Robotics: Science and Systems 2022
New York City, NY, USA, June 27-July 1, 2022

DiPCAN: Distilling Privileged Information for Crowd-Aware Navigation

Gianluca Monaci, Michel Aractingi, Tomi Silander
NAVER LABS Europe, Grenoble, France
Email: firstname.lastname@naverlabs.com

Abstract—**Mobile robots need to navigate in crowded environments to provide services to humans. Traditional approaches to crowd-aware navigation decouple people motion prediction from robot motion planning, leading to undesired robot behaviours. Recent deep learning-based methods integrate crowd forecasting in the planner, assuming precise tracking of the agents in the scene. To do this they require expensive LiDAR sensors and tracking algorithms that are complex and brittle. In this paper we propose a two-step approach to first learn a robot navigation policy based on privileged information about exact pedestrian locations available in simulation. A second learning step distills the knowledge acquired by the first network into an adaptation network that uses only narrow field-of-view image data from the robot camera. While the navigation policy is trained in simulation without any expert supervision such as trajectories computed by a planner, it exhibits state-of-the-art performance on a broad range of dense crowd simulations and real-world experiments. Video results at https://europe.naverlabs.com/research/dipcan.**

I. Introduction

If robots are to assist humans in everyday tasks, they must be capable to safely navigate in crowded dynamic environments. Traditionally, methods addressing social navigation decouple people motion prediction from robot motion planning [33]. Early approaches consider humans as non-responsive obstacles [5, 17, 56], resulting in unnatural behaviours that cause the robot to, e.g., block human paths [56] or move in a way that surprises the human, who in turn reacts unpredictably creating a short oscillatory interaction, the so-called “reciprocal dance” phenomenon [16]. Subsequent works have proposed to plan robot path after predicting future possible trajectories of nearby humans [3, 49]. These approaches often result in exploding complexity, with possible future trajectories of pedestrians filling the entire space and leading to undesired phenomena such as the “freezing robot” problem [49].

In recent years, promising deep learning-based methods have been developed to predict pedestrian movement and plan robot actions. A large body of work in this area focuses on forecasting trajectories of humans in crowds [25, 35, 55]. Deep Reinforcement Learning (RL) approaches have also been proposed to jointly model crowd motion and robot control [6, 13, 28]. While achieving high accuracy in predicting human paths, these methods rely on precise tracking of the agents in the scene, usually using combinations of RGB-D images and wide field-of-view (FOV) LiDAR information. Therefore, these methods typically face several difficulties when transferring to a robot platform. On the one hand, it is

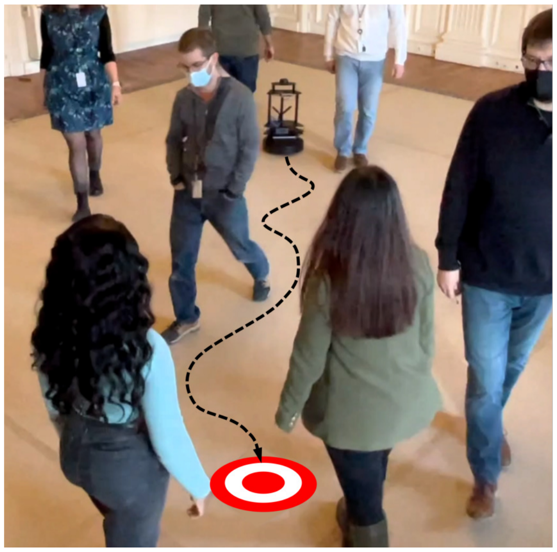


Fig. 1: The proposed DiPCAN architecture controls a robot (top) to reach a target location (bottom) while avoiding dense dynamic crowds using only the robot wheel odometry and narrow FOV depth camera. Videos of the experiments can be viewed at https://europe.naverlabs.com/research/dipcan.

difficult to precisely track moving pedestrians from the robot perspective: occlusions, motion blur, sensing and processing power limitations inevitably degrade the quality of the results, with errors accumulating in the downstream task. On the other hand, it is desirable to use an affordable sensor suite that does not require LiDAR and complex sensor fusion techniques for wide angle, accurate range estimation.

In this work we address the task of point-goal navigation with a LoCoBot robot [2] in a crowded environment (Figure 1). The robot has to reach a target location while avoiding collisions with dynamic and static pedestrians. Following the above desiderata, we aim to achieve that using only the narrow FOV image sensor on the robot. To address this challenging scenario we propose a novel two-step approach inspired by recent works on legged robot locomotion [26, 27]: DiPCAN, Distilling Privileged information for Crowd-Aware Navigation. The proposed system, shown in Figure 2, has three main components: the base policy π, the encoder μ and the adaptation module ϕ. The policy π and encoder μ are trained simultaneously with reinforcement learning in simulation us-

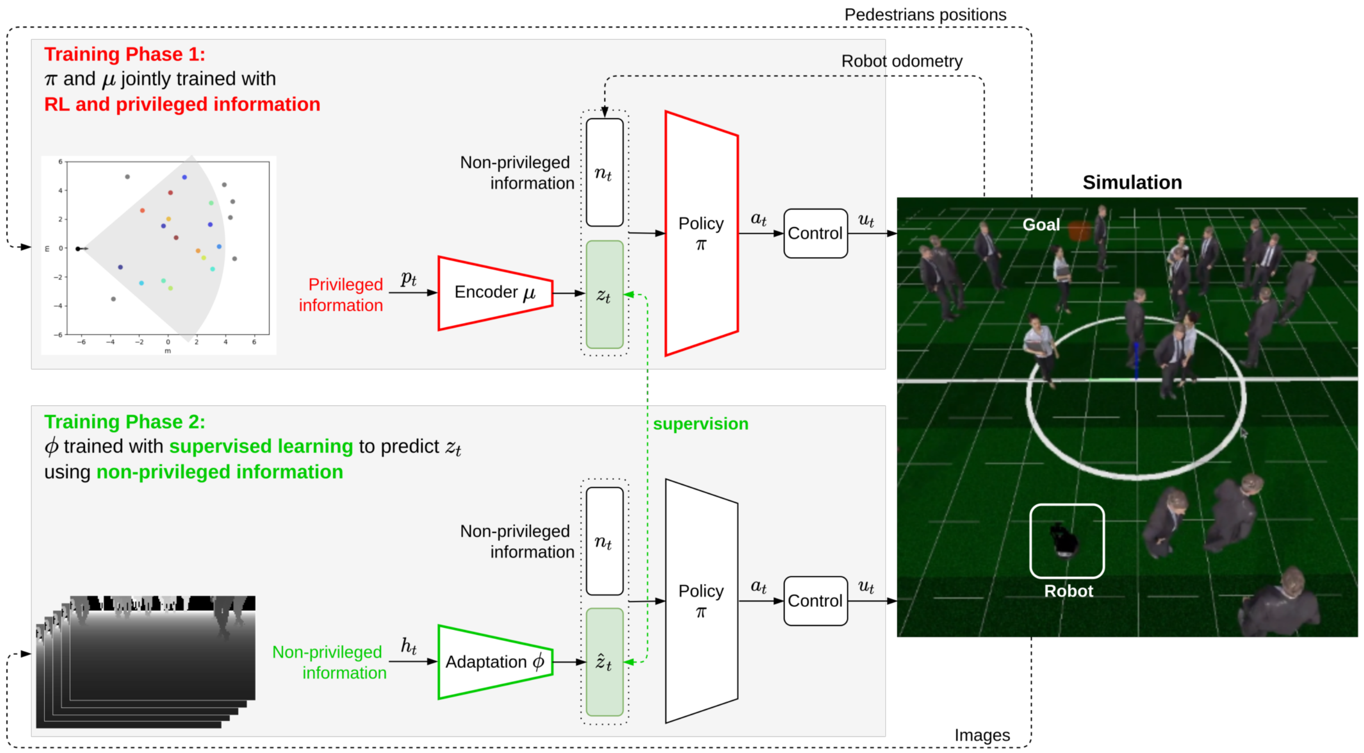


Fig. 2: Proposed DiPCAN system. The robot is trained in **simulation** (right) to reach a goal located across a densely crowded space. In **training phase 1** (top-left), the navigation policy π and the encoder μ are trained together with RL using privileged information about positions of pedestrians, shown on the top-left. The robot is the black dot and the coloured dots pinpoint exact people locations within the FOV of the robot, indicated with a grey cone. The grey dots are pedestrians outside of the FOV of the robot. In **training phase 2** (bottom-left) the adaptation module ϕ is trained using supervised learning to generate $\hat{z}_t$, an estimate of z_t, using only historic image data. On the bottom-left of the figure we show depth images used to train the adaptation network ϕ_D (see Section III-B3). At **deployment**, ϕ generates the latent vector $\hat{z}_t$ which is concatenated with the non-privileged information n_t and fed to the policy π to generate a_t.

ing *privileged information* about exact pedestrians positions, shown on the top left. The encoder μ embeds the privileged information into a low-dimensional feature vector z_t used by the policy π together with information about the robot state to select the next action a_t to take. Privileged information about exact pedestrian positions is hard to obtain at runtime, so the adaptation module ϕ is subsequently trained to generate $\hat{z}_t$ using *non-privileged image information* h_t. Since both images and z_t are available in simulation, the adaptation module ϕ is trained using supervised leaning by minimizing the difference between the embeddings $\hat{z}_t$ and z_t, effectively distilling the knowledge of the network μ into ϕ. At runtime the adaptation module ϕ generates the vector $\hat{z}_t$ that is concatenated with the non-privileged information n_t and fed to the base policy π to generate the robot controls.

The key idea here is to distill the knowledge of the encoder μ into the adaptation network ϕ that learns to reason about pedestrian motion using only historic image data and produces the low-dimensional context vector $\hat{z}_t$. Crucially, we do not need to accurately estimate pedestrians' tracks. However, we are neither training a policy using images in an end-to-end fashion, which is known to be complex because of the high data dimensionality, and hard to transfer because of the large simulation-to-real gap. As pointed out by Eysenbach et al. [14], conventional end-to-end RL agents might use too many bits of information from their environment which are irrelevant for decision making: compressing state information to latent representations relevant to the control policy, as done here by the encoder μ and the adaptation network ϕ, leads to more effective training while achieving superior performance.

To summarize, the main contributions of our work are:

- A new approach to bridge the gap between crowd trajectory prediction and realistic robot control using narrow FOV robot camera. The method is modular and makes it possible to use the encoder module from any state-of-the-art crowd modelling method to learn a context vector from privileged pedestrian position information. The modularity of the architecture allows the use of different adaptation networks and sensor modalities without modifying previously trained policies. Here we propose two adaptation modules, one based on depth images and one on people detections on RGB images.
- An environment and a procedure to train and evaluate our method. We demonstrate the DiPCAN approach in a

broad range of conditions and we benchmark against several baselines, achieving promising results in challenging simulation environments.
- The deployment of the proposed architecture on a LoCoBot robot for real-world navigation experiments.

II. Related Work

A. Navigation with pedestrian collision avoidance

Collision avoidance is the minimum requirement for mobile robots to safely navigate, and as such is a long-standing area of research in robotics. Seminal works on collision avoidance consider only static obstacles [5, 17, 56], leading to well-known undesired phenomena in dynamic environments such as robots freezing or "dancing" with nearby pedestrians. To overcome these limitations, subsequent works include pedestrian motion predictions in their models [3, 12, 49]. These approaches are computationally expensive [12] and predicted pedestrian trajectories quickly fill the entire space leading to the well-documented freezing robot problem [49]. In fact, Dynamic Window Approach (DWA) [17], and modern variants that use similar trajectory roll-out strategies such as Time Elastic Bands [41], are still the de-facto standard local planning methods for mobile robots [32].

Recently several methods based on deep learning have been developed to address the problem of collision avoidance with pedestrians [15, 22, 31, 37, 38, 43]. However, these methods use an expensive wide-FOV LiDAR sensor to accurately estimate the position of neighbouring pedestrians. Few works propose approaches to collision avoidance using imaging modalities with limited FOV, such as depth [8, 46], color images [47, 54], or grayscale images [52]. However, these approaches either do not account for the presence of dynamic obstacles, or consider scenarios with few pedestrians [8, 46, 47].

In contrast to existing methods, we address the problem of navigation with pedestrian collision avoidance in extremely dense crowds using only a narrow FOV imaging sensor.

B. Crowd trajectory prediction

Crowd trajectory modelling and prediction is an active area of research. While only few relevant methods are reviewed here, we refer the interested reader to recent comprehensive surveys on this topic [25, 42].

In their pioneering work, Helbing and Molnar [19] propose to model human interactions in crowds with attractive and repulsive forces and introduce the Social Force model of motion. Another successful model of motion is the Optimal Reciprocal Collision Avoidance (ORCA) model [50], which assumes that all agents in the scene use the same collision avoidance strategy. More recently, the community has focused on deep learning-based approaches [6, 13, 25, 28, 35, 48, 55], which provide flexible and computationally efficient solutions to crowd motion forecasting. Crucially, all these lines of research assume a fully observable state, e.g. the exact positions of all pedestrians in the scene are known [25]. While this is a reasonable assumption for video surveillance use cases where crowd footage is captured by a camera with top-down view over the scene, this limits the applicability of the methods to robotic platforms. Few of these approaches have been demonstrated on a robot [6, 13, 28], but they first need to accurately track pedestrians positions from the robot perspective, usually using a combination of RGB-D images and LiDAR scans.

Data-driven crowd trajectory prediction methods are typically composed of an interaction module to embed crowd dynamics information in a context vector, and a decoder to predict the next pedestrians positions. In our work we use the encoder part of one of the state-of-the-art methods reviewed in [25] to generate the context vector that models crowd motion. In contrast to previous work though, accurate pedestrians positions are only needed to train the encoder and are not used at deployment, when the context vector is estimated using only image data captured by the robot.

C. Learning with privileged information

Learning using privileged information [51] is a paradigm applied to a broad range of problems. Lately it has been successfully used for robot control [7, 23, 45, 47], where a student policy with limited information learns to imitate a teacher that has access to information not available at deployment. Our work differs from these approaches in the fact that we do not rely on an oracle policy, but we use privileged information to model properties of the environment.

Learning with privileged information is closely connected with the network distillation framework [20, 30], where knowledge is compressed and transferred across neural networks by setting the output of one network as target distribution for the other network. Specifically, the use of the adaptation network ϕ (Figure 2 and Section III-C2) relates to *cross-modal distillation* approaches which exploit at training information from a modality unavailable at testing e.g. by predicting this information through an auxiliary loss. For example, Hoffman et al. [21] propose an object detection model with a *hallucination* branch trained to mimic mid-level features from a depth convolutional network using only RGB images. Similar ideas have been applied into several fields such as action recognition [18], semantic segmentation [29] and visual question-answering [11]. While there is a clear connection between these lines of research and the proposed adaptation network, we are not aware of comparable approaches that distill privileged pedestrian trajectory information into an image-based network. Besides, the adaptation network, while important, is only a component of the overall system.

Our approach is more closely related to recent works on legged robot locomotion [26, 27], where an adaptation network that uses historic robot proprioception data is trained to reproduce a context vector representing properties of the environment. The context vector is then processed by a policy to generate robot controls. The idea also draws connections with the Asymmetric Actor Critic [39] approach, where privileged information not available at test time is provided to the critic during training. In the context of robot navigation, Choi et al. [8] propose an Asymmetric Actor Critic agent that uses privileged local map data to train a navigation policy using narrow FOV depth sensor.

III. Methods

A. Preliminaries

We consider the problem of point-goal navigation in crowded environments. The LoCoBot follows differential drive kinematics and its sensed state at time t, s_t, consists of its linear and angular velocities $s_t = [\dot{x}_t, \dot{\psi}_t]$. The goal position is given relative to the initial robot pose as $g = [\Delta x^g, \Delta y^g]$. The robot is controlled with the velocity commands $u_t = [v_t, \omega_t]$, where $v_t \in [-v_{max}, v_{max}]$ controls the linear velocity, and $\omega_t \in [-\omega_{max}, \omega_{max}]$ controls the angular velocity.

The non-privileged information vector at time t is then formed as:

$$n_t = [d_t, \cos(\theta_t), \sin(\theta_t), \dot{x}_t, \dot{\psi}_t], \tag{1}$$

where d_t is the Euclidean distance between the robot position at time t and the goal, θ_t is the angle to the goal relative to the robot's heading ψ_t, and $\dot{x}_t$ and $\dot{\psi}_t$ are the robot's linear and angular velocities. We do not rely on any form of localization to estimate n_t but we assume that precise odometry is available.

The robot is equipped with a front-facing RGB-D camera with resolution 360×640 pixels, and the depth channel is downsampled to 90×160 pixels. The objective of the robot is to get within 0.5 m of the goal location g in the given time budget T without colliding with pedestrians in the scene.

B. Model

The proposed model architecture is inspired by [26], with two main differences: here the encoder μ is specifically designed to capture pedestrians' movements and interactions, and it is derived from a state-of-the-art crowd modelling encoder [48]. The adaptation network ϕ is designed to use high-dimensional image data to reconstruct the context vector $\hat{z}$, as opposed to low-dimensional input data used in [26].

1) Base policy and controller: The base policy π takes as input the vector $n_t \in \mathbb{R}^5$ concatenated with the context vector $z_t \in \mathbb{R}^{16}$, which is computed starting from privileged information at time t. The policy π predicts the next action based on these inputs as:

$$a_t = \pi(n_t, \mu(p_t)). \tag{2}$$

We use five discrete actions, `MOVE_FORWARD`, `MOVE_BACKWARD`, `TURN_LEFT`, `TURN_RIGHT` and `STOP`. The controller converts these actions to robot velocity commands $u_t = [v_t, \omega_t]$ by associating 80% of the LoCoBot maximum linear or angular velocity to the corresponding action. The policy π is implemented as a 3-layer multi-layer perceptron (MLP) with hidden layers of size 32, and it is trained jointly with the encoder μ by maximizing the expected return using the Proximal Policy Optimization (PPO) algorithm [44]. More details about the training procedures will be presented in Section III-C1.

2) Encoder: The encoder μ is responsible for encoding the privileged information p_t about the crowd situation around the robot. As recommended in [25], we use a non-grid based method that uses as input the position, velocity and acceleration of neighbouring pedestrians relative to the robot. We follow the approach proposed in [48] and we form the input $p_t \in \mathbb{R}^{24}$ by concatenating 6-D vectors representing relative position, velocity and acceleration of the $N = 4$ closest neighbours in front of the robot within the horizontal FOV of 90° and closer than 10 m. The 6-D vector corresponding to each pedestrian is embedded using an MLP and these embeddings are then concatenated and fed to another MLP to generate the context vector $z_t \in \mathbb{R}^{16}$. We have selected this architecture because, despite its simplicity, it is among the best-performing methods benchmarked in [25], providing a good trade-off between complexity and performance. While benchmarking the best crowd trajectory prediction model is out of the scope of this work, the modular structure of the proposed approach allows to use any other crowd motion prediction model. In particular, since we use the same data structure of the TrajNet++ baselines, any alternative approach implemented in [25] could be employed.

3) Adaptation network: The privileged information necessary to encode the vector z_t is not available at deployment, hence we propose to estimate $\hat{z}_t$ using the history of sensor data available to the robot through its on-board camera. We train the adaptation network ϕ using supervised learning in the simulator. The network ϕ does not attempt to predict the privileged information p_t, but a low-dimensional representation z_t relevant to the policy π, which is a significantly simpler task and arguably transfers better from simulation to real-world [26]. Indeed, qualitative and quantitative results in Section IV-C demonstrate that our approach leads to better performance than explicitly estimating pedestrians' tracks.

Thanks to the modularity of the approach it is possible to use different adaptation networks without modifying the robot control policy. For example, depending on the sensor configuration available one can define different types of non-privileged information. Here we propose two distinct adaptation models: one based on history of depth images, and a second one that uses the history of people detections in RGB images. We propose these modalities, as opposed to raw RGB images, because we argue that the difference between real and synthetically generated data is smaller in the case of depth images or detections. Since illumination and texture effects are not present in depth images, the reality gap is easier to bridge than that existing between real and synthetic color images. People detectors, like the YOLOv4 [4] used here, are highly optimized for real-world scenes and empirically we observe that they exhibit comparable performance in real and simulated images. Also, since detections represent a higher level of abstraction, intuitively networks trained on those should transfer better from simulation to reality than networks trained on raw images.

The first adaptation network, ϕ_D, is implemented with an architecture inspired by the recent Video Transformer Network

(VTN) [36], which is designed to reduce training and inference runtime while maintaining high performance of Transformer-based methods. The input to ϕ_D is depth images $h_t \in \mathbb{R}^{90\times160}$ (Figure 2, bottom). We use a simple CNN-based 2D spatial backbone similar to the one proposed in [34] for feature extraction from depth images. This network has three layers with [num. input channels, num. output channels, kernel size, stride] equal to $[1, 32, 8, 4], [32, 64, 4, 2], [64, 32, 3, 1]$ respectively. The output is flattened into a 1-D vector which is fed to a temporal attention-based encoder with time window of length 10. The encoder output is processed by an MLP head to generate the vector $\hat{z}_t \in \mathbb{R}^{16}$.

The second adaptation network, ϕ_Y, features an architecture similar to the crowd encoder network μ: people are detected using a pre-trained YOLOv4 [4] model on RGB images captured by the robot camera. A fixed number of $N = 4$ detections with largest bounding box is considered, as we assume the largest bounding boxes to be associated to the closest pedestrians. The detections' bounding box height and width, centroid and confidence are used as input $h_t \in \mathbb{R}^{16}$ and embedded using an MLP. These embeddings are concatenated and fed to another MLP to extract a feature vector that is processed by a temporal attention-based encoder with time window of length 10. The output is processed by an MLP head to generate $\hat{z}_t \in \mathbb{R}^{16}$.

C. Training

The models are trained on the realistic iGibson simulation environment [53] based on the PyBullet physics engine [9] that uses the LoCoBot URDF model and simulates joint actuation. Dynamic pedestrian simulation is inspired by the implementation of the *2022 iGibson Social Navigation challenge* [1] and includes two motion models: pseudo-random time-correlated motion and ORCA model of motion [50].

1) Training Phase 1, Reinforcement Learning: The first training phase consists in learning jointly the base policy π and the encoder network μ. This ensures that the context vector generated by μ contains relevant information for the policy π. We use a random number of human-like agents into the environment. In 20% of the experiments all the agents are static, in 20% they move according to pseudo-random time-correlated motion and on 60% of the experiments they move according to the ORCA policy. In all the experiments with moving pedestrians, a random subset of those (up to 40% of the total number of pedestrians) is static. In all the random movement simulations and in 25% of the ORCA-controlled simulations humans do not perceive the robot, such that we can account for possible non-cooperative behaviours. Humans' target speed is randomized by uniformly sampling it for each pedestrian at each step from $v_{ped} \in [0.2, 1.2]$ m/s. All human agents are initialized in random positions in the environment within a $10\,\text{m} \times 10\,\text{m}$ square and they are given goals in a randomized location in the same area. Once humans reach their destination in the environment they are assigned new random goals. The robot start and goal locations are randomized but constrained to be at a minimum distance of 12 meters on the opposite sides of the $10\,\text{m} \times 10\,\text{m}$ square where the pedestrians move.

We train the networks using the Proximal Policy Optimization (PPO) algorithm [44] to maximize the expected return:

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_t \right], \tag{3}$$

where T is the episode's horizon, $\gamma = 0.99$ is the discount factor and the reward r_t at each time step is computed as:

$$r_t = \begin{cases} 0.5 & \text{if success;} \\ -0.5 & \text{if collision;} \\ -0.2 \cdot {r_t}^v + 0.1 \cdot {r_t}^p & \text{otherwise.} \end{cases} \tag{4}$$

An episode is successful if the robot gets within 0.5 m of the goal location within the time budget $T = 1200$ steps. ${r_t}^v$ is the space violation term and equals 1 if the robot gets closer than 0.5 m to a pedestrian and 0 otherwise. ${r_t}^p$ is the potential reward that encourages the robot to move towards the goal, and it is computed as ${r_t}^p = d_t - d_{t-1}$, where d_t is the Euclidean distance between the robot and the goal at time t.

Our agents have to be able to cope with a range of crowd densities. If the models are trained with a fixed number of pedestrians we observe that they cannot scale effectively to denser crowds. However, if the density of pedestrians in the scene is randomized over a broad range of values, convergence is slow and there is the risk to incur in catastrophic forgetting. To overcome these limitations a standard approach, used also in related works [26, 28], is to gradually increase the complexity of the training environment. Using a curriculum we train the networks π and μ for 10 million steps with a learning rate of $1e-4$ on environments with the number of pedestrians $\tilde{n}_{ped}$ sampled from a uniform random distribution in the interval $[0.7 \cdot n_{ped}, 1.3 \cdot n_{ped}]$. The training starts with $n_{ped} = 5$ and the number of pedestrian increases linearly every $600,000$ steps up until $n_{ped} = 20$.

2) Training Phase 2, Supervised Learning: In this second phase we train the adaptation module. We use either the past images captured by the robot depth camera or the past pedestrian detections as input, and z_t given by the encoder μ as supervising signal. Both are available in simulation, so the adaptation network ϕ can be trained via supervised learning to minimize: $\text{MSE}(z_t, \hat{z}_t) = \|z_t - \hat{z}_t\|_2$, where $z_t = \mu(p_t)$. The optimization process is run for 5 million steps, during which the MSE loss is minimized using the Adam optimizer [24] with a learning rate of $1e-5$. We apply geometric and appearance image augmentation [40] both to RGB and depth images to increase the robustness of the learned networks and facilitate transfer to the real robot.

IV. Experiments

We test our proposed approach, DiPCAN, on the point-goal task in environments with moving pedestrians. As mentioned in Section III-B3, we implement two adaptation networks that lead to two versions of DiPCAN. **DiPCAN-D** uses the adaptation network ϕ_D, trained using past depth images, together with the base policy π trained during training phase 1. **DiPCAN-Y** uses ϕ_Y, trained on past person detections, with the same base policy π.

A. *Baselines*

We compare DiPCAN-D and DiPCAN-Y to the following baselines:

1) DWA: The Dynamic Window Approach (DWA) [5, 17] is the default local planner in the ROS Navigation stack. Input to DWA are the goal coordinates and the exact positions of pedestrians within the robot FOV. At each step DWA rolls out a set of feasible trajectories and selects robot commands $u_t = [v_t, \omega_t]$ to execute the trajectory that maximizes robot velocity, clearance and proximity to the goal.

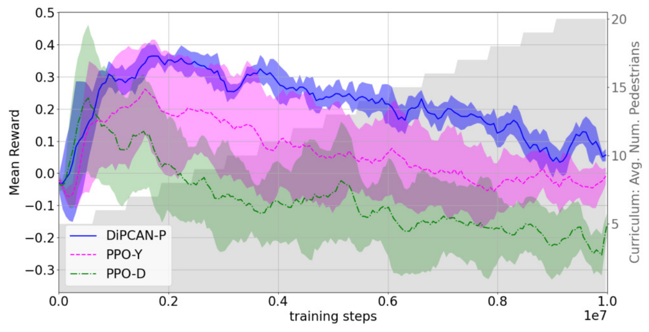
2) DiPCAN-P: The architecture depicted in Figure 2 (top) uses privileged information p_t about exact pedestrians position as input to the encoder μ to generate z_t. This model does not use any adaptation network and exploits privileged information unavailable at testing, thus representing an upper bound for DiPCAN. In practice this architecture is very similar to the method proposed in [28] for the pedestrian-only case that assumes perfect pedestrian localization and tracking.

3) Recon-D: This baseline uses the DiPCAN-P architecture just described and a reconstruction network ρ_D with the same VTN architecture of ϕ_D to generate $\hat{p}_t$, an estimate of the privileged pedestrian information p_t. ρ_D is trained using the same procedure of ϕ_D (Section III-C2). Performance comparison between DiPCAN-D and Recon-D sheds light on the importance of the intermediate low-dimensional embedding z_t and substantiates the claim that it is easier to estimate such embedding instead of the privileged information.

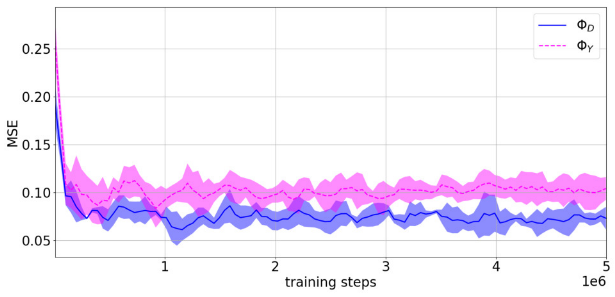
4) Recon-Y: Similarly, it uses the DiPCAN-P architecture where the privileged information $\hat{p}_t$ is estimated through a reconstruction network ρ_Y with the same architecture of ϕ_Y.

5) PPO-D: This agent has the same base policy architecture of DiPCAN (Section III-B1). The input to the policy is n_t concatenated with a context vector $\hat{z}_t$ generated using a sequence of depth images embedded by network ψ_D which has the same VTN architecture of the adaptation network ϕ_D (Section III-B3). The policy and embedding network ψ_D are jointly trained in one phase using PPO under the same training conditions and parameters as in our proposed approach. PPO-D can be seen as a standard end-to-end approach that does not use privileged information about pedestrian positions to generate the vector $\hat{z}_t$. The comparison of DiPCAN-D and PPO-D performances provides insights about the importance of privileged information to condition the learning of the context vector.

6) PPO-Y: This baseline has the same base policy architecture of DiPCAN. The input to the policy is n_t concatenated with the context vector generated using YOLO detections of pedestrians by ψ_Y, a network with the same architecture of our proposed ϕ_Y adaptation network (Section III-B3). Again, the difference between DiPCAN-Y and PPO-Y is that the latter is trained in one end-to-end phase without using privileged information about pedestrian positions to generate $\hat{z}_t$.



(a) PPO Rewards



(b) MSE in *Training Phase 2*

Fig. 3: (a) Average reward over 5 random seeds (left y-axis) for DiPCAN-P (blue), PPO-Y (magenta) and PPO-D (green). The graphs show the average and standard deviation (shaded areas) of the reward over 10 M training steps. The grey area shows the curriculum progression with average number of pedestrians increasing from 5 to 20 (right y-axis). (b) Average MSE and standard deviation (shaded areas) over 5 random seeds in *Training Phase 2* for ϕ_D (blue) and ϕ_Y (magenta).

B. *Analysis of the learned components*

1) RL policies: Training control policies using RL is notoriously complex [14]. We argue that using privileged information in DiPCAN to form a low-dimensional feature vector to guide the policy π has a positive impact on training. To study this hypothesis we analyze the training curves for the PPO-D and PPO-Y models and we compare with those of DiPCAN-P, the model trained with RL in *Training Phase 1* (Section III-C1). The average reward (solid line) and its standard deviation (shaded) over 5 random seeds over the course of training for the three architectures are shown in Figure 3a. Rewards are recorded every $50,000$ training steps by sampling the PPO policies, and the learning curves are smoothed by averaging over a 5-point window. The grey shaded area on the background shows the progression of the curriculum during training, with the average number of pedestrians on the scene linearly increasing from 5 to 20 every $600,000$ steps.

DiPCAN consistently achieves better overall performance than the other architectures. The guidance provided by privileged information forms a prior that leads to substantially stabler training and higher average rewards. PPO-D, trained with depth images, learns reasonable policies at the beginning of training, when very few pedestrians are present in the scene.

Method	10 pedestrians				20 pedestrians				30 pedestrians			
	SR ↑	STL ↑	PSC ↑	CR ↓	SR ↑	STL ↑	PSC ↑	CR ↓	SR ↑	STL ↑	PSC ↑	CR ↓
DWA	0.66	0.2369	**0.9983**	**0.12**	0.42	0.1408	**0.9950**	**0.14**	0.23	0.0746	**0.9920**	**0.22**
Recon-D	0.70	0.4918	0.9915	0.27	0.52	0.3399	0.9829	0.46	0.37	0.2269	0.9737	0.62
Recon-Y	0.39	0.2623	0.9875	0.33	0.32	0.1945	0.9832	0.45	0.23	0.1257	0.9769	0.57
PPO-D	0.36	0.1298	0.9934	0.26	0.24	0.0814	0.9905	0.41	0.18	0.0608	0.9836	0.51
PPO-Y	0.72	0.3569	0.9962	**0.12**	0.50	0.2199	0.9916	0.22	0.35	0.1437	0.9875	0.36
DiPCAN-D	**0.84**	**0.5643**	0.9960	**0.12**	**0.72**	**0.4394**	0.9918	0.21	**0.55**	**0.3164**	0.9832	0.35
DiPCAN-Y	0.76	0.5313	0.9918	0.22	0.59	0.3907	0.9040	0.37	0.42	0.2557	0.9754	0.52
DiPCAN-P	0.86	0.5836	0.9945	0.11	0.73	0.4532	0.9935	0.18	0.61	0.3462	0.9866	0.26

TABLE I: Average performance of our method and baselines over 2000 experiments in environments with 10, 20 and 30 pedestrians. Best performance is highlighted in **bold** and the second-best in blue. Baselines and experimental settings are described in Section IV-A and IV-C respectively. The proposed DiPCAN-D and -Y approaches outperform the alternatives while exhibiting small performance degradation compared to DiPCAN-P which has access to privileged information.

With the number of pedestrians increasing, PPO-D fails to converge to a meaningful policy. PPO-Y, which is trained starting from much lower dimensional features representing people detections, exhibits significantly better performance than PPO-D, however the variance of the obtained rewards is large and rewards are lower than DiPCAN across all seeds.

2) Adaptation networks: For each of the five DiPCAN-P models we train adaptation networks ϕ_D and ϕ_Y distilling the corresponding μ encoders. Figure 3b shows the average and standard deviation (shaded areas) of the MSE achieved during *Training Phase 2* for ϕ_D (blue) and ϕ_Y (magenta). The adaptation network using depth images consistently approximate the embedding z_t with lower error, leading to better performance of the corresponding DiPCAN-D model, as shown in Table I.

C. Simulation experiments

We test the proposed method and the baselines on environments with an average number of pedestrians $n_{ped} \in \{10, 20, 30\}$ with $n_{ped} = 30$ featuring significantly denser crowds than those encountered during training. The experimental setting is the same as for training: in 20% of the experiments agents are static, in 20% they move according to pseudo-random motion and in 60% of the experiments they move according to the ORCA policy. In all the experiments a random subset of pedestrians is static, and in all the random movement simulations and 25% of the ORCA-controlled simulations the humans do not perceive the robot. The pedestrians' target speed is sampled over a broader range than at training, with $v_{ped} \in [0.1, 1.4]$ m/s.

In order to quantify the performance of the navigation agents, we compare them based on the following metrics:

- **Success Rate (SR)**, the percentage of episodes in which the robot succeeds to reach the goal without collision.
- **Success weighted by Time Length (STL)** [1], computed as:
$$\text{STL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{p_i}{\max(p_i, t_i)},$$
where N is the number of episodes, S_i is a binary indicator of success in episode i and t_i is the length of the path taken by the agent in the episode. In [1] all pedestrians follow an ORCA policy and p_i is the time taken by an ORCA agent to reach the goal. Since not all our experiments involve ORCA agents and there is no obvious way to set an oracle policy, we fix the value of p_i to 400, which is a lower bound of the timesteps taken by any agent tested in our experiments to reach the goal.
- **Personal Space Compliance (PSC)** [1], the percentage of time steps in which the robot complies with pedestrians' personal space set as 0.5 m radius around each human-like agent.
- **Collision Rate (CR)**, the percentage of episodes in which the robot collides with a pedestrian.

The intuition behind the use of these metrics is that an agent needs to balance between taking as little time as possible to reach the goal (high STL) and incurring less personal space violation to the pedestrians (high PSC), while avoiding collisions (low CR).

Table I shows results for the proposed methods, DiPCAN-D and DiPCAN-Y, and for the baselines presented in Section IV-A in environments with an average of 10, 20 and 30 pedestrians. For each setting we perform the same 400 experiments. For DWA, scores are averaged over these 400 experiments. For the other methods, the reported values are averaged over 2000 experiments, i.e. 400 runs for each of the randomly initialized policies in Figure 3a: PPO-D, PPO-Y and DiPCAN-P. Recon-D and Recon-Y use the DiPCAN-P architecture and estimate the privileged input using the networks ρ_D and ρ_Y respectively. DiPCAN-D and DiPCAN-Y use the base policy π of DiPCAN-P, but the context vector is generated by ϕ_D and ϕ_Y respectively. Best performance is highlighted in bold and second-best in blue.

The proposed approaches, DiPCAN-D and DiPCAN-Y, clearly outperform alternative baselines in terms of success rate and speed to get to the goal, while achieving a good compromise in terms of safety and pedestrian comfort. Remarkably, performance degradation compared to DiPCAN-P is small, especially for DiPCAN-D that uses depth images. DiPCAN-D achieves better navigation metrics than DiPCAN-Y, arguably because the adaptation network ϕ_D

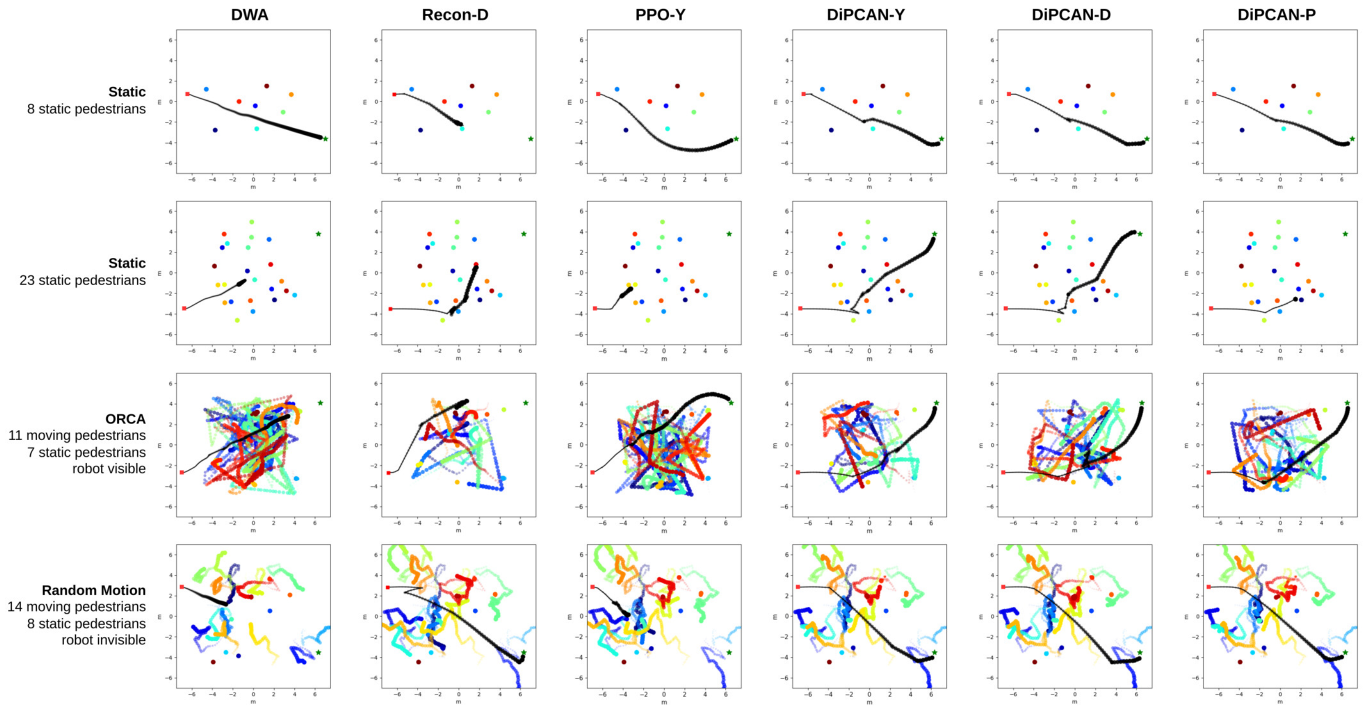


Fig. 4: Example results of robots controlled by DWA (column 1), Recon-D (column 2), PPO-Y (column 3) and the proposed DiPCAN-Y, -D and -P agents (columns 4, 5 and 6 respectively) navigating from their start position (🟥 on the left side of each sub figure) to the goal (★ on the right). The robot trajectory is drawn in black while pedestrians trajectories are colored, with thickness and opacity increasing with time. Each row shows results for a different experimental setup, detailed on the left.

approximates the context vector better than ϕ_Y (see Figure 3b).

As mentioned earlier, DiPCAN-P is very similar to the approach proposed by Liu et al. [28], and performance is indeed in line with one reported in [28], where their state-of-the-art navigation method that uses exact pedestrian tracking achieves 87% success rate on environments with on average 10 pedestrians.

As expected, DWA, that uses privileged exact information about pedestrians positions, runs a conservative strategy with low collision rates, but it clearly fails to successfully navigate in the more intricate scenarios.

Recon-D and Recon-Y make use of the DiPCAN-P model, thus the performance degradation is caused by estimation errors of pedestrian state estimation that propagate downstream in unpredictable ways, leading to low performance and notably high collision rates.

As pointed out in Section IV-B, we did not manage to train a meaningful control policy using the end-to-end PPO-D model. This is reflected in the poor results displayed in Table I. PPO-Y achieves significantly better scores than PPO-D, but still average performance is lower than that of DiPCAN methods, especially in complex scenarios. From these results it appears clear that capturing the underlying crowd dynamics using the context vector and learning to reproduce such context from images brings considerable benefits.

Example results of the baselines and proposed approaches are shown in Figure 4. We omit results for Recon-Y and PPO-D because these agents do not lead to meaningful control policies. In each panel the robot trajectory is drawn in black while pedestrian trajectories are colored, and their thickness and opacity increase with time. Colored dots pinpoint the positions of static pedestrians. The robot start and target positions are indicated with a red square and a green star respectively. The first row shows trajectories for a relatively simple environment with 8 static human-like agents. All agents but Recon-D (column 2) reach the goal: DWA (column 1) follows a very efficient trajectory, PPO-Y (column 3) reaches the target with a slightly longer path, and the DiPCAN models (columns 4, 5 and 6) follow closely the optimal DWA trajectory. Recon-D collides with a pedestrian, likely because of a distance estimation error. This example shows the drawback of the "classic" pipeline that first estimates the state of neighbouring obstacles and then plans the robot motion based on potentially faulty estimates. The second row displays results for an environment with 23 static pedestrians. This time only the DiPCAN-D and DiPCAN-Y agents manage to reach the goal, while surprisingly the policy with access to privileged information about exact people locations, DiPCAN-P, gets the robot stuck in between several pedestrians. On average DiPCAN-P performs slightly better than its non-privileged counterparts (see Table I), but it is interesting to observe the skills that the DiPCAN agents learn for navigating in such complex setting. The third row of figures shows results for an experiment with pedestrians moving according to an ORCA policy. The DiPCAN agents swiftly reach the goal, the PPO-Y agent achieves the result in a significantly longer time. The cautious DWA agent does not reach the goal in the assigned time budget of 1800 steps while the Recon-D agent collides with a moving pedestrian while backtracking. The fourth row displays trajectories for an environment with randomly-moving human agents that do not see the robot. Again the DiPCAN agents quickly get to the target location. DWA and PPO-Y struggle to navigate is such complex scenario and ultimately collide with uncooperative pedestrians that do not perceive the robot. Recon-D reaches the goal in this example, but performs an apparently unmotivated backing motion at the beginning of the trajectory. Additional results for the proposed DiPCAN-D agent in challenging scenarios with an average of 30 pedestrians can be viewed at https://europe.naverlabs.com/research/dipcan.

D. Experiments with noisy odometry

The proposed approach makes use of ground-truth robot odometry available in simulation. Since location, velocity and acceleration of obstacles are not explicitly estimated, the odometry precision should not have a significant impact on navigation results. Additionally, the trained system optimizes the minimum amount of information necessary for taking decisions and should be little sensitive to faulty odometry. To validate these intuitions we test DiPCAN-D on noisy environments. Gaussian noise $\mathcal{N}(0, \sigma_v^2)$ proportional to the robot velocities is added to ground-truth readings, with $\sigma_v = m_v \cdot v$, v the exact linear or angular velocity and $m_v \in \{0.1, 0.2\}$. Gaussian noise with standard deviation $\sigma_l = 30$ cm is also added to the robot ground-truth xy position. Using the same protocol described above we run 2000 experiments on environments with 20 pedestrians. Average performance of DiPCAN-D with exact and noisy odometry, displayed in Table II, shows that navigation metrics are not degraded by severe odometry perturbations.

Noise conditions	SR ↑	STL ↑	PSC ↑	CR ↓
No noise	0.72	0.4394	0.9918	0.21
$\sigma_v = 0.1 \cdot v$, $\sigma_l = 30$ cm	0.72	0.4442	0.9870	0.21
$\sigma_v = 0.2 \cdot v$, $\sigma_l = 30$ cm	0.73	0.4488	0.9908	0.21

TABLE II: Average performance of DiPCAN-D using ground-truth (first row) and noisy odometry (second and third rows).

E. Robot experiments

The proposed DiPCAN-D architecture is deployed and demonstrated on a LoCoBot robot [2]. As shown in Figure 1, the robot has to reach a target position located at a distance of 6 m along the robot x axis with respect to the starting robot pose. The system requires two inputs: n_t (Eq. (1)) and depth images that are processed by the adaptation network ϕ_D to generate $\hat{z}_t$. The vector n_t is computed using the robot wheel odometry, while depth images are captured by the onboard Intel RealSense D435 RGB-D camera with resolution 360×640 pixels and downsampled to 90×160. All computations are executed on the embedded Intel NUC with an i3-8109U CPU and 8GB Ram. We use the same π and ϕ_D networks trained in simulation and we only manually tune the velocity values that the controller associates to discrete actions. Additionally, velocity commands are smoothed by an exponential moving average filter with $\alpha = 0.3$ to achieve a safe robot behaviour. Robot controls are sent at a frequency of 10 Hz.

The videos at https://europe.naverlabs.com/research/dipcan show the behaviour of the LoCoBot controlled by the DiPCAN-D architecture in different settings. The navigation experiments demonstrate how the learned policy drives the robot towards its goal across dense crowds. The robot exhibits a range of behaviours to deal with these complex scenarios: it patiently waits and grabs opportunities to get clear paths, turns around to find promising trajectories towards the goal, backs off to leave space for pedestrians and avoids collisions even when people suddenly block its path.

V. Conclusions

In this work we present a novel architecture for pedestrian collision avoidance in dense crowds using only narrow FOV camera inputs. The navigation policy is trained distilling privileged information about pedestrian positions in a low-dimensional context vector that is reconstructed at test time using image data from the robot onboard camera. The policies are learned from scratch without expert supervision such as predefined trajectories computed by a planner, and they exhibit state-of-the-art performance and remarkable navigation competence on a broad range of dense crowd experiments in simulation and real-world. The approach is modular and allows for further exploration of different crowd trajectory modelling techniques or adaptation network strategies without having to modify other parts of the system.

One obvious limitation of the proposed approach is that it only works in open spaces. A reliable, realistic navigation agent needs to be able to cope equally well with pedestrians and with environmental elements such as walls, furniture and other obstacles. One promising direction that we plan to explore is to combine multiple navigation experts, learned or not, using a hybrid modular approach such as the one proposed by Dashora et al. [10].

References

[1] iGibson challenge 2022. URL http://svl.stanford.edu/igibson/challenge.html.

[2] LoCoBot, an open source low cost robot. URL http://www.locobot.org/.

[3] Maren Bennewitz, Wolfram Burgard, Grzegorz Cielniak, and Sebastian Thrun. Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31–48, 2005.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[5] Wolfram Burgard, Armin B Cremers, Dieter Fox, Dirk Hähnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner,

and Sebastian Thrun. The interactive museum tour-guide robot. In *Aaai/iaai*, pages 11–18, 1998.

[6] Changan Chen, Yuejiang Liu, Sven Kreiss, and Alexandre Alahi. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6015–6022. IEEE, 2019.

[7] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020.

[8] Jinyoung Choi, Kyungsik Park, Minsu Kim, and Sangok Seok. Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5993–6000. IEEE, 2019.

[9] Erwin Coumans and Yunfei Bai. PyBullet, a python module for physics simulation in robotics, games and machine learning, 2016–2019. URL https://pybullet.org/.

[10] Nitish Dashora, Daniel Shin, Dhruv Shah, Henry Leopold, David Fan, Ali Agha-Mohammadi, Nicholas Rhinehart, and Sergey Levine. Hybrid imitative planning with geometric and predictive costs in off-road environments. *NeurIPS Deep Reinforcement Learning Workshop*, 2021.

[11] Tuong Do, Thanh-Toan Do, Huy Tran, Erman Tjiputra, and Quang D Tran. Compact trilinear interaction for visual question answering. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 392–401, 2019.

[12] Noel E Du Toit and Joel W Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2011.

[13] Michael Everett, Yu Fan Chen, and Jonathan P How. Collision avoidance in pedestrian-rich environments with deep reinforcement learning. *IEEE Access*, 9:10357–10377, 2021.

[14] Ben Eysenbach, Russ R Salakhutdinov, and Sergey Levine. Robust predictable control. *Advances in Neural Information Processing Systems*, 34, 2021.

[15] Tingxiang Fan, Xinjing Cheng, Jia Pan, Pinxin Long, Wenxi Liu, Ruigang Yang, and Dinesh Manocha. Getting robots unfrozen and unlost in dense pedestrian crowds. *IEEE Robotics and Automation Letters*, 4(2):1178–1185, 2019.

[16] Franck Feurtey. Simulating the collision avoidance behavior of pedestrians. *Master's thesis, University of Tokyo, Department of Electronic Engineering*, 2000.

[17] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[18] Nuno C Garcia, Pietro Morerio, and Vittorio Murino. Modality distillation with multiple stream networks for action recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 103–118, 2018.

[19] Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[20] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *NIPS Deep Learning and Representation Learning Workshop*, 2014.

[21] Judy Hoffman, Saurabh Gupta, and Trevor Darrell. Learning with side information through modality hallucination. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 826–834, 2016.

[22] Jun Jin, Nhat M Nguyen, Nazmus Sakib, Daniel Graves, Hengshuai Yao, and Martin Jagersand. Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6979–6985. IEEE, 2020.

[23] Elia Kaufmann, Antonio Loquercio, René Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.040.

[24] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015*, 2015. URL http://arxiv.org/abs/1412.6980.

[25] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[26] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems XVII, Virtual Event, July 12-16, 2021*, 2021.

[27] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5 (47), 2020.

[28] Lucia Liu, Daniel Dugas, Gianluca Cesari, Roland Siegwart, and Renaud Dubé. Robot navigation in crowded environments using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5671–5677. IEEE, 2020.

[29] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019.

[30] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. In *4th International Conference on Learning Representations, ICLR 2016*, 2016. URL https://arxiv.org/abs/1511.03643.

[31] Björn Lütjens, Michael Everett, and Jonathan P How. Safe reinforcement learning with model uncertainty estimates. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8662–8668. IEEE, 2019.

[32] Steve Macenski, Francisco Martín, Ruffin White, and Jonatan Ginés Clavero. The marathon 2: A navigation system. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2718–2725. IEEE, 2020.

[33] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core Challenges of Social Robot Navigation: A Survey. *ArXiv*, Mar 2021. URL https://arxiv.org/abs/2103.05668v2.

[34] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. URL http://dx.doi.org/10.1038/nature14236.

[35] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020.

[36] Daniel Neimark, Omri Bar, Maya Zohar, and Dotan Asselmann. Video transformer network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3163–3172, October 2021.

[37] Claudia Pérez-D'Arpino, Can Liu, Patrick Goebel, Roberto Martín-Martín, and Silvio Savarese. Robot navigation in constrained pedestrian environments using reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1140–1146. IEEE, 2021.

[38] Mark Pfeiffer, Michael Schaeuble, Juan Nieto, Roland Siegwart, and Cesar Cadena. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1527–1533. IEEE, 2017.

[39] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *RSS*, 2018.

[40] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: an open source differentiable computer vision library for pytorch. In *Winter Conference on Applications of Computer Vision*, 2020. URL https://arxiv.org/pdf/1910.02190.pdf.

[41] Christoph Rösmann, Frank Hoffmann, and Torsten Bertram. Integrated online trajectory planning and optimization in distinctive topologies. *Robotics and Autonomous Systems*, 88:142–153, 2017.

[42] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020.

[43] Adarsh Jagan Sathyamoorthy, Jing Liang, Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha. Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11345–11352. IEEE, 2020.

[44] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[45] Maks Sorokin, Jie Tan, C. Karen Liu, and Sehoon Ha. Learning to navigate sidewalks in outdoor environments, 2021.

[46] Lei Tai, Jingwei Zhang, Ming Liu, and Wolfram Burgard. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1111–1117. IEEE, 2018.

[47] Varun Tolani, Somil Bansal, Aleksandra Faust, and Claire Tomlin. Visual navigation among humans with optimal control as a supervisor. *IEEE Robotics and Automation Letters*, 6(2):2288–2295, 2021. doi: 10.1109/LRA.2021.3060638.

[48] Antoine Tordeux, Mohcine Chraibi, Armin Seyfried, and Andreas Schadschneider. Prediction of pedestrian dynamics in complex architectures with artificial neural networks. *Journal of intelligent transportation systems*, 24(6):556–568, 2020.

[49] Pete Trautman, Jeremy Ma, Richard M Murray, and Andreas Krause. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015.

[50] Jur Van Den Berg, Stephen J Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics research*, pages 3–19. Springer, 2011.

[51] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural networks*, 22(5-6):544–557, 2009.

[52] Patrick Wenzel, Torsten Schön, Laura Leal-Taixé, and Daniel Cremers. Vision-based mobile robotics obstacle avoidance with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[53] Fei Xia, William B Shen, Chengshu Li, Priya Kasimbeg, Micael Edmond Tchapmi, Alexander Toshev, Roberto Martín-Martín, and Silvio Savarese. Interactive Gibson benchmark: A benchmark for interactive navigation in cluttered environments. *IEEE Robotics and Automation Letters*, 5(2):713–720, 2020.

[54] Linhai Xie, Sen Wang, Andrew Markham, and Niki Trigoni. Towards monocular vision based obstacle avoidance through deep reinforcement learning. In *RSS 2017 workshop on New Frontiers for Deep Learning in*

Robotics, 2017.

[55] Dapeng Zhao and Jean Oh. Noticing motion patterns: A temporal cnn with a novel convolution operator for human trajectory prediction. *IEEE Robotics and Automation Letters*, 6(2):628–634, 2020.

[56] Brian D Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J Andrew Bagnell, Martial Hebert, Anind K Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009.