Robot Navigation in Crowds by Graph Convolutional Networks with Attention Learned from Human Gaze

Yuying Chen*, Congcong Liu*, Ming Liu, Bertram E. Shi

Abstract-Safe and efficient crowd navigation for mobile robot is a crucial yet challenging task. Previous work has shown the power of deep reinforcement learning frameworks to train efficient policies. However, their performance deteriorates when the crowd size grows. We suggest that this can be addressed by enabling the network to identify and pay attention to the humans in the crowd that are most critical to navigation. We propose a novel network utilizing a graph representation to learn the policy. We first train a graph convolutional network based on human gaze data that accurately predicts human attention to different agents in the crowd as they perform a navigation task based on a top down view of the environment. We incorporate the learned attention into a graph-based reinforcement learning architecture. The proposed attention mechanism enables the assignment of meaningful weightings to the neighbors of the robot, and has the additional benefit of interpretability. Experiments on real-world dense pedestrian datasets with various crowd sizes demonstrate that our model outperforms state-of-art methods, increasing task completion rate by 18.4% and decreasing navigation time by 16.4%.

Index Terms—Autonomous Vehicle Navigation, Deep Learning in Robotics and Automation, Social Human-Robot Interaction

I. INTRODUCTION

W ITH the rapid development of artificial intelligence technologies, mobile robot navigation has many vital applications in crowded pedestrian environments such as hospitals, shopping malls, and canteens. In these scenarios with dense crowds, navigating robots safely and efficiently is a crucial, yet still challenging, problem [1].

Traditional approaches often treat pedestrians as simple dynamic obstacles and focus only on the next step [2], [3]. Since these approaches do not model human behavior, they result in robot behavior that can seem unnatural and shortsighted. To achieve better long-term navigation, many research efforts have included reasoning about human intention and prediction of human trajectories before planning [4], [5].

* These two authors contributed equally

All the authors are with the Hong Kong University of Science and Technology. {ychenco, cliubh, eelium, eebert}@ust.hk Digital Object Identifier (DOI): see top of this page.



Fig. 1: We represent the interactions among the humans and the robot as a graph whose node state and edge strength both vary over time. We aggregate crowd features using a graph convolutional network whose adjacency matrix is modulated by attention.

However, doing prediction and planning separately may cause the *freezing robot problem* when crowd density grows, because the planner believes every forward path will cause collision [6]. To address this problem, a key solution is to consider the impact of the motion of the robot on the crowds.

Current solutions can be divided into two categories: *model-based* and *learning-based*. Model-based methods mainly extend existing multi-agent collision avoidance solutions with explicit models of social interactions [7], [8]. However, the model parameters need to be tuned for different application scenarios. More recent research has used deep reinforcement learning (RL) successfully to learn efficient policies that model the cooperation and interactions implicitly [9]–[11].

Deep RL methods proposed for crowd navigation have been model-based. They use reinforcement learning to learn a deep neural network that estimates the value function of a given robot-crowd configuration. This value function is used in conjunction with a state transition function to perform action selection. Previously proposed approaches differ primarily in the structure of the networks used to encode the robot-crowd state and to estimate the corresponding value. There are several limitations of current models, which cause their performance to degrade when the crowd density increases [9], [10]. First, existing models have considered only pairwise interactions between the robot and each human in the crowd. By estimating these pairwise interactions independently, these approaches do not completely capture the global and dynamic nature of crowd interactions. Second, information about the crowd is obtained by combining information from these pairwise interactions by pooling [9], [12], [13], a maximum operation [11] or an LSTM that combines information about the humans in the

Manuscript received: September, 10, 2019; Revised November, 15, 2019; Accepted January, 14, 2020.

This paper was recommended for publication by Editor Dan Popa upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by the Hong Kong Research Grants Council under grant number 16211015 and was supported by the National Natural Science Foundation of China, under grant No. U1713211, the Research Grant Council of Hong Kong SAR Government, China, under Project No. 11210017, No. 21202816, and the Shenzhen Science, Technology and Innovation Commission (SZSTI) under grant JCYJ20160428154842603, awarded to Prof. Ming Liu.

crowd sequentially according to their proximity to the robot [10]. This does not completely capture important geometric information about the configuration of crowds and the robot.

This work addresses these shortcomings in two ways. First, we use a graph structure to represent the crowd state. Second, we use gaze data from humans performing a navigation task to learn a network that assigns different weights to different agents in the crowd according to their importance as measured by attention. Previous research has shown that deep networks trained for many visuomotor tasks such as driving [14], [15] and video games [16] can benefit from the guidance of human attention as measured by gaze.

As depicted in Fig. 1, for robot navigation in dynamic crowds, it is natural to use a graph structure, which captures the relations (edges) between agents (nodes) to represent the crowd state. Recently, Graph Convolutional Networks (GCNs) with arbitrarily structured graphs have been applied successfully in various areas, such as social networks [17] and citation networks [18]. A GCN takes a feature matrix that represents the attributes of each node as the input, and efficiently aggregates features from neighborhoods defined by an adjacency matrix. One advantage of using a GCN to encode the state is that the interactions between nodes can be modulated easily by changing the adjacency matrix.

There are three primary contributions of our work. First, we demonstrate that it is possible to train an attention network that accurately predicts human attention in crowd navigation scenarios. Second, we propose a novel graph-represented reinforcement learning method for the crowd navigation task and demonstrate the architecture has significant benefits in scalability and extensibility. Third, we show that incorporating adjacency weights based on the human attention model into the method results in performance that exceeds the state-ofthe-art on five real world pedestrian trajectory datasets.

II. RELATED WORK

A. Robot navigation in crowds

Previous researchers have proposed many methods to solve the navigation problem. The Social Force Model [19] is one of the representative methods that has been successfully applied and extended in different environments [7], [20], [21]. Reciprocal velocity obstacles (RVO), which consider communications with other agents, were proposed in multi-agent navigation scenarios [2]. More recently, ORCA enables multiple robots to avoid collisions when navigating in a cluttered workspace [3]. The main limitations of these model-based methods are that they require tedious parameter selection and that they may lead to unnatural robot behaviors, since they do not fully capture real human behavior.

Alternatively, imitation learning aims to learn optimal policies directly from human demonstration. In the context of robot navigation, previous work has used imitation learning to obtain policies supervisedly from raw 2D laser data [22] or depth inputs [23]. Inverse reinforcement learning has also been applied to model human cooperative navigation behavior through maximum entropy [24], [25].

Deep reinforcement learning algorithms learn policies while the robot interacts with the environment through trial-anderror. For robot navigation, recent work has used reinforcement learning methods to learn policies from raw sensor inputs [26] or agent-level representations of the environment [9], [10]. Learning from raw sensor representation has the benefit that static and dynamic obstacles can be considered together through a single neural network. However, an agent-level representation can provide a richer high-level representation of pedestrian intent, which is difficult to extract from raw sensor information. One challenge is the varying crowd size. Everett et al. [10] converted the state of a variable-sized crowd to a fixed-length vector using an LSTM module that processed each pedestrian's state in descending order of their distance from the robot. However, assigning importance according to distance is not always reasonable. For example, a pedestrian closely following a robot may be less important than a pedestrian farther in front of it. Chen et al. adopted a self-attention module to assign different relative importances to different parts of the crowd [9]. Here, we infer the relative importance of agents in the crowd to the robot by learning attention weights from gaze data collected from humans performing a crowd navigation task based on a top down view of the environment.

B. Graph representation learning

Graph Convolutional Networks (GCNs) have attracted much attention for graph representation learning since they generalize the convolution operation to graph-structured data efficiently. GCNs have achieved remarkable successes in various research areas such as citation networks [18], social networks [17] and material property prediction [27]. In the training of a GCN, a static binary adjacency matrix is commonly used. However, the entries in the adjacency matrix can also be continuous real valued functions of time, which enables adaptive and dynamic aggregation of neighbors' information.

A Graph Attention Network (GAT) is a GCN variant that adopts a self-attention mechanism allowing for the assignment of different importances to different nodes [28]. Similarly, the Edge Attention based multi-relational GCN replaces the adjacency matrix with a learned real-valued attention matrix, and has been used to optimize chemical property prediction [29]. Here, we modulate the interactions in the crowds using attention weights learned from human gaze data to determine the adjacency matrix.

C. Human attention in visuomotor learning

Human attention has been proved to be beneficial in learning effective policies for many visuomotor tasks, such as autonomous driving [14], [15] and Atari games [16]. Liu *et al.* [14] trained a conditional GAN to predict human attention and incorporated it into an end-to-end autonomous driving network. They added the predicted gaze map as an extra input to the network. Similarly, Zhang *et al.* [16] proposed an attention guided imitation learning network, which also treats the gaze map as additional image input in the task of eight Atari Games. Chen *et al.* proposed a novel form of dropout modulated by human gaze maps, and applied it to an imitation



Fig. 2: The network structure for graph based V-learning with learned human attention. The attention network generates attention weights for each node. These weights are normalized and used in the adjacency matrix of the GCN that aggregates information about the crowd. The aggregated crowd information and the robot state are used to predict the value \hat{V} of each possible state. Subscripts r and g indicate the robot and goal. Numerical subscripts index humans in the crowd.

learning network for autonomous driving [15]. The proposed gaze-modulated dropout leads to significant improvements in driving performance.

III. NETWORK ARCHITECTURE

Fig. 2 shows the proposed network structure for attentionbased value-learning (V-learning) using a graph representation. As shown in the top left of the framework, we first train a two-layer GCN to predict human attention to agents in the crowd. The learned attention weights are incorporated into the adjacency matrix of a second GCN, which aggregates the crowd information. Finally, the aggregated crowd features and robot state are combined to estimate the value of the crowd/robot state.

In this section, we first introduce the use of graph convolutional networks to process crowd information in the attention network and the crowd aggregation network. We then describe each of the three parts of Fig. 2 in detail. This section describes only the structure of the network. Training and testing are described in the next section.

A. Graph convolutional networks for crowds

Handling the varying numbers and large densities of humans has always been a problem for crowd robot navigation. However, the problem can be solved neatly with graph representations. Nodes in the graph correspond to the robot, the agents in the crowd, and the goal. For simplicity, we will refer to the agents in the crowd as humans, but the framework applies equally well if the crowd includes other robots. We assume the number of humans, N, is known.

We propose to use graph convolutional networks (GCNs) to process information about the crowd. To each node in the network, we associate a feature vector, which contains important information about the node. The graph convolutional layer is the key building block of a GCN. It takes input feature vectors for each node and transforms them to output feature vectors for each node by combining information both within

and across nodes. The set of input feature vectors for the nodes in layer l are contained in a matrix $\mathbf{H}^l \in \mathbb{R}^{M \times I}$, where M is the number of nodes in the graph and I is length of the input feature vector. These are transformed into a set of output vectors contained in a matrix $\mathbf{H}^{l+1} \in \mathbb{R}^{M \times O}$ (O is the output feature dimensionality) according to the layer-wise update rule:

$$\mathbf{H}^{l+1} = \sigma \left(\mathbf{A} \mathbf{H}^{l} \mathbf{W}^{l} \right) \tag{1}$$

 $\mathbf{W}^{l} \in \mathbb{R}^{I \times O}$ is a weight matrix. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{M \times M}$ defines the weights for combining the features from different nodes in the graph. We normalize it so that each row sums to one. $\sigma(\cdot)$ denotes the activation function. To increase complexity, successive graph convolutional layers can be cascaded.

The adjacency matrix reflects the topology of the graph structure. Zero values represent no connections between corresponding nodes. Larger values represent stronger connections. For the graph of interactions among the robot and humans, the meaning of the adjacency matrix is shown in Fig.3. The adjacency matrix is divided into three functional areas. The red and green areas are related to the human-robot interactions. Elements in the red area represent the importance that the robot assigns to the humans and itself, which can also be interpreted as attention. Elements in the green area indicate the influence of the robot on the humans. The blue area determines the interactions among the crowd.

Since we were primarily interested in modeling the robot's perspective, we used a star topology that included self-loops with the robot as the central connection point [18]. We focused on how to set the weights in the red area. For the green and blue areas, we set the values by first defining a binary array indicating connections, and the normalizing each row, resulting in the following assignments:

$$a_{ir} = 1/2 \tag{2}$$

$$a_{ij} = \begin{cases} 0, i \neq j \\ 1/2, i = j \end{cases}$$
(3)

	r	1		N
r	$\left(a_{rr} \right)$	a_{r1}	•••	a_{rN}
1	a_{1r}	a_{11}	•••	a_{1N}
	÷	÷	·	÷
N	a_{Nr}	a_{N1}	•••	a_{NN} /

Fig. 3: The adjacency matrix of the graph network for crowd aggregation. The red area determines the effect of the humans and the robot on the robot itself. The green area depends on the visibility of the robot to the human. The blue area models the interaction among the crowd. The subscript 'r' indicates the robot. Numerical subscripts indicate humans

where $i, j \in \{g, 1, ..., N\}$.

Note that we can easily extend the size of the GCN as the crowd size N grows, simply by increasing the number of nodes (M) and the size of the adjacency matrix. The size of the weight matrix is determined by the input and output feature dimensionality, and does not change with M.

B. Attention Network

We use a two-layer GCN to compute the attention weights. The robot, goal and humans are all included as nodes. Thus, the total number of nodes is M = N + 2, where N is the total number humans in the crowd.

The input features of each node are the corresponding position and velocity, expressed in robot-centric coordinate system where the x axis points in the direction of the goal. For node $i \in \{r, g, 1, ..., N\}$, the input features are

$$\mathbf{o}_i = [x_i, y_i, v_{xi}, v_{yi}]. \tag{4}$$

We set the output feature dimensionality of each node in the first layer to 128. The output of each node in the second layer is a scalar value corresponding to the attention of the robot to that node. For both layers, the activation function is the ReLU. The weights in the red part of the adjacency matrix A_a are all set to the same value, as shown in Fig. 2.

C. Crowd aggregation network

The robot and humans, but not the goal, were included as graph nodes for the crowd aggregation network. Thus, the total number of nodes in the GCN is M = N + 1.

The input features for each node included information about the robot, the goal and the corresponding agent. For node $i \in \{r, 1, ..., N\}$, the input features were:

$$\mathbf{s}_i = [\mathbf{p}, \mathbf{q}_i],\tag{5}$$

$$\mathbf{p} = [d_{\text{goal}}, v_{\text{pref}}, \theta, r_{\text{robot}}, v_{xr}, v_{yr}], \tag{6}$$

$$\mathbf{q}_{i} = [x_{i}, y_{i}, v_{xi}, v_{yi}, r_{i}, d_{i}, r_{\text{robot}} + r_{i}].$$
 (7)

The vector **p** contains information about the state of the robot and the goal, including distance between the robot and the goal d_{goal} , the robot's preferred velocity v_{pref} , the robot's moving direction θ , the radius of the robot r_{robot} and the robot's velocity (v_{xr}, v_{yr}) . **q**_i represents the state of the node, including its position (x_i, y_i) , velocity (v_{xi}, v_{yi}) , radius r_i , distance to the robot d_i and sum of the radii $r_{\rm robot} + r_i$. All positions and velocities are in a robot-centric coordinate system where the x axis points in the direction of the goal. We refer to the entire collection of input node features as the crowd/robot state, S_{cr} .

As shown in Fig.2, the input state of each node first passes through a two-layer multilayer perceptron (MLP) to increase the feature length to 100 for sufficient expressive power. The outputs of the MLP $\{e_i\}$ are then sent into a two-layer GCN. For both layers, the activation function is the ReLU.

In our experiments, we considered four ways to set the weights in the red part of the adjacency matrix A_v . In our proposed method, which we refer to as Gaze modulated GCN-based RL (*G-GCNRL*), we set the weights by first passing the outputs of the robot and human nodes of the attention network through a softmax function to normalize them, and then using the outputs of the softmax function as the corresponding adjacency weights. For comparison, we also trained three baseline models with the weights from the self-attention module of *SARL* [9] (*SA-GCNRL*), distance-related weights (*D-GCNRL*) and uniform weights (*U-GCNRL*). The distance-related weights capture the intuition that humans closer to the robot should exert a stronger influence on it. These weights decayed with the distance between the human and the robot according to

$$a_{ij} = \frac{e^{-d_{ij}^2/\sigma^2}}{\sum_{j=1}^n e^{-d_{ij}^2/\sigma^2}}, i = 1, 2, ..n$$
(8)

The parameter σ^2 was set to 2, based on a brute force search to maximize success rate of the *D*-*GCNRL* after imitation learning. The uniform weights were all equal and summed to one, resulting in an adjacency matrix similar to A_a .

D. Value network

The output features of the robot node in the second layer of the crowd aggregation network, c_r , are used to represent the influence of the crowd on the robot. There are concatenated with the robot state p before being sent into the value network, which is a four-layer MLP with (150,100,100,1) neurons in the layers. The scalar output of the network represents the value of the crowd/robot state and is used for action selection, as described in the next section.

IV. TRAINING AND TESTING

We learned the attention network \mathbf{G} with supervision from human gaze data.

To collect the gaze data, we reproduced overhead scenes from the *Students001* and *Students003* real world crowd datasets [30] in a simulator, and asked subjects to navigate a virtual robot through the crowd from a starting point to the goal using a joystick, while avoiding collisions. We used a Tobii Pro X60 remote eye tracker to collect gaze data at 60 Hz while subjects were performing the task. This experimental setup is similar to work collecting human gaze data while subjects played Atari games [16].

To generate the ground truth attention weights for each robot/crowd state configuration in the dataset, we first created

Algorithm 1: Deep V-learning **Input**: collected state-value pairs $\langle S_{cr}, V \rangle$ /* Imitation learning */ 1: for epoch = 1 to num of epochs do 2: $V = \mathbf{F}(S_{cr}, \mathbf{w})$ 3: $e = MSE(V, \hat{V})$ 4. $\mathbf{w} = \text{backprop}(e, \mathbf{w})$ 5: end for /* Reinforcement learning */ 6: **for** *episode* = 1 to num of episodes **do** 7: while not reach goal, collide or timeout do $u_t = \operatorname{argmax}_{u_t \in U} \mathbf{R} \left(\mathbf{S}_{cr}^t, u_t \right) + \gamma \mathbf{F} \left(\hat{\mathbf{S}}_{cr}^{t+1}, \mathbf{w} \right),$ 8: where \hat{S}_{cr}^{t+1} = propogate (u_t, S_{cr}^t) . Store $< \mathbf{S}_{cr}^{t}, \mathbf{R} (\mathbf{S}_{cr}^{t}, u_{t}) >.$ 9. S_{cr}^{t+1} = update (u_t, S_{cr}^t) ; t = t + 110: 11: end while Update replay buffer O. 12: Train value network with data from O. 13: 14: end for



Fig. 4: Datasets used in the simulation environment. The colored lines show pedestrian trajectories. For better visualization, the trajectories shown are down sampled by ten times. The black triangles show the starting positions and goals of the robot, which are 4m away from the center. The average number of humans per frame in the dataset is shown below each image.

a Gaussian mixture density over space by placing a Gaussian at each gaze point from a temporal window from -0.1 to +0.1 seconds around the current time point. The Gaussian standard deviation was set to two degrees of visual angle so that it covered the entire foveal region and part of the periphery. We assigned attention weights to each node by first sampling from this Gaussian mixture density at each node location, and then normalizing so that the attention weights of all nodes summed to one. We collected gaze data for multiple starts with varying crowd sizes and densities, resulting in around 1700 state-attention pairs.

We used the L1 loss function and trained the network for 400 epochs until convergence using the data from the *Students001* dataset. Dropout was applied after each graph convolutional layer with a drop probability of 0.5. The data collected from the *Students003* dataset was reserved for testing. The network trained on the *Students001* dataset was used to set the attention weights in all experiments.

We trained the crowd aggregation and value networks simultaneously following the deep reinforcement learning approach described in [9] [11]. For completeness, we describe the approach briefly here, and refer the reader to the original publications for more details.

The training algorithm is shown in the table titled Algorithm 1. The value function $\mathbf{F}(\cdot)$ is computed by the cascade of the crowd aggregation network and the value network, as described above. The reward function $\mathbf{R}(\cdot)$ awards the robot

for reaching its goal and penalizes it for getting too close to or colliding with humans [11].

Training consisted of two stages: imitation learning followed by reinforcement learning. All training was performed using the robot simulator with human data from the *Students001* dataset, example trajectories of which are shown in Fig. 4. Start frames were selected randomly, as were starting positions and goals (shown as opposite pairs of black triangles) We used the same unicycle robot model as used in [9]. Collected datasets were augmented by mirroring.

In the first stage, the crowd aggregation and value networks were trained with supervision. They learned to imitate the demonstrations of ORCA, which drives the robot through the crowd to the destination. We collected data in the simulation environment from the augmented crowd dataset, which contains real human trajectories with high crowd density. We simulated the sequential state update of humans as the robot navigated inside the crowd. Both successful cases and failure cases were recorded in the training dataset. For imitation learning, we collected 3000 episodes.

In the second stage, the networks were refined by reinforcement learning in the simulation environment. For each episode, we sampled multiple actions from the action space (U) at every step. For unicycle robot, U is defined to be a set of action pairs $\langle v, \theta \rangle$, where $v \in (0, v_{pref}], \theta \in [-\pi/4, \pi/4]$. We either selected the optimal action according to the current value function, or chose randomly to ensure exploration. The percentage of random actions decreased over time. States of the robot and humans were then updated. State-value pairs were stored in the replay buffer. We kept a size limit on the buffer so the old pairs were replaced by new ones gradually. For the state prediction (\hat{S}_{cr}^{t+1}) , we assumed the humans move following a constant velocity model. For the state update (S_{cr}^{t+1}) , we obtained the new states from the next frame of the dataset. The robot always moved with constant velocity (u_t) between timesteps. For reinforcement learning, we ran 20000 episodes and set the buffer size to 100000 state-value pairs. We followed the buffer size setting of SARL [9], and doubled the episodes for training, as the training environment is more complicated and we need more episodes to get a stable reward. Training was conducted in the Students001 dataset.

For testing of the entire system, we performed navigation experiments on the five real world pedestrian trajectories datasets shown in Fig. 4: the *Students003* dataset [30] (average crowd size 26), *NYC Grand Central* dataset [31] (average crowd size 30), the *ZARA2* dataset [30] (average crowd size 7), the *Hotel* dataset [32] (average crowd size 6), and the the *ETH* dataset [32] (average crowd size 6). These datasets cover a wide range of crowd sizes and traffic patterns.

We evaluated the performance of models by success rate and average navigation time in successful trials. For each dataset, we collected the results of each model over 2500 trials. During testing, we used greedy action selection (no exploration) to select the best action for each step.

V. EXPERIMENTAL RESULTS

This section evaluates the performance of our system with example trajectories and summary statistics: success rate and



Fig. 5: Estimated attention weights from our attention network are shown by the radius of the purple circle surrounding each human and by the numerical labels. Blue stars indicate the gaze points in the 0.2 second window used to compute ground truth attention weights. Blue lines link temporally adjacent gaze points. Red arrows indicate the instantaneous velocities.

average navigation time in successful trials. We also encourage readers to view the video¹ for a better understanding of the system dynamics.

A. Attention network evaluation

Fig. 5 compares the estimated attention weights with the gaze trajectory of the human operator for a test crowd scene from the *Students003* dataset. The figure shows a top down view similar to that shown by the simulator to the human operators.

The robot's path to the goal was blocked by two humans moving to the left. To avoid them, the human operator steered the robot so that it followed the human moving to the right. The human operator's gaze is concentrated on the two humans moving left. The learned attention weights are also the largest for those two. This suggests that the attention network has learned to infer correctly which humans in the crowd are most important for trajectory planning. Videos of this and more examples can be found on our website¹.

To evaluate the predicted attention weights quantitatively, we calculated the similarity between predicted attention weights and the ground truth weights using two standard metrics: the Kullback-Leibler divergence (KLD) and the Correlation Coefficient (CC). Smaller KL and larger CC denote better similarity. As shown in Table I, the attention weights predicted by our network closely match the ground truth weights. The KL divergence for the predicted attention weights is 50.5% smaller than for the distance related weights and 46.7% smaller than for the weights computed by *SARL*. Similarly, the CC for the predicted attention weights is 21.3% larger than for the weights and 17.5% larger than for the weights and 17.5% larger than for the weights computed by *SARL*.

B. System Performance

We first compare the performance of our proposed *G*-*GCNRL* network with the state of the art, *SARL*. We then give the results of ablation studies showing the impact of the use of the graph representation and the use of attention weights trained by human gaze. We conclude with an analysis of the performance of our system in different environments.

1) Comparison with SARL: Fig. 6 shows example trajectories generated by G-GCNRL and SARL in a sparse environment (Zara2) and a dense environment (NYC-GC). For both environments, G-GCNRL achieves the goal faster than SARL: 12.4 seconds vs. 14 seconds (sparse) and 9.2 seconds vs. 10 seconds (dense). SARL took more conservative policies, i.e. the robot waited somewhere or changed direction unnecessarily, whereas G-GCNRL drove the robot more smoothly. In the sparse environment, the SARL robot almost stopped between 6 and 8 seconds, whereas the velocity of the G-GCNRL robot remained fairly constant. In the dense environment, the path taken by the G-GCNRL robot after 6 seconds is much straighter than the path taken by the SARL robot.

We used two measures to quantify the smoothness of each robot trajectory: the standard deviation of the speed std(||v||) and the average absolute angular difference between two successive steps $mean(|\Delta\theta|)$. Both measures capture the idea that we wish the robot to move at a fairly constant speed with only small velocity variations. Lower values of both metrics suggest better smoothness. We compute the metrics over all the robot trajectories by our *G-GCNRL* and *SARL*. We find the average std(||v||) over all datasets for *G-GCNRL* (0.161) is significantly lower than for *SARL* (0.255). The average $mean(|\Delta\theta|)$ over all datasets for *G-GCNRL* (0.358) is significantly lower than *SARL* (0.426). Both have p-value < 0.001.

Table II compares the success rate and average navigation time of *G-GCNRL* with *SARL* on all of the testing environments. On average, *G-GCNRL* has an 18.4% higher success rate and a 16.4% shorter navigation time.

Both *G-GCNRL* approach and *SARL* assign different importance to different humans. There are two main differences in their approaches. *G-GCNRL* uses a GCN to estimate the value function and uses human gaze data to learn the attention weights. To evaluate the effect of these two differences, we implemented an intermediate model, *SA-GCNRL*, which used the GCN but the attention weights from *SARL*. *SA-GCNRL* achieves a 7.9% higher success rate and a 6.0% shorter navigation time on average than *SARL*, demonstrating the advantage of using the GCN to estimate the value function. *G-GCNRL* achieves a 9.7% higher success rate and an 11.1% shorter navigation time than *SA-GCNRL*, demonstrating the advantage of using attention weights trained from human gaze. Thus, both innovations are important in achieving the reported improvement.

2) Ablation studies: To evaluate the benefits of learning attention from human gaze, Table III compares the results of *G-GCNRL* with GCNRL where the weights in the adjacency matrices were set according to *SARL (SA-GCNRL)*, the distance from the robot (*D-GCNRL*) and uniformly (*U-GCNRL*). Using attention weights learned from human gaze resulted

¹Available from the supplementary files or the website: https://sites.google. com/view/gazenav

	KLD	CC
Predicted attention weights	0.49	0.74
Distance related weights	0.99	0.61
SARL [9] weights	0.92	0.63

TABLE I: Similarity of attention estimates. KLD denotes Kullback-Leibler divergence (smaller better) and CC denotes Correlation Coefficient (larger better).

in the best performance (10.5% higher than SA-GCNRL and D-GCNRL, and 5.0% higher than U-GCNRL). When crowd size is small (Zara2, Hotel, ETH), the gain in success rate by using attention (G-GCNRL vs. U-GCNRL) is around 1-2%. However, as crowd size increases, such as Students003 and NYC-GC, the gain in success rate of G-GCNRL reaches 10%. The navigation time of G-GCNRL is also shorter. This suggests that when crowd size grows larger, the network should allocate attention correctly.

To evaluate the benefits using the GCN structure to estimate the crowd state, Table IV compares the performance when using a *SARL*-like network to estimate the value function with the performance when using a GCN to estimate the value function. If uniform weights are used (*U-GCNRL* vs. *UARL*), using GCN improves success rate by 23.7% and reduces navigation time by 23.2%. If *SARL* weights are used (*SA-GCNRL* vs. *SARL*), success rate improves by 7.9% and navigation time reduces by 6.0%.

3) Performance in different environments: Comparing across Tables II-IV, we find all models had the worst performance on the NYC-GC dataset, which is the most complex one with on average 30 humans/frame and the most diversity in crowd movement. In this complex situation, the success rate of the models with uniform attention (UARL and U-GCNRL) performed the worst. For datasets with lower density crowds (Zara2, Hotel and ETH), the performance of different models was relatively even. Our intuition: when crowd size is small, it is relatively easy to keep track of all the humans, but when crowd size grows, attention becomes critical in preventing the system from becoming overwhelmed, by allowing it to focus on the most important parts of the crowd. Across all environments, G-GCNRL achieves highest success rate.

VI. DISCUSSION

Interestingly, we find *U*-GCNRL achieves shortest navigation time on the datasets with small crowd size (shown in Table III). The reason might be that when the crowd size becomes small, the attention is no longer a critical factor. This suggests that a hybrid strategy depending on the crowd size may achieve better results. For example, we could train multiple attention networks with different variances of the Gaussian distributions used to assign the ground truth attention weights. Small variances could focus attention on only a few humans near the gaze points whereas variances near infinity could allocate attention uniformly. Different attention networks could be used for different crowd sizes.

VII. CONCLUSION

This paper presents a crowd navigation method for mobile robots and demonstrates its efficacy on real-world dense pedestrian data. The proposed method outperforms the stateof-the-art. There are two key innovations in our work. The first innovation is the introduction of a graph convolutional network (GCN) for reinforcement learning to integrate information about the environmental context of the robot. The GCN makes our approach immediately extensible to varying crowd sizes in a simple and principled way. The influence of different agents can be controlled by changing the adjacency matrix. The second innovation is the introduction of an attention network trained using human gaze data for assigning adjacency values. The two innovations enhance the performance of the network, and may also be useful in other applications.

REFERENCES

- A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 1685–1692.
- [2] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008, pp. 1928–1935.
- [3] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal nbody collision avoidance," in *Robotics Research*. Springer, 2011, pp. 3–19.
- [4] V. V. Unhelkar, C. Pérez-D'Arpino, L. Stirling, and J. A. Shah, "Humanrobot co-navigation using anticipatory indicators of human walking motion," in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 6183–6190.
- [5] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. Lau, M. C. Lin, and D. Manocha, "Brvo: Predicting pedestrian trajectories using velocityspace reasoning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.
- [6] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2010, pp. 797–803.
- [7] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A socialforce based approach with human awareness-navigation in crowded environments," in 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2013, pp. 1688–1694.
- [8] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016, pp. 1190–1197.
- [9] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," arXiv preprint arXiv:1809.08835, 2018.
- [10] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 3052–3059.
- [11] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized noncommunicating multiagent collision avoidance with deep reinforcement learning," in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 285–292.
- [12] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2255–2264.
- [13] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "Sophie: An attentive gan for predicting paths compliant to social and physical constraints," in *Proceedings of the IEEE Conference* on Computer Vision and Pattern Recognition, 2019, pp. 1349–1358.
- [14] C. Liu, Y. Chen, L. Tai, H. Ye, M. Liu, and B. E. Shi, "A gaze model improves autonomous driving," in *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*. ACM, 2019, p. 33.
- [15] Y. Chen, C. Liu, L. Tai, M. Liu, and B. E. Shi, "Gaze training by modulated dropout improves imitation learning," *arXiv preprint* arXiv:1904.08377, 2019.
- [16] R. Zhang, Z. Liu, L. Zhang, J. A. Whritner, K. S. Muller, M. M. Hayhoe, and D. H. Ballard, "Agil: Learning attention from human for visuomotor tasks," in *Proceedings of the European Conference on Computer Vision* (ECCV), 2018, pp. 663–679.



Fig. 6: Trajectories of robot running in the simulated environments: a sparse environment (Zara2) and a dense environment (NYC-GC) show here. The highlighted green lines show the robot navigation. The other colors show the human trajectories. Each circle is timestamped.

		Su	iccess Rat	te			Navi	gation Tii	me			
	Students003	NYC-GC	Zara2	Hotel	ETH	AVG	Students003	NYC-GC	Zara2	Hotel	ETH	AVG
SARL	0.692	0.358	0.815	0.581	0.657	0.621	13.1	13.8	12.9	13.4	13.9	13.4
SA-GCNRL	0.616	0.431	0.838	0.683	0.782	0.670	12.0	12.0	13.2	12.8	13.0	12.6
G-GCNRL	0.753	0.453	0.936	0.703	0.831	0.735	11.2	11.8	10.9	11.1	11.1	11.2

TABLE II: Comparison with the state-of-the-art.

		Su	ccess Rat	e			Navi	gation Tir	ne			
	Students003	NYC-GC	Zara2	Hotel	ETH	AVG	Students003	NYC-GC	Zara2	Hotel	ETH	AVG
G-GCNRL	0.753	0.453	0.936	0.703	0.831	0.735	11.2	11.8	10.9	11.1	11.1	11.2
SA-GCNRL	0.616	0.431	0.838	0.683	0.782	0.670	12.0	12.0	13.2	12.8	13.0	12.6
D-GCNRL	0.556	0.405	0.876	0.699	0.790	0.665	12.7	13.8	11.5	9.3	10.3	11.5
U-GCNRL	0.671	0.387	0.928	0.687	0.827	0.700	11.2	12.0	10.1	9.3	10.2	10.6

TABLE III: Ablation study to show the advantage of attention weights trained with human gaze data.

		Su	ccess Rat	e			Navi	gation Tii	ne			
	Students003	NYC-GC	Zara2	Hotel	ETH	AVG	Students003	NYC-GC	Zara2	Hotel	ETH	AVG
SA-GCNRL	0.616	0.431	0.838	0.683	0.782	0.670	12.0	12.0	13.2	12.8	13.0	12.6
SARL	0.692	0.358	0.815	0.581	0.657	0.621	13.1	13.8	12.9	13.4	13.9	13.4
U-GCNRL	0.671	0.387	0.928	0.687	0.827	0.700	11.2	12.0	10.1	9.3	10.2	10.6
UARL	0.591	0.310	0.816	0.477	0.636	0.566	12.7	13.1	14.0	14.0	15.0	13.8

TABLE IV: Ablation study to show the advantage of the graph structure.

- [17] J. Chen, T. Ma, and C. Xiao, "Fastgen: fast learning with graph convolutional networks via importance sampling," arXiv preprint arXiv:1801.10247, 2018.
- [18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv preprint arXiv:1609.02907, 2016.
- [19] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, p. 4282, 1995.
- [20] G. Ferrer, A. G. Zulueta, F. H. Cotarelo, and A. Sanfeliu, "Robot social-aware navigation framework to accompany people walking sideby-side," *Autonomous Robots*, vol. 41, no. 4, pp. 775–793, 2017.
- [21] S. F. Chik, C. F. Yeong, E. L. M. Su, T. Y. Lim, F. Duan, J. T. C. Tan, P. H. Tan, and P. J. H. Chin, "Gaussian pedestrian proxemics model with social force for service robot navigation in dynamic environment," in *Asian Simulation Conference*. Springer, 2017, pp. 61–73.
- [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 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 1111–1117.
- [24] M. Pfeiffer, U. Schwesinger, H. Sommer, E. Galceran, and R. Siegwart, "Predicting actions to act predictably: Cooperative partial motion planning with maximum entropy models," in 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2016, pp. 2096–2101.
- [25] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially

compliant mobile robot navigation via inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

- [26] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 6252–6259.
- [27] T. Xie and J. C. Grossman, "Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties," *Physical Review Letters*, vol. 120, no. 14, p. 145301, 2018.
- [28] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.
- [29] C. Shang, Q. Liu, K.-S. Chen, J. Sun, J. Lu, J. Yi, and J. Bi, "Edge attention-based multi-relational graph convolutional networks," *arXiv* preprint arXiv:1802.04944, 2018.
- [30] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer Graphics Forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 655–664.
- [31] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," in 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012, pp. 2871–2878.
- [32] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in 2009 IEEE 12th International Conference on Computer Vision. IEEE, 2009, pp. 261–268.