Accurate High Fidelity Simulations for Training Robot Navigation Policies for Dense Crowds using Deep Reinforcement Learning

Jing Liang
Dept. of Computer Science
University of Maryland
College Park, USA

Utsav Patel
Dept. of Computer Science
University of Maryland
College Park, USA

Adarsh Jagan Sathyamoorthy
Dept. of Electrical Engg.
University of Maryland
College Park, USA

Dinesh Manocha
Dept. of Computer Science
University of Maryland
College Park, USA

***Abstract*—We present a novel high fidelity 3-D simulator that significantly reduces the sim-to-real gap in training collision avoidance policies based on a Deep Reinforcement Learning (DRL) for dense crowd scenarios. We make our simulations more accurate by modeling realistic crowd and pedestrian behaviors, along with friction, sensor noise, and delays in the simulated robot model. We also include a mechanism to incrementally control the randomness and complexity of the training scenarios to achieve better convergence and generalization capabilities in the collision-avoidance policy trained in our simulator. We have tested the capabilities and effectiveness of our simulator by training policies that fuse data from multiple perception sensors such as a 2-D lidar and a depth camera to detect pedestrians, and compute smooth, collision-free velocities for the robot. We observe that these policies outperform prior dynamic navigation techniques in terms of standard metrics such as rate of successfully reaching the goal, trajectory length, average time to reach the goal, and trajectory smoothness.**

***Index Terms*—Sim-to-real, Deep Reinforcement Learning, Robot Collision Avoidance.**

I. Extended Abstract

Recent developments in mobile robotics have allowed autonomous ground robots to be deployed in indoor and outdoor environments such as hospitals, hotels, malls, airports, warehouses, sidewalks, etc. These robots are used for surveillance, inspection, delivery, and cleaning, or as social robots. In such scenarios, robots are required to be able to navigate smoothly and avoid collisions, especially with dynamic agents or pedestrians. The crowd density in these scenarios is high and typically varies between 1-3 pedestrians per square meter.

There has been considerable work on learning-based collision avoidance for mobile robots operating in such dense scenarios. Specifically, Deep Reinforcement Learning-based (DRL) methods [1]–[3] have demonstrated better collision avoidance behaviors, lower time to reach the goal, and higher mean velocity of the robot in comparison to traditional Velocity Obstacle (VO) methods [4]–[6]. These methods typically use a single sensor such as a lidar, depth camera, or RGB camera for obstacle detection.

Such DRL-based collision avoidance policies are trained in a simulator and then transferred to a real robot. Due to this, the policies almost always suffer from a sim-to-real gap, i.e., their performance in the real world is worse than their performance in the simulation. A few reasons which cause the sim-to-real gap are 1) unmodeled sensor noise, friction and delays in the robot hardware [7], and 2) difficulty in recreating real-world robot-pedestrian interactions with variable complexity in the simulator. A policy trained in a simulator with such capabilities would also avoid overfitting.

Real world tests and simulation video: https://youtu.be/o9O-GEibJec

Apart from sim-to-real transfer issues, the performance of the policy also depends on what perception sensors are used to train it. For instance, algorithms that are trained using a single 2-D lidar [3] may perform well in dense scenarios but lack the ability to differentiate between animate and inanimate obstacles, and also cannot handle thin obstacles such as poles/legs of furniture. Methods which use several perception sensors [1], [2] exhibit good performance in moderately dense crowds but not in high-density crowds, and are susceptible to perception errors. Their trajectories also tend to be jerky and oscillatory. In addition, these methods are trained either in a simple 2.5D simulator or using generated trajectories, therefore lacking the scalability to train policies that use sensors such as RGB/depth cameras which generate high dimension data.

Main Results: We present a novel high-fidelity 3-D simulator which accurately models real-world crowd behavior, sensor noise, frictions and delays of the robot, to close the sim-to-real gap while training DRL policies for dense crowd navigation. Our simulator also allows us to incrementally adjust the complexity and randomness of the training scenarios. This simulator has been used to train works such as CrowdSteer [8], and DenseCAvoid [9], novel collision avoidance policies that fuse data from inexpensive perception sensors such as a 2-D lidar and a depth (RGB-D) camera to sense obstacle features. Our simulator helps the policies implicitly learn different kinds of robot interactions with the pedestrians, and significantly reduces the sim-to-real gap. The robot's velocities and behavior in the simulation and in real-world scenarios match over 80% of the time, demonstrating our simulator's sim-to-real transfer capabilities whereas previous methods perform well in simulations but exhibit oscillatory/jerky motions in real hardware.

We briefly discuss our simulator development in this extended abstract. For more details and results about our collision avoidance policies, we direct the readers to [8]–[10].

A. Our Simulation Approach

1) Background: Several works have discussed the reasons for the sim-to-real gap and methods to mitigate them [7], [11]. For training a crowd navigation policy in simulation, the simulated crowd's behavior must try to mimic the real-world

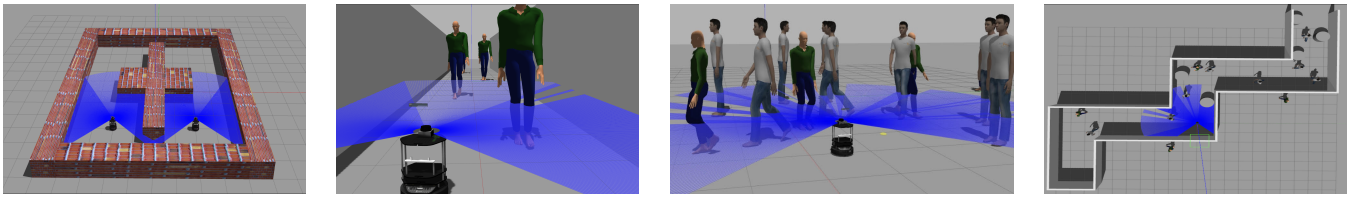

Figure 1: Left to right: Some of the training scenarios created in our simulator. From left: 1. Static scenarios with two Turtlebots, 2. Corridor with dynamic pedestrians, 3.Scenario with random number of standing and walking pedestrians, and 4.Scenario with high occlusions.

behavior. To this ends, works such as [12]–[15] have discussed several dynamics, physiological and psychological factors that should be accounted for. [16] demonstrates a modular structure for developing a crowd simulator. We extend the concepts from these works while developing our simulator.

2) Simulator Development: All the simulations are created using ROS Kinetic and Gazebo 8. The main components involved in the development of our simulator are: 1. Modeling pedestrian movement, and 2. Scenarios and modified robot model in the simulation. We briefly discuss them below.

Pedestrian Behavior Modeling: Pedestrian velocities in crowds depend on several complex factors such as the crowd's density, stride length of a person, and need for personal space. The fundamental diagram [12] provides an inverse relationship between individual pedestrian velocities and the crowd density. In order to create more accurate and realistic simulations, we account for the density-dependent behaviors based on the aforementioned factors.

We model each pedestrian in our simulation as a disk on a 2-D plane with the following state space: [r $\vec{p}$ $\vec{v}^{curr}$ $\vec{v}^{pref}$] $\in \mathbb{R}^7$. Here, r denotes the disc radius. $\vec{p}$, $\vec{v}^{curr}$ and $\vec{v}^{pref}$ represent the current 2-D position, current velocity and the preferred velocities of the pedestrians respectively. We relate a pedestrian's natural walking velocity (V) with physiological (pedestrian's height and stride length), and psychological (need for personal space) factors using the following equation:

$$V = min\left(||\vec{v}^{pref}||, \left(\frac{S\alpha}{H(1+\beta)}\right)^2\right) \tag{1}$$

where S is the available space in front of the pedestrian, H (height/1.72) is a height normalization factor, α and β are constants that account for stride of the pedestrian.

Scenario and Robot Model Development While developing a simulator and scenario for training navigation policies, it is important to strike a balance between: (i) the generality of the scenario such that overfitting is avoided, (ii) maintaining the complexity of the training scenarios such that the policy converges. We achieve this by training the policy, starting from simple scenarios, and then moving on to parallelly training in more complex scenarios (see Fig. 1). We vary each scenario's complexity and randomness by using a *complexity factor* ($\mathfrak{c}$) for the different scenarios, and by correlating it with a certain attribute of the scenario.

We classify our scenarios into three broad categories and explain the complexity factor for each category.

a) Static Scenario: This category of scenarios contains only immovable obstacles in an enclosed space of constant area, and the policy learns goal-reaching and static obstacle avoidance behaviors in these scenarios. We link the complexity factor $\mathfrak{c}$ to the number of obstacles in the scenario. As $\mathfrak{c}$ increases, obstacles are randomly placed in the scenario. This reduces the navigable free space available for the robot.

b) Random Static and Dynamic Obstacles: In this category of scenarios, $\mathfrak{c}$ is linked to the number of static obstacles and the magnitude of the velocity of dynamic pedestrians. The simulated pedestrians walk with the velocities given by equation 1. As ($\mathfrak{c}$) is increased, the training scenario has more randomly spawned static obstacles (less traversable space for the robot), and the robot needs to handle dynamic pedestrians that walk faster (by using higher than average values for the constants in equation 1).

c) Scenario with Occluded Obstacles: The robot learns to perform several sharp maneuvers and avoid pedestrians who can only be observed in close proximity due to the arrangement of walls (see Fig. 1) in this category of scenarios. Here, $\mathfrak{c}$ is correlated with the occlusion % in the environment, and the number of dynamic obstacles. The occlusion % is dependent on the FOV of the camera and is defined as $\left(\frac{\text{Angle not visible}}{\text{Camera's FOV}}\right) \times 100\%$. As $\mathfrak{c}$ increases, the occlusions become more severe and the robot faces more dynamic obstacles in random intervals.

The complexity factor $\mathfrak{c}$ gives more control to incrementally change the randomness and complexity of the scenarios. This aids in training convergence, and since all scenarios include some randomness, overfitting is prevented.

d) Robot Model: We use a modified model of Turtlebot 2 in our simulations for training our collision avoidance policies. We compare the velocities of a real Turtlebot with a laptop and sensors mounted on it with a simulated robot and use system identification techniques to deduce additional motor friction and inertia parameters that the simulated robot model must include to be more accurate. In addition, we include Gaussian noise $\mathcal{N}(0, 0.2)$ in the depth images, and the robot's goal location in the simulation and increase the delay in subscribing to the robot's current velocity observations in ROS. The position where the depth camera is mounted is also varied during training to provide different perspectives of the obstacles to the robot, which improves the generalization of the trained policy. These modifications help to more accurately recreate phenomena that happen only in hardware in our simulations.

In conclusion, we present a novel high fidelity 3-D simulator for training DRL-based collision avoidance policies for dense crowd scenarios that significantly reduces the sim-to-real gap in the trained policies. For more details on the results and implementation, we direct the reader to [8].

REFERENCES

[1] 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.

[2] 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.

[3] 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.

[4] F. Large, D. A. Vasquez Govea, T. Fraichard, and C. Laugier, "Avoiding Cars and Pedestrians using V-Obstacles and Motion Prediction," in *Proc. of the IEEE Intelligent Vehicle Symp.*, Pisa (IT), France, Jun. 2004, voir basilic : http://emotion.inrialpes.fr/bibemotion/2004/LVFL04/ note: Poster session address: Pisa (IT). [Online]. Available: https://hal.inria.fr/inria-00182054

[5] 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.

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

[7] M. Neunert, T. Boaventura, and J. Buchli, *Why Off-The-Shelf Physics Simulators Fail In Evaluating Feedback Controller Performance - A Case Study For Quadrupedal Robots: Proceedings of the 19th International Conference on CLAWAR 2016*, 10 2016, pp. 464–472.

[8] J. Liang, U. Patel, A. Jagan Sathyamoorthy, and D. Manocha, "Realtime Collision Avoidance for Mobile Robots in Dense Crowds using Implicit Multi-sensor Fusion and Deep Reinforcement Learning," *arXiv e-prints*, p. arXiv:2004.03089, Apr. 2020.

[9] A. Jagan Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "DenseCAvoid: Real-time Navigation in Dense Crowds using Anticipatory Behaviors," *arXiv e-prints*, p. arXiv:2002.03038, Feb. 2020.

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

[11] W. Yu, V. C. V. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion," *ArXiv*, vol. abs/1903.01390, 2019.

[12] S. Narang, A. Best, S. Curtis, and D. Manocha, "Generating pedestrian trajectories consistent with the fundamental diagram based on physiological and psychological factors," *PLOS ONE*, vol. 10, p. e0117856, 04 2015.

[13] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ser. SCA '07. Goslar, DEU: Eurographics Association, 2007, p. 99–108.

[14] R. Narain, A. Golas, S. Curtis, and M. C. Lin, "Aggregate dynamics for dense crowd simulation," *ACM Trans. Graph.*, vol. 28, no. 5, p. 1–8, Dec. 2009. [Online]. Available: https://doi.org/10.1145/1618452.1618468

[15] A. Braun, S. R. Musse, L. P. L. de Oliveira, and B. E. J. Bodmann, "Modeling individual behaviors in crowd simulation," in *Proceedings 11th IEEE International Workshop on Program Comprehension*, May 2003, pp. 143–148.

[16] S. Curtis, A. Best, and D. Manocha, "Menge: A modular framework for simulating crowd movement," *Collective Dynamics*, vol. 1, pp. 1–40, 2016. [Online]. Available: https://collective-dynamics.eu/index.php/cod/article/view/A1