2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning

Shuijing Liu*, Peixin Chang*, Weihang Liang†,
Neeloy Chakraborty†, and Katherine Driggs-Campbell

***Abstract*— Safe and efficient navigation through human crowds is an essential capability for mobile robots. Previous work on robot crowd navigation assumes that the dynamics of all agents are known and well-defined. In addition, the performance of previous methods deteriorates in partially observable environments and environments with dense crowds. To tackle these problems, we propose decentralized structural-Recurrent Neural Network (DS-RNN), a novel network that reasons about spatial and temporal relationships for robot decision making in crowd navigation. We train our network with model-free deep reinforcement learning without any expert supervision. We demonstrate that our model outperforms previous methods in challenging crowd navigation scenarios. We successfully transfer the policy learned in the simulator to a real-world TurtleBot 2i.**

I. Introduction

As mobile robots are becoming prevalent in people's daily lives, autonomous navigation in crowded places with other dynamic agents is an important yet challenging problem [1], [2]. Inspired by the recent applications of deep learning in robot control [3]–[6] and in graph modeling [7], we seek to build a learning-based graphical model for mobile robot navigation in pedestrian-rich environments.

Robot crowd navigation is a challenging task for two key reasons. First, the problem is decentralized, meaning that each agent runs its own policy individually, which makes the environment not fully observable to the robot. For example, other agents' preferred walking style and intended goals are not known in advance and are difficult to infer online [8]. Second, the crowded environment contains both dynamic and static agents, who implicitly interact with each other during navigation. The ways agents influence each other are often difficult to model [9], making the dynamic environment harder to navigate and likely to produce emergent phenomenon [10].

Despite these challenges, robot crowd navigation is well-studied and has had many successful demonstrations [11]–[13]. Reaction-based methods such as Optimal Reciprocal Collision Avoidance (ORCA) and Social Force (SF) use one-step interaction rules to determine the robot's optimal action [12], [14], [15]. Another line of works first predict other agents' future trajectories and then plan a path for the robot [16]–[19]. However, these two methods suffer from the *freezing robot problem*: in dense crowds, the planner decides that all paths are unsafe and the robot freezes, which is suboptimal as a feasible path usually exists [20].

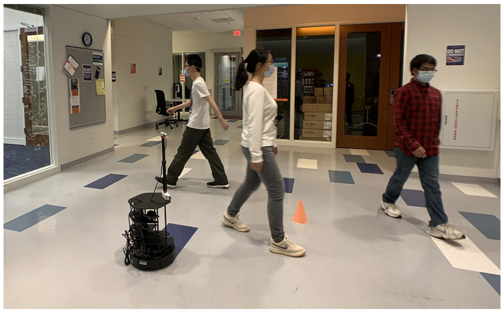


Fig. 1: **Real-world crowd navigation with a TurtleBot 2i.** The orange cone on the floor denotes the robot goal. The TurtleBot is equipped with cameras for localization and human tracking.

More recently, learning-based methods model the robot crowd navigation as a Markov Decision Process (MDP) and use Deep V-Learning to solve the MDP [13], [21]–[24]. In Deep V-Learning, the agent chooses an action based on the state value approximated by neural networks. However, Deep V-Learning is typically initialized by ORCA using supervised learning and, as a result, the final policy inherits ORCA's aforementioned problems. Moreover, to choose actions from the value network, the dynamics of the humans are assumed to be known to the robot and are deterministic, which can be unrealistic in real applications.

In this paper, we seek to create a learning framework for robot crowd navigation using spatio-temporal reasoning trained with model-free deep reinforcement learning (RL). We model the crowd navigation scenario as a decentralized spatio-temporal graph (st-graph) to capture the interactions between the robot and multiple humans through both space and time. Then, we convert the decentralized st-graph to a novel end-to-end decentralized structural-RNN (DS-RNN) network. Using model-free RL, our method directly learns a navigation policy without prior knowledge of any agent's dynamics or expert policies. Since the robot learns entirely from its own experience, the resulting navigation policy easily adapts to dense human crowds and partial observability and outperforms previous methods in these scenarios.

We present the following contributions: (1) We propose a novel deep neural network architecture called DS-RNN, which enables the robot to perform efficient spatio-temporal reasoning in crowd navigation; (2) We train the network using model-free RL without any supervision, which both simplifies the learning pipeline and avoids the network from converging to a suboptimal policy too early; (3) Our method

*,† denote equal contribution.

S. Liu, P. Chang, W. Liang, N. Chakraborty and K. Driggs-Campbell are with the Department of Electrical and Computer Engineering at the University of Illinois at Urbana-Champaign. emails: {sliu105,pchang17,weihang2,neeloyc2,krdc}@illinois.edu

978-1-7281-9077-8/21/$31.00 ©2021 IEEE

demonstrates better performance in challenging navigation settings compared with previous methods[1].

This paper is organized as follows: We review previous related works in Section II. We formalize the problem and propose our network architecture in Section III. Experiments and results in simulation and in the real world are discussed in Section IV and Section V, respectively. Finally, we conclude the paper in Section VI.

II. Related Works

A. Reaction-based methods

Robot navigation in dynamic environments has been studied for over two decades [11], [25]–[27]. A subset of these works specifically focuses on robot navigation in pedestrian-rich environments or crowd navigation [1], [28].

Reaction-based methods such as Reciprocal Velocity Obstacle (RVO) and ORCA model other agents as velocity obstacles to find optimal collision-free velocities under reciprocal assumption [12], [14], [29]. Another method named Social Force models the interactions in crowds using attractive and repulsive forces [15]. However, these algorithms suffer from the *freezing robot problem* [20]. In addition, since the robot only uses the current states as input, the generated paths are often shortsighted and unnatural.

In contrast, we train our network with model-free RL to mitigate the freezing problem. Also, our network contains RNNs that take a sequence of trajectories as input to encourage longsighted behaviors.

B. Trajectory-based methods

Trajectory-based methods predict other agents' intended trajectories to plan a feasible path for the robot [16]–[19], [30]–[32]. Trajectory predictions allow the robot planner to look into the future and make long-sighted decisions. However, these methods have the following disadvantages. First, predicting trajectory sequences and searching a path from a large state space online are computationally expensive and can be slow in real time [33]. Second, the predicted trajectories can make a large portion of the space untraversable, which might make the robot overly conservative [34].

C. Learning-based methods

With the recent advancement of deep learning, imitation learning has been used to uncover policies from demonstrations of desired behaviors [35], [36]. Another line of works use Deep V-Learning, which combines supervised learning and RL [13], [21]–[24], [30]. Given the state transitions of all agents, the planner first calculates the values of all possible next states from a value network. Then, the planner chooses an action that leads to the state with the highest value. To train the value network, Deep V-Learning first initializes the network by supervised learning using trajectories generated by ORCA, and then fine-tunes the network with RL. Using a single rollout of the policy, Monte-Carlo value estimation calculates the ground-truth values for both supervised learning and RL.

Deep V-Learning has demonstrated success in simulation and/or in the real world but still suffers from the following drawbacks: (1) Deep V-Learning assumes that the state transitions of all surrounding humans are known and well-defined, which are in fact highly stochastic and difficult to model; (2) Since the networks are pre-trained with supervised learning, they share the same disadvantages with the demonstration policy, which are hard to be corrected by RL; (3) Monte-Carlo value estimation is not scalable with increasing time horizon; and (4) To achieve the best performance, Deep V-Learning needs state information of all humans. If applied to real robots, a real-time human detector with a 360° field of view is required, which can be expensive or impractical.

To tackle these problems, we introduce a policy network trained with model-free RL, which does not need state transitions, Monte-Carlo value estimation, or expert supervision. Further, we show that incorporating both spatial and temporal reasoning in our network improves performance in challenging navigation environments over prior methods.

D. Spatio-temporal graphs and structural-RNN

St-graph is a type of conditional random field [37] with wide applications [7], [38]–[40]. St-graphs use nodes to represent the problem components and edges to capture the spatio-temporal interactions [7]. With each node or edge governed by a factor function, st-graph decomposes a complex problem into many smaller and simpler factors.

Jain *et al* propose a general method called structural-RNN (S-RNN) that transforms any st-graph to a mixture of RNNs that learn the parameters of factor functions end-to-end [7]. S-RNNs have been applied to research areas such as human tracking and human trajectory prediction [41], [42]. However, the scope of these works is restricted to learning from static datasets. Applying st-graph to crowd navigation poses extra challenges in data collection and decision-making under uncertainty. Although some works in crowd navigation have used graph convolutional network [43] to model the robot-crowd interactions [24], [30], to the best of our knowledge, our work is the first to combine S-RNN with model-free RL for robot crowd navigation.

III. Methodology

In this section, we first formulate the robot decision making in crowd navigation as an RL problem. Then, we present our approach to model crowd navigation scenario as an st-graph, which leads to the derivation of our DS-RNN network architecture.

A. Problem formulation

Consider a robot interacting with an episodic environment with other humans. We model this interaction as an MDP, defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathcal{S}_0 \rangle$. Suppose that all agents move in a 2D Euclidean space. Let $\mathbf{w}^t$ be the robot states and $\mathbf{u}_i^t$ be the i-th human's states observable by the robot. Then, the state $s_t \in \mathcal{S}$ for the MDP is $s_t =$

[1] Our code is available at `https://github.com/Shuijing725/CrowdNav_DSRNN`.

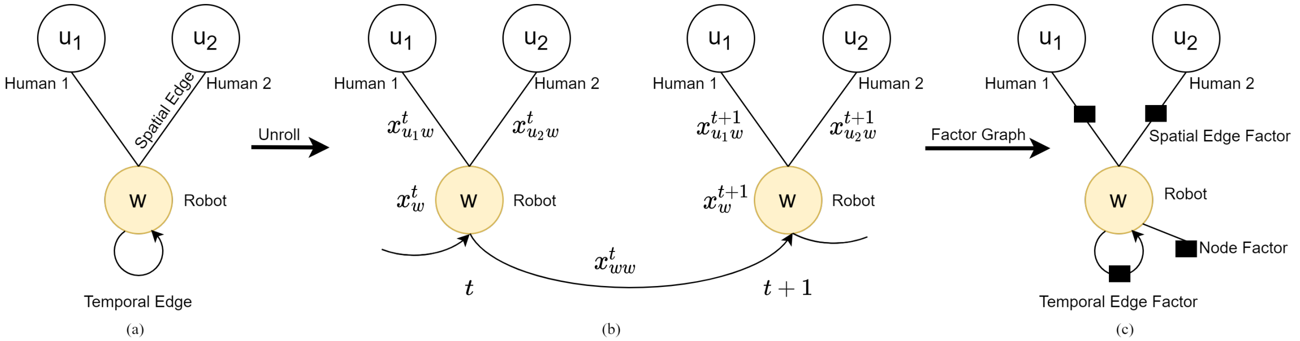


Fig. 2: **Conversion from the st-graph to the factor graph.** (a) St-graph representation of the crowd navigation scenario. We use w to denote the robot node and u_i to denote the i-th human node. (b) Unrolled st-graph for two timsteps. At timestep t, the node feature for the robot is x^t_w. The spatial edge feature between the i-th human and the robot is $x^t_{u_iw}$. The temporal edge feature for the robot is x^t_{ww}. (c) The corresponding factor graph. Factors are denoted by black boxes.

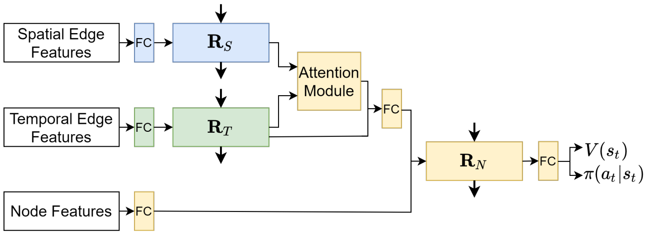


Fig. 3: **DS-RNN network architecture.** The components for processing spatial edge features, temporal edge features, and node features are in blue, green, and yellow, respectively. Fully connected layers are denoted as FC.

$[\mathbf{w}^t, \mathbf{u}^t_1, ..., \mathbf{u}^t_n]$ assuming a total number of n humans was involved at the timestep t. The robot state $\mathbf{w}^t$ consists of the robot's position (p_x, p_y), velocity (v_x, v_y), goal position (g_x, g_y), maximum speed v_{max}, heading angle θ, and radius ρ. Each human state $\mathbf{u}^t_i$ consists of the human's position (p^i_x, p^i_y). In contrast to previous works [13], [14], [21], [30], the human state $\mathbf{u}^t_i$ does not include human's velocity and radius because they are hard to be measured accurately in the real world.

In each episode, the robot begins at an initial state $s_0 \in \mathcal{S}_0$. At each timestep t, the robot takes an action $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|s_t)$. In return, the robot receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $P(\cdot|s_t, a_t)$. 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 t exceeds the maximum episode length T, the robot reaches its goal, or the robot collides with any humans.

Let $\gamma \in (0, 1]$ be the discount factor. Then, $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the total accumulated return from timestep t. The goal of the robot is to maximize the expected return from each state. The value of state s under policy π, defined as $V(s) = \mathbb{E}[R_t|s_t = s]$, is the expected return for following policy π from state s.

B. Spatio-Temporal Graph Representation

We formulate the crowd navigation scenario as a decentralized st-graph. Our graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$ consists of a set of nodes $\mathcal{V}$, a set of spatial edges $\mathcal{E}_S$, and a set of temporal edges $\mathcal{E}_T$. As shown in Fig. 2a, the nodes in the st-graph represent the agents, the spatial edges connect two different agents at the same timestep, and the temporal edges connect the same nodes at adjacent timesteps. We prune the edges and nodes not shown in Fig. 2a as they have little effect on the robot's decisions[2]. The corresponding unrolled st-graph is shown in Fig. 2b.

The factor graph representation of the st-graph factorizes the robot policy function into the robot node factor, spatial edge factors, and robot temporal edge factor. At each timestep, the factors take the node or edge features as inputs and collectively determine the robot's action. In Fig. 2c, factors are denoted by black boxes and have parameters that need to be learned.

We choose $x^t_{u_iw}$ to be the vector pointing from humans' position to the robot position, $(p^i_x - p_x, p^i_y - p_y)$, x^t_{ww} to be robot velocity (v_x, v_y), and x^t_w to be $\mathbf{w}^t$. To reduce the number of parameters, all spatial edges share the same factor. This parameter sharing is important for the scalability of our st-graph because the number of parameters is kept constant with an increasing number of humans [7].

C. Network Architecture

As shown in Fig. 3, we derive our network architecture from the factor graph representation of the st-graph motivated by [7]. In our network, we represent each factor with an RNN, referred to as spatial edgeRNN $\mathbf{R}_S$, temporal edgeRNN $\mathbf{R}_T$, and nodeRNN $\mathbf{R}_N$ respectively. We use W and f to denote trainable weights and fully connected layers throughout this section.

The spatial edgeRNN $\mathbf{R}_S$ captures the spatial interactions between humans and the robot. $\mathbf{R}_S$ first applies a non-linear transformation to each spatial feature $x^t_{u_iw}$ and then feeds the transformed results to the RNN cell:

$$h^t_{u_iw} = \mathrm{RNN}\left(h^{t-1}_{u_iw}, f_{\text{spatial}}(x^t_{u_iw})\right) \qquad (1)$$

where $h^t_{u_iw}$ is the hidden state of the RNN at time t for i-th human and the robot. Due to the parameter sharing mentioned in Section III-B, the spatial edge features between all human-robot pairs are fed into the same spatial edgeRNN.

The temporal edgeRNN $\mathbf{R}_T$ captures the dynamics of the robot's own trajectory. Similar to $\mathbf{R}_S$, $\mathbf{R}_T$ applies a linear

[2] From experiments, we find that a network derived from a full st-graph as in [7] performs very similarly to DS-RNN.

transformation to the temporal edge feature and processes the results with its RNN cell:

$$h^t_{ww} = \text{RNN}\left(h^{t-1}_{ww},\ f_{\text{temporal}}(x^t_{ww})\right) \tag{2}$$

where h^t_{ww} is the hidden state of the RNN at time t.

The outputs of two edgeRNNs are fed into an attention module which assigns attention weights to each spatial edge. The attention mechanism is similar to the *scaled dot product attention* in [44]. Let V^t be the output of $\mathbf{R}_S$ at time t, $V^t = [h^t_{u_1 w}, ..., h^t_{u_n w}]^\top$, where n is the number of spatial edges or humans. Both V^t and h^t_{ww} are first put through linear transformations:

$$Q^t = V^t W_Q, \quad K^t = h^t_{ww} W_K \tag{3}$$

where $Q^t \in \mathbb{R}^{n \times d_k}$, $K^t \in \mathbb{R}^{1 \times d_k}$, and d_k is a hyperparameter for the attention size. The attention weight at time t, α^t, is calculated as

$$\alpha^t = \text{softmax}\left(\frac{n}{\sqrt{d_k}} Q^t (K^t)^\top\right) \tag{4}$$

The output of the attention module at time t, v^t_{att}, is the weighted sum of spatial edges:

$$v^t_{\text{att}} = (V^t)^\top \alpha^t \tag{5}$$

The nodeRNN $\mathbf{R}_N$ uses the robot state, x^t_w, the weighted hidden states of $\mathbf{R}_S$, v^t_{att}, and the hidden states of temporal edgeRNN, h^t_{ww}, to determine the robot action and state value at each time t. The nodeRNN concatenates v^t_{att} and h^t_{ww} and embeds the concatenated results and the robot state with linear transformations:

$$e^t = f_{\text{edge}}([v^t_{\text{att}}, h^t_{ww}]), \quad n^t = f_{\text{node}}(x^t_w) \tag{6}$$

Both e^t and n^t are concatenated and fed into $\mathbf{R}_N$ to get the nodeRNN hidden state.

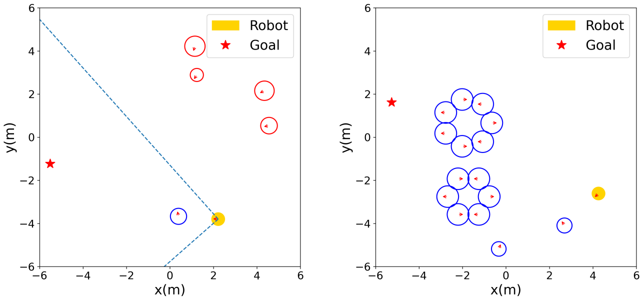
$$h^t_w = \text{RNN}\left(h^{t-1}_w, [e^t, n^t]\right) \tag{7}$$

Finally, the h^t_w is input to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$. We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [45] and we adopt the PPO implementation from [46]. To accelerate and stabilize training, we run twelve instances of the environment in parallel for collecting the robot's experiences. At each policy update, 30 steps of six episodes are used.

By identifying the independent components of robot crowd navigation, we split the complex problem into smaller factors, and use three RNNs to efficiently learn the parameters of the corresponding factors. By combining all components above, the end-to-end trainable DS-RNN network performs spatial and temporal reasoning to determine the robot action.

IV. Simulation Experiments

In this section, we describe the simulation environment for training and present our experimental results in simulation.



(a) (b)

Fig. 4: **Illustration of our simulation environment.** In a $12m \times 12m$ $2D$ plane, the humans are represented as circles, the orientation of an agent is indicated by a red arrow, the robot is the yellow disk, and the robot's goal is the red star. We outline the borders of the robot FoV with dashed lines. The humans in the robot's FoV are blue and the humans outside are red.

A. Simulation environment

Fig. 4 shows our 2D simulation environments adapted from [13]. We use holonomic kinematics for each agent, whose action at time t consists of the desired velocity along the x and y axis, $a_t = [v_x, v_y]$. All humans are controlled by ORCA with randomized maximum speed and radius. We assume that humans react only to other humans but not to the robot. This invisible setting prevents our model from learning an extremely aggressive policy in which the robot forces all humans to yield while achieving a high reward. We also assume that all agents can achieve the desired velocities immediately, and they will keep moving with these velocities for Δt seconds. We define the update rule for an agent's position p_x, p_y as follows:

$$\begin{aligned} p_x[t+1] &= p_x[t] + v_x[t]\Delta t \\ p_y[t+1] &= p_y[t] + v_y[t]\Delta t \end{aligned} \tag{8}$$

1) Environment configurations: Fig. 4a shows the FoV Environment, where the robot's field view (FoV) is within 0° to 360° and remains unchanged in each episode. The robot assumes that the humans out of its view proceed in a straight line with their last observed velocities. There are always five humans, whose starting and goal positions are randomly placed on a circle with radius $6m$. The FoV Environment simulates the limited sensor range of a robot, since deploying several sensors to obtain a 360° FoV is usually expensive and unrealistic in the real world.

Fig. 4b shows the Group Environment, where the robot's FoV is 360° but the number of humans is large and remains the same in each episode. Among these humans, some form circle groups in random positions and do not move while the rest of them move freely. The Group Environment simulates the scene of a dense crowd with both static and dynamic obstacles. We use this environment to evaluate whether the robot policies have the *freezing robot problem*.

To simulate the variety of complexities in real-world crowd navigation, we add the following randomness and features not included in the original simulator from [13]. When an episode begins, the robot's initial position and goal are chosen randomly. In addition, all humans occasionally change

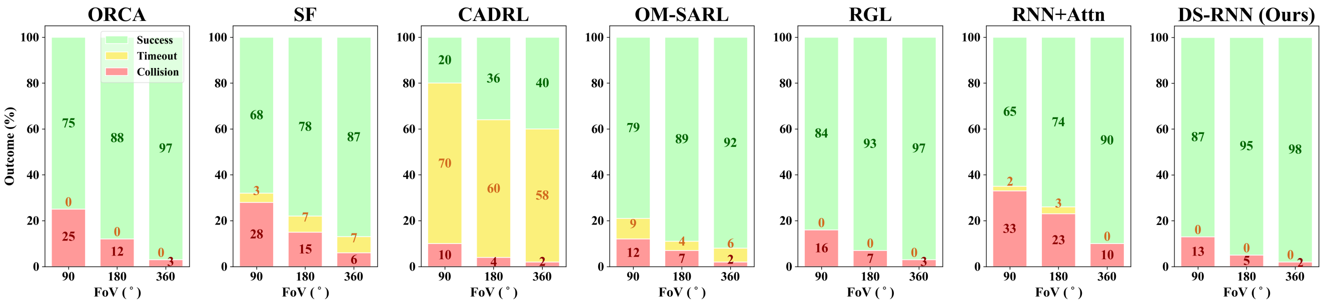


Fig. 5: Success, timeout, and collision rates w.r.t. different FoV. The numbers on the bars indicate the percentages of the corresponding bars.

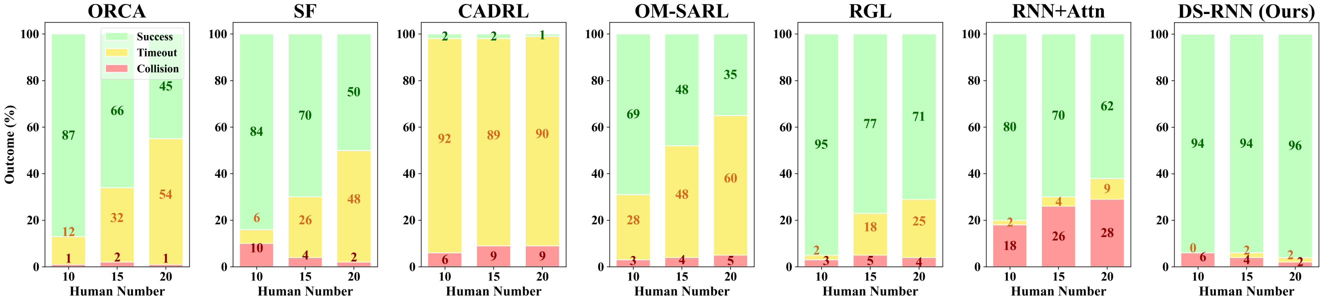


Fig. 6: Success, timeout, and collision rates w.r.t. different number of humans.

TABLE I: Navigation time (second) in two environments.

Method	FoV			Number of Humans		
	90°	180°	360°	10	15	20
ORCA	**9.12**	9.96	10.40	15.94	19.09	19.66
SF	20.86	24.28	23.96	24.60	30.26	31.12
CADRL	30.84	27.54	33.46	32.62	36.81	41.75
OM-SARL	18.32	13.70	21.04	27.25	23.79	29.24
RGL	9.54	**9.48**	**9.59**	**13.22**	**14.75**	16.44
RNN+Attn	16.57	14.00	10.96	16.01	21.31	25.55
DS-RNN	11.83	10.99	11.79	13.51	15.64	**15.52**

their goal positions within an episode. Finally, to simulate a continuous human flow, immediately after humans arrive at their goal positions, they will move to new random goals instead of remaining stationary at their initial destinations.

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

$$r(s_t, a_t) = \begin{cases} -20, & \text{if } d^t_{min} < 0 \\ 2.5(d^t_{min} - 0.25), & \text{if } 0 < d^t_{min} < 0.25 \\ 10, & \text{if } d^t_{goal} \leq \rho_{robot} \\ 2(-d^t_{goal} + d^{t-1}_{goal}), & \text{otherwise.} \end{cases} \quad (9)$$

where d^t_{min} is the minimum separation distance between the robot and any human at time t, and d^t_{goal} is the $L2$ distance between the robot position and goal position at time t. Intuitively, the robot gets a high reward when it approaches the goal while maintaining a safe distance from all humans.

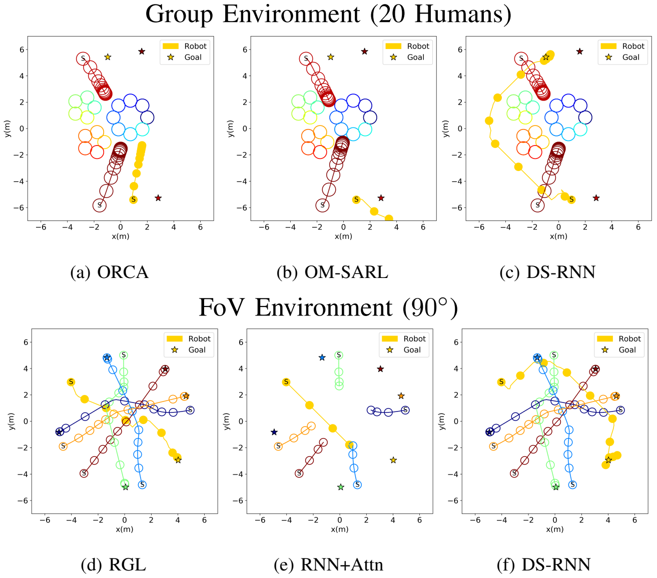
B. Experiment setup

1) Baselines and Ablation Models: We compare the performance of our model with the representatives of the three types of methods in Section II. We use ORCA and SF as the baselines for reaction-based methods; Relational Graph Learning (RGL) [30] as a baseline for both trajectory-based methods and Deep V-Learning; and CADRL [21] and OM-SARL [13] as the baselines for Deep V-Learning.

To remove the performance gain caused by other factors such as model-free RL and RNN, we also implement an ablation model, called RNN+Attn, by adding an RNN to the end of OM-SARL network. For RNN+Attn, the attention module assigns attention weights on the state features of humans. The weighted human features are then concatenated with robot state features to form the joint state features which are passed to an RNN network with the same size and sequence length as the robot nodeRNN in our model. Both networks are trained using PPO with the same hyper-parameters and thus the results serve as a clean comparison to highlight the benefits of our DS-RNN.

2) Training: We use the same reward as defined in Equation 9 for CADRL, OM-SARL, RGL, RNN+Attn, and DS-RNN. The network architectures of all methods are kept the same in all experiments. We train DS-RNN and RNN+Attn for 1×10^7 timesteps with a learning rate 4×10^{-5}. We train all baselines as stated in the original papers.

3) Evaluation: We evaluate the performance of all the models with six experiments: for the FoV Environment, the FoV of the robot is 90°, 180°, or 360°; for the Group Environment, the number of humans is 10, 15, or 20. For each of the six experiments, we test all the models with 500 random unseen test cases. We measure the percentage of success, collision, and timeout episodes, as well as the average navigation time of the successful episodes.



(a) ORCA (b) OM-SARL (c) DS-RNN

(d) RGL (e) RNN+Attn (f) DS-RNN

Fig. 7: **Trajectory comparisons of different methods with the same test cases.** Letter "S" denotes moving agents' starting positions, and stars denote moving agents' goals. The yellow filled circle denotes the robot. For the Group Environment (top), static humans are grouped in three circles.

C. Results

1) Spatio-temporal reasoning: We show the effectiveness of our DS-RNN architecture by comparing it with RNN+Attn. In Fig. 5 and Fig. 6, compared with RNN+Attn, our model exhibits higher success rates and lower collision and timeout rates in all settings. In Table I, our model has shorter navigation time because DS-RNN often finds a better path (Fig. 7e and 7f). We believe the main reason is that by formulating the crowd navigation into an st-graph, we decompose the robot decision making into smaller factors and feed each RNN with only relevant edge or node features. In this way, the three RNNs are able to learn their corresponding factors more effectively. By combining all factors (RNNs), the robot is able to explicitly reason about the spatial relationships with humans and its own dynamics to take actions. In contrast, RNN+Attn does not have such spatio-temporal reasoning and learns all factors with one single RNN, which explains its lower performance.

2) Comparison with traditional methods: We compare the performance of our model with those of ORCA and SF. As shown in Fig. 6, in the Group Environment, ORCA and SF exhibit high timeout rates, which increases significantly as the number of humans increases. This observation indicates that the *freezing robot problem* is prevalent in these mixed static and dynamic settings (Fig. 7a). Also, as Table I suggests, the large navigation times show that both methods are overly conservative in dense crowds. In the FoV Environment, our method also outperforms ORCA and SF in most metrics, as shown in Fig. 5 and Table I, because our method explores the environment and learns from the past experience during RL training. Combined with spatio-temporal reasoning, our method is able to better adapt to dense and partially observable environments. In addition, with our method, the robot is long-sighted because RL optimizes the policy over cumulative reward and the RNNs takes a sequence of trajectories to make decisions while ORCA and SF only consider the current state (Fig. 7c).

3) Comparison with Deep V-Learning: We compare model-free RL training with Deep V-Learning used by CADRL, OM-SARL, and RGL. In Fig. 6, all three baselines exhibit large timeout rates. The reason is that the value networks are initialized by a suboptimal expert (ORCA) in supervised learning and are insufficient to provide good state value estimates, resulting in policies that inherit OCRA's drawbacks. In contrast, model-free RL enables RNN+Attn and DS-RNN to learn from scratch and prevents the network from converging to a suboptimal policy too early. In addition, despite the unknown state transitions of all agents, RNN+Attn and our method still perform better in all metrics compared with Deep V-Learning.

As shown in Fig. 7d and 7f, RGL is competitive to our method in some cases, because the relational graph can perform spatial reasoning and human trajectory predictions make RGL long-sighted. However, in RGL, the relational graph and the robot planner are separated modules, while our network is trained end-to-end and jointly learns the robot-human interactions and decision making.

V. Real-world Experiments

We evaluate our trained model's performance on a TurtleBot 2i mobile platform as shown in Fig. 1. An Intel RealSense depth camera D435 with an approximately 69.4° FoV is used to obtain human positions. We use YOLOv3 [47] for human detection and Deep SORT [48] for human tracking (our implementation is adopted from [49]). The human detection and tracking are combined with the camera depth information to calculate human positions. An Intel RealSense tracking camera T265 is used to localize the robot and obtain the robot orientation. We run the above perception algorithms and our decision-making model on a remote host computer. The communication between the robot and the host computer is established by ROS. A video demonstration is available at `https://youtu.be/bYO-1IAjzgY`, where the robot successfully reaches the goals with maneuvers to maintain a safe distance with humans in various scenarios.

VI. Conclusion and Future Work

We propose a novel DS-RNN network that incorporates spatial and temporal reasoning into robot decision making for crowd navigation. We train our DS-RNN with model-free deep RL without any supervised learning or assumptions on agents' dynamics. Our experiments shows that our model outperforms various baselines in challenging simulation environments and show promising results in the real world. Possible directions to explore in future work include (1) utilizing mutual interactions between the robot and humans to improve our model, and (2) enabling our network to take raw camera images as inputs to simplify detection and localization in the real world.

Acknowledgements

We thank Zhengguan Dai for setting up the baseline code and Zhe Huang for feedback on paper drafts.

References

[1] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[2] P. Du, Z. Huang, T. Liu, K. Xu, Q. Gao, H. Sibai, K. Driggs-Campbell, and S. Mitra, "Online monitoring for safe pedestrian-vehicle interactions," in *International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[4] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[5] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2616–2625.

[6] P. Chang, S. Liu, H. Chen, and K. Driggs-Campbell, "Robot sound interpretation: Combining sight and sound in learning-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[7] 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.

[8] Z. Huang, A. Hasan, and K. Driggs-Campbell, "Intention-aware residual bidirectional lstm for long-term pedestrian trajectory prediction," *arXiv:2007.00113*, 2020.

[9] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2255–2264.

[10] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer, "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 789–795.

[11] 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.

[12] 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.

[13] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6015–6022.

[14] 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.

[15] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[16] 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," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

[17] 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.

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

[19] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Robotics: science and systems (RSS)*, 2012.

[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] 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.

[22] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1343–1350.

[23] 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.

[24] 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.

[25] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1999, pp. 35–40.

[26] 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.

[27] 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.

[28] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[29] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

[30] 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.

[31] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative rnns and monte carlo tree search," in *International Conference on Robotics and Automation (ICRA)*, 2020.

[32] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5551–5557.

[33] K. Driggs-Campbell, V. Govindarajan, and R. Bajcsy, "Integrating intuitive driver models in autonomous planning for interactive maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3461–3472, 2017.

[34] K. Driggs-Campbell, R. Dong, and R. Bajcsy, "Robust, informative human-in-the-loop predictions via empirical reachable sets," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 300–309, 2018.

[35] 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.

[36] 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.

[37] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[38] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[39] L. Fan, W. Wang, S. Huang, X. Tang, and S.-C. Zhu, "Understanding human gaze communication by spatio-temporal graph reasoning," in *IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 5724–5733.

[40] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 670–681, 2018.

[41] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 300–311.

[42] 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.

[43] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[44] 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 (NIPS)*, 2017, pp. 5998–6008.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[46] I. Kostrikov, "Pytorch implementations of reinforcement learning algorithms," https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[47] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[48] 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)*. IEEE, 2017, pp. 3645–3649.

[49] Z. Pei, "Deep Sort with PyTorch," 2020. [Online]. Available: https://github.com/ZQPei/deep_sort_pytorch