Relational Graph Learning for Crowd Navigation

Changan Chen*†[1,2], Sha Hu*[2], Payam Nikdel[2], Greg Mori[2], Manolis Savva[2]
[1]UT Austin, [2]Simon Fraser University

***Abstract*— We present a relational graph learning approach for robotic crowd navigation using model-based deep reinforcement learning that plans actions by looking into the future. Our approach reasons about the relations between all agents based on their latent features and uses a Graph Convolutional Network to encode higher-order interactions in each agent's state representation, which is subsequently leveraged for state prediction and value estimation. The ability to predict human motion allows us to perform multi-step lookahead planning, taking into account the temporal evolution of human crowds. We evaluate our approach against a state-of-the-art baseline for crowd navigation and ablations of our model to demonstrate that navigation with our approach is more efficient, results in fewer collisions, and avoids failure cases involving oscillatory and freezing behaviors.**

I. INTRODUCTION

Inferring the underlying relations between components of complex dynamic systems can inform decision making for autonomous agents. One natural system with complex dynamics is crowd navigation (i.e., navigation in the presence of multiple humans). The crowd navigation task is challenging as the agent must predict and plan relative to likely human motions so as to avoid collisions and remain at safe and socially appropriate distances from people. Some prior work predicts human trajectories using hand-crafted social interaction models [1] or by modeling the temporal behavior of humans [2]. Although these methods can estimate human trajectories, they do not use the prediction to inform the navigation policy. Other recent works [3]–[5] use deep reinforcement learning (RL) to learn a socially compliant policy. These policies either do not leverage the human interactions or approximate it with heuristics. They also simplify the human motion prediction problem with unrealistic assumptions such as linear human motion, and typically consider only the current state of humans instead of incorporating predicted human motion to inform the policy.

More broadly, interacting systems have been studied extensively in recent work [6]–[8] and Graph Neural Networks (GNNs) are one of the most powerful tools for modeling objects and their relations (interactions). A variant of GNNs is Graph Convolutional Networks (GCNs) [9] where relations between nodes are defined as an adjacency matrix. Whereas in GCNs the relations between all nodes are given, Wang et al. [10] and Grover et al. [11] propose to learn the relations between objects and use learned attention to compute new features. Inspired by this work on relational reasoning and GNN models, we propose a relational graph learning model for crowd navigation that reasons about relations between agents (robot and humans) and then use a GCN to compute interactions between the agents. With the predicted interaction features of both the robot and humans, our approach jointly plans efficient robot navigation and predicts the motion of present humans. Figure 1 illustrates how interaction reasoning between all agents and explicit prediction can yield a farsighted navigation policy.

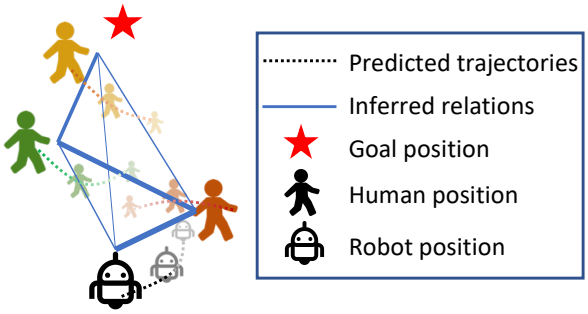


Fig. 1: Illustration of our relational graph learning approach (relation graph in blue). We infer interactions among the robot and humans and predict trajectories (line thickness indicates the strength of interaction; large/medium/small figures show current positions, the predicted position in next step, and the predicted position in two steps). By inferring a strong relation between the robot and the red human as well as the green and red human, and planning conditional on predicted human trajectories, we can find a safe path towards the goal.

Planning and prediction of multi-agent states is fundamental in many problems, including the general case of decision making in an environment with N non-cooperative agents. In this paper, we address this problem with a relational graph learning approach for model-based RL, which predicts future human motions and plans for crowd navigation simultaneously. We show that our model outperforms baselines from prior work and carry out ablation studies to measure the planning budget on navigation performance. We also qualitatively demonstrate that our approach mitigates undesirable robot behaviors in challenging scenarios. The code of our approach is publicly available at `https://github.com/ChanganVR/RelationalGraphLearning`.

II. RELATED WORK

Crowd navigation. Mobile robot navigation in crowds is challenging due to the complex ways in which human intent and social interactions determine human motions. Prior work has used rule-based algorithms to characterize the

*CC and SH contributed equally
†work done as an undergrad at Simon Fraser University

interactions between agents. Helbing et al. [1] proposed to model the interactions as "social forces". RVO [12] and ORCA [13] solve for collision avoidance under reciprocal assumptions. Interacting Gaussian Process (IGP) [14]–[16] model the trajectory of each agent as an individual Gaussian Process and propose an interaction potential term to couple the individual GP for interaction. Rule-based methods rely heavily on hand-crafting the model for navigation. Social-LSTM [2] used an LSTM to predict human trajectories directly from large scale datasets. However, for crowd navigation, forecasting models do not directly yield an action policy. Prediction-based models [17], [18] perform motion prediction and planning sequentially but face the freezing robot problem in complex environments. Recently, Chen et al. [3], [19] propose to use deep RL for crowd navigation by learning a value network encoding the state values.

LM-SARL [5] improved on previous work by learning the robot state representation with attentive pooling over pairwise interaction features. However, this model is limited by partial modeling of crowd interactions, due to significant simplifying assumptions for the underlying state transition without explicitly modeling human motions. Most recently, LeTS-Drive [20] used online belief-tree search to learn a value and policy function for autonomous driving in a crowded space. Although this approach models intentions and interactions between the vehicle and humans, the interaction is coarse-grained, utilizing Convolutional Neural Networks (CNNs) to process stacked frames of the environment, the history and intended path of the vehicle. In contrast, we include pairwise interactions among all agents, which coupled with our graph learning, explicitly captures relations between agents and models higher-order interactions.

Relational reasoning. Relational reasoning [21], [22] aims to capture relationships between entities such as decision-making agents [23], [24], image pixels [10], words[25], or humans and objects [26], [27]. The relations and entities are typically represented as a connected graph [28], [29], and standard tools for graph-based learning such as Graph Neural Nets(GNNs) [9], [30]–[32] are applied. GNNs are a class of neural networks that learn functions on graphs, representing objects as nodes and relations as edges. Knowledge about object dynamics can be encoded in the GNN node update function, and interaction dynamics can be encoded in the edge update function. Most current GNNs operate on interacting systems with known and fixed graph inputs, such as human skeletons or particles connected by springs [8], [9]. However, many interacting systems have unknown relations. For example, humans in sports [33] and crowds [34]. Thus, it is important to infer relations among the entities and learn based on the inferred relations. In crowd navigation, the temporal dynamics of these relations are useful for planning safe and efficient crowd navigation (e.g., understanding the joint motion of a pair of friends as they cross the robot's path while walking close to each other, in contrast to the motion of a pair of strangers walking in opposite directions and passing each other). Inspired by Kipf et al. [7], who estimate the graph connectivity map from trajectories using an auto-encoder architecture, our model dynamically infers the crowd-robot relation graph at each time step and learns the state representation for each agent based on this graph. Recent work [35] proposed to use graph convolutional networks in navigation and used human gaze data to train the network. The use of human gaze data helps the network to learn more human-like attention but it is also limited to the robot's attention. In our proposed method, the GCN not only captures the attention of the robot but also inter-human attention, which is subsequently leveraged by a human motion prediction model.

MPC, MCTS and model-based RL. Model predictive control (MPC) is a family of control algorithms that leverage models of the system to predict state changes and plan control accordingly. Traditional MPC [36], [37] usually assumes access to a known environment model, which is frequently unrealistic. Monte-Carlo Tree Search (MCTS) has been used for decision-time planning by estimating action values based on many simulated trajectories in complex search problems such as the game of Go [38]. More recent model-based RL methods first acquire a predictive model of the world, and then use that model to make decisions. Finn et al. [39] learned a state transition model by predicting future frames to achieve goal-directed robotic manipulation. VPN [40] and Predictron [41] learned to predict future abstract states that are informative for planning in Atari Games. In contrast, our model's predictive relational graph takes a set of raw human states (e.g., positions, velocities) as input and predicts multiple interacting human trajectories. To our knowledge, we are the first to integrate relational learning with a model-based RL algorithm for crowd navigation.

III. Approach

We first describe how we formulate the crowd navigation problem with deep RL, then introduce our relational graph learning model for modeling interactions in the crowd. In addition, we show how this model can be augmented by a planning algorithm (simplified MCTS) at both training and test time. Figure 2 shows an overview of our approach.

A. Deep Reinforcement Learning for Crowd Navigation

In this work, we address the crowd navigation task where the robot navigates through a crowd of N humans to a goal position as efficiently and safely as possible. This task is formulated as a sequential decision making problem in recent works [3]–[5]. Each agent (either human or the robot) observes others' observable state, including position $\mathbf{p} = [p_x, p_y]$, velocity $\mathbf{v} = [v_x, v_y]$ and radius r (an abstract measure of size). Each agent also maintains an internal state, such as a goal position $\mathbf{p_g}$ and preferred speed v_{pref}. We assume actions are instantaneous and the agent always arrives at the target position in the next time step. We use s_0^t and s_i^t to denote the robot state and the observed state of human i at time t, respectively. The robot input state is defined as $\mathbf{S}^t = \{s_0^t, s_1^t, ..., s_N^t\}$. The optimal policy mapping state $\mathbf{S}^t$ to

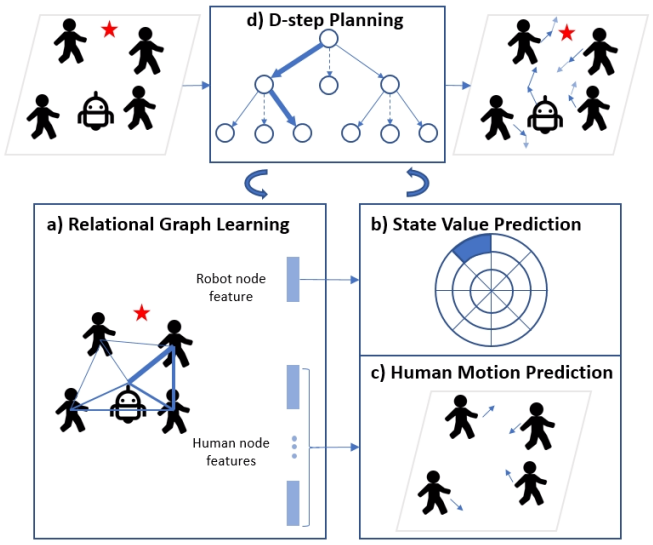


Fig. 2: Given the crowd state as input, our approach uses d-step planning to discover the best action sequence for safe and efficient crowd navigation: a) a relational graph learning model reasons about relations between agents and encodes local interactions. b) a value network predicts the value of the robot state representation. c) a motion network predicts future human states. d) with the learned value estimation and motion prediction models, our approach rolls out d steps into the future and searches for the best action sequence.

action $\mathbf{a}^t$ at time t, $\pi^* : \mathbf{S}^t \mapsto \mathbf{a}^t$, is to maximize the expected return: Which is V*()

$$\pi^*(\mathbf{S}^t) = \underset{\mathbf{a}^t}{\operatorname{argmax}}\ R(\mathbf{S}^t, \mathbf{a}^t) + \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{S}^{t+\Delta t}} P(\mathbf{S}^t, \mathbf{a}^t, \mathbf{S}^{t+\Delta t}) V^*(\mathbf{S}^{t+\Delta t}) d\mathbf{S}^{t+\Delta t} \quad (1)$$

$$V^*(\mathbf{S}^t) = \sum_{k=t}^{T} \gamma^{k \cdot v_{pref}} R^k(\mathbf{S}^k, \pi^*(\mathbf{S}^k)),$$

where $R(\mathbf{S}^t, \mathbf{a}^t)$ is the reward received at time t, $\gamma \in (0,1)$ is the discount factor, V^* is the optimal value function, $P(\mathbf{S}^t, \mathbf{a}^t, \mathbf{S}^{t+\Delta t})$ is the transition probability from time t to time $t+\Delta t$. And the preferred velocity v_{pref} is used as a normalization term in the discount factor for numerical reasons [3].

In the above equation, $P(\mathbf{S}^t, \mathbf{a}^t, \mathbf{S}^{t+\Delta t})$ represents the system dynamics and the knowledge of the world (e.g., how state changes depend on agent actions) and is usually unknown to the agent. Some recent work [3], [5] assumes the state transition function to be known during training time and models it with simple linear models at test time. This assumption strongly reduces the complexity of the problem since the agent can basically solve the navigation problem by searching the next state space. In this work, we remove the assumption of knowing the state transition, and instead use a model-based approach to learn to predict human motions.

We follow the formulation of the reward function defined in Chen et al. [5], which awards accomplishing the task while penalizing collisions or uncomfortable distances.

This problem statement can be applied to a set of more general tasks where there are N non-cooperative agents and the decision-maker only receives their observable states but does not know about their intents or hidden policies.

so this paper's model is 'given' non inferred info

B. Model Predictive Relational Graph Learning

The interactions (i.e., spatial relations) between humans are important for robot navigation and for predicting future human motion. Previous work does not learn such interaction features for the robot and humans simultaneously. Here, we model the crowd as a graph, reason about the relations between all agents and use a GCN to compute the robot and human state representations. Using the graph model as a building block, we further construct two other modules: a *value estimation module* $f_V()$ which estimates the value of the current state and a *state prediction module* $f_P()$ which predicts the state at the next time step.

Relational graph learning. The key challenge in crowd navigation is to learn a good representation of the crowd encoding interactions among all agents. Chen et al. [5] show that attentive crowd aggregation improves both interpretation and performance by modeling one-way human-robot interactions. This motivates us to model the crowd and the robot as a directed graph $G^t = (V^t, E^t)$ where $|V| = N+1$. The edge $e_{i,j} \in E$ indicates how much attention agent i pays to agent j or the importance of agent j to agent i. This pairwise relation is not known a priori, so it is inferred with a pairwise similarity function (relation inference). After the relations between all agents are inferred, a graph neural network propagates information from node to node and computes the state representations for all agents (interaction modeling) as shown in Figure 2 a).

how is it inferred?

LM-SARL [5] can be viewed as an approximation of our Relational Graph Learning (RGL) formulation, in that it learns robot interaction features with respect to all humans and uses attentive pooling to aggregate the interaction features. Our RGL formulation not only learns the attention of the robot to humans but also from humans to humans. Apart from learning the robot state representation, RGL also learns the state representation for other agents simultaneously and propagates these features to the robot node by message passing. In this way, RGL also models higher-order interactions. For example, $l = 2$ fully models human-human interaction rather than using a local map (as in LM-SARL) to approximate the local interactions of humans. This approach is also favorable compared to LeTS-Drive [20] in that LeTS-Drive doesn't reason about pairwise state interactions or model human-human interactions.

1) Relation Inference: the initial values of vertices V are the state values for the agents: $S^t = \{s_i^t\}_{i=0...N+1}$. Since robot and human states have different dimensions, we use two multilayer perceptrons (MLPs) $f_r()$ and $f_h()$ to embed them into a latent space, resulting in a matrix X, where the first row is the latent state of the robot and the remaining rows are the latent states of humans. Given the feature matrix X, a relation matrix is computed using a pairwise similarity function. Following Wang et al. [10], we use an

what to the MLPs do? autoencoder ?

so a matrix of latent state of robot on row 1, the rest of the rows is the latent state for humans. this is X
X and a pairwise similarity function form a RELATION_MATRIX

embedded Gaussian as the similarity function. The pairwise form is given by $f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)}$ and the matrix form is given by $A = \text{softmax}(XW_aX^T)$ where $x_i = X[i,:]$, $\theta(x_i) = W_\theta x_i$, $\phi(x_i) = W_\phi x_i$ and $W_a = W_\theta W_\phi^T$. A learned relation is illustrated in Figure 1 where the thickness of the line indicates the strength of pairwise interactions.

2) Interaction Modeling: with the feature matrix X and relation matrix A, we use a GCN to compute the pairwise interaction features. The message passing rule is defined by $H^{(l+1)} = \sigma(AH^{(l)}W^{(l)}) + H^{(l)}$ where $W^{(l)}$ is a layer-specific trainable weight matrix, $H^{(l)}$ is the node-level feature of layer l, and σ is an activation function. The feature of node i at level $l+1$ aggregates its neighbor node features at level l weighted by the relations stored in matrix A.

Let $H^{(0)} = X$ and after L layer propagations, we have state representation matrix $Z^t = H^{(L)}$ for S^t, and $Z^t[i,:]$ is the state representation for agent i encoding its local interactions.

Relational value estimation. Our value estimation module f_V consists of two models: a relational graph model to infer the robot state representation $Z^t[0,:]$ and a subsequent value network to predict the value of the state $v = f_v(Z^t[0,:])$. The value network f_v is an MLP and predicts values over a discretized action space by looking ahead as shown in Figure 2 b).

State prediction. We assume the action of the robot can be perfectly executed and the next robot state $\hat{s}_0^{t+1}$ can be computed directly from the current state s_0^t and action a^t. Our state prediction module models interactions between human states to predict future human states, as shown in Figure 2 c). In the action selection phase, previous works [3], [5] rely on either querying the ground truth or off-the-shelf algorithms to approximate the next human states, which does not benefit from end-to-end learning. Our state prediction module $f_P()$ consists of two models: first a relational graph model predicts relations between all agents and their l^{th} layer interaction features, then a successor motion prediction model uses the human state representations to predict their next state $\hat{s}_i^{t+1} = f_m(Z^t[i,:])$, where $\hat{s}_i^{t+1}$ is the predicted state for human i with $1 \le i \le n$. The motion prediction network f_m is modelled by an MLP. In addition, to simplify the prediction task, we use a hand-crafted function to estimate the reward based on the prediction of human states, denoted by $\hat{R}(S^t, a^t, \hat{S}^{t+1})$. By jointly learning state values and predicting human motions, the policy learning benefits from encoding human motions and better addresses the freezing robot problem in [17], [18].

C. Relational Forward Planning and Learning

In this section, we demonstrate how relational graph learning can be augmented with a simplified MCTS approach to provide bootstrapped targets in the learning phase and a far-sighted policy at decision time. Imperfect learned value functions can lead to suboptimal actions due to local minima. To leverage the prediction ability of our model, we further simulate d-steps into the future to provide a better estimate of the state values. Furthermore, by controlling the depth d of this simulation, we can trade off computation for better performance. Our approach is visualized in the tree search diagram shown in Figure 2 d).

We follow the d-step planning method proposed in Oh et al. [40], performing rollouts using the crowd state prediction and value estimation up to d steps in the future, and select the action with the maximum d-step predicted return, defined as follows:

$$V^d(S^t) = \begin{cases} f_V(S^t), & \text{if } d = 1 \\ \frac{1}{d} V^1(S^t) + \frac{d-1}{d} \max_{a^t} (\\ \quad \hat{R}(S^t, a^t, \hat{S}^{t+1}) + \gamma V^{d-1}(\hat{S}^{t+1})) & \text{otherwise} \end{cases} \tag{2}$$

where $\hat{S}^{t+1} = f_P(S^t, a^t)$.

With d-step planning, computation increases exponentially with search depth and width. Due to our large action space, we use action space clipping to reduce the computational cost. Intuitively, the value function estimates the quality of entering a state. Using this estimate, we recursively search the top-w next states with one-step lookahead. Compared to only considering the top action, d-step planning provides a better state value estimate when the agent encounters unseen states. Figure 2 d) shows one toy example of tree search with depth $d = 2$ and clipping width $w = 2$.

D. Joint Value Estimation and State Prediction Learning

Pseudocode for the joint state prediction and value estimation learning scheme is in Algorithm 1. Similar to the training scheme in Chen et al. [5], we first use imitation learning with collected experience from a demonstrator ORCA [13] policy to initialize the model, and then use RL to refine the policy. Imitation learning is important for policy initialization due to the sparse rewards in navigation, without which RL can't converge. We train f_V using RL and f_P with supervised learning. To stabilize training, we also use separate the graph models for these two functions. By integrating the planning into the learning phase, our learning algorithm is able to uses bootstrapped state value estimations as the targets to learn a more accurate value function than Chen et al. [5].

IV. Experimental Results

Implementation details. The hidden units of $f_r(\cdot)$, $f_h(\cdot)$, $f_v(\cdot)$, $f_m(\cdot)$ have dimensions (64, 32), (64, 32), (150, 100, 100), (64, 32) and the output dimension of $W_a^{(l)}$ is 32 for all $l = 1...L$. For fair comparison with the baseline method, we use the value network for the baseline as reported in Chen et al. [5]. All the parameters are trained using Adam [42], and the learning rate is 0.001. The discount factor γ is set to be 0.9. The exploration rate of the ε-greedy policy decays from 0.5 to 0.1 linearly in the first $5k$ episodes. We assume holonomic kinematics for the robot, i.e. it can move in any direction. The action space consists of 80 discrete actions: 5 speeds exponentially spaced between (0, v_{pref}] and 16 headings evenly spaced between [0, 2π).

Simulation setup. We use the CrowdNav[1] simulation environment. In this simulation, humans are controlled by

[1] https://github.com/vita-epfl/CrowdNav

Method	Success ↑	Collision ↓	Extra Time ↓	Avg. Return ↑	Max Diff. ↓
ORCA [12]	0.43 ± 0.00	0.57 ± 0.00	2.93 ± 0.00	0.081 ± 0.000	0.604 ± 0.000
LM-SARL-Linear [5]	0.90 ± 0.02	0.09 ± 0.02	3.15 ± 0.24	0.506 ± 0.018	0.179 ± 0.018
RGL-Linear (Ours)	0.92 ± 0.02	0.04 ± 0.03	2.35 ± 0.13	0.541 ± 0.014	0.144 ± 0.014
MP-RGL-Onestep (Ours)	0.93 ± 0.02	0.03 ± 0.02	2.15 ± 0.13	0.551 ± 0.025	0.134 ± 0.025
MP-RGL-Multistep (Ours)	$\mathbf{0.96} \pm 0.02$	$\mathbf{0.02} \pm 0.01$	$\mathbf{1.86} \pm 0.07$	$\mathbf{0.591} \pm 0.009$	$\mathbf{0.094} \pm 0.009$

TABLE I: Quantitative results in circle crossing with five humans. The metrics are defined as follows: "Success": the rate of robot reaching its goal without a collision; "Collision": the rate of robot colliding with humans; "Extra Time": extra navigation time to reach goal in seconds; "Avg. Return": returns averaged over steps across episodes and all test cases. "Max Diff.": the difference between actual average returns and the upper bound of average returns. $\pm$ indicates standard deviation measured using five independently seeded training runs.

Algorithm 1 Learning for f_P and f_V

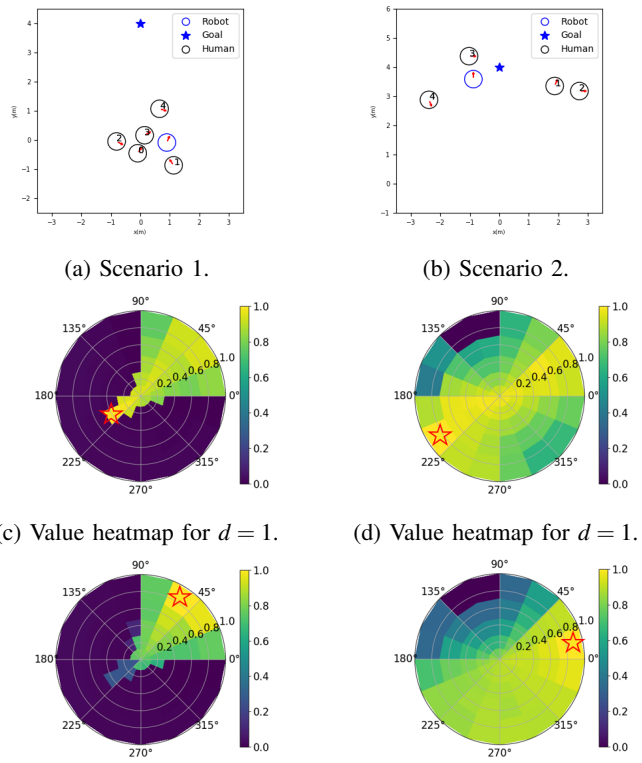
Initialize f_P, f_V with demonstration D
Initialize target value network $\hat{f}_V \leftarrow f_V$
Initialize experience replay memory E $\leftarrow$ D
for episode = 1, M **do**
  Initialize random sequence S^0
  repeat
    $a_t \leftarrow \text{argmax}_{a_t \in A} \hat{R}(S^t, a^t, \hat{S}^{t+1}) + \gamma^{\Delta t \cdot v_{pref}} V^d(\hat{S}^{t+1})$ where $\hat{S}^{t+1} = f_P(S^t, a^t)$
    Execute a_t and obtain r_t and $S^{t+\Delta t}$
    Store tuple $(S^t, a^t, r^t, S^{t+\Delta t})$ in E
    Sample random minibatch tuples from E
    Set target for value network: $y_i = r_i + \gamma^{\Delta t \cdot v_{pref}} \hat{V}^d(S_{i+1})$
    Update f_V by minimizing $L_1 = ||f_V(S_i) - y_i||$
    Set target for prediction network: S_{i+1}
    Update f_P by minimizing $L_2 = ||f_P(S_i, a_i) - S_{i+1}||$
  until terminal state s_t or $t \geq t_{max}$
  Update target value network $\hat{f}_V \leftarrow f_V$
end for
return f_P, f_V

ORCA [13], the parameters of which are sampled from a Gaussian distribution to introduce behavioral diversity. We use circle crossing scenarios for our experiments. Circle crossing scenarios have $N = 5$ humans randomly positioned on a circle of radius $4m$ with random perturbation added to their x,y coordinates. The maximum speed for the agent, v_{pref} is 1 m/s, and the goal is fixed so that it can be reached in a minimum time of 8 seconds. To fully evaluate the effectiveness of the proposed model, we look into the simulation setting where the robot is invisible to humans. As a result, the simulated humans react only to humans but not to the robot. This setting is a clean testbed for validating the model's ability to reason about human-robot and human-human interaction without affecting human behaviors. All models are evaluated with 500 random test cases.

Quantitative Evaluation. As expected, the ORCA method fails badly in the invisible setting due to the violation of the reciprocal assumption. The state-of-the-art method LM-SARL [5] assumes that the next ground truth state is given during training. To demonstrate the effectiveness of our relational graph learning model, we use a linear motion model (agents keep velocities as in the last state) in LM-SARL as well as serving as our state prediction model so that the comparison is purely in robot state representation learning. These two models are indicated by LM-SARL-Linear and RGL-Linear, respectively. We refer to our full model as MP-RGL-Multistep and the model with one-step lookahead as MP-RGL-Onestep. For all RGL-based models, we use a two-layer GCN. For MP-RGL-Multistep, we let $d = 2$ and $w = 2$. We do not compare with LeTS-Drive [20] as it focuses on components other than interaction modeling and assumes different inputs. Table I reports the rates of success, collision, the average extra navigation time (defined as extra navigation time beyond the minimum possible 8 seconds) as well as average return, which is the cumulative discounted rewards averaged over all steps in the 500 test cases. To provide a clearer interpretation of the average return metric, we add one more metric, that is the difference between the average return and its upper bound. The upper bound only exists in imaginary cases where there are no other humans and the agent can head straight to the goal all the time.

As expected, the performance of LM-SARL-Linear drops compared to the one reported in [5] after replacing the unrealistic assumption of access to the next ground truth state with a linear motion model. The first comparison between LM-SARL-Linear and RGL-Linear reflects the improvement brought by modeling the crowd interaction with graph neural networks. Then the comparison between RGL-Linear and MP-RGL-Onestep shows that using a state prediction model leads to a better estimate of human motions and thus yields a better navigation policy. Finally, the best model in the table is our MP-RGL-Multistep. Comparing it to the one-step version, we see that d-step planning improves the success rate, reduces extra time to reach the goal and increases the average return. Even the best-performing model is not collision-free. The robot is invisible to humans and the policy does not have access to the next ground truth state, making some collisions unavoidable (e.g., when the robot is surrounded by humans converging towards it).

Effect of planning budget. In our d-step planning, both the tree depth d and action space clipping width w influence the computation time and planning performance. With larger d, our approach is able to look further into the future and plan further ahead. With larger w, we consider more actions at each depth and can predict a wider range of possible outcomes, potentially reaching a better path. We study how the planning budget in d-step planning influences performance. We tested the MP-RGL-Onestep model in Table I with various test-time planning budgets. By simply setting $d = 2, w = 2$, the extra time decreases from 2.15 to 2.06 and the average return improves from 0.551 to 0.572. With larger



(a) Scenario 1. (b) Scenario 2.

(c) Value heatmap for $d = 1$. (d) Value heatmap for $d = 1$.

(e) Value heatmap for $d = 2$. (f) Value heatmap for $d = 2$.

Fig. 3: Value estimates for two scenarios using different planning depths d. (a), (c), (e) for Scenario 1 and (b), (d), (f) for Scenario 2. The top row shows the top-down view of the crowd, and the two heatmaps below visualize the estimated values over the action space of the robot. The red star in the value map indicates the action with the highest value.

d, w, the performance is also further improved. From these two experiments, we conclude that our d-step planning leads to better performance both at learning and decision-making.

Investigation of computation time. The GNN-based state feature extractor is computationally efficient compared to attention-based crowd navigation work [5] due to sequential pairwise interaction feature extraction. The GNN computation amounts to a matrix multiplication, with negligible change when the number of agents N is relatively small.

Qualitative evaluation. We further investigate how our full model handles two challenging scenarios. The first one is when shortsighted behaviors lead to collisions, and the second one shows the robot freezing when it is close to both the goal and potential danger.

The scenario shown in Figure 3a is a typical example of the first scenario. The one-step lookahead policy assigns almost equally high values to actions in the direction of 30° and 210°, with 210° being the highest. This is reasonable for a shortsighted policy since no human occupies either of these two directions. However, taking action around 210° will result in a collision with human #0. By looking two steps ahead, the policy anticipates an unavoidable collision in two steps if it moves in the direction 210° now. The advantage of the multi-step lookahead policy also validates the effectiveness of encoding higher-order human interactions. By observing the strong relation between human #0 and human #3, who are moving towards the same direction and are very close to each other, the robot predicts human #3 will move to a direction of 15° and predicts negative rewards for taking an action in the direction of 210°. Thus, the two-step lookahead policy assigns low values to the direction of 210° and avoids collisions by taking action in the direction of 30°.

The scenario shown in Figure 3b is a typical example of the second challenging scenario, where human #3 is standing near the robot's goal. The one-step lookahead policy assigns the highest value to the action in the direction of 225° and a lower value to the action in the direction of 45°. This is because taking an action towards 45° will result in a discomfort penalty from stepping into the comfort zone of human #3. The two-step lookahead policy can predict the big positive reward after taking an action in the direction of 45°, and the positive reward two steps later can compensate the discomfort penalty in one step. Thus, the two-step lookahead policy assigns the highest value to the action towards 45° and makes a non-myopic decision.

Fig. 4: Pioneer robot used for real-world demonstrations.

V. Real-world Demonstrations

We deploy our trained policy on a Pioneer robotic platform equipped with an Intel RealSense ZR300 camera and a Hokuyo 2D LiDAR (Figure 4). Test episodes are in the video. We first apply YOLO [43] on depth data to obtain 2D human positions. We then utilize an extended Kalman filter to track the human and compute human velocities. This demonstration shows the potential of our method to model complex relations among interacting people in real scenes. Video: `https://youtu.be/U3quW30Eu3A`.

VI. Conclusion

In this paper, we address crowd navigation with a relational graph learning approach. By formulating human-robot and human-human interactions as a graph and using GCNs to compute interaction features, our model can estimate state values as well as predict human motion. Augmented by d-step planning, our model explicitly plans into the future under a specific search budget. We show our model outperforms baseline methods by a large margin and can handle challenging navigation scenarios. The relational graph learning approach we proposed can be extended in several ways. For example, we do not model the temporal dynamics of past human trajectories, which can help infer the intent of individuals and group structure among humans.

Acknowledgements This research is partially supported by an NSERC Discovery Grant.

REFERENCES

[1] D. Helbing and P. Molnr, "Social force model for pedestrian dynamics," in *Phys. Rev. E*, vol. 51, no. 5, pp. 4282–4286.

[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 961–971.

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

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

[5] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *ICRA*, 2018.

[6] Z. Deng, A. Vahdat, H. Hu, and G. Mori, "Structure inference machines: Recurrent neural networks for analyzing relations in group activity recognition," in *CVPR 2016*.

[7] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *ICML*, 2018.

[8] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," in *NIPS*, 2016.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[10] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *CVPR*, 2018.

[11] A. Grover, M. Al-Shedivat, J. K. Gupta, Y. Burda, and H. Edwards, "Learning policy representations in multiagent systems," in *ICML*, 2018.

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

[13] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Springer Berlin Heidelberg, pp. 3–19.

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

[15] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *2013 IEEE International Conference on Robotics and Automation*, pp. 2153–2160.

[16] P. Trautman, "Sparse interacting gaussian processes: Efficiency and optimality theorems of autonomous crowd navigation," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 327–334.

[17] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," in *Auton Robot*, vol. 35, no. 1, pp. 51–76.

[18] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," in *The International Journal of Robotics Research*, vol. 24, no. 1, pp. 31–48.

[19] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IROS*, 2017.

[20] P. Cai, Y. Luo, A. Saxena, D. Hsu, and W. S. Lee, "LeTS-drive: Driving in a crowd by learning from tree search," in *RSS*, 2019.

[21] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, C. Gulcehre, F. Song, A. Ballard, J. Gilmer, G. Dahl, A. Vaswani, K. Allen, C. Nash, V. Langston, C. Dyer, N. Heess, D. Wierstra, P. Kohli, M. Botvinick, O. Vinyals, Y. Li, and R. Pascanu, "Relational inductive biases, deep learning, and graph networks," in *arXiv*, 2018.

[22] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[23] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart, M. Shanahan, V. Langston, R. Pascanu, M. Botvinick, O. Vinyals, and P. Battaglia, "Relational deep reinforcement learning," in *arXiv*, 2018.

[24] A. Agarwal, S. Kumar, and K. Sycara, "Learning transferable cooperative behavior in multi-agent teams," in *ICML*, 2019.

[25] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *ICLR*, 2017.

[26] Y.-W. Chao, Z. Wang, Y. He, J. Wang, and J. Deng, "HICO: A benchmark for recognizing human-object interactions in images," in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, pp. 1017–1025.

[27] G. Gkioxari, R. Girshick, P. Dollr, and K. He, "Detecting and recognizing human-object interactions," in *CVPR*, 2018.

[28] D. Xu, Y. Zhu, C. B. Choy, and L. Fei-Fei, "Scene graph generation by iterative message passing," in *CVPR*, 2017.

[29] P. Velikovi, G. Cucurull, A. Casanova, A. Romero, P. Li, and Y. Bengio, "Graph attention networks," in *ICLR*, 2018.

[30] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," in *IEEE Data Engineering Bulletin*, 2017.

[31] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "HOW POWERFUL ARE GRAPH NEURAL NETWORKS?" in *ICLR*, 2018.

[32] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *NIPS*, 2017.

[33] M. S. Ibrahim and G. Mori, "Hierarchical relational networks for group activity recognition and retrieval," in *Computer Vision ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Springer International Publishing, vol. 11207, pp. 742–758.

[34] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *Proceedings of the International Conference on Robotics and Automation (ICRA) 2018 (Best Paper Award in Cognitive Robotics)*, May 2018.

[35] Y. Chen, C. Liu, M. Liu, and B. E. Shi, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," in *ICRA*, 2020.

[36] P. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*, pp. 137–142.

[37] D. Q. Mayne, M. M. Seron, and S. V. Rakovi, "Robust model predictive control of constrained linear systems with bounded disturbances," vol. 41, no. 2, pp. 219–224.

[38] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," in *Nature*, vol. 529, no. 7587, 2016, pp. 484–489.

[39] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *ICRA*, 2017.

[40] J. Oh, S. Singh, and H. Lee, "Value prediction network," in *NIPS*, 2017.

[41] D. Silver, H. van Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, and T. Degris, "The predictron: End-to-end learning and planning," in *ICML*, 2017.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

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