hand-tune model parameters, making them difficult to implement and generalize to new scenarios.

B. Supervised Learning-based Approaches

With the success of deep learning in the computer vision area, many researchers started to apply learning-based approaches to other problems, including robot navigation. Unlike the traditional model-based approaches with multi-stage procedures, several recent works use deep neural networks to learn end-to-end control policies that generate steering commands directly from raw sensor data. Pfeiffer et al. [20] create a data-driven end-to-end motion planner that uses a convolutional neural network (CNN) to generate a linear and angular velocity from raw lidar data. Tai et al. [21] also use a CNN to select from one of five discrete robot control commands using raw depth camera data as input. Loquercio et al. [22] use a similar approach with raw camera image data to train a control policy that enables a drone to drive safely in the streets. Similarly, Kahn et al. [23] also use the raw camera image data to train an end-to-end learning-based mobile robot navigation system that can navigate in real-world environments with geometrically distracting obstacles. However, most of these end-to-end approaches only focus on static environments.

For dynamic environments, Pokle et al. [24] use multiple CNN networks to learn multimodal high-level features from raw sensor data. A CNN-based local planner fuses these features to generate velocity commands. However, their learned feature-based policy was only tested in a relatively open simulated environment with no more than 3 moving pedestrians. For highly crowded dynamic environments with up to 50 moving pedestrians, our previous work [5] proposes a CNN-based policy that combined a short history of lidar data with kinematic data about nearby pedestrians.[1] However, as we will show, the policy trained in a supervised setting is not as successful as the approach proposed in this work, particularly when applied to new environments or crowd sizes.

C. Reinforcement Learning-based Approaches

Compared with supervised learning-based methods, DRL-based approaches are more commonly used for navigation in dynamic environments. Using the proximal policy optimization (PPO) algorithm [25], Long et al. [26], [27] first propose a fully decentralized multi-robot collision avoidance framework using 2D raw lidar range data. Following this line of thought, Guldenring et al. [6] investigate the different state representations of 2D lidar data, and use the PPO training algorithm to train a human-aware navigation policy. Arpino et al. [28] present a multi-layout training regime that uses raw lidar data as its input to train an indoor navigation policy. Huang et al. [29] construct a multi-modal late fusion network to fuse raw lidar data and raw camera image data. Although these sensor-level approaches can quickly and easily generate control policies, they cannot distinguish between static or dynamic obstacles, nor can they utilize high-level information such as the positions and velocities of obstacles.

To address the limitations of using only raw sensor data, many other works leverage information about the pedestrian motion to enable safe navigation in dynamic environments.

[1] We utilize a very similar network architecture in this current paper.

Everett et al. [30]–[32] directly feed the relative position and velocity of pedestrians into the neural network, and generate socially aware collision avoidance policies for pedestrian-rich environments. Chen et al. [33] focus on jointly modeling human-robot and human-human interactions by a self-attention mechanism. Immediately afterwards, they also propose a relational graph learning network to infer robot-human interactions and predict their trajectories [34]. Similarly, another work [35] encodes the crowd state with a graph convolutional network (GCN) and predicts human attention. Recently, Liu et al. [36] use a decentralized structural-recurrent neural network (RNN) to model the interaction between the robot and pedestrians. One major drawback of these agent-level approaches is that they are only suitable for solving crowded navigation problems in open spaces and ignore static obstacles and geometric constraints in the layout. To solve this issue, Liu et al. [37] expand the work of [33] by adding additional static obstacle maps, but they need two parallel network models to handle the presence/absence of pedestrians. Sathyamoorthy et al. [38] expand the PPO training policy of [26] by using a late fusion network to combine raw lidar data with pedestrian trajectory predictions. Dugas et al. [39] use unsupervised learning architectures to predict and reconstruct future lidar latent states. However, most of the above-described works only focus on exploiting the sensor perception information and its data representations, but ignore the reward function design.

Towards this, Xu et al. [40] propose a knowledge distillation reward term to encourage the robot to learn expert behaviors. Patel et al. [41] propose a square warning region-based reward function to encourage the robot to navigate in the direction opposite to the heading direction of obstacles. However, the effect of knowledge distillation reward largely depends on the quality of the expert dataset, and the reward function based on the square warning region is too conservative, resulting in slow actions.

D. Comparative Analysis

There is an abundance of studies that focus on robot navigation problems, with solutions ranging from traditional model-based approaches to supervised learning-based approaches to reinforcement learning-based approaches. Typical model-based approaches compute efficient paths and the parameters are easily interpretable, but require manually adjusting model parameters for different scenarios [4], [9]–[19]. 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], [20]–[24]. Reinforcement learning-based approaches are experience-driven and similar to human learning, but require carefully designing a reward function [6], [26]–[41]. Although each type of approach has its own advantages and disadvantages, we choose the reinforcement learning-based framework because it is difficult to manually design a general model or collect effective training data in uncontrolled and human-filled environments.

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. Model-based approaches usually preprocess the raw sensor data and use the extracted high-level information, such as the obstacle costmaps [4], [9], the relative velocity of obstacles [10]–[13], and the environment dynamics [14], [15]. On the other hand, end-to-end learning approaches focusing on static environments directly feed the raw sensor data, *e.g.,* lidars [20] or cameras [21]–[23], and goal information into CNNs. DRL-based approaches are more varied, with some also feeding raw lidar scans (and goal points) into CNNs [6], [26]–[29] while others extract and utilize the position and velocity of pedestrians [30]–[32], model robot-human interactions [33], [35]–[37], predict pedestrian trajectories [34], [38], predict latent states [39], or construct obstacle cost matrices [41]. The long-term success of model-based approaches and the results of the recent DRL studies demonstrate the utility of preprocessed data representations over raw sensor data, as they contain more useful information about pedestrian motion which, in turn, 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 [5]. A significant advantage of this data representation is that it is much simpler and more easily interpretable than the complex pedestrian predictions or interactions used in other DRL-based works [33]–[38]. Also, all of the data is represented entirely within the robot's local coordinate frame, making the solution robust to errors in the localization, which can be common in densely crowded dynamic environments.

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 [6], [26]–[39]. Although this simple reward function design can work well, their sparse collision avoidance reward function only gives a large negative reward to penalize collisions, and does not give a direct and effective reward signal to guide the robot to actively avoid obstacles. In fact, the 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* of pedestrians (and other objects) is more important since people often follow one another closely in dense crowds and give more space when walking in opposite directions. One attempt at this is the reverse reward function from [41], which prioritizes safety but leads to slow motion and can cause deadlock in dense crowds.

We propose a novel reward function that uses first-order velocity obstacles to penalize headings that are likely to lead to collisions, thereby guiding the robot to continuously tune its heading direction to actively avoid crowds of pedestrians while making progress towards the goal. Note that Han et al. [42] simultaneously developed a reciprocal velocity obstacle (RVO)-based reward function for distributed multi-robot navigation. This is similar to our velocity obstacle concept but has two main differences. First, our work and theirs solve two distinct, albeit related, problems. We seek to prevent *robot-human* collisions by using VOs, which place the burden of collision avoidance entirely on the robot as we cannot directly

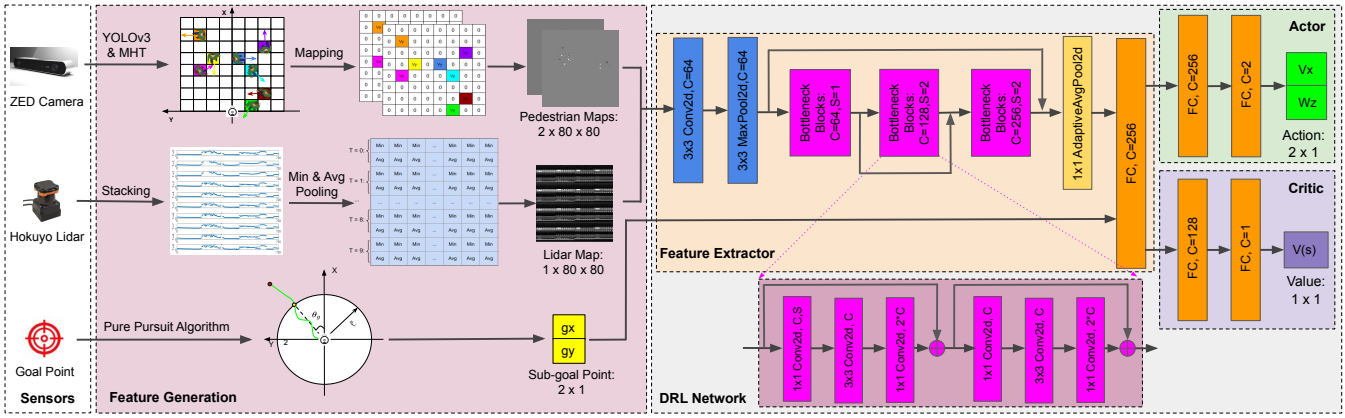


Fig. 2. The overall system architecture of our DRL-VO control policy. Raw sensor data from the ZED camera and Hokuyo lidar, as well as the goal point, are fed into a feature generation module to create intermediate features. Three 80×80 intermediate-level feature maps, one from lidar and two containing 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. The feature extractor module consists of bottleneck residual blocks [44], while the critic and actor modules consist of fully connected layers. "3×3" or "1×1" denote the kernel size, and "C" and "S" denote the number of output channels and the stride length, respectively.

model or control human behavior. On the other hand, Han et al. [42] seek to prevent *robot-robot* collisions by using RVOs, which split the burden of collision avoidance between the agents based on the assumption that both agents run similar control policies. Second, our approach encourages the robot to actively steer towards the goal while avoiding potential future human-robot collisions while theirs encourages the robot to focus only on potential robot-robot collisions, as judged by the expected collision time, and ignore the goals. Sun et al. [43] also use RVOs in tandem with a DRL-based control policy for multi-robot navigation. However, instead of using RVOs in the DRL reward function, they use it as a parallel control policy, switching from DRL to RVO when a collision is imminent.

Inspired by the problems and approaches mentioned above, this paper starts with two core issues of sensor data representation and reward function design to solve the problem of autonomous navigation in dynamic crowds. We use a unique combination of preprocessed sensor data as the input to our control policy. Specifically, we have information about static obstacles and the geometric layout of the environment from a short history of lidar data, kinematic information about dynamic obstacles from a multi-object tracker, and information about how to reach the destination from a sub-goal point. In designing our novel reward function, we leverage the utility of model-based approaches (*i.e.,* velocity obstacles) to create a navigation policy that does not have the same drawbacks as velocity obstacles. Specifically, our policy does not require the robot to have perfect information about pedestrian kinematics. This new term will allow our robot to learn a robust navigation policy with a good balance between collision avoidance and speed.

III. Navigation Policy

In this section, we formulate the navigation problem, with a focus on safety and speed through dynamic environments. We then detail our DRL-based approach, 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 autonomously 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 complicated navigation decision-making process can be formulated as a partially observable Markov decision process (POMDP), which is 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 solving large POMDPs is deep reinforcement learning [45]. The general objective $L(\theta)$ of DRL is to maximize the expected discounted return by optimizing the neural network-based policy π_θ:

$$L(\theta) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t R^t\right], \tag{2}$$

where γ is the discount rate in the range of $[0, 1]$, $\mathbb{E}_{\pi_\theta}[\cdot]$ denotes the expected value of a random variable assuming the agent follows the neural network-based policy π_θ. Figure 2 outlines our DRL network, which consists of four modules: feature generation, feature extractor, actor, and critic. The feature generation module generates our intermediate features, *i.e.,* the partial observation $\mathbf{o}^t$, from the raw sensor data. The

feature extractor module then extracts high-level features from $\mathbf{o}^t$, the critic module generates the state value, and the actor module generates steering action $\mathbf{a}^t$.

B. Observation Space

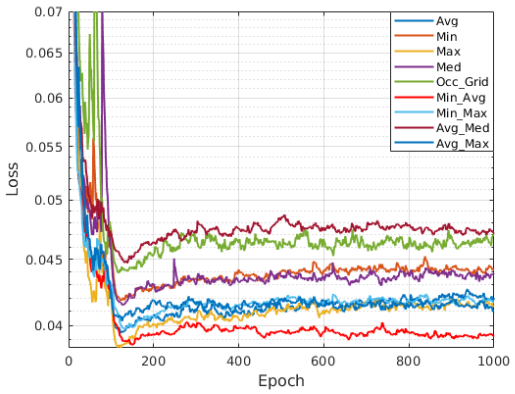
We use the same partial observation $\mathbf{o}^t = [\mathbf{l}^t, \mathbf{p}^t, \mathbf{g}^t]$ as our previous work using supervised learning [5], which consists of three preprocessed components: lidar history ($\mathbf{l}^t$), pedestrian kinematics ($\mathbf{p}^t$), and the sub-goal position ($\mathbf{g}^t$). All the data in the observation $\mathbf{o}^t$ are expressed in the local reference frame of the robot. This allows our method to be robust to errors in robot localization with respect to a global reference frame, which happens more frequently in densely-packed dynamic environments as there are fewer stationary landmarks for the robot to localize itself against. Additionally, relative data is more natural for planning purposes since navigating in a dense crowd is more about going with the flow of traffic than meeting some absolute velocity constraints.

We will briefly summarize the process for extracting the observation data $\mathbf{o}^t$ from the raw sensor data below, with the full details available in [5]. One of the key features of this previous work was the use of handcrafted intermediate features which allowed us to set the size of the control policy. We do this by using a consistent data format for all input types, which is an 80×80 grid in our case.

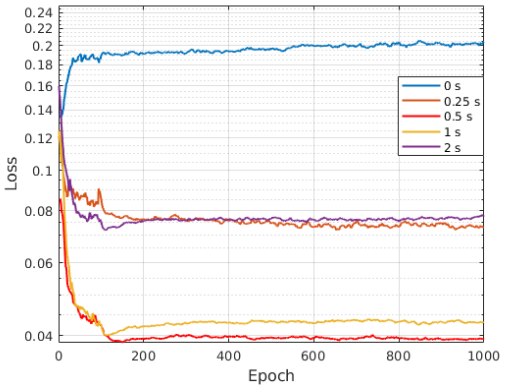
1) Pedestrian Kinematics: Unlike complex CAGE representations for crowd flow prediction [46], which only encode crowd location information, our proposed simple pedestrian kinematic map representation encodes both the location and velocity information of detected pedestrians. Specifically, we encode the pedestrian kinematics in a pair of 80×80 grids that contain information about both the relative locations and velocities of all pedestrians in a 20×20 m area in front of the robot. To create these grids, we first feed the RGB image and 3-D point cloud coming from a depth camera (*i.e.,* ZED camera) into the YOLOv3 [47] detector to detect pedestrians and extract their relative locations. Once we extract these instantaneous estimates, we use a multiple hypothesis tracker (MHT) [48] to track the *relative* positions and velocities of pedestrians. The velocity values are stored in the grid, using the position values to determine the cell index, as the ZED Camera track in the Feature Generation block of Fig. 2 shows.

2) Lidar History: The lidar data $\mathbf{l}^t$ consists of a history of 0.5 s data (*i.e.,* 10 scans), where we apply a combination of minimum and average pooling to each scan. The resulting pooled data is stacked 4 times to create an 80×80 array, as the Hokuyo Lidar track in the Feature Generation block of Fig. 2 shows. This data format is identical to our previous work [5], but here we present the results of ablation studies used to select this data format and the corresponding parameters, such as the history length.

First, motivated by the lidar data downsampling operation from [49], we explore the use of different lidar preprocessing operations, including reshaping the raw lidar data, projecting scan points into an occupancy grid map, and pooling (minimum, average, median, and maximum). We use each of these different operations, and several combinations thereof,



(a) Lidar preprocessing steps



(b) Lidar history length

Fig. 3. Validation loss curves of different lidar data configurations. The loss is the MSE between the expert velocity and the learned velocity of the resulting control policy.

to train our navigation policy in a supervised manner using the dataset from [5]. Figure 3(a) shows the resulting mean square error (MSE), *i.e.,* the loss, between the learned and expert velocities for each resulting control policy. We found that using a combination of minimum pooling and average pooling to extract two separate distance measurements per scan region, as Fig. 3(a) shows, yielded the best performance.

Once we determined the most informative way to structure the lidar data, we then studied how long of a history of scans to keep. We considered time windows of 0, 0.25, 0.5, 1, and 2 seconds,[2] using the same training and validation procedure as we did when studying the different lidar preprocessing operations. Figure 3(b) shows the resulting MSE loss curve for each resulting control policy. We see that a short history of lidar data contains more useful information than only the current scan (*i.e.,* 0 s), while keeping data from beyond a certain time period resulted in no improvement or even degraded performance. We found the best time window to be 0.5 s (*i.e.,* 10 lidar scans), which we conceptually think of as the effective time constant for pedestrian information.

3) Goal Position: Lastly, the robot needs to know the relative position of the goal. Instead of feeding the final goal

[2]Note: we sample the lidar data at 20 Hz, so each time corresponds to a different number of scans.

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 and avoid issues with normalization as the distance to the sub-goal is constant. To select the sub-goal $\mathbf{g^t}$, we use the pure pursuit algorithm [50], which calculates the point along the full path that is a specified distance (2 m in our case) ahead of the robot. By tuning this distance, the robot can look far ahead or right nearby. Figure 2 illustrates this process in the Goal Point track of the Feature Generation block. We hypothesize that this sub-goal information will enable the robot to more accurately navigate to the final goal position.

4) Normalization: We use the Max-Abs scaling procedure to normalize all observation data to $[-1, 1]$ before feeding them into our control policy network:

$$\mathbf{o}^t = 2\frac{\mathbf{o}^t - \mathbf{o}^t_{\min}}{\mathbf{o}^t_{\max} - \mathbf{o}^t_{\min}} - 1, \tag{3}$$

where $\mathbf{o}^t_{\max}$ and $\mathbf{o}^t_{\min}$ are the maximum value and minimum value of the data (*e.g.,* for the lidar history $\mathbf{l}^t$, they are the minimum and maximum range value of the lidar sensor).

C. Action Space

The action $\mathbf{a}^t = [v^t_x, \omega^t_z]$ of our DRL control policy has two terms, the forward (v^t_x) and rotational (ω^t_z) velocities in the local robot frame. Note, we use a continuous action space as we believe this gives the robot more accurate control. We limit the range of the forward velocity v^t_x to $[0, 0.5]$ m/s and the rotational velocity ω^t_z to $[-2, 2]$ rad/s based on the limitations of the Turtlebot2, our target hardware platform. Note that similar to observation space, we also normalize the action space to [-1, 1] using the Max-Abs scaling procedure.

D. Network Architecture

The feature extractor network is identical to the backbone CNN network of our previous work [5], which fuses the lidar historical observation $\mathbf{l}^t$ and pedestrian kinematics observation $\mathbf{p}^t$. After extracting high-level features, a single fully connected layer is used to fuse those high-level features with the sub-goal information. Both the actor module and the critic module are simply constructed from two fully connected layers. Note that each convolutional layer in our feature extractor module is followed by a batch normalization layer and a ReLU action layer. Additionally, the last fully connected layer in our feature extractor is followed by a ReLU action layer while the last fully connected layers of the actor and critic modules have no ReLU action layers.

Our architecture is an example of a middle fusion network, as data from different input streams are fused in the middle of the chain between raw data input and output. Feng et al. [51] provide a detailed discussion about different fusion architectures (in a general setting), which are typically categorized as early, middle, or late fusion. They note that there is no solid evidence to support which fusion network is the best choice, though there is a connection between the computational load of a neural network and its structure as the number of layers and their sizes affect both the number of computations that must be done to produce and output and the memory required to store the parameters of the neural network. To examine this, we used the same dataset and supervised training manner as our previous work [5] to train two networks. The first, which we call Middle, fuses the pedestrian kinematics and lidar data at the input to the feature extractor block, as we do above, and the second, called Late, fuses all data just before the output of the feature extractor block. As we can see in Table I, the Early network is about half the size of the Late network structure, which nearly doubles the processing rate and yields slightly better regression performance, with smaller root mean square error (RMSE) and explained variance ratio (EVA).

TABLE I
ABLATION EXPERIMENT RESULTS WITH DIFFERENT FUSION STRUCTURES

Network Structures	RMSE	EVA	# of Params	FPS
Middle fusion	**0.17**	**0.15**	**28.94 M**	**255.13**
Late fusion	0.18	0.17	57.86 M	140.53

Like our network, most other neural network-based control policies [24], [28], [29], [38] also use late or middle fusion architectures as this allows for flexible design because the different input sources each yield some learned intermediate representations which are all combined downstream. The key difference between these and our architecture is that we use handcrafted features as our intermediate representation, much like a traditional model-based algorithm, rather than learned features, as are typical in learning-based algorithms. One benefit of this approach is that our data representations, *i.e.,* the pedestrian map, are independent of the specific object detection and tracking tools used. This allows us to easily swap out YOLO and the MHT for different approaches with no need for retraining.

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 behaviors. Navigation tasks have 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 or moving objects. Thus, we design a multi-objective reward function:

$$r^t = r^t_g + r^t_c + r^t_w + r^t_d, \tag{4}$$

where r^t_g rewards making progress towards the goal, r^t_c penalizes passively approaching or colliding with an obstacle, r^t_w penalizes rapid changes in direction, and r^t_d rewards actively steering to avoid obstacles and point towards the sub-goal.

Note that most previous works [6], [26]–[39] only use such a zero-order passive collision avoidance reward r^t_c, a position-based heuristics, to guide robots to passively avoid obstacles.

While these standard terms are useful to learn good navigation behavior, we want to allow robots to indirectly learn from successful model-based navigation policies. To do this, we introduce a new reward term r^t_d that uses velocity obstacles [10] to set a desired navigation direction. This term allows the robot to use first-order kinematic information (*i.e.,* velocity) to

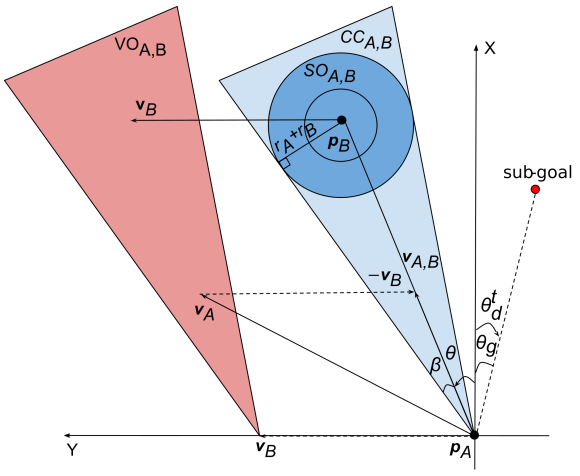


Fig. 4. The pictogram for velocity obstacle $VO_{A,B}$ (red cone), collision cone $CC_{A,B}$ (light blue cone), and special occupancy $SO_{A,B}$ (blue circle). A represents the robot with radius r_A, position $\mathbf{p}_A$, and velocities $\mathbf{v}_A$. B represents the moving pedestrian with radius 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.

proactively avoid collisions while making progress towards the goal.

1) Reaching the Goal: The reward is given by

$$r_g^t = \begin{cases} r_{\text{goal}} & \text{if } \|p_g^t\|) < g_m \\ -r_{\text{goal}} & \text{else if } t \geq t_{\max} \\ r_{\text{path}}(\|p_g^{t-1}\| - \|p_g^t\|) & \text{otherwise,} \end{cases} \tag{5}$$

where p_g^t is the goal position (in the robot's frame) at time t.

We use $r_{\text{goal}} = 20$, $r_{\text{path}} = 3.2$, $g_m = 0.3\,\text{m}$, and $t_{\max} = 25\,\text{s}$.

2) Passive Collision Avoidance: The reward is given by

$$r_c^t = \begin{cases} r_{\text{collision}} & \text{if } \|p_o^t\| \leq d_r \\ r_{\text{obstacle}}(d_m - \|p_o^t\|) & \text{else if } \|p_o^t\| \leq d_m \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

where p_o^t is obstacle position 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

$$r_w^t = \begin{cases} r_{\text{rotation}}|\omega_z^t| & \text{if } |\omega_z^t| > \omega_m \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

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

4) Active Heading Direction: The reward is given by

$$r_d^t = r_{\text{angle}}(\theta_m - |\theta_d^t|), \tag{8}$$

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 [10]. 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 region:

$$SO_{A,B} = \{\mathbf{p}_S \mid d(\mathbf{p}_S, \mathbf{p}_B) < r_A + r_B\}, \tag{9}$$

where $d(\mathbf{p}_S, \mathbf{p}_B)$ is the distance between $\mathbf{p}_S$ and $\mathbf{p}_B$. Then, the collision cone is defined as

$$CC_{A,B} = \{\mathbf{v}_{A,B} \mid \exists t, \mathbf{v}_{A,B}t \cap SO_{A,B} \neq \emptyset\}. \tag{10}$$

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

$$VO_{A,B} = CC_{A,B} \oplus \mathbf{v}_B, \tag{11}$$

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 a future time. From the perspective of the direction angle of $\mathbf{v}_{A,B}$, the collision cone $CC_{A,B}$ can also be defined as

$$CC_{A,B} \in [\theta - \beta, \theta + \beta], \tag{12}$$

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

$$\theta = \arctan 2\left(\frac{\mathbf{p}_{B_y}}{\mathbf{p}_{B_x}}\right), \qquad \sin\beta = \frac{r_A + r_B}{\|\mathbf{p}_B\|}. \tag{13}$$

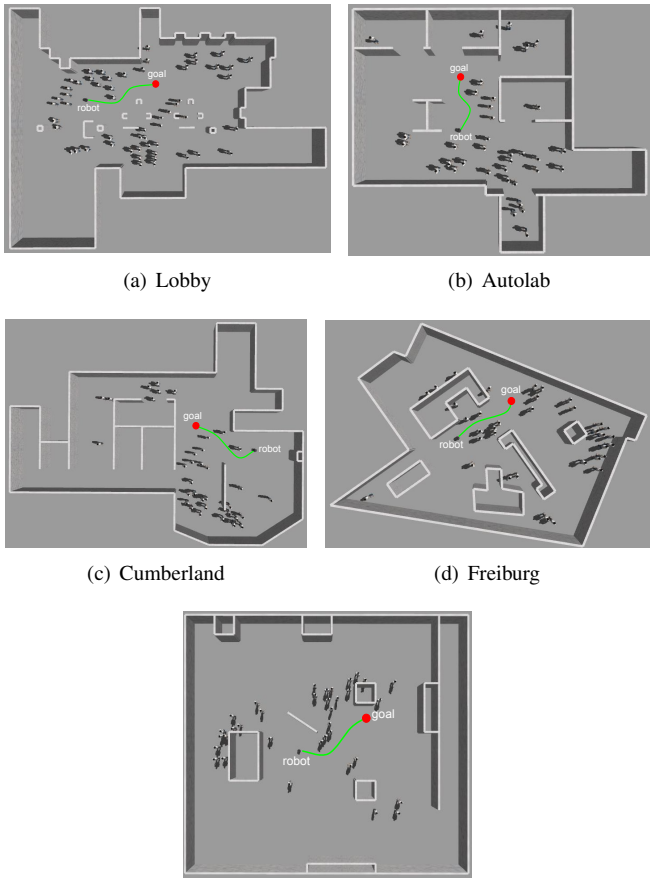
Using the collision cone $CC_{A,B}$, the robot can know which heading direction angles will cause collisions with all of the moving pedestrians tracked by MHT. The robot then uses these collision cones and the direction of the sub-goal (θ_g) to find

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: θ_d^t ← π/2
if B_peds ≠ ∅ then
    θ_min ← ∞
    for i = 1, 2, ..., N do
        θ_u ← sample from [−π, π]
        free ← True
        for B in B_peds do
            θ_{v_{A,B}} ← atan2((v_{A_x} sin(θ_u) − v_{B_y}) / (v_{A_x} cos(θ_u) − v_{B_x}))
            θ, β ← from (13) using B
            if θ_{v_{A,B}} ∈ [θ − β, θ + β] then
                free ← False
                break
        if free then
            if ‖θ_u − θ_g‖ < θ_min then
                θ_min ← ‖θ_u − θ_g‖
                θ_d^t ← θ_u
else
    θ_d^t ← θ_g
return θ_d^t
```



(a) Lobby (b) Autolab

(c) Cumberland (d) Freiburg

(e) Square

Fig. 5. Gazebo simulation environments. The Lobby world has two configurations with 34 pedestrians and 5-55 (sampling interval 10) pedestrians respectively. The Autolab world, Cumberland world, Freiburg world, and Square world only have one configuration with 25-35 (sampling interval 10) pedestrians.

the desired heading direction angle θ_d^t using Algorithm 1. This sampling-based search algorithm is motivated by [11].

F. Deep Reinforcement Learning Algorithm

While there are many deep reinforcement learning algorithms, we use the proximal policy optimization (PPO) algorithm [25] to train our DRL network. A major advantage of the PPO algorithm is that it is much simpler to implement and can achieve comparable or better results than other state-of-the-art algorithms (*e.g.,* A2C [52], TRPO [53] and ACER [54]). In addition, many other robot navigation works [6], [26], [27], [41] use PPO and have found that it works well on navigation problems. As suggested by [25], we use Adam optimizer [55], a stochastic gradient descent method, to find the optimal policy parameters θ^*.

IV. Crowded Dynamic Environments

To demonstrate the efficacy and performance of our proposed control policy, we first use a Gazebo simulator [56] and the PEDSIM library [57] to conduct a set of 3D simulated experiments, and then use a Turtlebot2 robot to conduct the real-world experiments. This section describes the experimental setup, training procedure, and results.

A. Experimental Setup

1) Robot Configuration: For the simulated and hardware tests, we use a Turtlebot2 robot with a ZED stereo camera and a Hokuyo UTM-30LX lidar, as shown in Fig. 1. The maximum velocity of the Turtlebot2 is limited to 0.5 m/s. The depth range of the 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 $0.25°$.

2) Pedestrian Configuration: PEDSIM is a microscopic pedestrian crowd simulation library, which uses the social forces model [16], [58] to guide the motion of individual pedestrians:

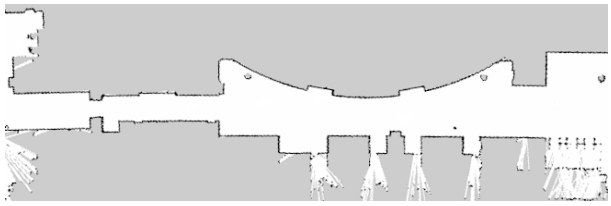
$$\mathbf{F}_p = \mathbf{F}_p^{\text{des}} + \mathbf{F}_p^{\text{obs}} + \mathbf{F}_p^{\text{per}} + \mathbf{F}_p^{\text{rob}}, \tag{14}$$

where $\mathbf{F}_p$ is the resultant force that determines the motion of a pedestrian p; $\mathbf{F}_p^{\text{des}}$ pulls a pedestrian towards a destination; $\mathbf{F}_p^{\text{obs}}$ pushes a pedestrian away from static obstacles; $\mathbf{F}_p^{\text{per}}$ models interactions with other pedestrians (*e.g.,* collision avoidance or grouping); and $\mathbf{F}_p^{\text{rob}}$ pushes pedestrians away from the robot, modeling the way people would naturally avoid collisions and thereby allowing our control policy to learn this behavior. The first three terms are part of the standard model while the last term is new.

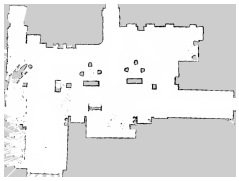
3) Simulation Configuration: Figure 5(a) shows the main Gazebo simulation environment, a replica of the lobby in the College of Engineering building at Temple University, that we used to train and test our control policies. This environment is roughly 25×10 m in size and is filled with a number of static obstacles (*e.g.,* chairs, tables, a security desk, waste bins, and pillars). The second environment, shown in Fig. 5(b), is a replica of the Autolab building that is about 20×10 m in size. The third environment, shown in Fig. 5(c), is a replica of the Cumberland building that is about 20×10 m in size. The fourth environment, shown in Fig. 5(c), is a replica of the Freiburg building that is about 20×10 m in size. The fifth environment, shown in Fig. 5(e), is an artificially created environment that is about 20×20 m in size. Using these five worlds, we set up six types of scenarios:

1) **Lobby world with 34 pedestrians**: train our DRL control policies
2) **Lobby world with 5-55 pedestrians**: test generalization across different crowd densities
3) **Autolab world with 25-35 pedestrians**: test generalization to unseen environments/crowd densities
4) **Cumberland world with 25-35 pedestrians**: test generalization to unseen environments/crowd densities
5) **Square world with 25-35 pedestrians**: test generalization to unseen environments/crowd densities
6) **Freiburg world with 25-35 pedestrians**: test generalization to unseen environments/crowd densities

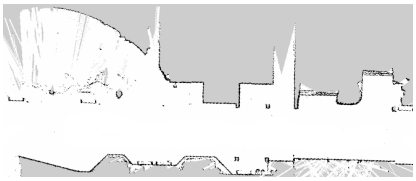
We use an Nvidia DGX-1 server with 40 CPU cores, 8x Tesla V100 GPUs with 16 GB memory, and 512 GB of RAM to train our DRL networks. We use a desktop computer with an Intel i7-6800K CPU, a GeForce GTX 1080 GPU with 8 GB memory, and 16 GB of RAM to run all the simulations. Both the training server and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



(a) Indoor hallway



(b) Indoor lobby



(c) Outdoor hallway

Fig. 6. Floorplans of real-world environments. All of these three environments are at Temple University.

4) Hardware Configuration: In the hardware tests, the Turtlebot2 is equipped with an NVIDIA Jetson AVG Xavier embedded computer containing a cluster of 8 Carmel ARM cores, a 512 Core Volta GPU, and 16 GB RAM memory. The Xavier runs Ubuntu 18.04 system and ROS Melodic Morenia, and we use the 30 W power mode.

We test our robot in three environments, shown in Fig. 6. Figure 6(a) is the indoor hallway connecting the Engineering building and Science building, which is about 66×5 m in size. Figure 6(b) is the indoor structured lobby in the Engineering building, which is about 20×10 m in size. This is the same environment that we replicated in Gazebo to use for training our policy. Figure 6(c) is the outdoor hallway outside of the Science building, which is about 60×8 m in size.

We conduct two sets of hardware experiments. First, we test the real-world applicability in different crowd sizes using the indoor hallway environment (Fig. 6(a)). Second, we test the applicability in different environments using all three test environments.

B. Training Procedure

We use the stable-baselines3 framework [59] to implement the PPO algorithm and train our DRL networks. We use the step decay with an initial learning rate of 10^{-3} as the learning rate schedule and set the mini-batch size to 512 during the training process. To train our control policies, we use the Lobby environment (Fig. 5(a)) 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 (4) (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 [6] (*i.e.,* A1-RD and A1-RC), the CNN-based policy [5], and the DWA planner [4], in each of the last five types of scenarios from Sec. IV-A3. Note that we use these baseline policies directly from their papers [4]–[6] without any retraining or parameter tuning. We compare our algorithms to these baselines using four commonly used metrics from the navigation literature:

1) **Success rate**: the fraction of collision-free trials
2) **Average time**: the average travel time
3) **Average length**: the average distance traveled
4) **Average speed**: the average travel speed

We run four trials for each experimental configuration and control policy, where each trial consists of the robot navigating through a predefined series of 25 goal points around the corresponding test environment. Note that we use the `amcl` ROS package to provide the robot localization service in known maps. All of these four trials use the same initial conditions, however, the results vary due to randomized sensor noise and variations in the pedestrian motion from the social force model.

1) Different Crowd Densities: We first focus on the Lobby environment with 5-55 (sampling interval 10) pedestrians used for testing the generalization across different crowd sizes. Table II presents the results obtained from these trials, where we observe three key phenomena. First, our DRL-VO policy has a much higher success rate than our DRL policy in each crowd size, while having a similar average speed. This shows that the proposed VO-based heading direction reward is beneficial and plays a key role in enabling the robot to maintain a good balance between collision avoidance and speed. 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 [5], 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 [4], we see that our planner takes a slightly longer route to the goal, indicating a preference for giving a wider berth to pedestrians.

2) Different Unseen Environments: We next consider the navigation behavior of our DRL-VO policy in different unseen environments, namely the Autolab, Cumberland, Freiburg, and Square environments from Fig. 5. In each environment, we test two different crowd densities: 25 and 35 pedestrians. Tables III and IV summarize these results. As can be seen, our novel DRL-VO policy still has both a higher success rate and slightly higher average speed than our DRL policy, with the only exception being a slightly lower speed in Square world. This further demonstrates the utility of our VO-based reward term to learn a more robust navigation policy and achieve a better trade-off between collision avoidance and speed.

Our DRL-VO policy also has the highest success rate out of all the algorithms in the lobby, Autolab, and Freiburg worlds. Additionally, our policy yields a faster and more consistent speed than other policies and tends to have the shortest path, aside from the DWA planner. This indicates that our policy has strong generalizability to different unseen environments.

TABLE II
NAVIGATION RESULTS AT DIFFERENT CROWD DENSITIES

Environment	Method	Success Rate	Average Time (s)	Average Length (m)	Average Speed (m/s)
Lobby world, 5 pedestrians	DWA [4]	0.93	12.10	**5.08**	0.42
	CNN [5]	-	-	-	-
	A1-RD [6]	-	-	-	-
	A1-RC [6]	0.91	14.11	6.14	0.44
	DRL	0.84	13.22	6.08	0.46
	DRL-VO	**0.94**	**11.41**	5.28	**0.46**
Lobby world, 15 pedestrians	DWA [4]	0.94	12.21	**5.09**	0.42
	CNN [5]	-	-	-	-
	A1-RD [6]	0.94	15.69	5.78	0.37
	A1-RC [6]	0.94	14.36	6.40	0.45
	DRL	0.80	13.66	6.24	0.46
	DRL-VO	**0.95**	**11.34**	5.26	**0.46**
Lobby world, 25 pedestrians	DWA [4]	0.82	13.49	**5.12**	0.38
	CNN [5]	0.80	19.31	6.16	0.32
	A1-RD [6]	0.90	17.87	6.07	0.34
	A1-RC [6]	0.88	14.18	6.26	0.44
	DRL	0.81	13.73	6.24	0.45
	DRL-VO	**0.92**	**11.37**	5.29	**0.47**
Lobby world, 35 pedestrians	DWA [4]	0.82	14.18	**5.15**	0.36
	CNN [5]	0.81	14.30	5.40	0.38
	A1-RD [6]	0.86	17.82	6.06	0.34
	A1-RC [6]	0.77	16.81	6.89	0.41
	DRL	0.75	14.30	6.46	0.45
	DRL-VO	**0.88**	**11.42**	5.31	**0.46**
Lobby world, 45 pedestrians	DWA [4]	0.77	15.39	**5.16**	0.34
	CNN [5]	0.79	16.65	5.62	0.34
	A1-RD [6]	0.76	23.16	6.61	0.29
	A1-RC [6]	0.77	14.65	6.28	0.43
	DRL	0.69	13.96	6.41	0.46
	DRL-VO	**0.81**	**11.65**	5.37	**0.46**
Lobby world, 55 pedestrians	DWA [4]	0.67	16.05	**5.22**	0.33
	CNN [5]	0.70	19.13	5.94	0.31
	A1-RD [6]	0.67	26.90	6.58	0.25
	A1-RC [6]	0.66	16.36	6.73	0.41
	DRL	0.60	13.39	6.13	0.46
	DRL-VO	**0.79**	**11.76**	5.39	**0.46**

The success rate of our policy drops, however, both in absolute and relative terms, in the Cumberland and Square worlds. Both of these environments have larger open spaces compared to the rest in Fig. 5. Given this, two potential causes of this decrease in success rate are: 1) the policy was trained in the more structured lobby and so adding in more varied environments during training would further improve the generalizability or 2) our DRL-VO policy with VO-based reward function is more suitable for structured environments rather than open spaces.

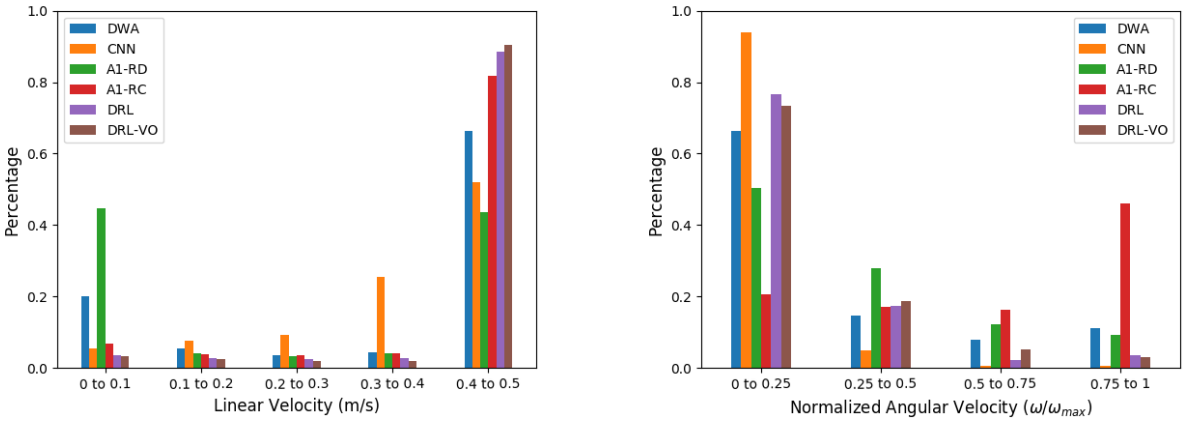
The A1-RD policy [6] has the highest success rate in the Cumberland and Square environments. We found this to be primarily due to its "stop and wait" strategy, where a robot that detects an obstacle in front of it will turn to a free space, stop, and wait for the obstacles to disappear. However, this behavior is the same regardless of the number of obstacles or whether these obstacles are static or dynamic. This can cause the robot to become frozen when it encounters a "C-shaped" obstacle because its policy only used raw lidar data and it was trained using a typical passive collision avoidance strategy based only on distance. This causes a robot using A1-RD to fail to reach the goal in both the Autolab and Freiburg environments with 35 pedestrians. In contrast to this, our DRL-VO policy can distinguish between 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 the goal point. See the videos in Multimedia Extension 1 to see these differences in the behavior of A1-RD and DRL-VO detailed.

3) Output Velocity Distribution: We also analyze the distribution of command velocities from each tested control policy to better understand why our DRL-VO policy has a faster average speed than other model-based and learning-based control policies. To do this we make histograms of the linear and angular velocity, using bins of size 0.1 m/s $0.25\,\omega_{\max}$, respectively. Note that we take the absolute value and normalize the angular velocity to ignore the direction of the turn and to account for different maximum values across the different control policies.

We observe two interesting trends in the data in Fig. 7. First, the majority of the linear velocities generated by all policies lie at either extreme of the range. Our DRL and DRL-VO policies have the highest percentage at the upper end of the range (about 90%), which is why we achieve a consistently higher average speed. On the other hand, the A1-RD policy has the highest percentage in the lower end of the range (over 40%), showing the prevalence of the "stop and wait" strategy noted above. The only other policy with a significant percentage of

TABLE III
NAVIGATION RESULTS AT DIFFERENT UNSEEN ENVIRONMENTS WITH 25 PEDESTRIANS

Environment	Method	Success Rate	Average Time (s)	Average Length (m)	Average Speed (m/s)
Lobby world, 25 pedestrians	DWA [4]	0.82	13.49	**5.12**	0.38
	CNN [5]	0.80	19.31	6.16	0.32
	A1-RD [6]	0.90	17.87	6.07	0.34
	A1-RC [6]	0.88	14.18	6.26	0.44
	DRL	0.81	13.73	6.24	0.45
	DRL-VO	**0.92**	**11.37**	5.29	**0.47**
Autolab world, 25 pedestrians	DWA [4]	0.88	13.50	**5.47**	0.41
	CNN [5]	0.78	21.59	6.61	0.31
	A1-RD [6]	0.88	18.92	6.54	0.35
	A1-RC [6]	0.77	15.63	7.35	0.43
	DRL	0.75	15.66	7.03	0.45
	DRL-VO	**0.91**	**12.79**	5.97	**0.47**
Cumberland world, 25 pedestrians	DWA [4]	0.78	15.16	**5.53**	0.37
	CNN [5]	0.63	27.80	7.73	0.28
	A1-RD [6]	**0.91**	17.47	6.63	0.38
	A1-RC [6]	0.81	16.31	6.93	0.43
	DRL	0.66	14.28	6.38	0.45
	DRL-VO	0.87	**12.42**	5.78	**0.47**
Freiburg world, 25 pedestrians	DWA [4]	0.83	13.90	**5.75**	0.41
	CNN [5]	0.48	30.38	8.30	0.27
	A1-RD [6]	0.81	17.95	6.61	0.37
	A1-RC [6]	0.74	17.06	7.35	0.43
	DRL	0.53	15.68	7.06	0.45
	DRL-VO	**0.84**	**12.98**	5.93	**0.46**
Square world, 25 pedestrians	DWA [4]	0.94	19.62	**8.48**	0.43
	CNN [5]	0.65	53.35	14.72	0.28
	A1-RD [6]	**0.95**	21.39	9.05	0.42
	A1-RC [6]	0.89	21.44	9.74	0.45
	DRL	0.83	22.62	10.77	**0.48**
	DRL-VO	0.86	**18.19**	8.61	0.47



(a) Linear velocity distribution

(b) Angular velocity distribution

Fig. 7. Histograms of the output velocities for different control policies. Data was collected from the lobby simulation world with 35 pedestrians.

the output near 0 is DWA, which occurs due to deadlock in particularly crowded regions.

Second, most of the control policies have similar angular velocity distributions, with a majority of commands being small changes to update the heading angle of the robot (*i.e.,* between $[0, 0.25]\,\omega_{\max}$). However, there are some key differences. First, the two outliers are the CNN policy, which yields very few commands outside of this minimum range, and the A1-RC policy, which generates a majority of commands at the upper range. Our DRL and DRL-VO policies generate fewer commands with high angular velocities, which is a result of the path smoothness term in our reward function, r_w.

4) Failure Cases and Solutions: Although our proposed DRL-VO policy is robust to different crowd densities and unseen environments, there are still some failure cases. One is that it can become difficult to find a collision-free direction in very high crowd 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 is caused by three reasons: 1) our simulated pedestri-

TABLE IV
NAVIGATION RESULTS AT DIFFERENT UNSEEN ENVIRONMENTS WITH 35 PEDESTRIANS

Environment	Method	Success Rate	Average Time (s)	Average Length (m)	Average Speed (m/s)
Lobby world, 35 pedestrians	DWA [4]	0.82	14.18	**5.15**	0.36
	CNN [5]	0.81	14.30	5.40	0.38
	A1-RD [6]	0.86	17.82	6.06	0.34
	A1-RC [6]	0.77	16.81	6.89	0.41
	DRL	0.75	14.30	6.46	0.45
	DRL-VO	**0.88**	**11.42**	5.31	**0.46**
Autolab world, 35 pedestrians	DWA [4]	0.81	16.52	**5.55**	0.34
	CNN [5]	0.68	22.73	6.65	0.29
	A1-RD [6]	-	-	-	-
	A1-RC [6]	0.73	17.11	6.93	0.41
	DRL	0.59	14.40	7.01	0.46
	DRL-VO	**0.84**	**12.88**	5.98	**0.46**
Cumberland world, 35 pedestrians	DWA [4]	0.74	16.28	**5.57**	0.34
	CNN [5]	0.60	24.25	7.08	0.29
	A1-RD [6]	**0.88**	18.04	6.69	0.37
	A1-RC [6]	0.77	15.37	6.66	0.43
	DRL	0.56	15.79	6.97	0.44
	DRL-VO	0.78	**12.62**	5.84	**0.46**
Freiburg world, 35 pedestrians	DWA [4]	0.70	18.15	**5.78**	0.32
	CNN [5]	0.57	20.09	6.66	0.33
	A1-RD [6]	-	-	-	-
	A1-RC [6]	0.65	17.11	6.93	0.41
	DRL	0.39	18.21	7.61	0.42
	DRL-VO	**0.76**	**13.37**	6.16	**0.46**
Square world, 35 pedestrians	DWA [4]	0.93	19.81	**8.51**	0.43
	CNN [5]	0.85	23.81	9.23	0.39
	A1-RD [6]	**0.96**	21.85	9.06	0.41
	A1-RC [6]	0.89	21.29	9.67	0.45
	DRL	0.86	21.48	10.27	**0.48**
	DRL-VO	0.87	**18.32**	8.66	0.47

ans occasionally bump into one another due to the imperfect social force model, 2) there are occasionally missed pedestrian tracks in the MHT due to occlusion and ambiguous data association, and 3) the safety guarantees of VOs do not extend to the resulting learned policy because we cannot explicitly define collision constraints and apply them to the black box network. There are many possible ways to alleviate these issues in future work, including continuing to train the policies (learned in simulation) in real-world environments, improving the robustness of the MHT, adding probabilistic predictions of the future state of the environment, adding a control barrier function as a safety constraint like [60], or simply add a safety module as a backup: when the obstacle is too close to the robot, the safety module will override the learned policy to stop and rotate the robot until it finds a collision-free space (much like the "stop and wait" in the A1-RD policy).

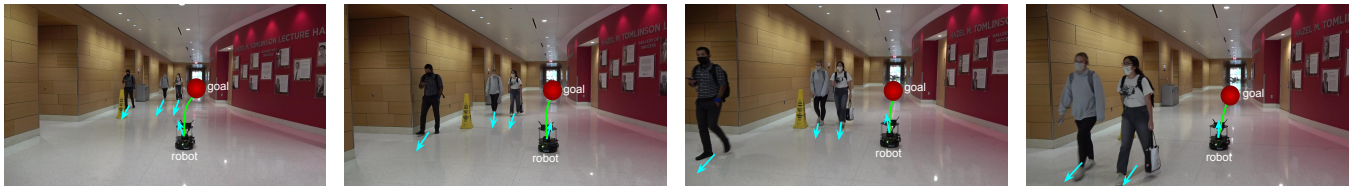
D. Hardware Results

Besides the simulated experiments, we also conduct a series of real-world experiments to demonstrate the applicability of our DRL-VO policy. Instead of performing hardware experiments in the laboratory under carefully controlled conditions as [6], [32]–[34], [36], we focus on robotic navigation in real and difficult environments by testing our system during the peak periods of pedestrian traffic in the intervals between classes. We test in three different parts of campus, an indoor hallway, an indoor lobby, and an outdoor hallway environment, shown in Fig. 6. In each environment, the robot passes through a predefined series of goal points with students of different crowd sizes naturally walking around and causing the robot to adjust its path, as Fig. 8-12 shows.

The real Turtlebot2 robot directly uses our DRL-VO policy trained from our Gazebo simulation platform without any fine-tuning. The computational complexity of our DRL-VO policy is relatively low, with up to 60 FPS inference speed on the Jetson AVG Xavier embedded computer, far above the 40 Hz of the sensors. The space complexity of our DRL-VO policy is also low, with only 32.137 M parameters. This means our DRL-VO policy is clearly capable of real-time processing and is suitable for robots with limited resources.

1) Different Crowd Densities: To test the real-world generalization across different crowd sizes, we let the robot navigate through a sequence of 15 goal points in the indoor hallway shown in Fig. 6(a). We tested the robot at different times, getting results with different crowd densities. Figure 8-10 and Multimedia Extension 2 shows the schematic diagrams of robot navigation behavior and experimental videos for low, medium, and high crowd density.

In the low density crowd, Multimedia Extension 2 and Fig. 8 show how our robot safely drives a total of 131.24 m in the nearly empty hallway at an average speed of 0.41 m/s. It is apparent that in this case, our robot can easily navigate to its goals safely and quickly by tuning its heading direction towards sub-goal points and following the nominal path. Similarly, our robot is also able to safely and quickly navigate to its goal points through the medium-density crowds, traveling a total of 137.28 m at an average speed of 0.43 m/s without any collisions, as shown in Multimedia Extension 2 and Fig. 9.

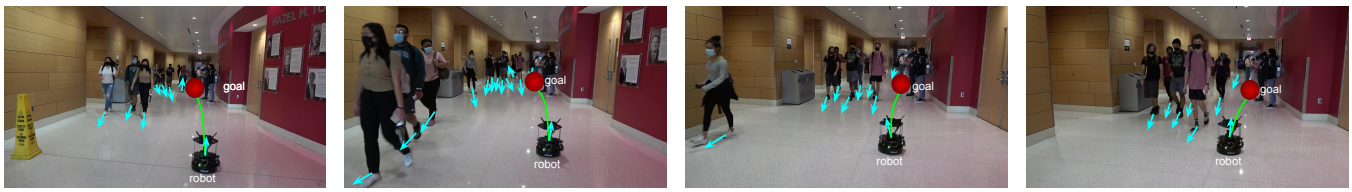


(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 8. Robot reactions to moving pedestrians in the indoor hallway with the low crowd density at different times.

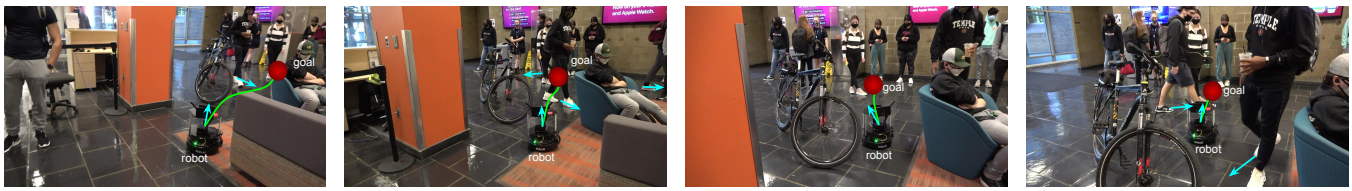
(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 9. Robot reactions to moving pedestrians in the indoor hallway with the medium crowd density at different times.



(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 10. Robot reactions to moving pedestrians in the indoor hallway with the high crowd density at different times.



(a) t (b) t+1 (c) t+2 (d) t+3

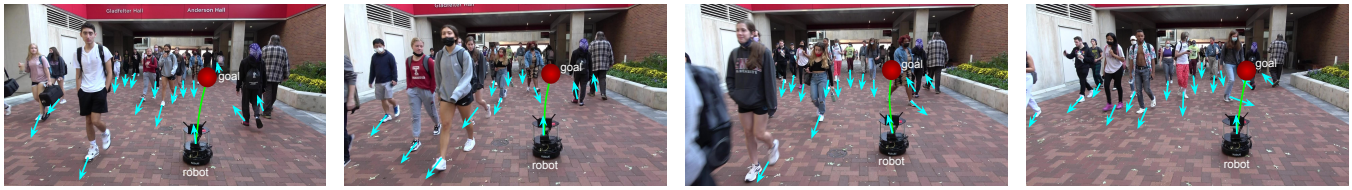
Fig. 11. Robot reactions to moving pedestrians in the indoor lobby with the high crowd density at different times.

The high density crowd, shown in Multimedia Extension 2 and Fig. 10, is more complex than the first two cases as there are stationary pedestrians in addition to the groups of moving pedestrians. Even in such a complex dynamic environment, our robot is still able to actively avoid collisions with moving students going in or going out of classes, safely avoid static students standing or sitting in the hallway, and quickly reach predefined goals, traveling a total length of 133.40 m and an average speed of 0.41 m/s.

These results demonstrate that our navigation policy is able to generalize to yet another new environment with varying crowd densities and, more importantly, is able to seamlessly transition from simulation to the real world. Furthermore, we see that the trend of maintaining a high irrespective of the crowd density is consistent with the simulations in Sec. IV-C1.

2) Different Environments: We also investigate the ability of our DRL-VO control policy to generalize across different real-world environments. Aside from the indoor hallway environment, we also let the robot navigate through different sequences of 10 goal points in both the indoor lobby and outdoor hallway environments, shown in Fig. 6(b) and Fig. 6(c), respectively. Note that all these three environments are tested under high crowd density, forming two sets of comparative experiments. One is the comparison of the structured indoor lobby and the relatively open indoor hallway. Another is the comparison of the indoor hallway and the outdoor hallway.

Multimedia Extension 3 video and Fig. 11 show the navigation behavior of our robot in the indoor lobby environment, where the robot successfully navigates through 10 goal points and travels a total of 68.96 m at an average speed of 0.37 m/s. Compared with the relatively open indoor hallway environment, this structured lobby environment is more complex and unpredictable, with static chairs, tables, bicycles and trash bins as well as many dynamic pedestrians, resulting in a



(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 12. Robot reactions to moving pedestrians in the outdoor hallway with the high crowd density at different times.

narrow and crowded free space. This structure yielded a more winding nominal path for the robot to follow, which frequently changed as pedestrians cut of different narrow passageways. Despite these complexities, we found no significant navigation differences between these two environments, and our robot was still able to safely and quickly reach its goals. This qualitative comparison demonstrates that our DRL-VO policy is able to generalize to relatively open environments and structured environments in the real world.

We next analyze robot navigation in the outdoor hallway environment. As can be seen from Multimedia Extension 3 and Fig. 12, our robot navigates smoothly in the dense crowds and goes to its predefined goals without any collisions. It travels a total of 102.73 m at an average speed of 0.43 m/s through this open space environment. Note that our outdoor hallway is more open than that indoor hallway, with no static obstacles but many more pedestrians walking around. Compared with navigating in the indoor hallway, the navigation behavior of our robot again shows no significant observable difference. This outdoor experiment indicates that our DRL-VO policy can also work reliably in outdoor environments and it is robust to both indoor and outdoor environments.

One interesting observation from these real-world experiments is the curiosity of the pedestrians, who frequently stopped to observe the robot's behavior, a phenomenon that we did not include in the training data but which the robot was able to easily handle. We also observed that most people actively avoided the robot when they noticed it, giving it a wide berth. Recall we did model this behavior by adding an additional social force component to push people away from the robot and that this repulsion force was stronger than that between two pedestrians.

In summary, these real-world experiments and results demonstrate that our DRL-VO policy can smoothly transfer from simulation to reality and be applied to real-world tasks. We believe there are three main reasons for this. First, different from other works using 2D simulators [6], [26], [30]–[34], [36], [37], [39], [40], we use a 3D simulator to train our control policy, which shrinks the sim-to-real gap by modeling anthropomorphic 3D pedestrian and precise robot kinematics to give more realistic information than the point pedestrian and robot models. Second, we preprocessed the sensor data before inputting it into our navigation policy, turning the raw sensor data into the kinematic and lidar grids. These data structures are less sensitive to the differences between simulation and reality than raw data is (*e.g.,* simulated camera images are much different than real images due to the lack of detailed texture in the environment). Third, the VO-based heading direction reward function is a dense reward and always motivates the robot to actively move both towards goal points and away from potential collisions. All of these factors reduce the gap between simulation and reality, allowing our policy learned entirely in simulation to work in the real world without any retraining or fine-tuning.

V. Highly Constrained Environments

We further test our policy's ability to generalize to other environments and robot models. To do this, we entered our DRL-VO control policy in the ICRA 2022 BARN Challenge [7], where the goal is to navigate through unknown and highly constrained static environments, such as those shown in Fig. 13. The competition consisted of two phases, simulation and hardware.

A. Experimental Setup

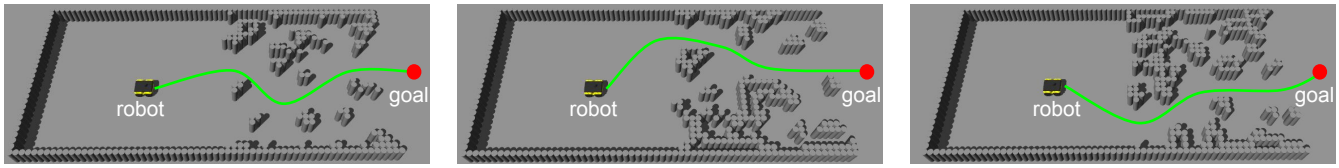
1) Robot Platform: The BARN challenge uses a Jackal robot equipped with a Hokuyo UTM-10LX lidar. Compared to the Turtlebot2 platform used in Sec. IV, the maximum velocity of the Jackal is much greater (2 m/s vs. 0.5 m/s) and the sensing range and angular resolution of the UTM-10LX are both smaller (10 m and 0.375° per beam vs. 30 m and 0.25° per beam, respectively).

2) Simulation: The simulation portion of the challenge used 300 publicly available training Gazebo environments and 50 test environments known only to the competition managers. Each map is filled with cylindrical obstacles generated by the BARN environment generator [61], and the object density varies greatly across environments. The competition evaluated all policies on a desktop computer with an Intel Xeon Gold 6342 CPU @ 2.80 GHz (without GPU support).

3) Hardware: There were three different test environments, each about 10×10 m in size: wide "S", pointed "S", and narrow "S". Unlike the semi-closed BARN simulation environments, these three physical test environments are complex mazes with only one reasonable path, many sharp turns, gap traps, and narrow passages, as Fig. 14 shows. The physical Jackal robot was equipped with an onboard computer with an Intel i3-9100TE CPU and 16 GB RAM memory (with no GPU) running Ubuntu 18.04 and ROS Melodic Morenia.

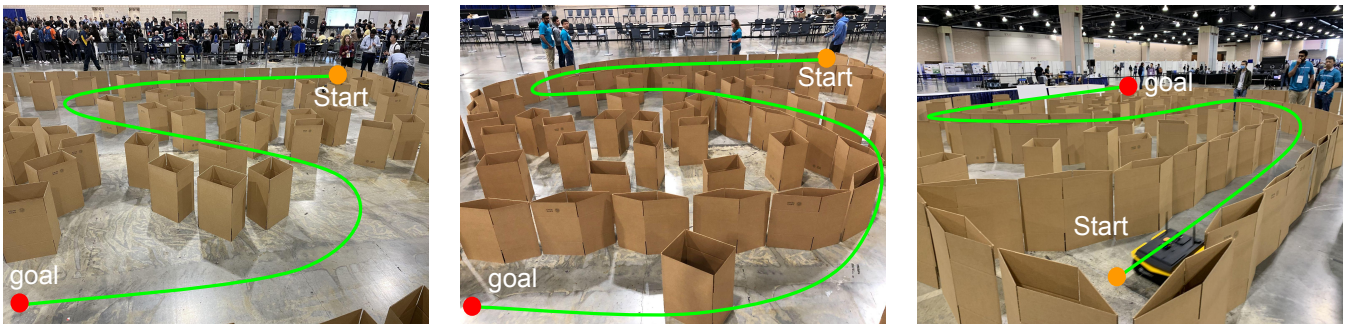
B. Deployment Adjustment

We utilized the same DRL-VO policy from Sec. IV-B (trained using the Turtlebot2) in the BARN Challenge. However, to successfully deploy policy on the Jackal robot and



(a) World 62 (b) World 120 (c) World 285

Fig. 13. Example BARN simulated environments.



(a) Wide "S" shaped environment (b) Pointed "S" shaped environment (c) Narrow "S" shaped environment

Fig. 14. Real-world BARN test environments.

meet the requirements of the BARN challenge, we needed to adjust several settings of our overall system.

First, the BARN challenge maps are all unknown to the robot. Therefore, we removed the localization module (*i.e.,* AMCL) and adapted our navigation system to operate using only odometry (*i.e.,* without a map).

Second, we need to adjust the input observation data $\mathbf{o}^t = [\mathbf{l}^t, \mathbf{p}^t, \mathbf{g}^t]$ of the DRL-VO network. For the lidar data $\mathbf{l}^t$, since the UTM-10LX has only 720 scan points instead of 1080 scan points, we use the full 270° FOV (instead of the 180° FOV for the Turtlebot2) to construct the 80×80 lidar historical map $\mathbf{l}^t$. We set the pedestrian data $\mathbf{p}^t = 0$ since the maps only contain stationary obstacles. Lastly, we use a sub-goal point with a look-ahead distance of 1 m instead of 2 m to generate the goal $\mathbf{g}^t$ as this allows the robots to better avoid obstacles in the highly constrained environments.

Finally, we found that directly mapping the normalized velocity from the control policy to the speed range of the Jackal led to very aggressive behavior. We hypothesize that this is an artifact of the Turtlebot2 being slower than many pedestrians so it nearly always moves at (close to) maximum speed. However, this caused the Jackal to frequently bump into obstacles. To solve this problem, we designed a simple maximum velocity switching mechanism:

$$v_{\max} = \begin{cases} 2\,\mathrm{m/s} & \text{if } d^t_{\mathrm{obs}} \geq d_f \\ 0.5\,\mathrm{m/s} & \text{otherwise,} \end{cases} \tag{15}$$

where $v_{\max}$ is the maximum velocity limit of the robot, d^t_{obs} is the distance to the obstacle closest to the robot at time t, and $d_f (= 2.2\,\mathrm{m})$ is a threshold value. Intuitively, this equation sets the maximum velocity to 2 m/s only when there are no obstacles in front of the robot and uses a slower velocity near obstacles (matching the training velocity). We open-sourced the DRL-VO policy code with these adjustments at https://github.com/TempleRAIL/nav-competition-icra2022-drl-vo.

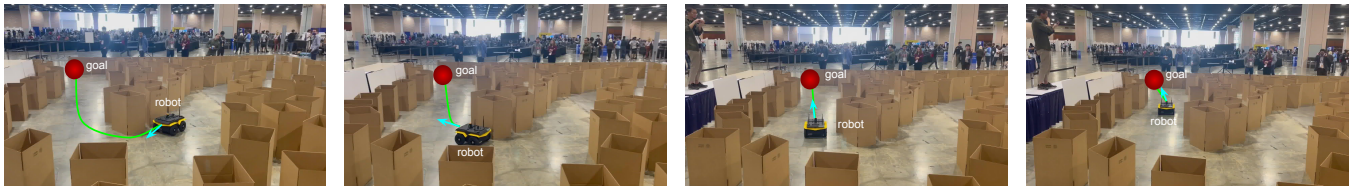
C. Results

We competed in both the simulation and hardware phases of the BARN competition.

1) Simulation: The BARN challenge simulation competition uses 50 unseen Gazebo environments to test navigation policies, running 10 trials for each environment. All trials use the same start and goal points, with the goal 10 m in front of the start, which allows the robot to plan a reasonable nominal path in the absence of an environmental map. Policies were scored using a metric s that considers the success rate, travel time, and environment difficulty and yields a value in the range $[0, 0.25]$ (where 0.25 is best) [61].

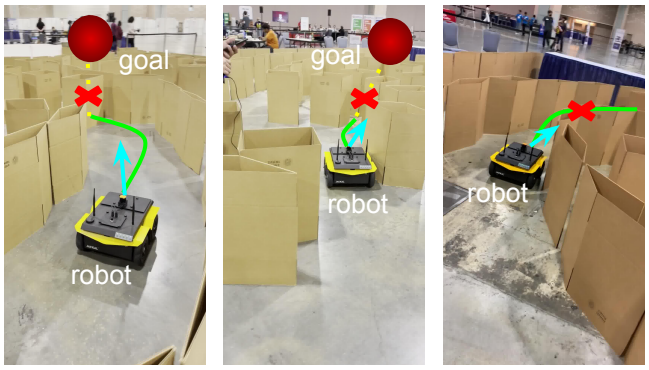
A total of 11 policies (*e.g.,* model-based, learning-based, and hybrid) participated in the BARN challenge simulation competition. Our DRL-VO policy has the average highest navigation score among all policies at 0.2415, and achieved a score of 0.25 on all maps that it successfully completed (meaning the average speed was at least 0.5 m/s). The other competitors' scores ranged from 0.1627 to 0.2334. See Multimedia Extension 4 to see our policy in these highly constrained environments. We did not receive full data about the test environments, but based on our experience in the training environments we saw that failures happened when the robot would just barely bump into an obstacle near very small gaps and sharp turns. We believe that this is because the Jackal has a larger physical size than the Turtlebot2, so the policy likely would have been safe with a Turtlebot2.

2) Hardware: Due to the excellent navigation performance achieved by our DRL-VO policy in the BARN simulation competition, we were invited to participate in the physical competition (along with two other teams, ARML and



(a) t (b) t+1 (c) t+2 (d) t+3

Fig. 15. Successful navigation behavior of the Jackal robot in the wide "S" shaped environment at different times.



(a) Sharp turns (b) Gap traps (c) Narrow passages

Fig. 16. Typical failure cases in the pointed/narrow "S" shaped environment. The yellow dotted lines are the unreasonable nominal paths given by the odometry-based global planner due to the limited FOV of the robot. The red crosses indicate potential collision areas.

DRAGON). The structure of the physical competition was to conduct three runs in each of the three test environments shown in Fig. 14. The team that completed the most runs (out of 9 total) won, with the average time used a tiebreaker. The test environments were generated by the competition managers that day and were not known to competitors beforehand. As we can see, these environments are more challenging than the simulation test environments, with many sharp turns and gap traps where the robot's field of view is highly occluded. Additionally, some of the passageways were so narrow that the Jackal robot could not turn around without hitting a wall.

We placed third in the physical competition, completing 2 out of 9 runs successfully (both in the first environment, Fig. 14(a)). Multimedia Extension 5 and Fig. 15 show one of these successful runs through the first test environment (Fig. 14(a)), which had relatively wide passages. However, we were not able to complete a full run in either of the other two environments due to two primary reasons: 1) there were no reasonable nominal paths to guide the robot through sharp turns or gap traps due to unpredictable obstacles limiting the robot's view, and 2) our DRL-VO policy is a bit aggressive and lacks precise control directives in the narrow passages, as it is trained in an environment without narrow passages and on a robot with a smaller physical size. See Extension 5 and Fig. 16 for examples of these failures. In the future, we plan to alleviate these issues by using a map-based navigation system to give a better nominal global path to the robot and by training our DRL-VO policy in multiple different environments. Additionally, relative to the model-based approaches used by our competition, we could quickly retrain or fine-tune our learning-based model to work in each new environment.

D. Discussion

Participating in the simulated and hardware BARN competition showed that our DRL-VO control policy, which was trained in a single crowded dynamic Lobby environment with 34 pedestrians using a Turtlebot2 robot, was able to generalize to a new navigation task (*i.e.,* highly constrained static environments) and work with different robot platforms and sensor configurations. We believe that this generalizability can be explained by the fact that we use preprocessed data representations (Sec. III-B) rather than the raw sensor data. This creates a level of abstraction between the hardware/environment and the policy that allows it to be highly flexible. We will continue to explore the transferability of our policy in future work, particularly looking at how to account for changes in the physical properties of a robot. Two examples that we encountered in this work were the differences in the physical size of a robot (which we believe led to many of the collisions in the narrow passageways) and in the speed of the robot (since a Jackal robot moving in a crowd of people should vary its speed based on the people rather than going full speed).

VI. Lessons Learned

Neural networks are, in essence, complicated tools to approximate functions. As such, they are particularly well suited to solve problems that are too complex for traditional rule-based methods, such as control policies to enable robots to navigate through dense crowds of pedestrians. There are three primary aspects to a neural network: the input data, the network structure, and the training methodology. In this section, we will discuss some of the broad lessons we have learned from our work in each of the above aspects.

A. Data Representations

We argue that creating intermediate data representations (*i.e.,* hand-crafted features or learned features) from the raw sensor data allows us to learn navigation policies that better generalize to new scenarios than policies that directly use raw sensor data alone. As we conclude in Sec. II-D, most robot navigation methods (*i.e.,* model-based [4], [9]–[15] and learning-based [30]–[39], [41]) utilize the preprocessed intermediate data as the input to their control policies, with most of the learning-based control policies also utilizing learned

intermediate features. However, learning useful intermediate features from raw sensor data requires the use of photorealistic simulators and/or real data, both of which present significant challenges during the learning phase. Additionally, even photorealistic simulators differ (subtly) from reality, which can create a sim-to-real gap as the learned intermediate features depend on the data used to train the network.

Instead, we carefully handcrafted features in an attempt to concisely summarize the salient information needed for navigation while abstracting away the complexities of the real world. In our case, we choose two inputs to encode relevant data about the structure of the environment and the relative kinematics of pedestrians (or other moving objects). Specifically, we compress the lidar data by using minimum and average pooling to encode information about the basic structure of the space. This step allows our policy to avoid overfitting to the specific training environment, based on the findings of [49]. Similarly, we encode the RGB-D camera data by detecting objects from the RGB value, measuring positions using the depth data, and fusing these in the MHT to get information about pedestrian kinematics. Overall, these preprocessing steps form a compact and useful representation that is far less sensitive to pixel-level differences between individual images than policies that use raw sensor data as input. This choice also affects the network design and training methodology, as we will discuss below.

B. Computational Constraints

The goal of much machine learning research is simply to improve accuracy and/or reward. However, when using machine learning in the context of robotics it is critical to remember that the resulting algorithm will be deployed on a mobile robot with limited computational resources. Therefore, in practice, one must often make a trade-off between accuracy/reward and computational efficiency. Based on our experience, we believe that having a short reaction time is more important than using a policy that is incrementally better (in the sense of lower training loss or higher reward in a simulator that runs slower than real-time) but that runs much slower and cannot be processed in real-time. In other words, we believe one should attempt to maximize reward subject to a constraint on processing time rather than the other way around. Future work will study this hypothesis in greater detail.

C. Training Methodology

In a deep reinforcement learning setting, there are two key aspects to training: the environment and the reward design.

1) Training Environment: In general, reinforcement learning must take place in simulation due to the large number of iterations needed for the policy to converge and the need to explore potentially unsafe regions of the control space before convergence. One direction of work in the navigation research community is to develop photorealistic simulators [62]. We agree that these are necessary to learn control policies that use raw images as their input. However, our input data structures provide a level of abstraction between the visual scene and the control policy by encoding the detail (or lack thereof) in the world in a consistent set of representations. This allows us to avoid using a computationally complex simulator and instead have an environment with flat visual textures and identical people-shaped rigid bodies that float along the ground. As a result, each iteration takes less time while still effectively learning a policy that generalizes well to new environments, including going from a simulated world to the real world without the need for any modifications.

Another important aspect of the training environment is the realism of human behavior. There are three types of pedestrian crowd models used by navigation-related researchers: social force-based model [5], [6], [9], [17], velocity obstacle-based model [28], [30]–[34], [36]–[39], and pedestrian dataset-based model [18], [35], [40]. These works, especially those driven by pedestrian datasets, only modeled pedestrian dynamics and completely ignore any human-robot interactions.

However, during our real-world experiments, we observed three distinct types of human reactions to the robot, each of which affect the robot in a different way. The first, and most common, was for pedestrians to slightly adjust their path to avoid the robot but to continue on their way. Second, some pedestrians were curious about our robot and would often stop and watch or follow the robot to observe its behavior. Third, and least commonly, some pedestrians were afraid of the robot and would try to get away from the robot quickly. This list is by no means exhaustive, and there are likely many more human-robot interactions that affect navigation. Instead, this list is meant to demonstrate that approaches that completely ignore human reactions to robots will be less successful and have a larger gap between simulation and reality. Our approach, also used by several other groups [6], [17], is a simplistic one: add an additional social force to repel people and robots away from one another. We believe that one fruitful direction for future work will be to explore the range of human behaviors, determine what effect they have on the robot, and develop methods to (approximately) reconstruct this behavior in a simulated setting.

2) Reward Design: The question of how to design an effective reward function for a specific application is still an open problem. For our robot navigation application, we want the prioritize collision avoidance while still reaching the goal in the shortest time possible. As we previously discussed, most DRL-based navigation policies use sparse reward functions, including a zeroth-order (*i.e.,* distance-based) collision avoidance term and a goal-attainment term, which guide the robot to avoid collisions and reach the goal, respectively. However, these zeroth-order collision avoidance terms force the robot to reactively avoid collisions instead of proactively taking evasive action. This was the main motivating factor in using velocity obstacles to create a first-order collision avoidance term, as the velocity information guides the robot to pick actions that will be safe both now *and in the future*. Our experiments on different crowd densities and unseen environments show that our novel first-order active reward function greatly improves the safety and speed of robot navigation compared to the zero-order passive reward function (DRL-VO vs. DRL in Tables II–IV). Based on these results, another avenue for future exploration will be designing other higher-order reward functions

to improve robots' ability to proactively avoid collisions in highly dynamic scenes.

A secondary motivation for using velocity obstacles was to allow the robot to indirectly learn from successful model-based approaches. For example, controllers based on velocity obstacles can guarantee collision avoidance under the assumption of perfect knowledge of the environment. However, in practice, that assumption is never valid. Therefore, instead of directly applying velocity obstacles to generate control actions, we use it as a reward term to guide behavior that is (at least close to) safe. Another avenue of future work will be to examine the effectiveness of using other model-based control approaches to guide the design of similar reward terms that combine safety and progress towards the goal into a single term.

VII. Conclusion

In this paper, we proposed the DRL-VO control policy to enable autonomous robot navigation in crowded dynamic environments with both static obstacles and dynamic pedestrians, as well as in highly constrained static environments. From the detailed background material analysis, our DRL-VO control policy has two key novelties. First, we propose a new combination of preprocessed data representations, which can work well in crowded dynamic environments with up to 55 pedestrians and bridge the appearance gap between an imperfect simulation and reality. Specifically, the robot fuses a short history of lidar data, current pedestrian kinematics (*i.e.,* position and velocity stored in the two grids), and a sub-goal point. All of this data is represented within the robot's local coordinate frame, making our navigation policy robust to errors in localization that are common in crowded, dynamic environments. Second, we design a novel velocity obstacle-based reward function used to train the policy, which can be easily applied to multi-robot navigation policies. This reward uses first-order information about pedestrians to guide the robot to actively steer towards the goal while avoiding potential future collisions.

We demonstrate that our DRL-VO policy generalizes better and maintains a better balance between collision avoidance and speed to different crowd sizes and different unseen environments than other state-of-the-art model-based, supervised learning-based, and DRL-based policies, achieving an almost constant average speed but with a higher success rate over a series of 3D simulation experiments. We also demonstrate through extensive hardware experiments that our DRL-VO policy overcomes the sim-to-real gap and works reliably in real-world indoor and outdoor environments with different crowd sizes, and that it is able to do so without any fine-tuning or modification. Furthermore, by participating in the ICRA 2022 BARN simulation and real-world challenge competitions and placing 1st and 3rd respectively, we demonstrated that our DRL-VO policy can also generalize well to highly constrained environments, different input sensors, and different types of robots. Finally, we summarize several lessons learned from our work to help other researchers improve the generalizability of learning-based algorithms, choose network architectures for use on mobile robot platforms, bridge the gap between simulation and reality, and design appropriate reward functions.

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] J.-u. Kim, "Keimyung hospital demonstrates smart autonomous mobile robot," https://www.koreabiomed.com/news/articleView.html?idxno=10585, Mar 2021, (Accessed on 08/24/2021).

[2] J. Blyler, "One big 2020 robot trend that's hard to miss," https://www.designnews.com/automation/2020-robot-trend-could-explode-2021-and-beyond, Dec 2020, (Accessed on 08/24/2021).

[3] B. Marr, "Demand for these autonomous delivery robots is skyrocketing during this pandemic," https://www.forbes.com/sites/bernardmarr/2020/05/29/demand-for-these-autonomous-delivery-robots-is-skyrocketing-during-this-pandemic/, May 2020, (Accessed on 08/24/2021).

[4] 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.

[5] 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. [Online]. Available: https://doi.org/10.1109/IROS51168.2021.9636102

[6] 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.

[7] X. Xiao, Z. Xu, Z. Wang, Y. Song, G. Warnell, P. Stone, T. Zhang, S. Ravi, G. Wang, H. Karnan, J. Biswas, N. Mohammad, L. Bramblett, R. Peddi, N. Bezzo, Z. Xie, and P. Dames, "Autonomous ground navigation in highly constrained spaces: Lessons learned from the Benchmark Autonomous Robot Navigation Challenge at ICRA 2022," *IEEE Robotics & Automation Magazine*, vol. 29, no. 4, pp. 148–156, 2022. [Online]. Available: https://doi.org/10.1109/MRA.2022.3213466

[8] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[9] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot interaction contexts," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 5817–5824.

[10] 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.

[11] 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.

[12] 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.

[13] S. H. Arul and D. Manocha, "V-rvo: Decentralized multi-agent collision avoidance using voronoi diagrams and reciprocal velocity obstacles," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8097–8104.

[14] 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.

[15] X. Shen, E. L. Zhu, Y. R. Stürz, and F. Borrelli, "Collision avoidance in tightly-constrained environments without coordination: a hierarchical control approach," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 2674–2680.

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

[17] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1688–1694.

[18] M. Sebastian, S. B. Banisetty, and D. Feil-Seifer, "Socially-aware navigation planner using models of human-human interaction," in *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2017, pp. 405–410.

[19] M. Boldrer, M. Andreetto, S. Divan, L. Palopoli, and D. Fontanelli, "Socially-aware reactive obstacle avoidance strategy based on limit cycle," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3251–3258, 2020.

[20] 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.

[21] 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.

[22] 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.

[23] G. Kahn, P. Abbeel, and S. Levine, "Badgr: An autonomous self-supervised learning-based navigation system," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312–1319, 2021.

[24] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese *et al.*, "Deep local trajectory replanning and control for robot navigation," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 5815–5822.

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

[26] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6252–6259.

[27] 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.

[28] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1140–1146.

[29] X. Huang, H. Deng, W. Zhang, R. Song, and Y. Li, "Towards multi-modal perception-based navigation: A deep reinforcement learning method," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4986–4993, 2021.

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

[31] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3052–3059.

[32] ——, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[33] 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.

[34] 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.

[35] 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.

[36] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 3517–3524.

[37] 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.

[38] 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.

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

[40] P. Xu and I. Karamouzas, "Human-inspired multi-agent navigation using knowledge distillation," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8105–8112.

[41] 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 *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 6057–6063.

[42] R. Han, S. Chen, S. Wang, Z. Zhang, R. Gao, Q. Hao, and J. Pan, "Reinforcement learned distributed multi-robot navigation with reciprocal velocity obstacle shaped rewards," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 5896–5903, 2022.

[43] L. Sun, J. Zhai, and W. Qin, "Crowd navigation in an unknown and dynamic environment based on deep reinforcement learning," *IEEE Access*, vol. 7, pp. 109 544–109 554, 2019.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[45] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[46] S. S. Sohn, H. Zhou, S. Moon, S. Yoon, V. Pavlovic, and M. Kapadia, "Laying the foundations of deep long-term crowd flow prediction," in *European Conference on Computer Vision*. Springer, 2020, pp. 711–728.

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

[48] 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.

[49] M. Pfeiffer, S. Shukla, M. Turchetta, C. Cadena, A. Krause, R. Siegwart, and J. Nieto, "Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423–4430, 2018.

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

[51] D. Feng, C. Haase-Schütz, L. Rosenbaum, H. Hertlein, C. Glaeser, F. Timm, W. Wiesbeck, and K. Dietmayer, "Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1341–1360, 2020.

[52] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[53] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*. PMLR, 2015, pp. 1889–1897.

[54] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample efficient actor-critic with experience replay," *arXiv preprint arXiv:1611.01224*, 2016.

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

[56] 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.

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

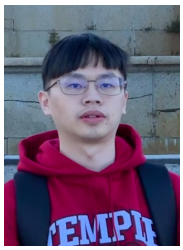
[58] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz, "Experimental study of the behavioural mechanisms underlying self-organization in human crowds," *Proceedings of the Royal Society B: Biological Sciences*, vol. 276, no. 1668, pp. 2755–2762, 2009.

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

[60] Y. Chen, H. Peng, and J. Grizzle, "Obstacle avoidance for low-speed autonomous vehicles with barrier function," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 1, pp. 194–206, 2017.

[61] D. Perille, A. Truong, X. Xiao, and P. Stone, "Benchmarking metric ground navigation," in *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2020, pp. 116–121.

[62] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.



Zhanteng Xie received his B.Eng. degree in Electronic Information Engineering from Zhengzhou University in 2015 and his M.Eng. degree in Information and Communication Engineering from Harbin Institute of Technology in 2018. He is currently working towards his Ph.D. degree at the Temple Robotics and Artificial Intelligence Lab (TRAIL) in the Department of Mechanical Engineering at Temple University. His research interests lie at the intersection of robotics and machine learning, with a focus on environment prediction and autonomous robot navigation in crowded dynamic scenes.

Philip Dames received both his B.S. (summa cum laude) and M.S. degrees in Mechanical Engineering from Northwestern University in 2010 and his Ph.D. degree in Mechanical Engineering and Applied Mechanics from the University of Pennsylvania in 2015. From 2015–2016 he was a Postdoctoral Researcher in Electrical and Systems Engineering at the University of Pennsylvania. Since 2016, Philip has been an Assistant Professor of Mechanical Engineering at Temple University, where he directs the Temple Robotics and Artificial Intelligence Lab (TRAIL), https://sites.temple.edu/trail. He is the recipient of an NSF CAREER award. His research aims to improve robots' ability to operate in complex, real-world environments to address societal needs.