Structured Graph Network for Constrained Robot Crowd Navigation with Low Fidelity Simulation

Shuijing Liu, **Kaiwen Hong**, **Neeloy Chakraborty**, **Katherine Driggs-Campbell**
University of Illinois, Urbana-Champaign
{sliu105, kaiwen2, neeloyc2, krdc}@illinois.edu

Abstract—**We investigate the feasibility of deploying reinforcement learning (RL) policies for constrained crowd navigation using a low-fidelity simulator. We introduce a representation of the dynamic environment, separating human and obstacle representations. Humans are represented through detected states, while obstacles are represented as computed point clouds based on maps and robot localization. This representation enables RL policies trained in a low-fidelity simulator to deploy in real world with a reduced sim2real gap. Additionally, we propose a spatio-temporal graph to model the interactions between agents and obstacles. Based on the graph, we use attention mechanisms to capture the robot-human, human-human, and human-obstacle interactions. Our method significantly improves navigation performance in both simulated and real-world environments. Video demonstrations can be found at https://sites.google.com/view/constrained-crowdnav/home.**

I. Introduction

To co-exist and collaborate with people seamlessly, robots must navigate through dynamic environments with both moving agents and static obstacles. Such environments are usually constrained: in indoor environments, walls and furniture are common, while in outdoor environments, robots must stay on their lanes or sidewalks.

Robot crowd navigation has received much attention since last century [8, 10, 28, 37, 19]. Model-based approaches have explored various mathematical models, such as velocity obstacles [33, 32], dynamic windows [8], and forces [9], to optimize robot actions. Although these models consider both humans and obstacles, they are prone to failures such as the freezing problem due to unrealistic assumptions on human behaviors [30, 15]. In addition, the hyperparameters are sensitive to different environments and thus need to be hand-tuned to ensure good performance [18].

Another line of work trains more expressive crowd navigation policies using imitation learning and reinforcement learning (RL) [5, 3, 15, 16, 21]. However, these works focus on navigation in open spaces and ignore obstacles and constraints. To address this issue, other learning-based approaches use raw sensor images or point clouds to represent environments [23, 7, 25, 37]. These end-to-end (e2e) pipelines have made promising progress with very few assumptions about the environment. However, to deploy the learned policies to the real world, these e2e methods need either a large amount of demonstration data from the real world or a high-fidelity simulator, both of which are expensive to obtain and prone to domain shifts between training and testing scenarios [20, 26].

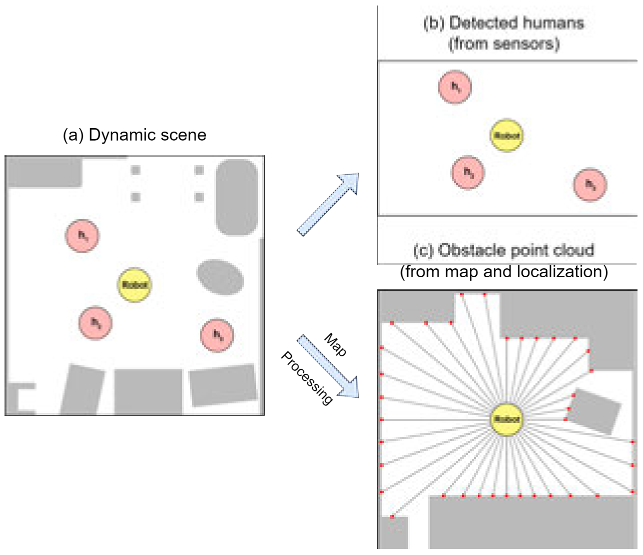
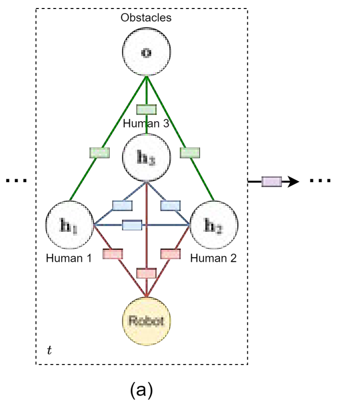

(a) Dynamic scene
(b) Detected humans (from sensors)
(c) Obstacle point cloud (from map and localization)
Map Processing
Robot

Fig. 1: **A split representation of constrained navigation scenario.** In a dynamic scene, human information is obtained from detections by sensors. For obstacle information, we remove all humans and compute a point cloud from a known map and the robot's location. In this way, we can learn a robot policy with smaller sim2real gaps with a cheap low-fidelity simulator.

In this paper, we ask the following question: Is it possible to deploy an RL policy for constrained crowd navigation with a cheap and low-fidelity simulator? The first step is to come up with an environment representation that is robust to sim2real gaps from perception. As shown in Fig. 1, we propose to split the human and obstacle representation and leverage on processed inputs instead of raw sensor inputs. We represent humans with detected states and obstacles as computed point clouds from map and robot localization. With processed states of humans and objects, the split representation is less affected by inaccurate simulations of human gaits, visual appearance, and 3D shapes. As a result, we can train the robot policy in a low-fidelity simulator, such as the one in Fig. 3, with a much smaller sim2real gap compared with previous e2e methods.

With the proposed scene representation, we learn a robot policy that reasons about interactions among different entities. In constrained environments, agents have limited traversable spaces and thus interact with each other as well as obstacles frequently. To capture these subtle interactions, we propose spatio-temporal (st) graph and derive a novel policy network from the st-graph. We use three separate attention networks to address the different effects of robot-human, human-human, and human-obstacle interactions. After training, the attention



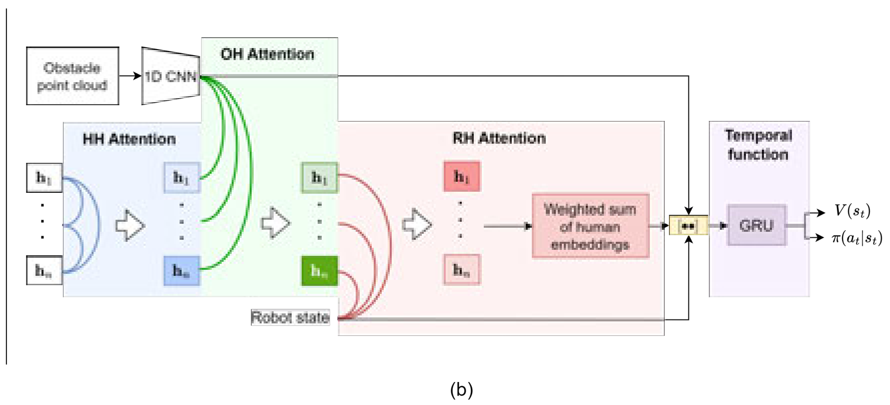


Fig. 2: **The spatial-temporal interaction graph and the network architecture.** (a) Graph representation of crowd navigation. The robot node is in yellow, the i-th human node is u_i, and the obstacle node is o. HH edges and HH functions are in blue, OH edges and OH functions are in green, and RH edges and RH functions are in red. The temporal function is in purple. (b) Our network. Three attention mechanisms are used to model the human-human, human-obstacle, and robot-human interactions. We use a GRU as the temporal function.

networks enable the robot to pay more attention to important interactions, which ensures good performance when the number of humans increases and the landscape becomes complex.

II. Preliminaries

A. MDP formulation

We model the constrained crowd navigation scenario as a Markov Decision Process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \mathcal{S}_0 \rangle$. Let $\mathbf{w}^t$ be the robot state which consists of the robot's position (p_x, p_y), velocity (v_x, v_y), goal position (g_x, g_y), and heading angle θ. Let $\mathbf{h}_i^t$ be the current state of the i-th human at time t, which consists of the human's position and velocity $(p_x^i, p_y^i, v_x^i, v_y^i)$. Let $\mathbf{o}^t$ be the current observations of the static obstacles and walls. We define the state $s_t \in \mathcal{S}$ of the MDP as $s_t = [\mathbf{w}^t, \mathbf{o}^t, \mathbf{h}_1^t, ..., \mathbf{h}_n^t]$ if a total number of n humans are observed at the timestep t, where n may change within a range in different timesteps.

In each episode, the robot begins at an initial state $s_0 \in \mathcal{S}_0$. According to its policy $\pi(a_t|s_t)$, the robot takes an action $a_t \in \mathcal{A}$ at each timestep t. In return, the robot receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $\mathcal{P}(\cdot|s_t, a_t)$. Please refer to Sec. VI-A in Appendix for the definition of reward function. Meanwhile, all humans also take actions according to their policies. The process continues until the robot reaches its goal, t exceeds the maximum episode length T, or robot collision.

III. Methodology

A. Scene representation

High-dimensional raw sensor representation suffer from large sim2real gaps due to the presence of humans and complex obstacles and landscapes. Instead of investing in expensive high-fidelity simulators or laborious dataset collection, we use low-fidelity simulators. To circumvent sim2real gaps, our scene representation leverages processed information from perception, maps, and robot localization, which are relatively easier to obtain and robust to domain shifts.

As shown in Fig. 1, at each timestep t, we split a dynamic scene into a human representation denoted as $\mathbf{h}_1^t, ..., \mathbf{h}_n^t$, as well as an obstacle and constraint representation denoted as $\mathbf{o}^t$. In human representation, the position and velocity of each human is detected from off-the-shelf human detectors [13, 27, 36]. By representing each human as a low-dimentional state vector, we abstract away detailed information such as gaits and appearance, which are difficult to model accurately [31, 20]. To obtain obstacle representation, as shown in Fig. 4 in Appendix, we first map the environment and process the map by combining close-by obstacles and approximating obstacle shapes as polygons. The map processing step is shown from Fig. 1(a) to Fig. 1(c). During navigation, assuming robot localization is available, we can compute a "fake" point cloud by performing a ray tracing algorithm centered at the robot location, as shown in Fig. 1(c). The "fake" point cloud contains approximate information on obstacles and constraints, which is sufficient for robot navigation. It is worth noting that compared to real point clouds from sensors, our obstacle representation is not affected by the presence of humans and is less sensitive to inaccuracies simulations of object appearance or shapes.

B. Structured interaction graph

Interactions among different entities contain essential information for multi-agent problems [35, 11, 3, 15, 1]. We formulate constrained crowd navigation as a spatio-temporal (st) graph, which breaks the problem into smaller components in a structured fashion [12]. In Fig. 2(a), the robot, all observed humans, and the observed static environment are nodes in the st-graph $\mathcal{G}_t$. At each timestep t, the edges that connect different nodes denote the spatial interactions among nodes. Different interactions have different effects on robot decision-making. Specifically, since we have control of the robot but not the humans, robot-human interactions have direct effects while human-human interactions have indirect effects on the robot actions. Since the agents are movable but the obstacles are static, interactions among agents are mutual while the influence of static obstacles on agents is one-way. Thus, we categorize the spatial edges into three types: human-human (HH) edges (blue in Fig. 2), obstacle-human (OH) edges (green), and robot-human (RH) edges (red). The three types of edges allow us to factorize the spatial interactions into HH function, OH function, and RH function. Each function is a neural network that has learnable parameters. Compared with the previous works that ignore some edges [3, 15, 16], our method allows the robot to reason about all observed spatial interactions that exist in constrained crowded environments.

Since the movements of all agents cause the visibility of humans and obstacles to change dynamically, the set of nodes and edges and the parameters of the interaction functions may change correspondingly. To this end, we integrate the temporal correlations of the graph $\mathcal{G}_t$ at different timesteps using another function denoted by the purple box in Fig. 2a. The temporal function connects the graphs at adjacent timesteps, which enables long-term decision-making of the robot.

The same type of edges share the same function parameters. This parameter sharing is important for the scalability of our st-graph because the number of parameters is kept constant with an increasing number of humans [12].

C. Structured attention network

In Fig. 2b, we derive our network architecture from the st-graph. We represent the HH, OH, and RH functions as feedforward networks with attention mechanisms, referred as HH attn, OH attn, and RH attn respectively. We represent the temporal function as a gated recurrent unit (GRU). We use W and f to denote trainable weights and fully connected layers.

1) Attention mechanism: The attention modules assign weights to all edges that connect to a node, allowing the node to attend to important edges or interactions. The 3 attention networks are similar to the scaled dot-product attention [34], which computes attention score using a query Q and a key K, and applies the normalized score to a value V.

$$\text{Attn}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{1}$$

where d is the dimension of the queries and keys.

In HH attention, the current states of humans are concatenated and passed through linear layers to obtain $Q^t_{HH}, K^t_{HH}, V^t_{HH} \in \mathbb{R}^{n\times d_{HH}}$, where d_{HH} is the attention size for the HH attention.

$$\begin{aligned} Q^t_{HH} &= [\mathbf{u}^t_1, ..., \mathbf{u}^t_n]^\top W^Q_{HH} \\ K^t_{HH} &= [\mathbf{u}^t_1, ..., \mathbf{u}^t_n]^\top W^K_{HH} \\ V^t_{HH} &= [\mathbf{u}^t_1, ..., \mathbf{u}^t_n]^\top W^V_{HH} \end{aligned} \tag{2}$$

We obtain the human embeddings $v^t_{HH} \in \mathbb{R}^{n\times d_{HH}}$ from Eq. 1, and the number of attention heads is 8.

In OH attention, the obstacle point cloud is fed into a 1D CNN, which outputs an obstacle embedding: $K^t_{OH} = f_{CNN}(\mathbf{o}^t)$, where $K^t_{OH} \in \mathbb{R}^{1\times d_{OH}}$. $Q^t_{RH}, V^t_{RH} \in \mathbb{R}^{n\times d_{RH}}$ are linear embeddings of the weighted human features from HH attention v^t_{HH}.

$$Q^t_{OH} = v^t_{HH} W^Q_{OH},\ K^t_{OH} = v^t_O W^K_{OH},\ V^t_{OH} = v^t_{HH} W^V_{OH} \tag{3}$$

We compute the attention score from Q^t_{OH}, K^t_{OH}, and V^t_{OH} to obtain the twice weighted human features $v^t_{OH} \in \mathbb{R}^{1\times d_{OH}}$ as in Eq. 1. The number of attention heads is 1.

Similarly, in RH attention, we first embed the robot with a linear layer: $K^t_{RH} = f_R(\mathbf{w}^t)$, where $K^t_{RH} \in \mathbb{R}^{1\times d_{RH}}$. $Q^t_{RH}, V^t_{RH} \in \mathbb{R}^{n\times d_{RH}}$ are linear embeddings of the weighted human features from OH attention v^t_{OH}.

$$Q^t_{RH} = v^t_{OH} W^Q_{RH},\ K^t_{RH} = \mathbf{w}^t W^K_{RH},\ V^t_{RH} = v^t_{OH} W^V_{RH} \tag{4}$$

We compute the attention score from Q^t_{RH}, K^t_{RH}, and V^t_{RH} to obtain the weighted human features for the third time $v^t_{RH} \in \mathbb{R}^{1\times d_{RH}}$ as in Eq. 1. The number of attention heads is 1.

In all three attention networks, we use binary masks that indicate the visibility of each human to prevent attention to invisible humans. The masks provide unbiased gradients to the networks, which stabilizes and accelerates the training.

2) GRU: We concatenate the robot embedding, the obstacle embedding, and the weighted human features and fed them into the GRU. Finally, the hidden state of the GRU is input to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$. We train the entire network with Proximal Policy Optimization (PPO) [29].

IV. Simulation Experiments

A. Simulation environment

Developed with PyBullet [6], our simulator consists of two scenarios as shown in Fig. 3. We conduct simulation experiments in *random environment* in Fig. 3(a). In each episode, obstacles are initialized with random shapes and random poses. The initial positions of the humans and the robot are also randomized. The human goals are set on the opposite side of their initial positions so that they cross each other in a circle. The number of humans varies from 2 to 4 and the number of obstacles varies from 7 to 9.

To simulate a continuous human flow, humans will move to new random goals immediately after they arrive at their goal positions. All humans are controlled by ORCA [33]. 80% of humans do not react to the robot and 20% of humans react to the robot. This mixed setting prevents our network from learning an extremely aggressive policy in which the robot forces all humans to yield while achieving a high reward, while maintaining a enough number of reactive humans to resemble the real crowd behaviors.

We use unicycle kinematics for the robot. The action of the robot consists of desired translational and rotational accelerations $a_t = [a_{trans}, a_{rot}]$. The robot action space is discrete: the translational acceleration $a_{trans} \in \{-0.05\,m/s^2, 0\,m/s^2, 0.05\,m/s^2\}$ and the rotational acceleration $a_{rot} \in \{-0.1\,rad/s^2, 0\,rad/s^2, 0.1\,rad/s^2\}$. The translational velocity is clipped within $[0\,m/s, 0.5\,m/s]$ and rotational velocity is within $[-1\,rad/s, 1\,rad/s]$. The robot motion is governed by the dynamics of TurtleBot 2i. We use holonomic kinematics for humans. The speed of humans is limited to 0.5 m/s to accommodate the speed of the robot.

B. Experiment setup

1) Baselines and ablation models: To validate the effectiveness of the proposed scene representation, we compare our method with the following baselines:

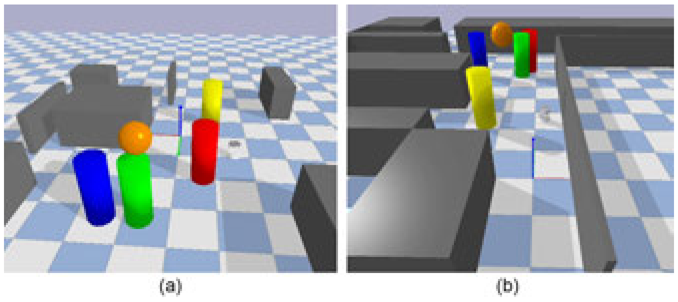


Fig. 3: **Two PyBullet simulation scenarios.** (a) Random environment with random obstacles and circle-crossing humans. (b) Sim2real environment with fixed obstacles and the random human flow is designed based on the layout.

- Object-centric: The human representation is the same as our method, while obstacles are represented by the coordinates of their polygon vertices. The obstacle embedding network is a multi-layer perception instead of 1D CNN.
- Raw point cloud: The humans and obstacles are represented as raw point clouds from a LiDAR. No human detection or fake point cloud is available. The point cloud is fed into a 1D CNN. No attention network is used since all entities are mixed as a single point cloud.

To validate the effectiveness of the structured graph network, we experiment with ablations of different attention models to justify the effect of 3 types of spatial interaction networks. Everything else except the presence of attention network(s) is kept the same as our method.

- Ours, RH: The network has only RH attention and does not have HH or OH attention, similar to previous works such as [3, 14, 15].
- Ours, RH+OH: The network has only RH and OH attention and does not have HH attention.
- Ours, RH+HH+OH: The full version of our proposed network with all 3 attention modules.

2) *Training:* The policy is trained for 6×10^7 steps in total. We run 16 parallel environments to collect the robot's experiences. The learning rate is 8×10^{-5} and decays linearly.

3) *Evaluation metrics:* We test all methods with 500 random unseen test cases. Our metrics include success rate (Success), collision rate (Collision), timeout rate (Timeout), and average navigation time (NT) in seconds.

C. Results

In Table I and the videos, among the baselines, raw point cloud performs the worst because the network needs to extract both human and obstacle features from raw sensor inputs, which slows down the convergence. Object-centric achieves better results due to the presence of low-level state information but is outperformed by our method, because the vertex representation is sparse and not always useful for navigation. For example, among all vertices of a long and thin wall, the occluded vertices or the vertices outside of the robot field-of-view are fed into the network, yet none of them affects navigation. In addition, the vertex representation is sensitive to the permutation order of vertices, which poses extra challenges for learning [2]. In contrast, our representation leverages low-level human states from detectors, which accelerates the convergence and thus achieves better performance with the same amount of training budget. For the obstacle representation, the "fake" point cloud is denser and more relevant to robot decision-making compared with vertex representation.

TABLE I: Navigation results in random environment.

Method	Success↑	Collision↓	Timeout↓	Time↓
Object-centric	0.63	0.27	0.09	12.02
Raw point cloud	0.58	0.27	0.15	12.02
Ours, RH	0.56	0.28	0.16	12.02
Ours, RH+OH	0.68	0.26	0.06	**11.28**
Ours, RH+HH+OH	**0.75**	**0.15**	**0.10**	11.75

Among ablated models, we observe that if we remove HH attention, the success rate drops by 7% because the interactions among humans are dense due to the presence of obstacles and environmental constraints. As a result, a human constantly changes its trajectories due to other humans. If we further remove OH attention, the success rate drops by another 12% because the obstacles limit the traversable regions of agents, and thus human trajectories are also directly affected by obstacles. Thus, we conclude that reasoning about both human-human and human-robot interactions plays an important role in robot collision avoidance, in addition to robot-human interactions from previous works.

V. Real-world Experiments

We train our method in *sim2real environment* in Fig. 3(b) and transfer the policy to a TurtleBot 2i in a real constrained indoor environment with pedestrians in a university building. We define 1 to 3 of human routes and robot routes based on the environment layout and randomly choose a route for each agent in each episode. In addition, 1 to 3 static humans are added. The poses of 11 obstacles are fixed.

We use an Intel RealSense tracking camera T265 to obtain the pose of the robot. With an RPLIDAR-A3 laser scanner, we first remove non-human obstacles on a map, and then use a 2D LIDAR people detector [13] to estimate the positions of humans. From results (see videos), we observe that the robot can achieve goals without collisions no matter the pedestrians react to the robot or ignore the robot. However, the robot fails if the pedestrians intentionally block its path, since this kind of adversarial behavior is not simulated during training.

VI. Conclusion

In conclusion, to enable robots to navigate in constrained crowded environments, we propose a structured scene representation from preprocessed information. Then, we formulate the scenario as a st-graph, which leads to the derivation of a robot policy network that reasons about different interactions during navigation. We train the network with RL and demonstrate good results in both simulation and the real world. For limitations and future work, please refer to Sec. VI-B in Appendix.

References

[1] Neeloy Chakraborty, Aamir Hasan, Shuijing Liu, Tianchen Ji, Weihang Liang, D. Livingston McPherson, and Katherine Driggs-Campbell. Structural attention-based recurrent variational autoencoder for highway vehicle anomaly detection. In *IFAAMAS International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

[2] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

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

[4] Haonan Chen, Tianchen Ji, Shuijing Liu, and Katherine Driggs-Campbell. Combining model-based controllers and generative adversarial imitation learning for traffic simulation. In *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, pages 1698–1704, 2022.

[5] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 285–292, 2017.

[6] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[7] Daniel Dugas, Olov Andersson, Roland Siegwart, and Jen Jen Chung. Navdreams: Towards camera-only rl navigation among humans. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2504–2511, 2022.

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

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

[10] Michael Hoy, Alexey S Matveev, and Andrey V Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(3):463–497, 2015.

[11] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[12] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *IEEE conference on computer vision and pattern recognition (CVPR)*, pages 5308–5317, 2016.

[13] Dan Jia, Alexander Hermans, and Bastian Leibe. DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10270–10277, 2020.

[14] Edouard Leurent and Jean Mercat. Social attention for autonomous decision-making in dense traffic. In *Machine Learning for Autonomous Driving Workshop at Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[15] Shuijing Liu, Peixin Chang, Weihang Liang, Neeloy Chakraborty, and Katherine Driggs-Campbell. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3517–3524, 2021.

[16] Shuijing Liu, Peixin Chang, Zhe Huang, Neeloy Chakraborty, Kaiwen Hong, Weihang Liang, D Livingston McPherson, Junyi Geng, and Katherine Driggs-Campbell. Intention aware robot crowd navigation with attention-based interaction graph. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 12015–12021, 2023.

[17] Ziyuan Liu and Georg von Wichert. Extracting semantic indoor maps from occupancy grids. *Robotics and Autonomous Systems*, 62(5):663–674, 2014.

[18] Pinxin Long, Tingxiang Fan, Xinyi Liao, Wenxi Liu, Hao Zhang, and Jia Pan. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 6252–6259, 2018.

[19] Christoforos Mavrogiannis, Krishna Balasubramanian, Sriyash Poddar, Anush Gandra, and Siddhartha S. Srinivasa. Winding through: Crowd navigation via topological invariance. *IEEE Robotics and Automation Letters*, 8(1): 121–128, 2023.

[20] Christoforos Mavrogiannis, Francesca Baldini, Allan Wang, Dapeng Zhao, Pete Trautman, Aaron Steinfeld, and Jean Oh. Core challenges of social robot navigation: A survey. *ACM Transactions on Human-Robot Interaction*, 12(3), 2023.

[21] Ye-Ji Mun, Masha Itkina, Shuijing Liu, and Katherine Driggs-Campbell. Occlusion-aware crowd navigation using people as sensors. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 12031–12037, 2023.

[22] Anay Pattanaik, Zhenyi Tang, Shuijing Liu, Gautham Bommannan, and Girish Chowdhary. Robust deep reinforcement learning with adversarial attacks. *arXiv preprint arXiv:1712.03632*, 2017.

[23] Claudia Pérez-D'Arpino, Can Liu, Patrick Goebel, Roberto Martín-Martín, and Silvio Savarese. Robot navigation in constrained pedestrian environments using reinforcement learning. In *IEEE International Confer-*

ence on Robotics and Automation (ICRA), pages 1140–1146, 2021.

[24] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning (ICML)*, pages 2817–2826, 2017.

[25] Ashwini Pokle, Roberto Martín-Martín, Patrick Goebel, Vincent Chow, Hans M Ewald, Junwei Yang, Zhenkai Wang, Amir Sadeghian, Dorsa Sadigh, Silvio Savarese, et al. Deep local trajectory replanning and control for robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5815–5822, 2019.

[26] Amir Hossain Raj, Zichao Hu, Haresh Karnan, Rohan Chandra, Amirreza Payandeh, Luisa Mao, Peter Stone, Joydeep Biswas, and Xuesu Xiao. Rethinking social robot navigation: Leveraging the best of two worlds. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[27] Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[28] Andrey V Savkin and Chao Wang. Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation. *Robotics and Autonomous Systems*, 62(10):1568–1580, 2014.

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

[30] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803, 2010.

[31] Nathan Tsoi, Alec Xiang, Peter Yu, Samuel S. Sohn, Greg Schwartz, Subashri Ramesh, Mohamed Hussein, Anjali W. Gupta, Mubbasir Kapadia, and Marynel Vázquez. Sean 2.0: Formalizing and generating social situations for robot navigation. *IEEE Robotics and Automation Letters*, pages 1–8, 2022.

[32] Jur Van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1928–1935, 2008.

[33] 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.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5998–6008, 2017.

[35] Anirudh Vemula, Katharina Muelling, and Jean Oh. Social attention: Modeling attention in human crowds. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2018.

[36] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649, 2017.

[37] Zhanteng Xie and Philip Dames. Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles. *IEEE Transactions on Robotics*, 39(4):2700–2719, 2023.

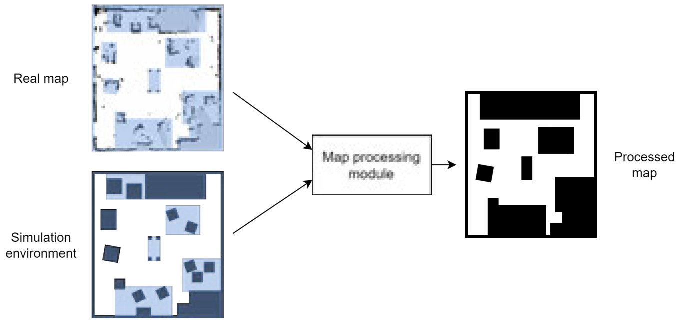


Fig. 4: **Illustration of map processing.** Using off-the-shelf map processing techniques [17], we can combine and smooth the edges of obstacles with irregular shapes. As a result, the processed map produces the obstacle point cloud representation, which introduces very small sim2real gaps. In the two raw maps on the left, we overlay the processed map on top of them for visualization purposes.

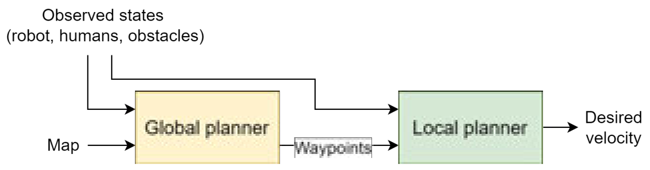


Fig. 5: **A two-level hierarchical planner.** To enable long-horizon navigation, we can treat our method as a local planner and combine it with a global planner.

Appendix

A. Reward function

The reward function awards the robot for reaching its goal and penalizes the robot for collisions with or getting too close to humans or obstacles. In addition, we add a potential-based reward shaping to guide the robot to approach the goal:

$$r(s_t, a_t) = \begin{cases} -20, & \text{if } d^t_{min} < 0 \\ 2(d^t_{min} - 0.25), & \text{if } 0 < d^t_{min} < 0.25 \\ 20, & \text{if } d^t_{goal} \leq \rho_{robot} \\ 2(-d^t_{goal} + d^{t-1}_{goal}), & \text{otherwise.} \end{cases} \quad (5)$$

where d^t_{min} is the minimum distance between the robot and any human or obstacle at time t, and d^t_{goal} is the $L2$ distance between the robot and its goal at time t. Intuitively, the robot gets a high reward when it approaches the goal while maintaining a safe distance from dynamic and static obstacles.

B. Limitations and future work

Our work encompasses the following limitations, which opens up opportunities for future work:

1) The robot only achieves a 75% success rate when it only navigates for 3 to 4 meters from start to goal. The task horizon and success rate are not enough for robot deployment in real applications such as last-mile delivery. To address this issue, as shown in Fig. 5, we plan to adopt a hierarchical planner, which consists of a global planner that outputs waypoints, and a local planner that takes waypoints and performs low-level control. The current model can be used as the local planner. Possible options for the global planner include sample-based planners such as A^* and RRT and learning-based policies.
2) As Sec. V discussed, the robot policy is overfitted to the simulated human behaviors. However, the simulated humans do not capture all the nuanced behavior patterns of real humans. A more realistic human motion model is necessary, which might need to be learned from real pedestrian data [4]. In addition, adversarial RL training may also improve the robustness of the robot policy with respect to changes in human behaviors [22, 24].