2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

DenseCAvoid: Real-time Navigation in Dense Crowds using Anticipatory Behaviors

Adarsh Jagan Sathyamoorthy[1], Jing Liang[1], Utsav Patel, Tianrui Guan, Rohan Chandra, and Dinesh Manocha

Abstract— **We present DenseCAvoid, a novel algorithm for navigating a robot through dense crowds and avoiding collisions by anticipating pedestrian behaviors. Our formulation uses visual sensors and a pedestrian trajectory prediction algorithm to track pedestrians in a set of input frames and compute bounding boxes that extrapolate to the pedestrian positions in a future time. Our hybrid approach combines this trajectory prediction with a Deep Reinforcement Learning-based collision avoidance method to train a policy to generate smoother, safer, and more robust trajectories during run-time. We train our policy in realistic 3-D simulations of static and dynamic scenarios with multiple pedestrians. In practice, our hybrid approach generalizes well to unseen, real-world scenarios and can navigate a robot through dense crowds ($\sim$1-2 humans per square meter) in indoor scenarios, including narrow corridors and lobbies. As compared to cases where prediction was not used, we observe that our method reduces the occurrence of the robot freezing in a crowd by up to 48%, and performs comparably with respect to trajectory lengths and mean arrival times to goal.**

I. INTRODUCTION

Mobile robots are increasingly being used in different scenarios that are crowded with pedestrians and other obstacles. For instance, robots are being used for room service in hotels, package and food delivery, or as caretakers in hospitals. Furthermore, they are used for surveillance in public places such as malls, airports, etc. These environments can be populated with high pedestrian density (e.g., 1-2 pedestrians per square meter). In such scenarios, a robust and efficient collision avoidance method is crucial to ensure the safety of the robot and the humans.

The problem of robot navigation among dynamic obstacles has been well studied in robotics and related areas. There is a large body of work on *classic navigation techniques* based on potential fields, velocity obstacles, and dynamic windows [1], [2], [3], [4], [5]. Recently, many collision avoidance methods based on machine learning have been proposed [6], [7], [8], [9], [10], [11] and have shown considerable promise in real-world scenarios. These *learning-based methods* can be directly integrated with existing 2-D or 3-D lidars or cameras and are robust to sensing inaccuracies in the states of the obstacles.

In practice, dense crowds pose several challenges for robot collision avoidance. First, pedestrian motions in such crowds can be highly non-smooth [12]. Second, the robot must be able to react to sudden changes in pedestrian motion to avoid collisions. Current learning-based collision avoidance methods [6], [9], [13] work well for sparse or moderately dense crowds, but either result in collisions or oscillations

This work was supported in part by ARO Grants W911NF1910069 and W911NF1910315, and Intel.

[1] Authors contributed equally. All authors are with the University of Maryland, College Park.

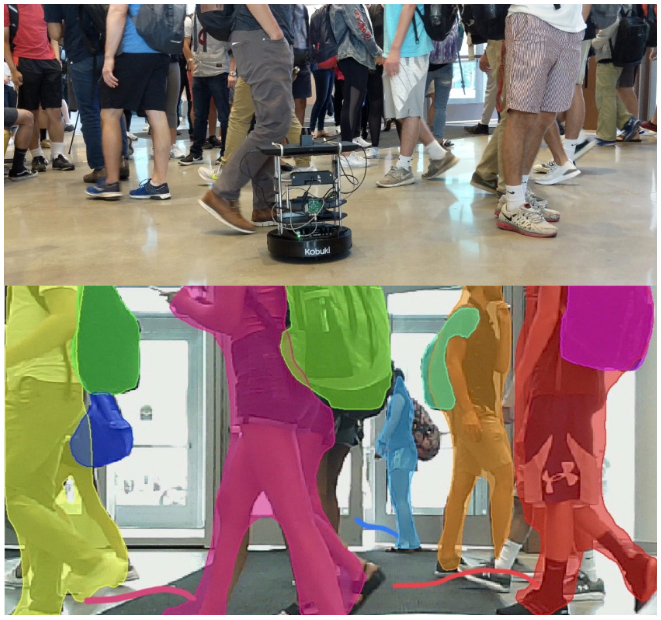

Fig. 1: **[Top]** Our method implemented on a Turtlebot navigating through a dense crowd. **[Bottom]** We explicitly track and predict pedestrian motions (marked for the three pedestrians) and use it to train a Deep Reinforcement Learning-based collision avoidance policy. Our network implicitly learns to reduce the occurrence of the freezing robot problem and generates smooth robot trajectories in dense crowds.

[7] as the crowd density increases. Another common phenomenon in such cases is the freezing robot problem [14], [15], [16], where the navigation method completely halts the robot, declaring that all forward velocities lead to a collision. Moreover, if the pedestrians obstructing the robot do not move to give way, the robot could stall indefinitely.

Main Results: We present a novel algorithm (DenseCAvoid) for safe robot navigation in dense crowds. Our approach is hybrid and combines techniques based on Deep Reinforcement Learning (DRL) with navigation methods that use pedestrian trajectory prediction. As a result, our approach provides the benefits of learning-based methods in terms of handling noisy sensor data, along with the benefits of a navigation method that explicitly predicts the trajectory and behavior of each pedestrian in an anticipatory manner. The latter enables our hybrid approach to robustly deal with new or unforeseen scenarios, which are quite different from the synthetic training data.

Our new DRL-based algorithm includes a modified network, a novel reward function and simulated training scenarios with static and dynamic pedestrians and other obstacles. We use a state-of-the-art pedestrian trajectory prediction method [17] that can handle dense scenarios. Our collision avoidance policy is trained using these explicitly predicted pedestrian motions, and a policy gradient method known as Proximal Policy Optimization (PPO)[18]. During run-time, our method uses raw sensor data from a lidar, a depth camera, the robot's odometry, and pedestrian prediction to generate smooth, collision-free trajectories. Our main contributions include:

- A new end-to-end DRL-based collision avoidance policy that is combined with a human motion prediction algorithm to anticipate pedestrian motion and generate smooth trajectories in dense crowds. This results in an increase by up to 74% in rates of reaching the goal when compared to times when prediction was not used.
- A motion prediction algorithm that is general and can handle dense scenarios, that is more robust than a simple linear motion model.
- A novel network structure and reward function that take multiple sensor inputs and pedestrian prediction data to train a policy using PPO. This results in a reduction by up to 48% in the occurrence of the freezing robot problem.
- Complex 3-D simulations of indoor environments with pedestrians and static obstacles for training and benchmarking DRL methods.

II. PRIOR WORK AND BACKGROUND

In this section, we briefly cover prior work in pedestrian tracking, motion prediction and learning-based collision avoidance methods. We also provide details regarding the preliminaries of our work.

A. Pedestrian Tracking and Motion Models

Object and pedestrian detection has been widely studied in computer vision. Some of the most accurate methods are based on deep learning including R-CNN [19] and its faster variants [20], [21], [22], which use a selective search area to optimize the object detection problem. Other works in learning-based tracking include [23], [24], [25], [26]. Many of these learning-based methods lack real-time performance that is needed for navigation in dense environments. Moreover, highly accurate methods such as [27] and [28] require high-quality detection features for reliable performance.

Several motion models have been used to improve pedestrian tracking accuracy [29], [30], [31], [32]. However, most of these methods assume a constant linear velocity or acceleration models for the pedestrians. These methods cannot characterize pedestrian dynamics accurately in dense settings [33]. Non-linear motion models such as RVO [1] and its variants have been shown to work well for tracking in dense crowd videos. Social Force model [34], LTA [35], and ATTR [36] are other non-linear motion models that have been used for pedestrian tracking in low to medium density crowds.

YOLOv3: In our approach, we use YOLOv3 [37], a real-time high accuracy variant of YOLO [38] for pedestrian detection and tracking in dense crowd. Given an RGB or a depth image and an object of interest (pedestrians, in our case), YOLOv3 outputs the bounding box coordinates over all the detected objects in the image.

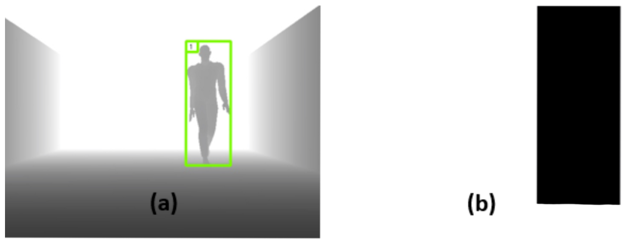

Fig. 2: (a) Tracking a moving pedestrian in a depth image. (b) Prediction output for a future time step. The white space in the output is the admissible free space, and the black bounding box denotes the space the pedestrian would occupy in the future.

B. Pedestrian Trajectory Prediction and Navigation

There has been extensive research in predicting object or pedestrian trajectories in computer vision and robotics. Early works include formulations such as Bayesian [39], Monte Carlo simulation [40], Hidden Markov Models [41], and Kalman Filters [42]. Deep learning-based prediction methods mostly utilize Recurrent Neural Networks (RNNs) [43] and Long Short-Term Memory (LSTM). Hybrid methods using a combination of RNNs and other deep learning architectures such as Convolutional Neural Networks (CNNs), Generative Adversarial Networks (GANs) and LSTMs have also been proposed. For instance, GANs have been used for pedestrian trajectory prediction [44] and CNNs have been used for traffic prediction [45]. There has also been extensive work on accurately modeling crowd behavior [46], [47], [48] which could aid robot navigation.

Our Pedestrian Prediction: We modify a state-of-the-art traffic trajectory prediction algorithm called RobustTP [17] to predict pedestrian motions. An image with the bounding box coordinates of the detected pedestrians (from YOLOv3) is fed as input into RobustTP. This algorithm uses a combination of CNNs and LSTMs to predict the positions of the detected pedestrians in the next frame immediately after the input image. The pedestrian tracking by YOLOv3 is shown in Fig.2 (a), and the position as predicted by RobustTP is shown in Fig.2(b) as a black bounding box on a white background. RobustTP can also be modified to predict the trajectories of any generic obstacle. The computation time of RobustTP depends on the level of accuracy required in prediction. For mobile robot navigation in dense crowds, we set this accuracy level to $< 90\%$ to ensure real-time performance.

Navigation using prediction: Pedestrian prediction using Bayesian estimation and modeling their motions using RVO is presented in [49]. Long-term path prediction based on Bayesian learning and personality trait theory for socially-aware robot navigation is presented in [50]. Other techniques use a Partially Observable Markov Decision Process (POMDP) to model the uncertainties in the intentions of pedestrians. [51] presents a POMDP-based planner to estimate the pedestrians' goals for autonomous driving. The planner was then augmented with an ORCA-based pedestrian motion model [52]. The resulting POMDP planner runs in near real-time. Our approach is complimentary and can be combined with these navigation methods.

C. Learning-Based Collision Avoidance with Pedestrians

In recent years, several works have used different learning methods for navigation in dense scenes. GAIL (Generative Adversarial Imitation Learning) [9] used raw depth camera images to train a socially acceptable navigation method for a differential drive robot. Similarly, CNNs with RGB images have been used to train an end-to-end visuomotor navigation system [8] and a deep double-Q network (D3QN) has been used to predict depth information from RGB images for static obstacle avoidance [53]. A method to use expert demonstrations in simulation to train a method for mapless navigation [54] has also been demonstrated. These methods, however, may not work well on dense crowds.

A decentralized, scalable, sensor-level collision avoidance method was trained in [7]. It was extended to a hybrid learning architecture [55], which switched policies based on the obstacle density in the environment. This method was further augmented to learn localization recovery points in the environment to solve the loss of localization and freezing robot problems simultaneously [14]. Cooperative behaviors between humans and robots have been modeled using a value network for better collision avoidance [6]. This formulation was extended to observe an arbitrary number of pedestrians in the surroundings using LSTMs [13]. [10] presented an approach to model interactions within a crowd which indirectly affect robot navigation. A novel collision avoidance method that identified previously unseen scenarios to carefully navigate around pedestrians is presented in [56].

D. ***DRL-based Collision Avoidance***

Our learning-based algorithm is based on deep reinforcement learning. The underlying objective is to train a policy π_θ that drives the robot to its goal while avoiding all the obstacles in the environment. There are three important components in DRL policy training, namely, **(i)** robot's observation space, **(ii)** action space, and **(iii)** the reward function. We briefly describe our observation and action spaces here and describe our novel reward function in Section III.

Observation and Action Spaces: Our observation space at any time instant t can be represented as $\mathbf{o}^t = [\mathbf{o}^t_{percep}, \mathbf{o}^t_{odom}, \mathbf{o}^t_{goal}]$, where $\mathbf{o}^t_{percep}$ is the observation from the perception sensors, $\mathbf{o}^t_{odom}$ is the odometry observation (which includes the current velocity of the robot), and $\mathbf{o}^t_{goal}$ denotes the goal position relative to the robot. The action space of the robot is a continuous space consisting of the linear and angular velocities of the robot, represented as $\mathbf{a}^t = [v^t, \omega^t]$.

The robot performs a certain action until it receives the observation for the next time instant $\mathbf{o}^{t+1}$. For optimizing the policy, we use the minimization of the mean arrival time of the robot to its goal as our objective function:

$$\underset{\pi_\theta}{\operatorname{argmin}} \mathbb{E}[\frac{1}{N}\sum_{i=1}^{N} t_i^g | \pi_\theta]. \tag{1}$$

1) ***Proximal Policy Optimization****:* To train our collision avoidance policy, we use a policy gradient method [57] called Proximal Policy Optimization (PPO) [18]. Policy gradient methods (in contrast with value-based methods) directly modify the policy during training, which is more suitable for navigation applications and continuous action spaces. In addition, PPO bounds the update of parameters θ to a trust region [58], thereby ensuring that the policy does not diverge between two consecutive training iterations. This guarantees stability during the training phase.

III. Our Hybrid Approach: DenseCAvoid

In this section, we present our hybrid collision avoidance method that combines DRL with explicit pedestrian trajectory prediction for navigation. We present our network architecture that is used to train our collision avoidance policy with anticipatory behaviors. We also discuss our reward function design and the complex 3-D training scenarios to handle dense crowds.

A. Anticipating Pedestrian Behavior

In densely crowded scenarios, pedestrian motion is highly non-smooth. In our case, we choose to explicitly model pedestrian behavior, similar to classic navigation methods. As mentioned in Section II, we use an explicit pedestrian trajectory predictor and use this information to generate non-oscillating, non-jerky navigation in dense scenarios. In addition, since our trained policy *knows* where each pedestrian is headed in the immediate future, we can also make the robot avoid regions where several pedestrians might be heading. Therefore, the robot tends to avoid scenarios that could possibly lead to the freezing robot problem.

As shown in Fig.2, the prediction frame extracted using [17] has black bounding boxes at locations where the pedestrians could be in the future. These boxes are placed over a white background, which represents the collision-free free-space. We provide this *future* free-space representation while training our network, which makes our collision avoidance policy training converge faster. This is due to the fact that our free-space representation provides a more direct way to infer the direction the robot should move towards and to learn the dynamic properties or behaviors of the pedestrians in different settings.

In our end-to-end formulation, we integrate the pedestrian prediction outputs with the DRL collision avoidance network as a new observation. Formally, $\mathbf{o}^t = [\mathbf{o}^t_{lid}, \mathbf{o}^t_{cam}, \mathbf{o}^t_g, \mathbf{o}^t_v, \mathbf{o}^t_{pred}]$, where $\mathbf{o}^t_{lid}$ denotes raw data from a 2-D lidar, $\mathbf{o}^t_{cam}$ denotes the raw image data from either a depth or an RGB camera, $\mathbf{o}^t_g$ refers to the relative position of the goal with respect to the robot, $\mathbf{o}^t_v$ denotes the robot's current velocity, and $\mathbf{o}^t_{pred}$ refers to the predicted positions of pedestrians in the next frame (shown in Fig. 2). Once the policy is trained, we sample a collision-free action from $\mathbf{a}^t$ at each time instant as:

$$\mathbf{a}^t \sim \pi_\theta(\mathbf{a}^t | \mathbf{o}^t). \tag{2}$$

Note that it is also possible to directly combine the prediction output with the action $\mathbf{a}^t$. This could be useful in scenarios where the robot encounters a situation that is quite different from the training data.

B. Network Architecture

Our network (Fig.3) consists of four branches, each processing a component of the observation o^t. Two 1-D layers and three 2-D layers are used respectively for processing the 2-D lidar data and depth image data, which are followed by fully-connected layers that modify the dimensions of

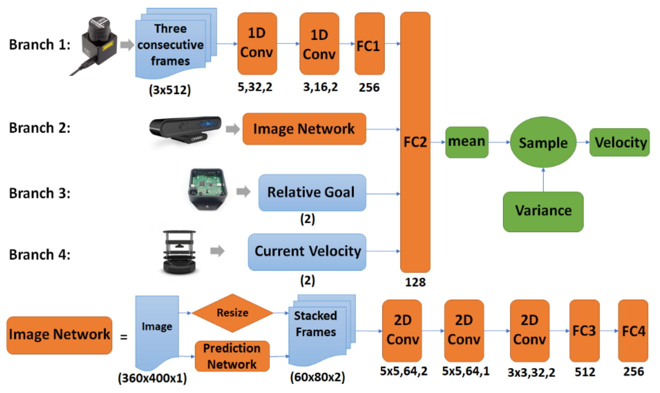

Fig. 3: Architecture of our anticipatory collision avoidance network with four branches to process different observations. The input layer is marked in blue and, the hidden layers are marked in orange and the fully-connected layers in the network are marked as FCn. The green layer represents the output layer. The three values underneath each hidden layer denote the kernel size, number of filters, and stride length respectively. We stack a raw depth image with a the prediction frame and use it directly for training our collision avoidance policy.

the outputs of the two branches to match each other. In branch 2, the depth image from the camera is first passed into our prediction algorithm. We then stack the computed prediction frame behind a resized version of the original depth image before passing these frames through a set of three 2-D convolutional layers. ReLU activation is applied to the outputs of all the hidden layers in branches 1 and 2. Branches 3 and 4 feed the relative position of the goal and the robot's current velocity to the fully-connected layer FC2.

We apply a sigmoid activation to restrict the robot's linear velocity between (0.0, 1.0) m/s and a tanh activation to restrict the angular velocity between (-0.4, 0.4) rad/s in the output layer. The output velocity is sampled from a Gaussian distribution that uses the mean value outputted from the fully connected layer FC2, which is updated during training. We address the imbalance between the dimensions of the perception sensors, and the robot's current velocity and goal using appropriate training scenarios (Section III.D). Our training scenarios ensure that the velocity and goal inputs are used in the initial stages of training to learn goal reaching capabilities with reduced oscillations.

C. Reward Function

Our purpose during policy training is to avoid collisions while moving towards the goal and to reduce oscillations or freezing behavior during navigation. Therefore, reaching the goal and colliding with obstacles are assigned high values of reward and penalty, respectively. To obtain smooth trajectories during run-time, we penalize sudden, large changes in the angular velocity. In addition, we use intermediate waypoints to guide the policy away from obstacles and towards the goal, which results in faster convergence.

Formally, the total reward collected by a robot at time instant t can be given as:

$$r^t = (r_g)^t + (r_c)^t + (r_{osc})^t + (r_{safedist})^t, \tag{3}$$

where the reward for reaching the goal or an intermediate waypoint $(r_g)^t$ is given as:

$$(r_g)^t = \begin{cases} r_{wp} & \text{if } ||\mathbf{p}^t - \mathbf{p}_{wp}|| < 0.2, \\ r_{goal} & \text{if } ||\mathbf{p}^t - \mathbf{g}|| < 0.3, \\ 2.5(||\mathbf{p}^{t-1} - \mathbf{g}|| - ||\mathbf{p}^t - \mathbf{g}||) & \text{otherwise.} \end{cases} \tag{4}$$

Here, $\mathbf{p}^t$ and $\mathbf{p}_{wp}$ denotes the position of the robot at time t and the position of the waypoint respectively, and $\mathbf{g}$ denotes the robot's goal location. The collision penalty $(r_c)^t$ is given as:

$$(r_c)^t = \begin{cases} r_{collision} & \text{if } ||\mathbf{p}^t - \mathbf{p}_{obs}|| < 0.3, \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The oscillatory behaviors (i.e. choosing sudden large angular velocities) are penalized as:

$$(r_{osc})^t = -0.1|\Delta\omega^t| \qquad \text{if } |\Delta\omega^t| > 0.3. \tag{6}$$

Here, ω^t is the angular velocity of the robot. The penalty for moving too close to an obstacle is given by:

$$(r_{safedist})^t = -0.1|d_{thresh} - d_{rob}| \quad \text{if } d_{thresh} > d_{rob}, \tag{7}$$

where d_{thresh} and d_{rob} denote the *threshold* distance that the robot needs to maintain from an obstacle at any time, and the actual distance that the robot maintains from an obstacle, respectively. For multiple obstacles present in the robot's vicinity, the penalties in equations 5 and 7 are applied for each obstacle. We set r_{wp} = 10, r_{goal} = 20, and $r_{collision}$ = -20 in our formulation.

D. Training Scenarios

The policy training is carried out in multiple stages to ensure fast convergence of the total accumulated reward. When the robot trains in more complicated training scenarios, we also run the robot in the simpler training scenarios simultaneously to ensure that previously learned capabilities are not overwritten. We designed several training scenarios to suit our pedestrian prediction network by including walking pedestrian models in the dynamic scenes in our training. Furthermore, we include dense pedestrian scenarios with sudden changes in the pedestrian motion. Our training scenarios are as follows:

- **Random Goal:** The robot is given a random goal in an empty world, and actions leading the robot towards the goal are rewarded. In this scenario, the partially trained model learns basic goal-reaching capabilities.
- **Dense-Static:** The robot is given a random goal in a world cluttered with static obstacles. During training, the policy augments its previously learned goal-reaching capabilities with basic static obstacle avoidance.
- **Random-Pedestrians:** The policy from the previous scenario is now trained in a world with randomly walking pedestrians. The pedestrian prediction observations now play a major role in training the policy for dynamic obstacle avoidance. We vary the positions, trajectories, and the densities of the pedestrians.
- **Dense-Random-Pedestrians:** In this scenario, the robot needs to navigate through a dense crowd of randomly walking pedestrians before reaching its goal.

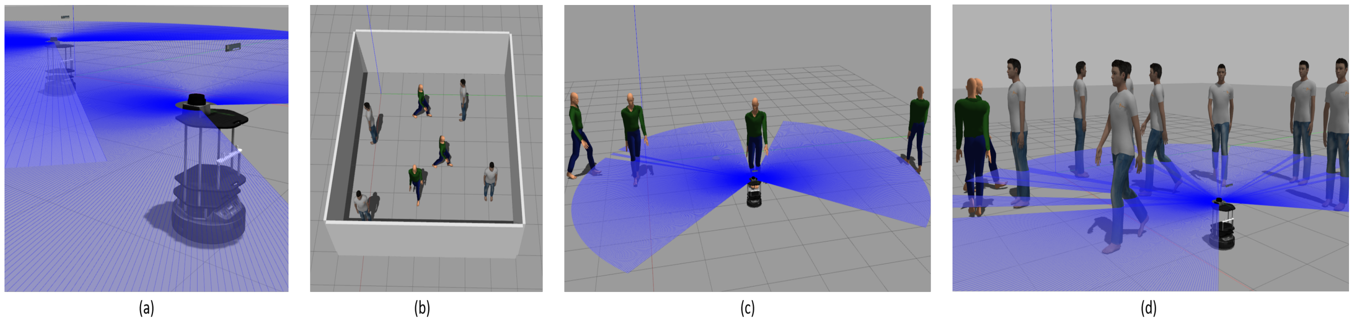

Fig. 4: Different training scenarios used for training our algorithm from simplest to complex. **(a)**: Empty scenario with random goals; **(b)**: Dense-Static scenario with random goal; **(c)**: Robot moves through a few pedestrians walking randomly to reach the goal; **(d)** Robot moves through a dense crowd to reach its goal.

Metrics	Sensor Configuration	Dense-Static	Random-Sparse-Ped	Dense-Ped
Success Rate	Depth Camera	0.26	0.73	0.55
	Depth Camera + Lidar	0.6	0.733	1
	DenseCAvoid	0.93	0.87	1
Avg Trajectory Length	Depth Camera	N/A	5.5	16.3
	Depth Camera + Lidar	N/A	5.11	15.51
	DenseCAvoid	N/A	6.39	16.87

TABLE I: Relative performance of our DenseCAvoid hybrid method versus learning-based methods that do not use explicit pedestrian prediction. In the latter category, we use two combinations of sensors (i.e. only depth camera and depth camera + lidar). The trajectory length for Dense-Static case is not measured, since we assign the robot's goals randomly. These numbers highlight the benefit in terms of success rate (higher is better) of DenseCAvoid. However, the explicit use of prediction can slightly increase the trajectory length (lower is better) to the goal, because the robot may not take the shortest straight line path.

Metrics	Distance	Depth Camera	Depth camera + lidar	DenseCAvoid
Freezing Robot %	$<$ 1.0 meters	100%	100%	100%
	1 - 1.5 meters	53%	33%	5%
	1.5 - 2 meters	27%	0%	0%

TABLE II: The performance of different methods in avoiding freezing robot scenarios, i.e. the number of instance the robot freezes (lower is better), tested in the Robot Freezing scenario. Our DenseCAvoid method considerably improves the performance when the robot is more than 1m away from an obstacle. In these cases, our explicit prediction of pedestrian trajectory improves the navigation capability and the robot does not freeze.

IV. Results and Evaluations

In this section, we describe our implementation and highlight the performance of DenseCAvoid in different scenarios. We also compare our navigation algorithm with policies trained without trajectory prediction and highlight the benefits of explicitly modeling the anticipatory pedestrian behavior. The convergence of the logarithm of our reward function versus the number of iterations is shown in Fig. 4.

A. Implementation

Our policy is trained in simulations created using ROS Kinetic and Gazebo 8.6 on a workstation with an Intel Xeon 3.6 GHz processor and an Nvidia GeForce RTX 2080Ti GPU. We use Tensorflow, Keras, and Tensorlayer to create our network. We simulate sensor data using models of the Hokuyo 2-D lidar and the Orbbec Astra depth camera in Gazebo during training and testing. The 2-D Hokuyo lidar has a maximum range of 4 meters and a field of vision (FOV) of 240°, and provides 512 range values per scan. The Orbbec Astra has a minimum range of 1.4 meters and a maximum sensing range of 5 meters. We use depth images of size 60×80 as inputs to our policy training network. We use a low resolution image to reduce the latency for processing the data.

Each pixel value in the depth image varies from 0 to 255. For pedestrian prediction, we normalise the depth images to have values between 1.4 to 5 (corresponding to the camera's range) before passing it into a pre-trained YOLOv3 and RobustTP network. We observe about 85% prediction accuracy in our dense benchmarks. We mount the same sensors on a Turtlebot 2 robot to test our model in real-world scenarios such as densely crowded corridors with non-smooth pedestrian trajectories. During run-time, we ensure that the rate at which the lidar scan data is received, and the rate at which the depth images are processed by RobustTP are comparable.

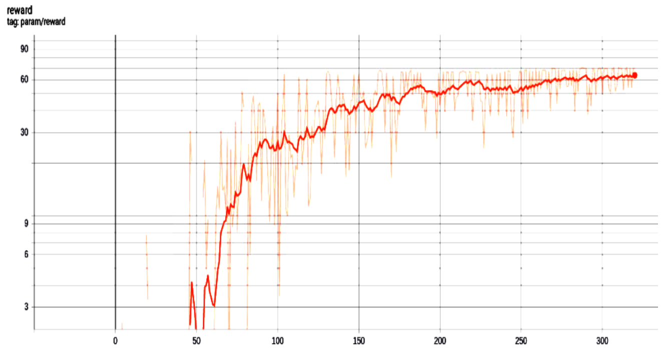

Fig. 5: Convergence of our reward function (shown for positive reward values) vs the number of iterations. Explicit pedestrian prediction helps our training converge faster and smoother when compared with the case with no prediction.

B. *Testing scenarios*

We compare our policy trained with pedestrian prediction with two policies which were trained without pedestrian prediction, but with the same reward function and training scenarios: (i) Policy trained only with depth camera observations, and (ii) Policy trained with lidar and depth camera observations. This comparison would clearly highlight the benefits of including pedestrian prediction when all other factors are the same. We consider four different test scenarios that have more challenging static and dynamic scenes, as compared to our training scenarios. This demands tight maneuvers from the robot to reach its goal. The scenarios we consider are:

1. **Dense-Static**: Scenario cluttered with static obstacles, with random goals provided to the policy.
2. **Random-Sparse-Ped**: Scenario where the robot must pass through 10 randomly walking pedestrians to reach its goal.
3. **Dense-Ped**: Scenario where the robot must move against the direction of 15 walking pedestrians in a narrow corridor to reach its goal.
4. **Robot Freezing**: Several $(3-4)$ pedestrians are suddenly spawned at different distances $(1-2$ meters) in front of the robot to simulate the freezing-robot scenario. Most prior benchmarks fail in such cases and we highlight the benefits of explicit trajectory prediction.

C. *Performance Benchmarks and Metrics*

We use the following metrics to evaluate the performance of different navigation algorithms:

- **Success Rate** - The number of times that the robot arrived at its goal without colliding with an obstacle over the total number of attempts.
- **Average Trajectory Length** - The trajectory length that the robot travels before reaching the goal, calculated as the sum of linear segments over small time intervals over the total number of attempts.
- **Robot Freezing %** - The number of times the robot got stuck or started oscillating indefinitely, while avoiding sudden obstacles over total number of attempts. A lower value is better.

D. *Analysis and Comparison*

We present our results in Table I. We observe that DenseCAvoid consistently has a higher success rate as the testing scenarios get more complicated. Using only observations from the depth camera for navigation works well for sparse scenarios, however, performs poorly in dense scenarios. This is mainly due to the low field of vision of the depth camera when compared to a lidar. Using observations from depth camera and a 2-D lidar performs slightly better than a single sensor. However, explicit modeling of the pedestrian future trajectory and behavior improves the results in these scenarios. However, the use of trajectory prediction can increase the trajectory length, as the robot may take a larger turn during collision avoidance.

The main benefit of DenseCAvoid arises in terms of dealing with the freezing robot problem, as can be observed from Table II. All methods fail in scenarios where a pedestrian suddenly appears within 1 meters from the robot. This is a consequence of limiting the angular velocity of the robot between (-0.4, 0.4) rad/sec to avoid high oscillations during navigation. Although a higher angular velocity range (say -1 to 1 rad/sec) could avoid such sudden pedestrians, it also leads to undesirable oscillations in some instances. It also prolongs the training time due to the increase in the oscillation penalty earned by the robot. Prior deep reinforcement learning methods can't deal with such situations and the robots tend to freeze. However, DenseCAvoid is able to handle sudden pedestrians about 1 to 1.5 meters away much better than the other methods, leading to oscillations/freezing only 5% of the time. This highlights the benefit of using explicit pedestrian prediction as the classic navigation method along with learning-based methods.

V. CONCLUSIONS, LIMITATIONS, FUTURE WORK

We present a novel method for collision avoidance with pedestrians in dense scenarios. Our hybrid approach tends to combine the benefits of learning-based methods with classic navigation methods that explicitly perform trajectory prediction. Our approach has been implemented in highly dense crowds with pedestrian densities up to 1-2 pedestrian per square meter. Our prediction method does not make any assumptions regarding the motion model for the pedestrian, which results in stable behavior during training and runtime. We validate our work in simulation and in real-world scenarios using a Turtlebot and showed that our approach drastically reduces the freezing robot problem, when pedestrians suddenly appear in front of the robot. Our work has several limitations. While we improve the navigation capabilities in dense scenarios, our approach does not work robustly in all possible scenarios. The accuracy is also limited by the accuracy of trajectory prediction, which is not perfect. The overall performance is mostly governed by the synthetic datasets used during the training phase and the limitations of sim-to-real paradigm. As part of future work, we want to overcome these limitations. It may also be possible to combine classic navigation methods as a post-processing step to the action computed by Equation 2. We also need to account for the robot's dynamics constraints and test them in outdoor scenarios. We also need better perception techniques to handle transparent surfaces.

References

[1] J. van den Berg, Ming Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, May 2008, pp. 1928–1935.

[2] J. P. van den Berg, S. J. Guy, M. C. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *ISRR*, 2009.

[3] J. Alonso-Mora, A. Breitenmoser, M. Rufli, P. A. Beardsley, and R. Siegwart, "Optimal reciprocal collision avoidance for multiple non-holonomic robots," in *DARS*, 2010.

[4] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[5] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers, "A hybrid collision avoidance method for mobile robots," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No.98CH36146)*, vol. 2, May 1998, pp. 1238–1243 vol.2.

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

[7] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards Optimally Decentralized Multi-Robot Collision Avoidance via Deep Reinforcement Learning," *arXiv e-prints*, p. arXiv:1709.10082, Sep 2017.

[8] Y. Kim, J. Jang, and S. Yun, "End-to-end deep learning for autonomous navigation of mobile robot," in *2018 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2018, pp. 1–6.

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

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

[11] 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 *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 1527–1533.

[12] A. Bera and D. Manocha, "Reach - realtime crowd tracking using a hybrid motion model," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 740–747.

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

[14] T. Fan, X. Cheng, J. Pan, P. Long, W. Liu, R. Yang, and D. Manocha, "Getting robots unfrozen and unlost in dense pedestrian crowds," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1178–1185, 2019.

[15] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2010, pp. 797–803.

[16] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015. [Online]. Available: https://doi.org/10.1177/0278364914557874

[17] R. Chandra, U. Bhattacharya, C. Roncal, A. Bera, and D. Manocha, "Robusttp: End-to-end trajectory prediction for heterogeneous road-agents in dense traffic with noisy sensor inputs," *arXiv preprint arXiv:1907.08752*, 2019.

[18] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv e-prints*, p. arXiv:1707.06347, Jul 2017.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *ArXiv e-prints*, Nov. 2013.

[20] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[21] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *ArXiv e-prints*, June 2015.

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," *ArXiv e-prints*, Mar. 2017.

[23] A. Milan, S. H. Rezatofighi, A. Dick, I. Reid, and K. Schindler, "Online multi-target tracking using recurrent neural networks," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[24] S.-H. Bae and K.-J. Yoon, "Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 3, pp. 595–610, 2018.

[25] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," 2017.

[26] K. Fang, Y. Xiang, and S. Savarese, "Recurrent autoregressive networks for online multi-object tracking," *arXiv preprint arXiv:1711.02741*, 2017.

[27] N. Wojke, A. Bewley, and D. Paulus, "Simple Online and Realtime Tracking with a Deep Association Metric," *ArXiv e-prints*, Mar. 2017.

[28] S. Murray, "Real-time multiple object tracking-a study on the importance of speed," *arXiv preprint arXiv:1709.03572*, 2017.

[29] A. Milan, S. Roth, and K. Schindler, "Continuous energy minimization for multitarget tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 1, pp. 58–72, 2013.

[30] C. Kim, F. Li, A. Ciptadi, and J. M. Rehg, "Multiple hypothesis tracking revisited," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4696–4704.

[31] R. Henschel, L. Leal-Taixe, D. Cremers, and B. Rosenhahn, "Fusion of head and full-body detectors for multi-object tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1428–1437.

[32] H. Sheng, L. Hao, J. Chen, Y. Zhang, and W. Ke, "Robust local effective matching model for multi-target tracking," in *Pacific Rim Conference on Multimedia*. Springer, 2017, pp. 233–243.

[33] A. Bera and D. Manocha, "Realtime multilevel crowd tracking using reciprocal velocity obstacles," in *Pattern Recognition (ICPR), 2014 22nd International Conference on*. IEEE, 2014, pp. 4164–4169.

[34] A. Bera, N. Galoppo, D. Sharlet, A. Lake, and D. Manocha, "Adapt: real-time adaptive pedestrian tracking for crowded scenes," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 1801–1808.

[35] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, Sept 2009, pp. 261–268.

[36] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 1345–1352.

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

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[39] S. Lefèvre, C. Laugier, and J. Ibañez-Guzmán, "Exploiting map information for driver intention estimation at road intersections," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 583–588.

[40] S. Danielsson, L. Petersson, and A. Eidehall, "Monte carlo based threat assessment: Analysis and improvements," in *Intelligent Vehicles Symposium, 2007 IEEE*. IEEE, 2007, pp. 233–238.

[41] J. Firl, H. Stübing, S. A. Huss, and C. Stiller, "Predictive maneuver evaluation for enhancement of car-to-x mobility data," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*. IEEE, 2012, pp. 558–564.

[42] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[43] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[44] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," *ArXiv e-prints*, Mar. 2018.

[45] F.-C. Chou, T.-H. Lin, H. Cui, V. Radosavljevic, T. Nguyen, T.-K. Huang, M. Niedoba, J. Schneider, and N. Djuric, "Predicting motion of vulnerable road users using high-definition maps and efficient convnets," 2018.

[46] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin, "Composite agents," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 2008, pp. 39–47.

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

[48] S. J. Guy, J. Van Den Berg, W. Liu, R. Lau, M. C. Lin, and D. Manocha, "A statistical similarity measure for aggregate crowd dynamics," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 1–11, 2012.

[49] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics*

Research, vol. 34, no. 2, pp. 201–217, 2015. [Online]. Available: https://doi.org/10.1177/0278364914555543

[50] A. Bera, T. Randhavane, R. Prinja, and D. Manocha, "Sociosense: Robot navigation amongst pedestrians with social and psychological constraints," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 7018–7025.

[51] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 454–460.

[52] Y. Luo, P. Cai, A. Bera, D. Hsu, W. S. Lee, and D. Manocha, "Porca: Modeling and planning for autonomous driving among many pedestrians," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3418–3425, Oct 2018.

[53] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning," *arXiv e-prints*, p. arXiv:1706.09829, Jun 2017.

[54] 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," *arXiv e-prints*, p. arXiv:1609.07910, Sep 2016.

[55] T. Fan, P. Long, W. Liu, and J. Pan, "Fully Distributed Multi-Robot Collision Avoidance via Deep Reinforcement Learning for Safe and Efficient Navigation in Complex Scenarios," *arXiv e-prints*, p. arXiv:1808.03841, Aug 2018.

[56] B. Lötjens, M. Everett, and J. P. How, "Safe reinforcement learning with model uncertainty estimates," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8662–8668.

[57] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, ser. NIPS'99. Cambridge, MA, USA: MIT Press, 1999, pp. 1057–1063. [Online]. Available: http://dl.acm.org/citation.cfm?id=3009657.3009806

[58] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," *CoRR*, vol. abs/1502.05477, 2015. [Online]. Available: http://arxiv.org/abs/1502.05477