

# HEIGHT: HEterogeneous Interaction Graph Transformer for Robot Navigation in Crowded and Constrained Environments

Shuijing Liu, Haochen Xia, Fatemeh Cheraghi Pouria, Kaiwen Hong,  
Neeloy Chakraborty, and Katherine Driggs-Campbell

**Abstract**—We study the problem of robot navigation in dense and interactive crowds with environmental constraints such as corridors and furniture. Previous methods fail to consider all types of interactions among agents and obstacles, leading to unsafe and inefficient robot paths. In this article, we leverage a graph-based representation of crowded and constrained scenarios and propose a structured framework to learn robot navigation policies with deep reinforcement learning. We first split the representations of different components in the environment, and propose a heterogeneous spatio-temporal graph to model distinct interactions among humans, robots, and obstacles. Based on the heterogeneous st-graph, we propose HEIGHT, a novel navigation policy network architecture with different components to capture heterogeneous interactions among entities through space and time. HEIGHT utilizes attention mechanisms to prioritize important interactions and a recurrent network to track changes in the dynamic scene over time, encouraging the robot to avoid collisions adaptively. Through extensive simulation and real-world experiments, we demonstrate that HEIGHT outperforms state-of-the-art baselines in terms of success and efficiency in challenging navigation scenarios. Furthermore, we demonstrate that our pipeline achieves better zero-shot generalization capability than previous works when the densities of humans and obstacles change. More videos are available at <https://sites.google.com/view/crowdnav-height/home>.

## I. INTRODUCTION

Robots are increasingly prevalent in human-centric environments. In applications such as last-mile delivery and household robots, the ability to navigate among humans is crucial. For example, Fig. 1 shows a navigation scenario with abundant subtle interactions: Obstacles have a one-way effect on the paths of agents (i.e. humans and the robot), while the influence among agents is mutual. Among agents, humans may react to other humans and robots in different ways. To navigate, a robot directly participates in some interactions in its close proximity, and simultaneously, is indirectly affected by other interactions. These interactions are heterogeneous, dynamic, and difficult to infer, making navigation in such environments challenging.

Rising to these challenges, previous works have explored various approaches for robot crowd navigation [1]–[3]. However, these works typically have one of two limitations: (1)

S. Liu is with the Department of Computer Science at The University of Texas at Austin. Email: shuijing.liu@utexas.edu

H. Xia, F. Cheraghi Pouria, K. Hong, N. Chakraborty, and K. Driggs-Campbell are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. Emails: { hx17, fatemeh5, kaiwen2, neeloyc2, krdc }@illinois.edu

This material is based upon work supported by the National Science Foundation under Grant No. 2143435.

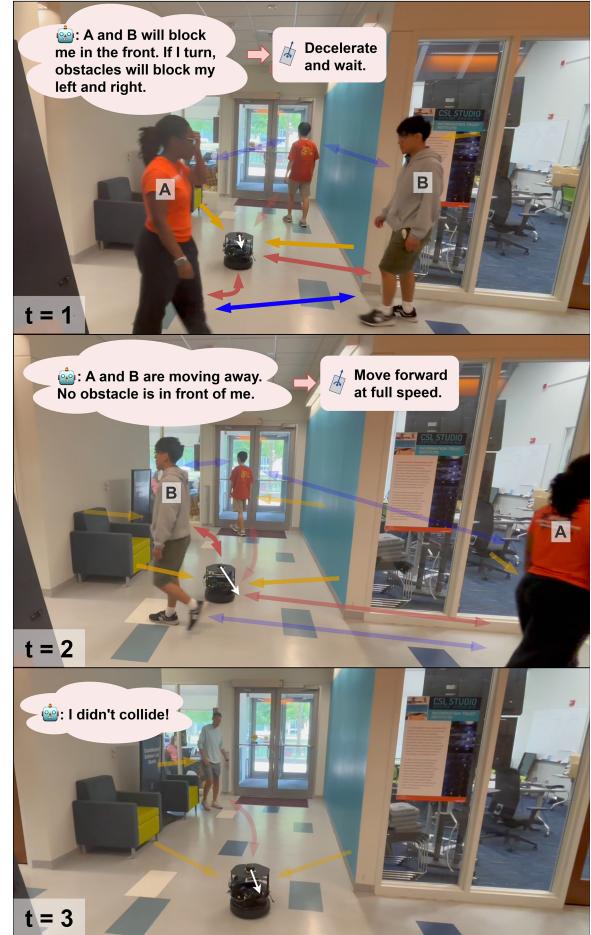


Fig. 1: A heterogeneous graph aids spatio-temporal reasoning when a robot navigates in a crowded and constrained environment. The colored arrows denote **robot-human (RH)**, **human-human (HH)**, and **obstacle-agent (OA)** interactions. The opaque arrows are the more important interactions while the transparent arrows are the less important ones. At each timestep  $t$ , the robot reasons about these interactions, focuses on the important ones, and makes decisions.

They assume agents move in an open space without obstacles, which are common in the real-world [3]–[5]; (2) They do not differentiate between various types of interactions, and thus the robot has difficulties taking adaptive strategies to avoid collisions with humans and obstacles [1], [6]–[8].

Our goal is to navigate a robot to a destination without colliding with humans and obstacles. To solve this problem, we ask the following research question: *How can a robot reason about diverse interactions in crowded and constrained*

*environments to adaptively avoid collisions during navigation?*

To address this question, we propose a framework that takes advantage of the heterogeneity of interactions in crowded and constrained scenarios. First, we split the environment into human and obstacle representations, which are processed and fed separately into the reinforcement learning (RL)-based navigation pipeline. Then, we decompose the scenario into a heterogeneous spatio-temporal (st) graph with different types of edges to represent different types of interactions among the robot, humans, and obstacles, as shown in the colored arrows in Fig. 1. Finally, we convert the heterogeneous st-graph into a **HE**terogeneous Interaction **GraPH** Transformer (HEIGHT), a robot policy network consisting of different modules to parameterize the various spatio-temporal interactions. Specifically, we use two separate multi-head attention networks to address the different effects of robot-human (RH) and human-human (HH) interactions. The attention networks enable the robot to pay more attention to the important interactions, leading to a low collision-rate even as the number of humans increases, and the graph becomes more complex. In addition, we use a multilayer perceptron (MLP) to model the single-directional obstacle-agent interactions and a recurrent network for temporal evolution of the scene. In response to the rapidly changing scenario (Fig. 1 bottom), HEIGHT captures the heterogeneous interactions among different components through space and time, enabling the robot to avoid collisions and approach its goal in an efficient manner.

This article is expanded from a contribution on attention graph network proposed by our previous work [9]. While our previous work focuses on crowd navigation in open spaces, this article incorporates static obstacles and constraints, which leads to significant modifications of scene representation and network architecture. Corresponding to these changes in methodology, new simulation and hardware experiments with new baseline comparisons are performed. In summary, the main contributions of this article are as follows.

- 1) We propose an input representation of crowded and constrained environments that treats humans and obstacles differently. The split scene representation naturally allows us to inject structures in the rest of the framework.
- 2) We propose a structured graph representation of crowded and constrained scenarios, named heterogeneous spatio-temporal graph (st-graph), to effectively model the pairwise interactions among all agents and entities.
- 3) From the heterogeneous st-graph, we use a principle approach to derive HEIGHT, a transformer-based robot navigation policy network with different modules to reason about all types of spatial and temporal interactions.
- 4) The experiments in simulation with dense crowds and dense obstacles demonstrate that our method outperforms previous state-of-the-art methods in unseen obstacle layouts. In addition, our method demonstrates better generalization to out-of-distribution environments with different human and obstacle densities.
- 5) We successfully transfer the robot policy learned in a low-fidelity simulator to challenging real-world, everyday crowded environments without finetuning.

## II. RELATED WORKS

In this section, we first review the literature of robot crowd navigation, which is divided into model-based and learning-based approaches. Then, we discuss previous crowd navigation efforts in constrained spaces. Finally, we review graph attention mechanism with a focus on the usage of attention networks in multi-agent interaction modeling.

### A. Model-based methods

Robot navigation in human crowds is particularly challenging and has been studied for decades [1], [10]–[12]. Model-based approaches have explored various mathematical models to optimize robot actions [1], [2], [13]–[16]. As an early example, ROS navigation stack [17] uses a cost map for global planning and dynamic window approach (DWA) for local planning [1]. DWA searches for the optimal velocity that brings the robot to the goal with the maximum clearance from any obstacle while obeying the dynamics of the robot. By treating humans as obstacles, DWA exhibits myopic behaviors such as oscillatory paths in dynamic environments [18]. As a step forward, optimal reciprocal collision avoidance (ORCA) and social force (SF) account for the velocities of agents. ORCA models other agents as velocity obstacles and assumes that agents avoid each other under the reciprocal rule [2], [13]. SF models the interactions between the robot and other agents using attractive and repulsive forces [14]. However, the hyperparameters of the model-based approaches are sensitive to crowd behaviors and thus need to be tuned carefully to ensure good performance [18], [19]. In addition, model-based methods are prone to failures, such as the freezing problem, if the assumptions such as the reciprocal rule are broken [20], [21]. In contrast, while our method also models these interactions, we learn the hyperparameters of the model from trial and error with minimal assumptions on human behaviors posed by model-based methods.

### B. Learning-based methods

Learning-based approaches have been widely used for navigation in dynamic environments to reduce hyperparameter tuning efforts and the number of assumptions. One example is supervised learning from expert demonstrations of desired behaviors, where the expert ranges from model-based policies [22]–[24], human teleoperators in simulators [25], to real pedestrians [26], [27]. Supervised learning does not require explorations of the state and action spaces of the robot, yet the performance of learned policy is limited by the quality of expert demonstrations.

Another line of work takes advantage of crowd simulators and learns policies with RL. Through trial and error, RL has the potential to learn robot behaviors that outperform model-based approaches and expert demonstrations [28]. For example, Deep V-Learning first uses supervised learning with ORCA as the expert and then uses RL to learn a value function for path planning [3]–[5], [29], [30]. However, Deep V-Learning assumes that state transitions of all agents are known without uncertainty. In addition, since the networks

are pre-trained with supervised learning, they share the same disadvantages with OCRA, which are hard to correct by RL [8]. To address these problems, more recent efforts leverage model-free RL without supervised learning or assumptions on state transitions [5], [8], [21], [31]. Since the state transition probability of humans is uncertain, directly learning a policy network without explicitly modeling state transitions is more suitable for navigation [32]. However, the aforementioned RL works have at least one of the following two problems: (1) They focus on navigation in open spaces and isolate agents from static constraints, posing difficulties when deploying robots in the real-world; (2) They ignore all or part of interactions among agents, which are important for robot navigation in dense crowds and highly constrained environments. We discuss how prior works address the above two problems in Sec. II-C and Sec. II-D respectively.

### C. Crowd navigation in constrained environments

Besides dynamic agents, real-world navigation environments usually consist of static constraints, such as walls, furniture, and untraversable terrains. To deal with these constraints and humans, some methods such as DWA [1] and DS-RNN [8] use groups of small circles to represent the contours of obstacles. While groups of circles is a straightforward representation of obstacles, as we will show in Sec. V, they are not scalable as the number of obstacles increases and can lead to overfitting problems for learning-based approaches.

Other learning-based approaches use raw sensor images or point clouds to represent the whole environment [7], [25], [33], [34]. These end-to-end (e2e) pipelines have made promising progress in simulation. However, generalizing these e2e methods to real-world scenarios is challenging due to domain gaps such as inaccurate human gait simulation [35], [36]. Despite the recent advancements in crowd navigation simulators [37]–[39] and datasets [26], [40], [41], learning a deployable e2e policy that outperforms human teleoperation in dense and interactive crowds remains an open challenge.

Furthermore, prior works that demonstrate strong real-world performance in dense crowds and obstacles usually leverage processed inputs such as detected human states and processed sensor readings [28], [32], [42]. As such, we develop a structured input representation of humans and obstacles, which splits human and obstacle representations and feeds processed states to the policy network. This input representation leads to strong performance in both simulation and real-world. In contrast to the above works with processed inputs that focus on pedestrian behavior prediction [32], reward design [28], and incorporating risks into map [42], our paper proposes a principled way for network architecture design from the structures of complex navigation scenarios.

### D. Graph attention networks for multi-agent interactions

In recent years, attention mechanisms have demonstrated success in various fields [43]–[45]. Vaswani et al. propose a transformer with self-attention mechanism that achieves state-of-the-art performance in machine translation [43]. Later, graph attention networks show the effectiveness of attention

on learning relationships of data with graphical structures such as citation networks [44]. Inspired by these works, researchers in trajectory prediction and crowd navigation have found that attention networks are also well-suited to capture interactions amongst agents and entities, which contain essential information for multi-agent tasks [3], [8], [45]–[47]. For each agent, these works compute attention scores for all neighboring agents. Due to the permutation invariance property, attention scores better capture the importance of pairwise relationships than combining the information of all agents with concatenation or an LSTM encoder [4], [29].

More specifically, in crowd navigation of robots and autonomous vehicles, a line of works uses a robot-human attention network to determine the relative importance of each human to the robot [3], [8], [48], [49]. However, interactions among humans, which can also influence the robot, are not explicitly modeled. To this end, other works use a homogenous graph network to include both RH and HH interactions [6], [50]. However, since these works feed RH and HH features to a single attention network, the resulting robot policy has difficulty reasoning the specificity of each type of feature, which limits the robot’s ability to adapt to different interactions, as demonstrated in [51] and our experiments in Sec. V. In addition, the works above only deal with open-world social navigation and thus ignore the interactions between agents and obstacles. To this end, Chen et al. treat the humans, the robot, and static objects as different types of nodes in a heterogeneous graph attention network [51].

The main difference from our work is that Chen et al. focus on semantic navigation, where a robot must navigate to an object in simulation. Our work focuses on point-goal navigation in crowded and constrained real-world environments. This difference leads to (1) Different representations of obstacles: the object representation in Chen et al. comes from a known semantic map of the environment, which is hard to obtain in the real-world. Thus, in our case, treating all obstacles as a 2D point cloud is more suitable for studying collision avoidance; (2) Chen et al. learn a value function and plan paths assuming simplified dynamics for agents, whereas we learn a model-free RL policy and assume unknown state transition for all agents, which is more suitable for sim2real transfer where the dynamics of pedestrians and robots are uncertain.

## III. PRELIMINARIES

In this section, we formulate constrained crowd navigation as a Markov Decision Process (MDP) and introduce the scene representation and the reward function.

### A. MDP formulation

We model the constrained crowd navigation scenario as a MDP, defined by the tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \mathcal{S}_0 \rangle$ . Let  $w^t$  be the robot state which consists of the robot’s position  $(p_x, p_y)$ , velocity  $(u_x, u_y)$ , goal position  $(g_x, g_y)$ , and heading angle  $\theta$ . Let  $h_i^t$  be the current state of the  $i$ -th human at time  $t$ , which consists of the human’s position and velocity  $(p_x^i, p_y^i, u_x^i, u_y^i)$ . Let  $o^t$  be the current observation of the static obstacles and walls, which is represented as a 2D point cloud. We define the

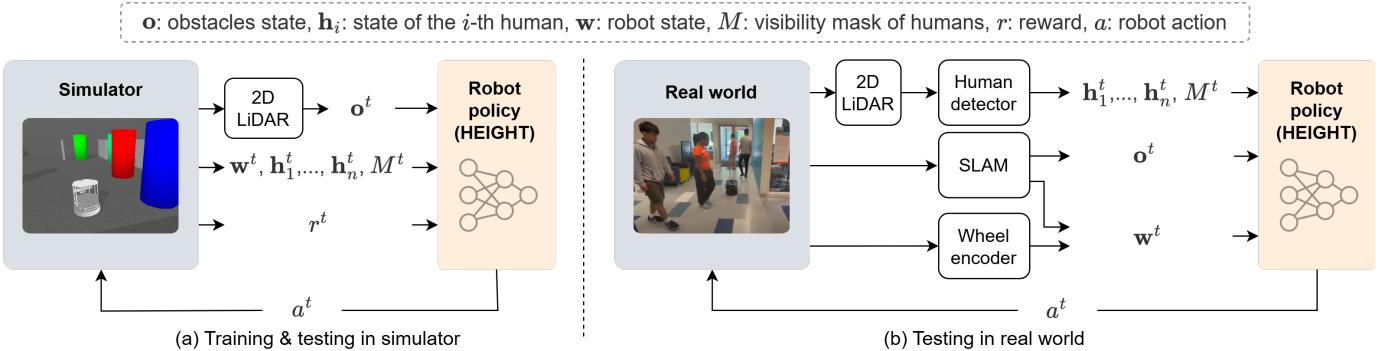


Fig. 2: **An overview of our pipeline in simulation and real-world.** (a) At each timestep  $t$  in training and testing, the simulator provides a reward  $r^t$  and the following observations of the environment: obstacle point cloud  $o^t$ , the robot state  $w^t$ , and the human states  $h_1^t, \dots, h_n^t$ , and masks  $M^t$  (Sec. IV-B1). These observations serve as inputs to HEIGHT, which outputs a robot action  $a^t$  that maximizes the future expected return  $R^t$ . The simulator executes the actions of all agents and the loop continues. (b) The testing loop in the real-world is similar to the simulator except the perception modules for obtaining the observations are different and the reward is absent.

state  $s^t \in \mathcal{S}$  of the MDP as  $s^t = [w^t, o^t, h_1^t, \dots, h_n^t]$  if a total number of  $n$  humans are observed at the timestep  $t$ , where  $n$  may change within a range in different timesteps. The humans are sorted by an increasing  $L_2$  distance to the robot. The state space  $\mathcal{S}$  is continuous and our choice of state representation is expanded in Sec. III-B.

In each episode, the robot begins at an initial state  $s^0 \in \mathcal{S}_0$ . As shown in Fig. 2(a), according to its policy  $\pi(a^t|s^t)$ , the robot takes an action  $a^t \in \mathcal{A}$  at each timestep  $t$ . The action of the robot consists of the desired translational and rotational accelerations  $a^t = [a_{trans}^t, a_{rot}^t]$ . The action space  $\mathcal{A}$  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 and rotational velocity is clipped within  $[-0.5 m/s, 0.5 m/s]$  and  $[-1 rad/s, 1 rad/s]$  respectively. The robot motion is governed by the dynamics of TurtleBot 2i. In return, the robot receives a reward  $r^t$  (see Sec. III-C for details) and transits to the next state  $s^{t+1}$  according to an unknown state transition  $\mathcal{P}(\cdot|s^t, a^t)$ . Meanwhile, all other humans also take actions according to their policies and move to the next states with unknown state transition probabilities. The process continues until the robot reaches its goal,  $t$  exceeds the maximum episode length  $T = 491$  steps, or the robot collides with any humans or static obstacles.

The goal of the robot is to maximize the expected return,  $R^t = \mathbb{E}[\sum_{i=t}^T \gamma^{i-t} r^i]$ , where  $\gamma$  is a discount factor. The value function  $V^\pi(s)$  is defined as the expected return starting from  $s$ , and successively following policy  $\pi$ .

### B. State representation

In our MDP, a state consists of a large and varying number of agents. To aid policy learning with such a complicated state space, we develop a structured representation of states that will be fed into the structured policy network. To reduce sim2real gaps caused by raw sensor readings, our scene representation leverages processed information from perception, maps, and robot localization, which are relatively robust to domain shifts.

In Fig. 3, at each timestep  $t$ , we split a scene into a human representation  $h_1^t, \dots, h_n^t$ , and an obstacle representation  $o^t$ . In human representation (Fig. 3(b)), the position and velocity of each human are detected using off-the-shelf human

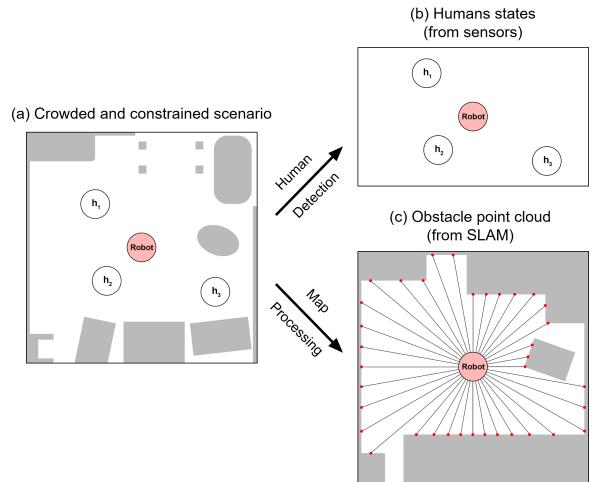


Fig. 3: **A split representation of a crowded and constrained navigation scenario.** In a dynamic scene, (b) low-level human states are detected by sensors and perception modules. (c) To obtain obstacle information, we remove all humans and compute a point cloud from a known map and the robot’s location. In this way, we can learn a robot policy with a small sim2real gaps using a cheap low-fidelity simulator.

detectors [52]–[54]. By representing each human as a low-dimensional state vector, we abstract away detailed information such as gaits and appearance, which are difficult to simulate accurately [35], [37]. We use a 2D point cloud as the obstacle representation. Instead of obtaining the point cloud from sensors, we take advantage of simultaneous localization and mapping (SLAM) and first create a map of the environment. During navigation, in Fig. 3(c), assuming robot localization on the map is known, an “artificial” point cloud that only contains static obstacles is computed from the map. Compared to real-time point clouds from sensors, our obstacle representation is not affected by the presence of humans and thus is less sensitive to inaccurate human simulations. The tradeoff is that our method requires a map and accurate localization. If a high-fidelity simulator is available, the “artificial” point cloud can be replaced by a real-time point cloud from sensors without changing the rest of our method described below.

### C. Reward function

Our reward function consists of three parts. The first and main part of the function awards the robot for reaching its goal

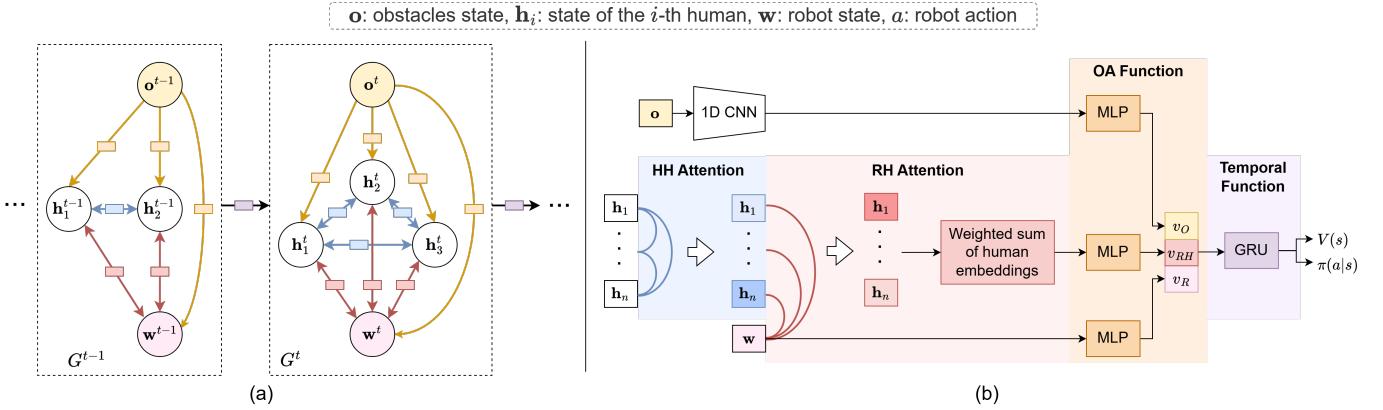


Fig. 4: **The heterogeneous st-graph and the HEIGHT network architecture.** (a) Graph representation of crowd navigation. The robot node is  $w$  (pink), the  $i$ -th human node is  $h_i$  (white), and the obstacle node is  $o$  (yellow). HH edges and HH functions are in blue, OA edges and OA functions are in orange, and RH edges and RH functions are in red. The temporal function is in purple. (b) HEIGHT network. Two attention mechanisms are used to model the HH and RH interactions. We use MLPs and a concatenation for obstacle-agent interactions, and a GRU for the temporal function. The superscript  $t$  that indicates the timestep and the human mask  $M$  is eliminated for clarity.

and penalizes the robot for colliding with or getting too close to humans or obstacles. In addition, we add a potential-based reward to guide the robot to approach the goal:

$$r_{main}(s^t, a^t) = \begin{cases} 20, & \text{if } d_{goal}^t \leq \rho_{robot} \\ -20, & \text{else if } d_{min}^t \leq 0 \\ d_{min}^t - 0.25, & \text{else if } 0 < d_{min}^t < 0.25 \\ 4(d_{goal}^{t-1} - d_{goal}^t), & \text{otherwise.} \end{cases} \quad (1)$$

where  $\rho_{robot}$  is the radius of the robot,  $d_{goal}^t$  is the  $L_2$  distance between the robot and its goal at time  $t$ , and  $d_{min}^t$  is the minimum distance between the robot and any human or obstacle at time  $t$ . The second part is a spinning penalty  $r_{spin}$  to penalize high rotational velocity:

$$r_{spin}(s^t, a^t) = -0.05\|\omega^t\|_2^2 \quad (2)$$

where  $\omega_t$  is the current rotational velocity of the robot. Finally, we add another small constant penalty  $r_{time} = -0.025$  at each timestep to encourage the robot to finish the episode as soon as possible.

In summary, the reward function of our MDP is the sum of the above three parts:

$$r(s^t, a^t) = r_{main}(s^t, a^t) + r_{spin}(s^t, a^t) + r_{time} \quad (3)$$

Intuitively, the robot gets a high reward when it approaches the goal with a high speed and a short and smooth path, while maintaining a safe distance from dynamic and static obstacles.

#### IV. METHODOLOGY

In this section, we present our approach to decompose the constrained crowd navigation scenario as a heterogeneous st-graph, which leads to the derivation of the HEIGHT architecture in a structured manner.

##### A. Heterogeneous Spatio-Temporal Graph

The subtle yet highly dynamic interactions among agents and entities are important factors that make crowd navigation difficult. To model these interactions in a structured manner,

we formulate the navigation scenario as a heterogeneous st-graph. In Fig. 4a, at each timestep  $t$ , our heterogeneous st-graph  $G^t = (\mathcal{V}^t, \mathcal{E}^t)$  consists of a set of nodes  $\mathcal{V}^t$  and a set of edges  $\mathcal{E}^t$ . The nodes include the detected humans  $h_1^t, \dots, h_n^t$  and the robot  $w^t$ . In addition, an obstacle node  $o^t$  represents the point cloud of all obstacles as a whole. At each timestep  $t$ , the spatial edges that connect different nodes denote the spatial interactions among nodes. Different spatial interactions have different effects on robot decision-making. Specifically, since we have control of the robot but not the humans, RH interactions have direct effects while HH interactions have indirect effects on the robot actions. As an example of indirect effects, if human A aggressively forces human B to turn toward the robot's front, the robot has to respond as a result of the interaction between A and B. Additionally, since the agents are dynamic 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: HH edges (blue in Fig. 4), obstacle-agent (OA) edges (orange), and RH edges (red). The three types of edges allow us to factorize the spatial interactions into HH, OA, and RH functions. Each function is parameterized by a neural network that has learnable parameters. Compared to previous works that ignore some edges [3], [8], [9], our method allows the robot to reason about all observed spatial interactions that exist in constrained and crowded environments.

Since the movements of all agents cause the visibility of each human to change dynamically, the set of nodes  $\mathcal{V}^t$  and edges  $\mathcal{E}^t$  and the parameters of the interaction functions may change correspondingly. To this end, we integrate the temporal correlations of the graph  $G^t$  at different timesteps using another function denoted by the purple box in Fig. 4(a). The temporal function connects the graphs at adjacent timesteps, which overcomes the short-sightedness of reactive policies and enables long-term decision-making of the robot.

To reduce the number of parameters, the same type of edges in Fig. 4(a) share the same function parameters. This parameter sharing is important for the scalability of our graph because the number of parameters is kept constant when the number of human changes [55].

## B. HEIGHT Architecture

In Fig. 4b, we derive our network architecture from the heterogeneous st-graph. We represent the HH and RH functions as feedforward networks with attention mechanisms, referred to as HH attn and RH attn respectively. We represent the OA function as an MLP with concatenation, and the temporal function as a gated recurrent unit (GRU). We use  $W$  and  $f$  to denote trainable weights and fully connected layers.

1) *Attention among agents*: The attention modules assign weights to all edges that connect to a robot or a human node, allowing the node to pay attention to important edges or interactions. The two attention networks are similar to the scaled dot-product attention with a padding mask [43], which computes the attention score using a query  $Q$  and a key  $K$ , and applies the normalized score to a value  $V$ , which results in a weighted value  $v$ .

$$v := \text{Attn}(Q, K, V, M) = \text{softmax} \left( \frac{(QK^\top + M)}{\sqrt{d}} \right) V \quad (4)$$

where  $d$  is the dimension of the queries and keys and acts as a scaling factor. The mask  $M$  is used to handle the changing number of detected humans at each timestep, as we will expand below.

**Human-human attention:** To learn the importance of each HH edge to the robot decision at time  $t$ , we first weigh each observed human w.r.t. other humans using an HH attention network, which is a self-attention among humans. In HH attention, the current states of humans  $\mathbf{h}_1^t, \dots, \mathbf{h}_n^t$  are concatenated and passed through linear layers with weights  $W_{HH}^Q$ ,  $W_{HH}^K$ , and  $W_{HH}^V$  to obtain  $Q_{HH}^t, K_{HH}^t, V_{HH}^t \in \mathbb{R}^{n \times d_{HH}}$ , where  $d_{HH}$  is the attention size for the HH attention.

$$\begin{aligned} Q_{HH}^t &= [\mathbf{h}_1^t, \dots, \mathbf{h}_n^t]^\top W_{HH}^Q = [q_1, \dots, q_n]^\top \\ K_{HH}^t &= [\mathbf{h}_1^t, \dots, \mathbf{h}_n^t]^\top W_{HH}^K = [k_1, \dots, k_n]^\top \\ V_{HH}^t &= [\mathbf{h}_1^t, \dots, \mathbf{h}_n^t]^\top W_{HH}^V = [v_1, \dots, v_n]^\top \end{aligned} \quad (5)$$

where  $q_i \in \mathbb{R}^{1 \times d_{HH}}$ ,  $k_i \in \mathbb{R}^{1 \times d_{HH}}$ , and  $v_i \in \mathbb{R}^{1 \times d_{HH}}$  are the query embedding, key embedding, and value embedding of the  $i$ -th human, respectively.

In addition, following Eq. 4, a mask  $M^t \in \mathbb{R}^{n \times n}$  indicates the visibility of each human to the robot at current time  $t$  and is obtained from the perception system of the robot. Assume the  $n$ -th human is not visible at time  $t$ . Then  $M^t$  is a matrix filled with zeros, except that every entry in  $n$ -th column is  $-\infty$ . The numerator of Eq. 4 can be express as

$$\begin{aligned} &Q_{HH}^t \cdot (K_{HH}^t)^\top + M^t \\ &= \begin{bmatrix} q_1 k_1 & \cdots & q_1 k_{n-1} & q_1 k_n \\ \vdots & \ddots & \vdots & \vdots \\ q_n k_1 & \cdots & q_n k_{n-1} & q_n k_n \end{bmatrix} + \begin{bmatrix} 0 & \cdots & 0 & -\infty \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & -\infty \end{bmatrix} \\ &= \begin{bmatrix} q_1 k_1 & \cdots & q_1 k_{n-1} & -\infty \\ \vdots & \ddots & \vdots & \vdots \\ q_n k_1 & \cdots & q_n k_{n-1} & -\infty \end{bmatrix} \end{aligned} \quad (6)$$

Let  $V_{HH}^t = [v_1, \dots, v_n]^\top$ , where  $v_i \in \mathbb{R}^{1 \times d_{HH}}$  is the value embedding of the  $i$ -th human. Then, based on Eq. 4, the weighted human embeddings  $v_{HH}^t \in \mathbb{R}^{n \times d_{HH}}$  is

$$\begin{aligned} v_{HH}^t &= \text{softmax} \left( \frac{Q_{HH}^t (K_{HH}^t)^\top + M^t}{\sqrt{d}} \right) \cdot V_{HH}^t \\ &= \begin{bmatrix} c_1 q_1 k_1 & \cdots & c_1 q_1 k_{n-1} & 0 \\ \vdots & \ddots & \vdots & \vdots \\ c_n q_n k_1 & \cdots & c_n q_n k_{n-1} & 0 \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ \vdots \\ v_{n-1} \\ v_n \end{bmatrix} \\ &= \begin{bmatrix} c_1 q_1 k_1 v_1 + \cdots + c_1 q_1 k_{n-1} v_{n-1} \\ \vdots \\ c_n q_n k_1 v_1 + \cdots + c_n q_n k_{n-1} v_{n-1} \end{bmatrix} \end{aligned} \quad (7)$$

where  $c_1, \dots, c_n$  are constants that reflect the effect of the scaling factor  $d$  and the softmax function. Thus, the value of the  $n$ -th missing human  $v_n$  is eliminated by the mask  $M^t$  and thus does not affect the resulting weighted human embedding  $v_{HH}^t$ . The mask that indicates the visibility of each human prevents attention to undetected humans, which is common in crowd navigation due to the partial observability caused by the limited robot sensor range, occlusions, imperfect human detectors, etc [31]. Additionally, the mask provides unbiased gradients to the networks, which stabilizes and accelerates the training [9], [56].

**Robot-human attention:** After the humans are weighted by HH attention, we weigh their embeddings again w.r.t. the robot with another RH attention network to learn the importance of each RH edge. In RH attention, we first embed the robot state  $w^t$  with a fully connected layer, which results in the key for RH attention  $K_{RH}^t \in \mathbb{R}^{1 \times d_{RH}}$ . The query and the value,  $Q_{RH}^t, V_{RH}^t \in \mathbb{R}^{n \times d_{RH}}$ , are the other two linear embeddings of the weighted human features from HH attention  $v_{HH}^t$ .

$$Q_{RH}^t = v_{HH}^t W_{RH}^Q, \quad K_{RH}^t = w^t W_{RH}^K, \quad V_{RH}^t = v_{HH}^t W_{RH}^V \quad (8)$$

We compute the attention score from  $Q_{RH}^t, K_{RH}^t, V_{RH}^t$ , and the mask  $M^t$  to obtain the weighted human features for the second time  $v_{RH}^t \in \mathbb{R}^{1 \times d_{RH}}$  as in Eq. 4.

2) *Incorporating obstacle and temporal information*: We first feed the point cloud that represents obstacles,  $o^t$ , into a 1D-CNN followed by a fully connected layer to get an obstacle embedding  $v_O^t$ . Then, we embed the robot states  $w^t$  with linear layers  $f_R$  to obtain a robot embedding  $v_R^t$ .

$$v_O^t = f_{CNN}(o^t), \quad v_R^t = f_R(w^t) \quad (9)$$

Finally, the robot and obstacle embeddings are concatenated with the twice weighted human features  $v_{RH}^t$  and fed into the GRU<sup>1</sup>:

$$h^t = \text{GRU}(h^{t-1}, ([v_{RH}^t, v_R^t, v_O^t])) \quad (10)$$

where  $h^t$  is the hidden state of GRU at time  $t$ . Finally, the  $h^t$  is input to a fully connected layer to obtain the value  $V(s^t)$  and the policy  $\pi(a^t | s^t)$ .

<sup>1</sup>From experiments, we find that adding a third obstacle-agent attention network, where the obstacle embedding is the key and the robot and human embeddings are the query and value, does not improve the performance. Thus, we keep the simple concatenation design to incorporate obstacle information.

3) *Training*: We train the entire network with Proximal Policy Optimization (PPO) [57] in a simulator as shown in Fig. 2(a) (see Sec. V-A for details of the simulator). At each timestep  $t$ , the simulator provides all state information that constitutes  $s^t$ , which is fed to the HEIGHT network. The network outputs the estimated value of the state  $V(s^t)$  and the logarithmic probabilities of the robot's action  $\pi(a^t|s^t)$ , both of which are used to compute the PPO loss and then update the parameters in the network. In training, the robot action is sampled from the action distribution  $\pi(a^t|s^t)$ . In testing, the robot takes the action with the highest probability  $a^t$ . The robot action  $a^t$  is fed into the simulator to compute the next state  $s^{t+1}$ , and then the loop continues.

Without any supervised learning, our method is not limited by the performance of expert demonstrations [3], [4], [50]. However, to improve the low training data efficiency, an inherent problem of RL, HEIGHT can also be trained with a combination of imitation learning and RL.

4) *Summary*: We present a structured and principled approach to design the robot policy network for crowd navigation in constrained environments. By decomposing the complex scenario into independent components, we split the complex problem into smaller functions, which are used to learn the parameters of the corresponding functions. By combining all components above, the end-to-end trainable HEIGHT allows the robot to perform spatial and temporal reasoning on all pairwise interactions, leading to better navigation performance.

## V. SIMULATION EXPERIMENTS

In this section, we present our environment, experiment setup, and results in simulation. Our experiments are guided by the following questions: (1) What is the advantage of our split scene representation compared with alternative representations? (2) What is the importance of the graph formulation and why do we differentiate types of edges with a heterogenous st-graph? (3) What is the importance of HH and RH attention in HEIGHT? (4) What are the failure cases of our method?

### A. Simulation environment

We conduct simulation experiments in *random environment* in Fig. 5(a) developed with PyBullet [58]. The robot, humans, and static obstacles are in a  $12\text{ m} \times 12\text{ m}$  arena. In each episode, rectangular obstacles are initialized with random shapes and random poses. The width and the length of each static obstacle are sampled from  $\mathcal{N}(1, 0.6^2)$  and are limited in  $[0.1, 5]$  meters. The initial positions of the humans and the robot are also randomized. The starting and goal positions of the robot are randomly sampled inside the arena. The distance between the robot's starting and the goal positions is between 5 m and 6 m. Some humans are dynamic and some are static. The goals of dynamic humans are set on the opposite side of their initial positions to create circle-crossing scenarios. In training, the number of humans varies from 5 to 9 and the number of static rectangular obstacles varies from 8 to 12. Among all humans, 0-2 of them are static and the rest are dynamic. In testing, the number of humans and obstacles are shown in the first column

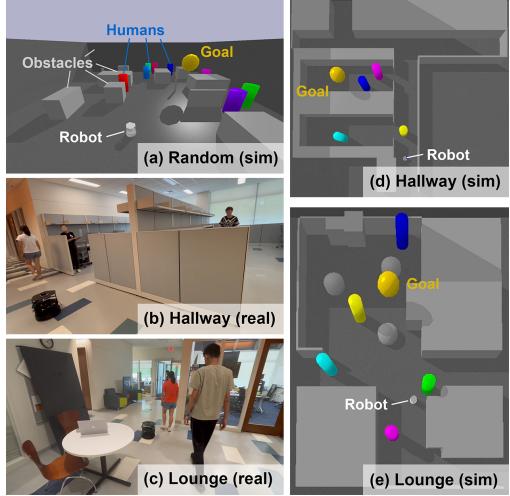


Fig. 5: **Simulation and real-world environments.** (a) In our simulation benchmark, the colored cylinders denote humans and gray objects denote obstacles. Each episode is randomized (Sec. V-A). (b) and (c) are everyday indoor environments for sim2real. Before real-world testing in (b) and (c), the robot policy is trained in (d) and (e) respectively (Sec. VI-A).

of Table I. The heights of humans and obstacles are all above the height of the robot LiDAR to ensure detectability.

To simulate a continuous human flow similar to [8], [9], [28], dynamic humans will move to new random goals immediately after they arrive at their goal positions or they get stuck in front of narrow passageways for more than 10 timesteps. All dynamic humans are controlled by ORCA [13]. 80% of dynamic 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. Meanwhile, the simulation maintains enough reactive humans to resemble the real crowd behaviors. We use holonomic kinematics for humans. The preferred speed of humans ranges from 0.4 m/s to 0.6 m/s to accommodate the speed of the robot. We assume that humans can achieve the desired velocities immediately, and they will keep moving with these velocities for the next  $\Delta t$  seconds. The simulation and control frequency  $\Delta t$  is 0.1 s. The maximum length of an episode is 491 timesteps or 49.1 s.

### B. Experiment setup

We now introduce the baselines and ablation models, training procedure, and evaluation metrics.

1) *Baselines*: We compare the performance of our method with the following baselines:

- **Dynamic window approach (DWA)** [1] searches for the optimal velocity that brings the robot to the goal with the maximum clearance from any obstacle. DWA is a model-based method that only considers the current state. In addition, DWA represents both humans and obstacles as groups of small circles.
- **A<sup>\*</sup>+CNN** [7] is a hybrid method. With a map of the environment, A<sup>\*</sup> is the global planner and generates 6 waypoints in the beginning of an episode. The inputs to the robot RL policy are a 2D LiDAR point cloud,

the robot state, the waypoints, and the robot’s final goal. No human detections are used and human features are included in the point cloud. The point cloud is passed through a CNN local planner, whose output is concatenated with the embedding of other inputs. In addition to the reward function in Eq. 1, the robot receives a small reward of 2 if it arrives at any waypoint in sequence.

- **Decentralized structural RNN (DS-RNN)** [8] is an RL-based method that represents static obstacles as groups of small circles. In network architecture, DS-RNN only contains the RH attention and the GRU.
- **Homogeneous graph attention network (HomoGAT)** [44] is RL-based and splits human and obstacle inputs. However, HomoGAT does not differentiate between 3 types of nodes and 3 types of edges in policy network. Instead, HomoGAT uses a single self-attention network to weigh humans, the robot, and the obstacle point cloud and feed the weighted sum of all embeddings into a GRU. HomoGAT is similar to Chen et al. [6] and Liu et al. [50] but the input, output, and training algorithm are kept the same as our method for a fair comparison.

In summary, DWA, A<sup>\*</sup>+CNN, and DS-RNN mix humans and obstacles in both input representations and navigation algorithm. HomoGAT only mix humans and obstacles in algorithm but not in input, while ours distinguishes them in both input and algorithm.

2) *Ablations*: To show the benefits of each attention network, we perform an ablation study on the two attention models. The ablated models are defined as follows:

- **No attn**: Both RH and HH attention networks are removed. No attn model only has the embedding layers for the inputs and the GRU.
- **RH**: The HH attention network is removed and the humans are weighted only once w.r.t. the robot. Everything else is the same as ours.
- **HH**: The RH attention network is removed and the humans are weighted only once w.r.t. other humans. Everything else is the same as ours.
- **RH+HH (ours)**: The full network as shown in Fig. 4(b).

3) *Training*: We train all RL methods, including all baselines and ablations except DWA, for  $2 \times 10^8$  timesteps with a learning rate  $5 \times 10^{-5}$ . The learning rate decays at a linear rate with respect to training timesteps. To accelerate and stabilize training, we run 28 parallel environments to collect the robot’s experiences. At each policy update, 30 steps of 6 episodes are used. We train and test all methods in a commercial desktop computer with an Intel Core i9-13900K processor, 32 GB memory, and a NVIDIA RTX 4090 GPU. Training our method takes approximately 48 hours.

4) *Evaluation*: We test all methods with the same 500 random unseen test episodes. For RL-based methods, we test the last 5 checkpoints (equivalent to the last 6000 training steps) and report the result of the checkpoint with the highest success rate. We conduct the testing in 5 different human and obstacle densities, as shown in the first columns of Table I and Table II. The first set of densities is the same as the training environment and is thus in *training distribution*. To test the

generalization of methods, we change the range of human or obstacle numbers in the remaining four environments to values that do not overlap with those used in training. Thus, these 4 environments are out-of-distribution (OOD).

All testing scenarios, including those in the *training distribution*, are unseen during training. This is because in each testing episode, a new seed that is not used in training determines the number of humans and obstacles, the starting and goal positions of all agents, and the size and pose of obstacles. Only the 4 arena walls do not change across different episodes.

The testing metrics measure the quality of the navigation and include the success rate, collision rate with humans and obstacles, timeout rate, the average navigation time of successful episodes in seconds, and path length of the successful episodes in meters.

### C. Results

1) *Effectiveness of scene representation*: To analyze the effects of input scene representations on crowd navigation algorithms, we compare ours and HomoGAT, the two methods that distinguish human and obstacle inputs, with baselines that mix humans and obstacles in input representation: the model-based DWA, the RL-based DS-RNN, and the hybrid planner A<sup>\*</sup>+CNN. The results are shown in Table I.

For DWA, treating humans as obstacles leads to the highest average human collision rates (0.54) and a freezing problem, indicated by the highest average timeout rates (0.21), in all environments. For example, the robot in Fig. 6 stays close to everything and thus fails to avoid the magenta human in time.

Similarly, by representing obstacles as groups of circles, DS-RNN performs better in the *More crowded* environment (Fig 7(d)) than in the *More constrained* environment (Fig 6(d)). In addition, Table I shows that DS-RNN achieves the highest average collision rate with obstacles (0.13) in all environments. Furthermore, the obstacle collision rate of DS-RNN increases in all 4 OOD environments, indicating an overfitting problem. Among the OOD environment, the percentage of obstacle collision increase is higher in environments with higher obstacle-to-human ratios. For example, in *Less crowded* with the highest percentage of obstacles, the obstacle collision rate (0.21) increases by 133% w.r.t. the obstacle collision rate in *training distribution* (0.09). On the contrary, in *More crowded* with the lowest percentage of obstacles, the obstacle collision rate (0.06) drops by 33%. The reason is that DS-RNN has trouble inferring the geometric shapes of obstacles from a large group of circles, and thus fails to avoid collision with them. Thus, treating both humans and obstacles as circles is not an ideal input representation for robot navigation algorithms.

For A<sup>\*</sup>+CNN, the A<sup>\*</sup> global planner does not account for humans and the CNN local planner represents all observations as a 2D point cloud. As a result, from Table I, A<sup>\*</sup>+CNN has the highest average timeout rate (0.10) and the longest average time (18.99) among RL-based methods in all environments, especially the two most challenging ones, for the following 2 reasons. (1) The waypoints can fail to guide the robot to reach the goal because the waypoints lose optimality as agents move.

TABLE I: Baseline comparison results with different human and obstacle densities in unseen environments

| Environment  | Method        | Success↑    | Collision↓  |             |              | Timeout↓    | Nav Time↓    | Path Len↓    |
|--|---------------|-------------|-------------|-------------|--------------|-------------|--------------|--------------|
|  |               |             | Overall     | w/ Humans   | w/ Obstacles |             |              |              |
| <b>Training distribution</b><br>5-9 humans<br>8-12 obstacles | DWA [1]       | 0.16        | 0.68        | 0.63        | 0.05         | 0.15        | 28.45        | 11.52        |
|  | A*+CNN [7]    | 0.64        | 0.29        | 0.28        | <b>0.01</b>  | 0.07        | 25.72        | 12.30        |
|  | DS-RNN [8]    | 0.80        | 0.20        | 0.11        | 0.09         | <b>0.00</b> | 19.90        | 10.78        |
|  | HomoGAT [44]  | 0.86        | 0.14        | 0.13        | <b>0.01</b>  | <b>0.00</b> | 18.66        | 10.36        |
|  | HEIGHT (ours) | <b>0.88</b> | <b>0.12</b> | <b>0.09</b> | 0.03         | <b>0.00</b> | <b>18.31</b> | <b>10.34</b> |
| <b>Less crowded</b><br>0-4 humans<br>8-12 obstacles          | DWA [1]       | 0.29        | 0.37        | 0.28        | 0.09         | 0.34        | 25.74        | 14.42        |
|  | A*+CNN [7]    | 0.84        | 0.12        | 0.11        | <b>0.01</b>  | 0.04        | 22.64        | 11.73        |
|  | DS-RNN [8]    | 0.72        | 0.28        | 0.07        | 0.21         | <b>0.00</b> | 17.21        | <b>9.56</b>  |
|  | HomoGAT [44]  | 0.95        | 0.05        | 0.03        | 0.02         | <b>0.00</b> | 16.64        | 10.17        |
|  | HEIGHT (ours) | <b>0.97</b> | <b>0.03</b> | <b>0.02</b> | <b>0.01</b>  | <b>0.00</b> | <b>16.32</b> | 10.04        |
| <b>More crowded</b><br>10-14 humans<br>8-12 obstacles        | DWA [1]       | 0.11        | 0.78        | 0.74        | 0.04         | 0.11        | 29.60        | <b>10.37</b> |
|  | A*+CNN [7]    | 0.47        | 0.42        | 0.39        | <b>0.03</b>  | 0.11        | 27.33        | 12.47        |
|  | DS-RNN [8]    | 0.76        | <b>0.22</b> | <b>0.16</b> | 0.06         | 0.01        | 23.08        | 11.67        |
|  | HomoGAT [44]  | 0.72        | 0.28        | 0.23        | 0.05         | <b>0.00</b> | 20.46        | 10.57        |
|  | HEIGHT (ours) | <b>0.78</b> | <b>0.22</b> | 0.19        | <b>0.03</b>  | <b>0.00</b> | <b>19.69</b> | 10.39        |
| <b>Less constrained</b><br>5-9 humans<br>3-7 obstacles       | DWA [1]       | 0.17        | 0.57        | 0.55        | 0.02         | 0.26        | 28.50        | 14.21        |
|  | A*+CNN [7]    | 0.66        | 0.29        | 0.28        | <b>0.01</b>  | 0.05        | 23.63        | 11.98        |
|  | DS-RNN [8]    | 0.66        | 0.34        | 0.20        | 0.14         | <b>0.00</b> | 18.01        | <b>9.76</b>  |
|  | HomoGAT [44]  | 0.88        | 0.12        | 0.10        | 0.02         | <b>0.00</b> | 17.66        | 10.37        |
|  | HEIGHT (ours) | <b>0.90</b> | <b>0.10</b> | <b>0.09</b> | <b>0.01</b>  | <b>0.00</b> | <b>17.20</b> | 10.23        |
| <b>More constrained</b><br>5-9 humans<br>13-17 obstacles     | DWA [1]       | 0.14        | 0.66        | 0.49        | 0.17         | 0.20        | 30.47        | 11.35        |
|  | A*+CNN [7]    | 0.48        | 0.29        | 0.23        | <b>0.06</b>  | 0.23        | 27.28        | 13.11        |
|  | DS-RNN [8]    | 0.71        | 0.23        | 0.09        | 0.14         | 0.06        | 23.45        | 11.58        |
|  | HomoGAT [44]  | 0.80        | 0.20        | 0.11        | 0.09         | <b>0.00</b> | 19.20        | <b>10.51</b> |
|  | HEIGHT (ours) | <b>0.84</b> | <b>0.15</b> | <b>0.07</b> | 0.08         | 0.01        | <b>18.79</b> | 10.65        |

Consequently, in OOD scenarios such as Fig. 6(b), the CNN policy is especially bad at long-horizon planning because it is overfitted with good waypoints. (2) By mixing humans and obstacles in its input, the robot policy sometimes has a hard time distinguishing dynamic and static obstacles. For example, in Fig. 6(b), the robot keeps a large distance from everything and thus is less agile and efficient compared with the robots in Fig. 6(e) and (f). Thus, for RL-based approaches, treating humans as obstacles leads to overfitting to specific types of scenarios, preventing the policies from generalizing to OOD scenarios which are common in the real-world.

By splitting human and obstacle input representations, HomoGAT and HEIGHT achieve the top 2 performance across most metrics and environments. Especially, HEIGHT learns to keep a larger distance from human paths yet a shorter distance from obstacles to balance safety and efficiency (Fig. 6(f)), indicated by the lowest overall collision rates (0.17) and the shortest average navigation time (18.10) in all environments. Therefore, we conclude that our split representation consisting of detected human states and obstacle point clouds is most suitable to learn robot navigation in crowded and constrained environments with RL, among all popular choices in Table I.

2) *Effectiveness of the heterogeneous st-graph*: To justify our heterogeneous st-graph formulation, we compare our method with A\*+CNN with no graph concept and HomoGAT with a homogenous graph attention network in Table I.

A\*+CNN lacks structure in input representation, and subsequently, lacks structure in network architecture. For this reason, as we discussed in Sec. V-C1, A\*+CNN shows much worse average success rate (0.62 v.s. 0.84), navigation time (25.32 v.s. 18.52), and path length (12.32 v.s. 10.40) than HomoGAT which has a simple st-graph in all environments.

A\*+CNN also exhibits larger variances in terms of success rate (0.019 v.s. 0.0060), navigation time (3.62 v.s. 1.70), and path length (0.22 v.s. 0.019) across different OOD environments. Especially in the challenging *More crowded* and *More constrained* environments, A\*+CNN exhibits a larger drop in average success rate compared with HomoGAT (0.165 v.s. 0.10). This finding shows that even a simple st-graph structure can lead to a performance gain, indicating the importance of spatial and temporal reasoning for robot crowd navigation.

However, occasional failures of HomoGAT still exist such as Fig. 6(b). Without differentiating RH and OA interactions, the robot maintains similar distances from humans and obstacles, yet fails because RH interactions require more space. As a step further, with a heterogeneous st-graph, HEIGHT treats the 3 types of edges with different network components. As a result, HEIGHT outperforms HomoGAT and achieves the best average success rate (0.87), navigation time (18.06), and path length (10.33) in all environments. The variance success rate (0.0040) and navigation time (1.40) are also the best, whereas the path length variance (0.040) is the runner-up across all environments. In addition, the average success rate drop in the two challenging environments of HEIGHT is smaller than HomoGAT (0.07 v.s. 0.10). The reason is that the heterogeneous components allow the robot to reason about the different effects of HH, RH, and OA spatial interactions. For example, in Fig. 6(f) and Fig. 7(e), HEIGHT chooses a path that avoids the most crowded region, yields to humans when the robot must encounter them, and stays closer to walls for efficiency. Therefore, we conclude that the spatial and temporal reasoning on different types of interactions is the key to ensuring good in-distribution performance and OOD generalization in crowded and interactive environments.

TABLE II: Ablation study results with different human and obstacle densities in unseen environments

| Environment  | Method       | Success↑    | Collision↓  |             |              | Timeout↓    | Nav Time↓    | Path Len↓    |
|--|--------------|-------------|-------------|-------------|--------------|-------------|--------------|--------------|
|  |              |             | Overall     | w/ Humans   | w/ Obstacles |             |              |              |
| <b>Training distribution</b><br>5-9 humans<br>8-12 obstacles | No attn      | 0.52        | 0.46        | 0.41        | 0.05         | 0.02        | 26.48        | 11.81        |
|  | RH           | 0.85        | 0.15        | 0.13        | <b>0.02</b>  | <b>0.00</b> | 18.86        | 10.41        |
|  | HH           | 0.87        | <b>0.12</b> | 0.10        | <b>0.02</b>  | 0.01        | <b>18.31</b> | 10.42        |
|  | RH+HH (ours) | <b>0.88</b> | <b>0.12</b> | <b>0.08</b> | 0.04         | <b>0.00</b> | 18.49        | <b>10.28</b> |
| <b>Less crowded</b><br>0-4 humans<br>8-12 obstacles          | No attn      | 0.72        | 0.27        | 0.24        | 0.03         | 0.01        | 21.78        | 10.94        |
|  | RH           | 0.89        | 0.07        | 0.04        | 0.03         | 0.04        | 17.34        | 10.66        |
|  | HH           | 0.94        | 0.06        | 0.05        | <b>0.01</b>  | <b>0.00</b> | <b>16.10</b> | <b>9.90</b>  |
|  | RH+HH (ours) | <b>0.97</b> | <b>0.03</b> | <b>0.02</b> | <b>0.01</b>  | <b>0.00</b> | 16.32        | 10.04        |
| <b>More crowded</b><br>10-14 humans<br>8-12 obstacles        | No attn      | 0.29        | 0.61        | 0.53        | 0.08         | 0.10        | 28.35        | 12.57        |
|  | RH           | 0.70        | 0.29        | 0.25        | 0.04         | 0.01        | 20.39        | 10.49        |
|  | HH           | 0.74        | 0.25        | <b>0.17</b> | 0.08         | 0.01        | 20.34        | 10.53        |
|  | RH+HH (ours) | <b>0.78</b> | <b>0.22</b> | 0.19        | <b>0.03</b>  | <b>0.00</b> | <b>19.69</b> | <b>10.39</b> |
| <i>Less constrained</i><br>5-9 humans<br>3-7 obstacles       | No attn      | 0.55        | 0.43        | 0.42        | <b>0.01</b>  | 0.01        | 24.64        | 11.75        |
|  | RH           | 0.86        | 0.14        | 0.12        | 0.02         | <b>0.00</b> | 18.17        | 10.33        |
|  | HH           | 0.88        | 0.12        | 0.10        | 0.02         | <b>0.00</b> | 17.24        | 10.25        |
|  | RH+HH (ours) | <b>0.90</b> | <b>0.10</b> | <b>0.09</b> | <b>0.01</b>  | <b>0.00</b> | <b>17.20</b> | <b>10.23</b> |
| <b>More constrained</b><br>5-9 humans<br>13-17 obstacles     | No attn      | 0.43        | 0.49        | 0.38        | 0.11         | 0.08        | 26.25        | 11.82        |
|  | RH           | 0.77        | 0.22        | 0.11        | 0.11         | 0.01        | 20.21        | 10.73        |
|  | HH           | <b>0.84</b> | 0.16        | <b>0.06</b> | 0.10         | <b>0.00</b> | 18.94        | 10.66        |
|  | RH+HH (ours) | <b>0.84</b> | <b>0.15</b> | 0.07        | <b>0.08</b>  | 0.01        | <b>18.79</b> | <b>10.65</b> |

3) *Importance of attention networks:* We use ablations to evaluate the contribution of RH and HH attention networks to the performance of HEIGHT, as shown in Table II.

First, in terms of all metrics in most environments, the model with both RH and HH attention shows the strongest results, followed by the models with only one attention, and finally by the model with no attention. For example, in Fig. 7, the ablations in (a), (b), and (c) fail yet our full model in (e) succeeds. Especially in the 2 most challenging *More crowded* and *More constrained* environments, the existence of either RH or HH attention significantly boosts the average success rate by 0.38 and 0.43 respectively, compared with No attn. For example, in Fig. 6 (e), the robot detours and goes dangerously close to the crowds in a constrained area, yet HH attention allows it to realize the danger and resume to a less crowded path. This result shows that reasoning about both HH and RH spatial relationships is essential for our problem, especially in *more crowded* or *more constrained* environments where the spatial interactions are also dense.

Second, by comparing RH and HH, we find that HH attention plays a more important role in all environments in terms of success rate and navigation time. This is because the number of HH edges is larger than the number of RH edges in most cases. Thus, the robot can observe and is affected by a relatively larger number of HH interactions yet a smaller number of RH interactions.

4) *Failure cases:* By visualizing the testing episodes across all environments, we find that HEIGHT typically collides when (1) a human arrives at its goal and suddenly changes directions to a new goal, as shown in Fig. 8(a), or (2) the robot surroundings are extremely crowded and constrained, and almost all free paths are blocked by humans (Fig. 8(b)). In these cases, due to its speed limit, the robot sometimes cannot switch to an alternative path in time to prevent collisions. To remedy the difficulty of RL in long-term decision making [59],

in future work, our method can be combined with long-term prediction and path planning algorithms that consider the stochasticity of pedestrian motions.

5) *Additional insights:* Besides answering the four research questions above, we provide additional insights obtained from simulation experiments below.

*Effectiveness of RL planning:* In Table I, DWA is a model-based approach that determines robot actions based on only the current state, which leads to the worst performance. Thus, reactive policies are not sufficient to solve our problem, justifying the necessity of long-sighted planning. In contrast, A<sup>\*</sup>+CNN combines planning and RL, yet relies on occupancy maps with only obstacles and not humans. As humans move, the increasingly inaccurate occupancy map reduces the optimality of waypoints, which negatively affects overall navigation. On the other hand, RL learns to maximize the expected long-term returns based on both the current and historical states of all components in the scene. Consequently, the remaining 3 RL methods, especially HomoGAT and HEIGHT, outperform A<sup>\*</sup>+CNN in most metrics and in most environments. Thus, to ensure the best navigation performance, it is important to optimize the entire planning system based on the task setting instead of optimizing only a part of it.

*Difficulty of scenarios:* From Table I and Table II, we observe that, besides the overfitted DS-RNN, all methods show the same trend of performance change in the 5 environments: *Less crowded* > *Less constrained* > *More constrained* > *More crowded*. Obviously, adding more humans or obstacles increases task difficulty. But interestingly, the change in the number of humans has a larger effect on the task difficulty than the change in the number of obstacles. This phenomenon shows that avoiding collisions with humans is more difficult than avoiding obstacles in nature.

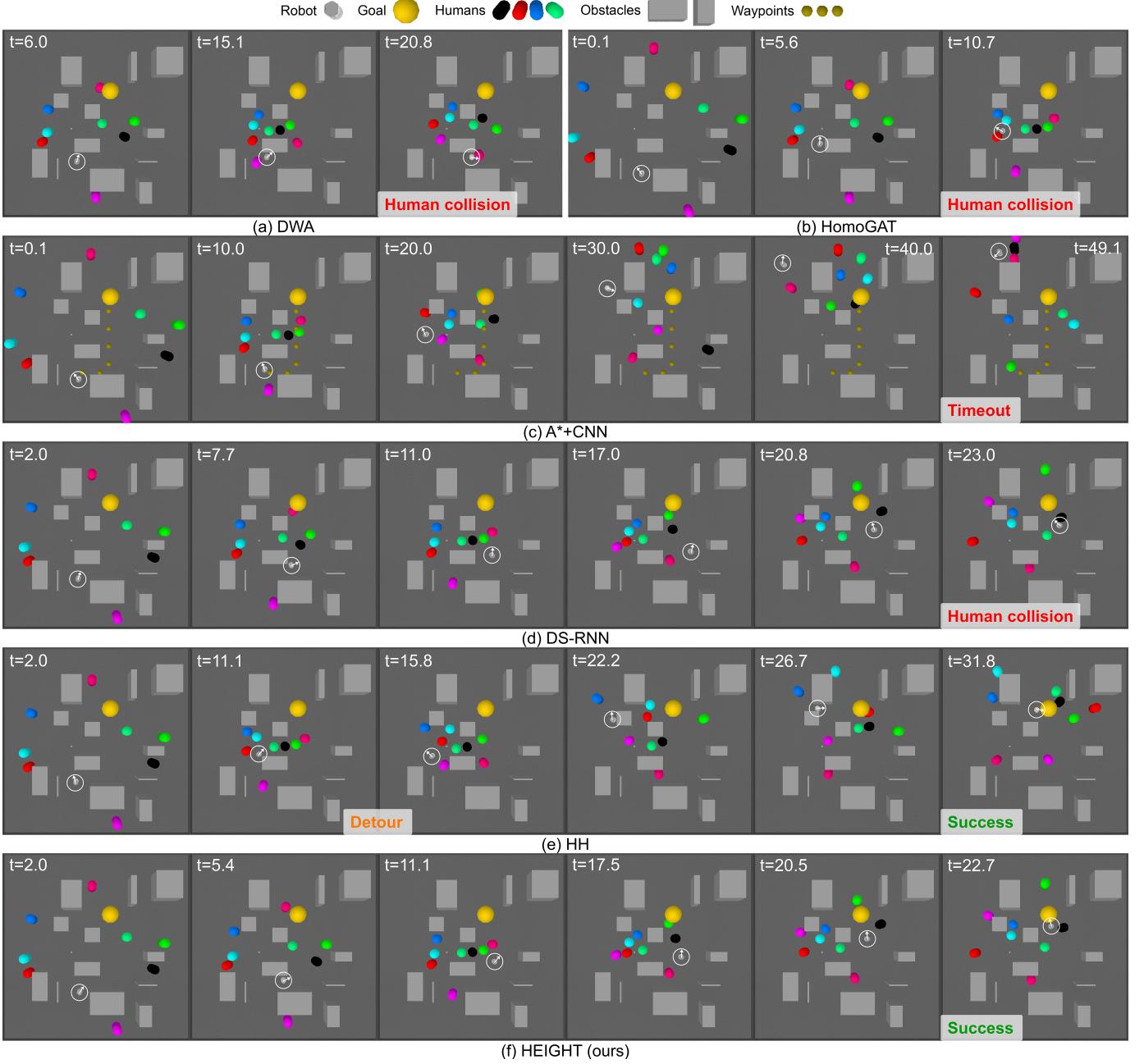


Fig. 6: **Comparison of different methods in the same testing episode in *More Constrained* environment.** The robot is centered in white circles and its orientation is denoted by white arrows. More qualitative results can be found in the video attachment and at <https://sites.google.com/view/crowdnav-height/home>.

## VI. REAL-WORLD EXPERIMENTS

In this section, we present our hardware setup and sim2real testing results in everyday environments with pedestrians and static constraints.

### A. Experiment setup

We train the sim2real policies in the simulators of a hallway (Fig. 5(b)) and a lounge (Fig. 5(c)) for  $2 \times 10^8$  timesteps with a decaying learning rate  $5 \times 10^{-5}$ . To learn robust policies, we inject noises into the agent positions and robot control. Then, as shown in Fig. 5 (d) and (e), we test the policies in the two corresponding everyday environments in a university building **without any additional training**. In the hallway environment where the free space is extremely narrow, we

test the robot's ability to handle constraints with low density crowds. In the lounge environment, we test the robot's ability to avoid dense crowds and obstacles with more diverse shapes. The distance between the starting and the goal position of the robot ranges from 6m to 11m. The pedestrians were told to react naturally to the robot based on their own preferences. In some testing episodes, other pedestrians who were unaware of our experiment also engaged with the robot.

The configuration of hardware testing is shown in Fig. 2(b). We use a TurtleBot 2i with an RPLIDAR-A3 laser scanner. We first use the ROS gmapping package to create a map of the environment. Then, we process the map to combine small obstacles and eliminate noises. To reduce sim2real gap of the LiDAR point clouds, we use artificial point clouds obtained

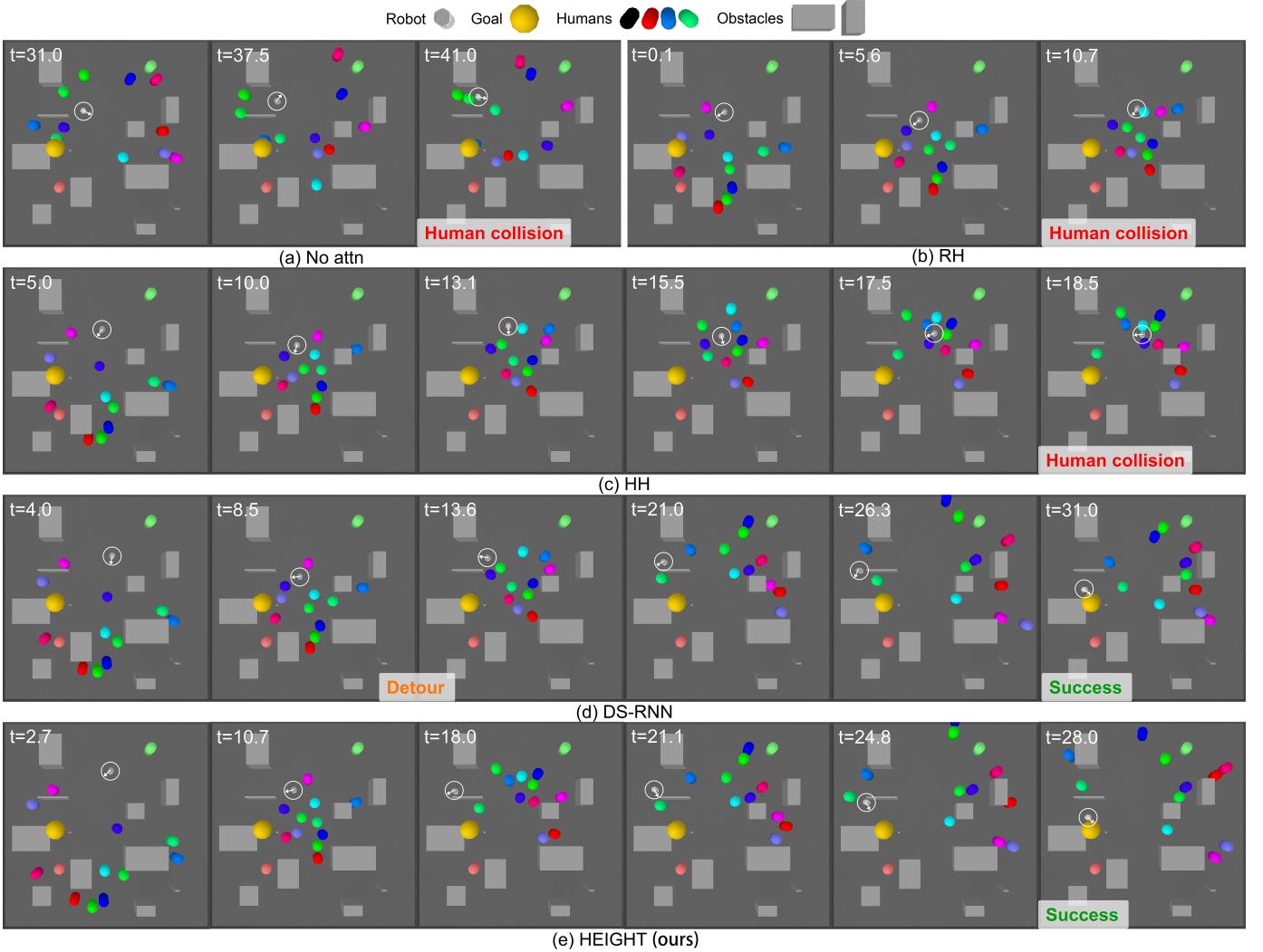


Fig. 7: Comparison of different methods in the same testing episode in *More Crowded* environment.

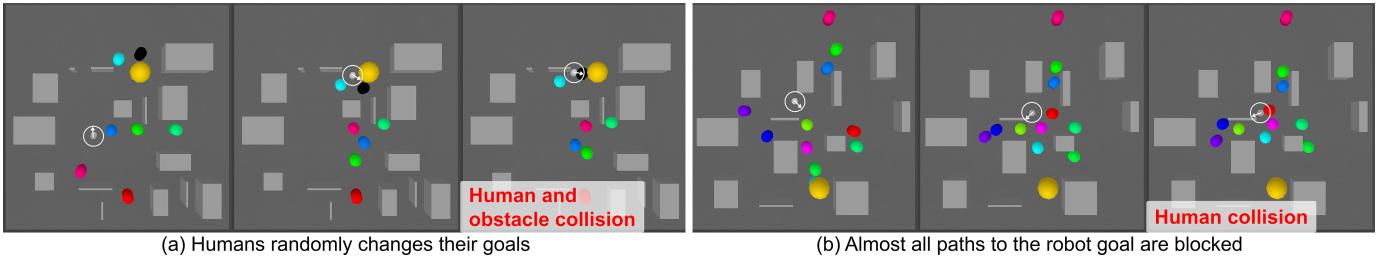


Fig. 8: **Failure cases of our method.** (a) The black human was going downwards for a while, but suddenly changes its direction toward the robot. (b) All efficient paths toward the goal are blocked or will soon be blocked by human crowds.

from localization and mapping, instead of the raw point clouds from LiDAR. The reason is that the artificial point clouds are cleaner and are not obstructed by humans. When a navigation trial begins, a user inputs the robot goal position through a keyboard. To detect human positions, we use an off-the-shelf people detector [52]. We use an Intel RealSense tracking camera T265 to obtain the pose of the robot ( $p_x, p_y, \theta$ ) and the robot wheel encoder to obtain its velocity ( $u_x, u_y$ ).

Our baseline is the ROS navigation stack, which uses the dynamic window approach (DWA) [1] as the local planner and A\* as the global planner. For each method, we run 30 trials in total. Among all trials, the start goal positions of the robot

are the same, while the number and trajectories of pedestrians are similar. We measure the success, collision, timeout rates, and navigation time of successful episodes as testing metrics.

## B. Results

In the highly constrained yet less crowded Hallway environment, ROS navigation stack often needs to spin in place to replan, as shown by the higher navigation time in Table III. Some of the spinning recovery attempts fail and result in timeouts. In ROS navigation stack, both global and local planners treat humans as obstacles. As a result, similar to the baselines in Sec. V-C1, the robot has difficulties distinguishing

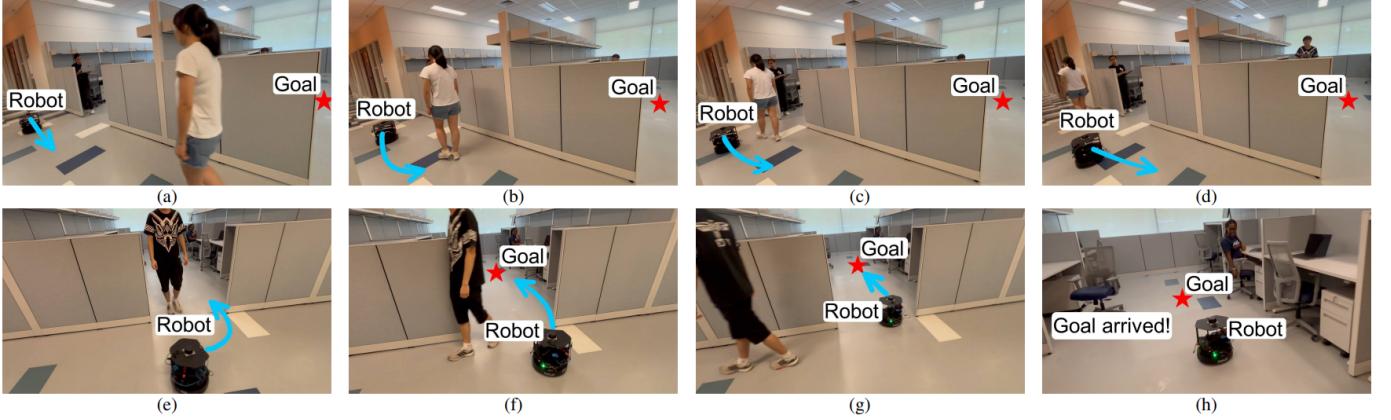


Fig. 9: A testing episode of our method in the real Hallway environment. The blue arrow denotes the robot path that results from its actions. The red star denotes the goal position. In this episode, the turtlebot avoids two pedestrians, one after another in a narrow corridor, enters a narrow doorway, and arrives at the goal. More qualitative results can be found in the video attachment and at <https://sites.google.com/view/crowdnav-height/home>.

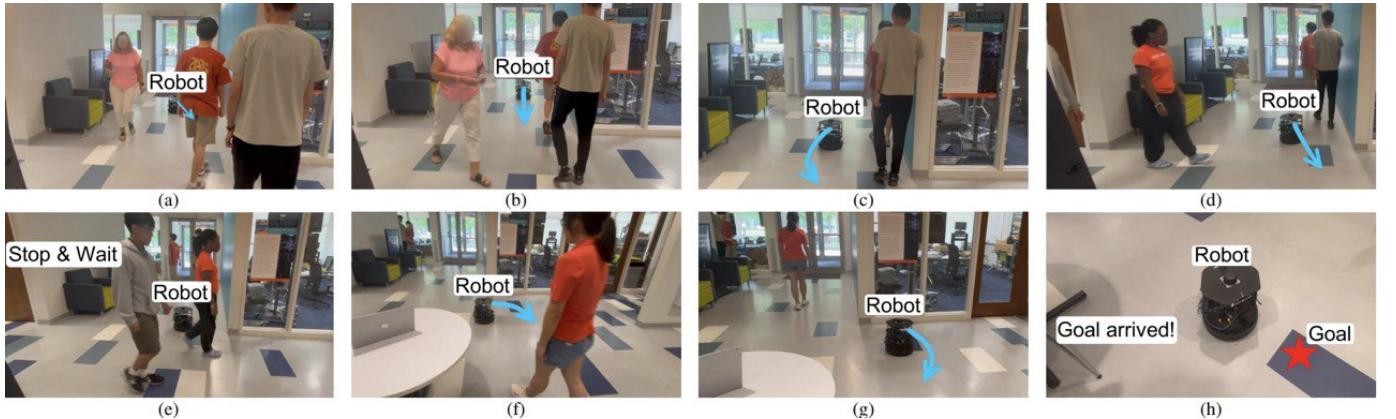


Fig. 10: A testing episode of our method in the real Lounge environment. The turtlebot avoids multiple groups of people who pass each other in different heading directions, avoids the walls and furnitures, and arrives at the goal.

TABLE III: Real-world results in 2 everyday environments

| Environment                  | # of trials | Method                            | Success↑            | Collision↓          | Timeout↓            | Nav Time↓             |
|------------------------------|-------------|-----------------------------------|---------------------|---------------------|---------------------|-----------------------|
| <b>Hallway</b><br>1-2 humans | 12          | Navigation Stack<br>HEIGHT (ours) | 0.72<br><b>1.00</b> | 0.06<br><b>0.00</b> | 0.22<br><b>0.00</b> | <b>16.71</b><br>22.36 |
| <b>Lounge</b><br>1-6 humans  | 18          | Navigation Stack<br>HEIGHT (ours) | 0.83<br>0.83        | 0.17<br>0.17        | 0.00<br>0.00        | 32.00<br><b>30.71</b> |

dynamic obstacles that will clear out and static obstacles that will always stay, causing failures in highly constrained spaces. On the contrary, by using different representations of humans and obstacles, our method is able to explore the different strategies to avoid them through trial and error during RL training. Consequently, the robot demonstrates high success rates in Table III. Qualitatively, the robot is able to safely pass a human in a narrow corridor (Fig. 9 (a)-(d)), takes a wide turn to give a human enough room, and enters an extremely narrow doorway (Fig. 9 (e)-(h)).

In the less constrained yet more crowded Lounge environment, ROS navigation stack and ours have the same success and collision rate, because the Lounge environment has enough space for the navigation stack to apply a “stop and wait” strategy for humans and obstacles. However, as a cost, “stop and wait” takes a longer average time to arrive at goals as shown in the last column of Table III. Different from this naïve strategy, HEIGHT adapts its navigation behaviors based on

different types of spatio-temporal interactions. For example, in Fig. 10 (a)-(c), since pedestrians walking in opposite directions are crossing each other, the robot first turns left to avoid the lady on its right, and then turns right to avoid the two males on its left. Then, in Fig. 10 (d)-(e) and (f)-(g), the robot chooses actions that deviate from the current and intended paths of pedestrians, walls, and the table to safely arrive at the goal. The failure cases of both methods are caused by the tables with irregular 3D shapes. Due to the height of the 2D LiDAR, some wide parts of the tables are not detected or mapped, and thus the robot cannot avoid them. Further pre-processing of the LiDAR point clouds or adding 3D sensors can mitigate this problem.

Furthermore, the successful sim2real transfer from a cheap and low-fidelity simulator to complex everyday environments demonstrates the robustness of our input representation and the cost-efficiency of our pipeline design.

## VII. DISCUSSIONS, LIMITATIONS, AND FUTURE WORK

Deep RL is a promising tool to solve robotic problems that are beyond the capabilities of traditional rule-based methods without large-scale real-world datasets. However, preventing the performance degradation of a deep RL policy when inevitable distribution shifts happen, especially in real-world, is challenging. In this section, we reflect on the key components of our framework, discuss the limitations of our approach, and propose directions for future research.

### A. Sim2real through Real2sim

To overcome sim2real gaps, the design of simulation pipelines needs to be guided by the constraints of hardware and environments in the real-world. On one hand, we determine the input representation of HEIGHT based on what could be obtained from sensors and off-the-shelf perception modules and how accurate they are. We find that intermediate features, such as detected human positions and processed point clouds, reduces sim2real gaps. On the other hand, we also ensure the consistency of the simulation and real-world, such as the robot action space, whenever possible. These design choices allow our policy to generalize to different simulation environments and deploy to challenging real-world scenarios.

However, although we have minimized the sim2real gaps through real2sim, certain gaps still exist. In real-world experiments, the difficulty of the task is reduced and the agility of the robot policy is only partially transferred from simulation. To further align the simulation and the real-world, we plan to explore the following directions for future work: (1) developing a more natural pedestrian model to replace the ORCA humans in the simulator, (2) revising our pipeline to enable self-supervised RL fine-tuning in the real-world [60], [61], (3) using a small amount of real-world data to automatically optimize the parameters of our simulator to match the real-world environment [62], [63].

### B. Scene representation

A good scene representation is tailored to the needs of its downstream task. Besides the above sim2real considerations, our scene representation is split due the different nature of humans and obstacles for robot collision avoidance. The size of humans are small and their shapes are simple. Therefore, to avoid humans, the robot only needs to treat them as circles with a heading direction. In contrast, obstacles have larger and more complicated surfaces. The part of the obstacle contours that faces the robot is more important for collision avoidance. Therefore, point clouds are the most intuitive way to represent such useful information about obstacles. Our experiments in Sec. V show that this split scene representation reduces robot collision avoidance with both dynamic and static obstacles.

A side effect of our scene abstraction is the loss of detailed information such as gaits of humans and 3D shapes of obstacles. However, we argue that this is a tradeoff to minimize sim2real gaps with limited simulation tools and computation resources. Another side effect is the cascading errors between the perception modules and robot policy, such as inaccurate

human detections or robot localization. To this end, it is worth exploring the joint optimization of the whole robotic stack from perception to control [64], [65].

### C. Structured neural network

Model-based approaches require low-fidelity data yet heavily rely on assumptions. In contrast, end-to-end learning approaches need few assumptions yet require high-fidelity data. Our structured learning method combines the best of both worlds: It requires low-fidelity data yet relies on minimal assumptions. By injecting structures into the neural network, we decompose a complex problem into smaller and relatively independent components. Note that our decomposition does not break the gradient flow, which keeps HEIGHT end-to-end trainable. We propose a principled way for network architecture design, increasing the transparency of these black-boxes. Our experiments demonstrate that the structured network outperforms both model-based methods and RL-based methods without structures, which empirically proves the effectiveness of structure learning for interactive tasks with multiple heterogeneous participants.

### D. Training method

Deep RL enables the robot to explore the environment and learn meaningful behaviors through trial and error. Without heuristics or demonstrations, the robot has to collect reward signals to improve its policy. Consequently, simulation development with a randomization scheme and a good reward design are indispensable to the performance of our method. Nevertheless, our method is subjective to the inherent limitations of RL, such as low training data efficiency and difficulties in long-horizon tasks. In future work, combining the RL policy with other traditional planners or imitation learning has the potential to alleviate these problems [36].

## VIII. CONCLUSION

In this article, we proposed HEIGHT, a novel structured graph network architecture for autonomous robot navigation in dynamic and constrained environments. Our approach takes advantage of the graphical nature and decomposability of the constrained crowd navigation problem, introducing the following two key novelties. First, we split and process the human and obstacle representations separately. This allows the robot to effectively reason about the different geometries and dynamics of humans and obstacles, improving its ability to navigate complex environments. Second, we propose a heterogeneous st-graph to capture various types of interactions among the robot, humans, and obstacles. The decomposition of the scenario as a heterogeneous st-graph guides the design of the HEIGHT network with attention mechanisms. Attention enables the robot to reason about the relative importance of each pairwise interaction, leading to adaptive and agile robot decision-making during navigation.

Our simulation experiments show that the HEIGHT model outperforms traditional model-based methods and other learning-based methods in terms of collision avoidance and

navigation efficiency. The HEIGHT model also demonstrates improved generalization in environments with varied human and obstacle densities. In real-world environments, HEIGHT is seamlessly transferred from simulation to everyday indoor navigation scenarios without additional training, showcasing its robustness and ability to overcome the sim-to-real gap.

Our work suggests that reasoning about subtle spatio-temporal interactions is an essential step toward smooth human-robot interaction. Furthermore, our work highlights the significance of uncovering the inherent structure of complex problems and injecting these structures into learning frameworks to solve the problems in a principled manner.

#### ACKNOWLEDGEMENTS

We thank Zhe Huang for helpful discussions and code of the A\* planner. We thank Aamir Hasan and Rutav Shah for feedback on paper drafts. We thank members in Human-Centered Autonomy Lab who participated in real-world experiments.

#### REFERENCES

- [1] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
- [2] J. Van den Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1928–1935.
- [3] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, “Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.
- [4] Y. F. Chen, M. Liu, M. Everett, and J. P. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285–292.
- [5] Y. Yang, J. Jiang, J. Zhang, J. Huang, and M. Gao, “St<sup>2</sup>: Spatial-temporal state transformer for crowd-aware autonomous navigation,” *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 912–919, 2023.
- [6] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, “Relational graph learning for crowd navigation,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.
- [7] C. Pérez-D’Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, “Robot navigation in constrained pedestrian environments using reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 1140–1146.
- [8] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, “Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3517–3524.
- [9] S. Liu, P. Chang, Z. Huang, N. Chakraborty, K. Hong, W. Liang, D. L. McPherson, J. Geng, and K. Driggs-Campbell, “Intention aware robot crowd navigation with attention-based interaction graph,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12015–12021.
- [10] M. Hoy, A. S. Matveev, and A. V. Savkin, “Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey,” *Robotica*, vol. 33, no. 3, pp. 463–497, 2015.
- [11] A. V. Savkin and C. 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*, vol. 62, no. 10, pp. 1568–1580, 2014.
- [12] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, “Core challenges of social robot navigation: A survey,” *arXiv preprint arXiv:2103.05668*, 2021.
- [13] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-body collision avoidance,” in *Robotics research*. Springer, 2011, pp. 3–19.
- [14] D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” *Physical review E*, vol. 51, no. 5, p. 4282, 1995.
- [15] L. Huber, J.-J. Slotine, and A. Billard, “Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds,” *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3113–3132, 2022.
- [16] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, and S. S. Srinivasa, “Winding through: Crowd navigation via topological invariance,” *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 121–128, 2023.
- [17] Open Source Robotics Foundation, “ROS Navigation Stack,” <http://wiki.ros.org/navigation>, 2007, accessed: 2024-11-11.
- [18] M. Dobrevski and D. Skočaj, “Dynamic adaptive dynamic window approach,” *IEEE Transactions on Robotics*, vol. 40, pp. 3068–3081, 2024.
- [19] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, “Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.
- [20] P. Trautman and A. Krause, “Unfreezing the robot: Navigation in dense, interacting crowds,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 797–803.
- [21] Z. Liu, W. Na, C. Yao, C. Liu, and Q. Chen, “Relaxing the limitations of the optimal reciprocal collision avoidance algorithm for mobile robots in crowds,” *IEEE Robotics and Automation Letters*, vol. 9, no. 6, pp. 5520–5527, 2024.
- [22] P. Long, W. Liu, and J. Pan, “Deep-learned collision avoidance policy for distributed multiagent navigation,” *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.
- [23] L. Tai, J. Zhang, M. Liu, and W. Burgard, “Socially compliant navigation through raw depth inputs with generative adversarial imitation learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1111–1117.
- [24] Z. Xie, P. Xin, and P. Dames, “Towards safe navigation through crowded dynamic environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 4934–4940.
- [25] A. Pokle, R. Martín-Martín, P. Goebel, V. Chow, H. M. Ewald, J. Yang, Z. Wang, A. Sadeghian, D. Sadigh, S. Savarese, et al., “Deep local trajectory replanning and control for robot navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5815–5822.
- [26] H. Karnan, A. Nair, X. Xiao, G. Warnell, S. Pirk, A. Toshev, J. Hart, J. Biswas, and P. Stone, “Socially compliant navigation dataset (scand): A large-scale dataset of demonstrations for social navigation,” *IEEE Robotics and Automation Letters*, 2022.
- [27] R. Chandra, H. Karnan, N. Mehr, P. Stone, and J. Biswas, “Towards imitation learning in real world unstructured social mini-games in pedestrian crowds,” *arXiv preprint arXiv:2405.16439*, 2024.
- [28] Z. Xie and P. Dames, “Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles,” *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2700–2719, 2023.
- [29] M. Everett, Y. F. Chen, and J. P. How, “Motion planning among dynamic, decision-making agents with deep reinforcement learning,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052–3059.
- [30] Y. Chen, C. Liu, B. E. Shi, and M. Liu, “Robot navigation in crowds by graph convolutional networks with attention learned from human gaze,” *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754–2761, 2020.
- [31] Y.-J. Mun, M. Itkina, S. Liu, and K. Driggs-Campbell, “Occlusion-aware crowd navigation using people as sensors,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12031–12037.
- [32] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, “Densecavoid: Real-time navigation in dense crowds using anticipatory behaviors,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 11345–11352.
- [33] D. Dugas, O. Andersson, R. Siegwart, and J. J. Chung, “Navdreams: Towards camera-only rl navigation among humans,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2504–2511.
- [34] Z. Zheng, C. Cao, and J. Pan, “A hierarchical approach for mobile robot exploration in pedestrian crowd,” *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 175–182, 2022.
- [35] C. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, “Core challenges of social robot navigation: A survey,” *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, 2023.
- [36] A. H. Raj, Z. Hu, H. Karnan, R. Chandra, A. Payandeh, L. Mao, P. Stone, J. Biswas, and X. Xiao, “Rethinking social robot navigation: Leveraging the best of two worlds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

- [37] N. Tsoi, A. Xiang, P. Yu, S. S. Sohn, G. Schwartz, S. Ramesh, M. Hussein, A. W. Gupta, M. Kapadia, and M. Vázquez, “Sean 2.0: Formalizing and generating social situations for robot navigation,” *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.
- [38] X. Puig, E. Undersander, A. Szot, M. D. Cote, T.-Y. Yang, R. Partsey, R. Desai, A. Clegg, M. Hlavac, S. Y. Min, V. Vondruš, T. Gervet, V.-P. Berge, J. M. Turner, O. Maksymets, Z. Kira, M. Kalakrishnan, J. Malik, D. S. Chaplot, U. Jain, D. Batra, A. Rai, and R. Mottaghi, “Habitat 3.0: A co-habitat for humans, avatars, and robots,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [39] L. Kästner, V. Shcherbyna, H. Zeng, T. A. Le, M. H.-K. Schreff, H. Osmaev, N. T. Tran, D. Diaz, J. Golebiowski, H. Soh, and J. Lambrecht, “Arena 3.0: Advancing social navigation in collaborative and highly dynamic environments,” in *Robotics: Science and Systems*, 2024.
- [40] R. Martin-Martin, M. Patel, H. Rezatofighi, A. Shenoi, J. Gwak, E. Frankel, A. Sadeghian, and S. Savarese, “Jrdb: A dataset and benchmark of egocentric robot visual perception of humans in built environments,” *IEEE transactions on pattern analysis and machine intelligence*, 2021.
- [41] N. Hirose, D. Shah, A. Sridhar, and S. Levine, “Sacson: Scalable autonomous control for social navigation,” *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 49–56, 2024.
- [42] H. Yang, C. Yao, C. Liu, and Q. Chen, “Rmrl: Robot navigation in crowd environments with risk map-based deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 8, no. 12, pp. 7930–7937, 2023.
- [43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
- [44] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, “Graph attention networks,” *International Conference on Learning Representations*, 2018.
- [45] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–7.
- [46] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, “Stgt: Modeling spatial-temporal interactions for human trajectory prediction,” in *IEEE International Conference on Computer Vision (ICCV)*, 2019.
- [47] A. Hasan, P. Sriram, and K. Driggs-Campbell, “Meta-path analysis on spatio-temporal graphs for pedestrian trajectory prediction,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 617–624.
- [48] E. Leurent and J. 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.
- [49] S. Liu, P. Chang, H. Chen, N. Chakraborty, and K. Driggs-Campbell, “Learning to navigate intersections with unsupervised driver trait inference,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.
- [50] L. Huajian, D. Wei, M. Shouren, W. Chao, and G. Yongzhuo, “Sample-efficient learning-based dynamic environment navigation with transferring experience from optimization-based planner,” *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 7055–7062, 2024.
- [51] B. Chen, H. Zhu, S. Yao, S. Lu, P. Zhong, Y. Sheng, and J. Wang, “Socially aware object goal navigation with heterogeneous scene representation learning,” *IEEE Robotics and Automation Letters*, vol. 9, no. 8, pp. 6792–6799, 2024.
- [52] D. Jia, A. Hermans, and B. 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)*, 2020, pp. 10270–10277.
- [53] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
- [54] N. Wojke, A. Bewley, and D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric,” in *2017 IEEE international conference on image processing (ICIP)*, 2017, pp. 3645–3649.
- [55] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, “Structural-rnn: Deep learning on spatio-temporal graphs,” in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 5308–5317.
- [56] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, “Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.
- [57] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
- [58] E. Coumans and Y. Bai, “Pybullet, a python module for physics simulation for games, robotics and machine learning,” <http://pybullet.org>, 2016–2019.
- [59] S. Nasiriany, H. Liu, and Y. Zhu, “Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022.
- [60] H. Zhu, J. Yu, A. Gupta, D. Shah, K. Hartikainen, A. Singh, V. Kumar, and S. Levine, “The ingredients of real world robotic reinforcement learning,” in *International Conference on Learning Representations (ICLR)*, 2020.
- [61] P. Chang, S. Liu, T. Ji, N. Chakraborty, K. Hong, and K. R. Driggs-Campbell, “A data-efficient visual-audio representation with intuitive fine-tuning for voice-controlled robots,” in *Conference on Robot Learning (CoRL)*, 2023.
- [62] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak, “Auto-tuned sim-to-real transfer,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, p. 1290–1296.
- [63] V. Lim, H. Huang, L. Y. Chen, J. Wang, J. Ichnowski, D. Seita, M. Laskey, and K. Goldberg, “Real2sim2real: Self-supervised learning of physical single-step dynamic actions for planar robot casting,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8282–8289.
- [64] Y. Hu, J. Yang, L. Chen, K. Li, C. Sima, X. Zhu, S. Chai, S. Du, T. Lin, W. Wang, L. Lu, X. Jia, Q. Liu, J. Dai, Y. Qiao, and H. Li, “Planning-oriented autonomous driving,” in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [65] H. Wang, K. Kedia, J. Ren, R. Abdullah, A. Bhardwaj, A. Chao, K. Y. Chen, N. Chin, P. Dan, X. Fan, G. Gonzalez-Pumariega, A. Komella, M. A. Pace, Y. Sharma, X. Sun, N. Sunkara, and S. Choudhury, “Mosaic: A modular system for assistive and interactive cooking,” 2024.