sensors

MDPI

Article

Crowd-Aware Mobile Robot Navigation Based on Improved Decentralized Structured RNN via Deep Reinforcement Learning

Yulin Zhang and Zhengyong Feng *

School of Electronic Information Engineering, China West Normal University, Nanchong 637009, China
* Correspondence: zhyfeng@cwnu.edu.cn

Abstract: Efficient navigation in a socially compliant manner is an important and challenging task for robots working in dynamic dense crowd environments. With the development of artificial intelligence, deep reinforcement learning techniques have been widely used in the robot navigation. Previous model-free reinforcement learning methods only considered the interactions between robot and humans, not the interactions between humans and humans. To improve this, we propose a decentralized structured RNN network with coarse-grained local maps (LM-SRNN). It is capable of modeling not only Robot–Human interactions through spatio-temporal graphs, but also Human–Human interactions through coarse-grained local maps. Our model captures current crowd interactions and also records past interactions, which enables robots to plan safer paths. Experimental results show that our model is able to navigate efficiently in dense crowd environments, outperforming state-of-the-art methods.

Keywords: robot navigation; deep reinforcement learning; RNN; spatio-temporal graphs; coarse-grained local maps

Citation: Zhang, Y.; Feng, Z. Crowd-Aware Mobile Robot Navigation Based on Improved Decentralized Structured RNN via Deep Reinforcement Learning. *Sensors* **2023**, *23*, 1810. https://doi.org/10.3390/s23041810

Academic Editors: Nir Shvalb and Amir Shapiro

Received: 13 December 2022
Revised: 27 January 2023
Accepted: 3 February 2023
Published: 6 February 2023

1. Introduction

With the development of artificial intelligence, mobile robots can be seen everywhere in daily life, and how to efficiently navigate in the unstructured environment of human life will become more and more important. Traditional methods generally only consider static obstacles, which often lead to unsafe and unnatural behaviors in dynamic crowd environments [1,2]. To solve this problem, we should make robots obey the cooperation rules of humans [3]. For example, the robot should move smoothly and not cut through the crowd.

Navigation with social etiquette is a challenging task. Since there is generally no explicit communication between the robot and other agents, it is necessary for the robot to understand and predict the behavior of other agents. However, there are both moving agents and temporarily static agents in a crowded environment. There are various explicit and implicit interactions among agents, which are usually difficult to model [4].

There are two main types of methods to solve this problem in early research. The first type is the model-based method, which selects the best action through one-step interaction rules, such as optimal reciprocal collision avoidance (ORCA) [5] and social force model (SFM) [6]. The second type is based on prediction methods, by predicting the future trajectories of other agents, and then planning a best path for the robot, such as [7,8]. However, in dynamic dense scenes, these methods will cause the problem of robot freezing. Usually, feasible paths still exist in this scenario [9,10], but the difficulty of planning will be greatly increased.

In recent years, with the rapid development of machine learning, deep reinforcement learning methods have been widely used in crowd navigation [11–16]. Since the

deep reinforcement learning strategy can implicitly encode the interaction and cooperation among agents, the navigation performance has been greatly improved. Although recent studies have made significant progress, these strategies still have two shortcomings: (1) Initialized by ORCA through imitation learning, the strategy inherits the drawbacks of ORCA [13,14]; (2) Only the interactions between robot and humans is considered, and the interactions between humans and humans is not considered [17]; (3) Only low-density dynamic environments are considered, and the performance will drop in high-density dynamic environments [13,14,17]. Due to these deficiencies, these methods become less effective in more challenging settings.

In this work, we propose to build more comprehensive interactions rather than just first-order Robot–Human interactions. We model Robot–Human interactions through spatio-temporal graphs, and Human–Human interactions through local coarse-grained maps. These interactions are processed through an attention mechanism to learn the relative importance of surrounding humans, as well as the relative importance of Human–Human interactions. Our method is trained via model-free reinforcement learning, does not require imitation learning with an expert policy, and thus does not converge to a locally optimal policy. We conduct extensive experiments in dynamic dense crowd environments, and the experimental results show that our model performs better than previous methods in terms of navigation success rate and time efficiency.

2. Related Works

2.1. Model-Based Methods

Early works have accomplished navigation in dynamic environments by designing specific interaction models. One is a reaction-based method, such as RVO [18] and ORCA [5]. This type of method is based on the speed obstacle area method [19], and can obtain the joint collision avoidance speed under the assumption of reciprocity. The other is the social force model [6], which simulates the interactions among agents through goal driving force, repulsive force, and attractive force. These model-based methods rely heavily on hand-crafted functions, which may require different parameters in different scenarios, and finding a suitable parameter for dynamic-dense scenes is difficult [20].

2.2. Learning-Based Methods

With the rapid development of machine learning, learning-based methods have been extensively studied, which combine supervised learning and reinforcement learning, and have shown promising results in crowd navigation [11–14]. These works use simulated environments to collect robot navigation experiences, and then use these experiences to update value policies. These strategies are given the state transition probabilities of all agents, then calculate the value of all possible next states, and finally choose the action that maximizes the value. When training the value network, imitation learning is first performed, that is, the trajectory generated by ORCA is used to initialize the network parameters. The network parameters are then tuned using reinforcement learning. These methods must know the state transition probabilities of all humans, but state transitions of humans are usually difficult to model. Furthermore, since such methods employ imitation learning for initialization, they often inherit the shortcomings of the demonstration strategy.

2.3. Spatio-Temporal Graph Methods

In recent years, some works have represented the problem into components and their spatio-temporal interactions through spatio-temporal graphs, which have achieved good results in fields such as trajectory prediction and human tracking [21–23]. Jain et al. [24] converted arbitrary spatio-temporal graphs into RNN networks, proposing a method called structured RNN. Liu et al. [17] proposed a decentralized RNN network, which simulates the spatio-temporal interactions between the robot and other agents through the nodes and edges of the spatio-temporal graph, and achieved good results in crowd navigation. However, this method only focuses on the interactions between the robot and

other agents, ignoring the interactions among other agents. Based on these models, we design a new neural network that considers not only Robot–Human interactions, but also Human–Human interactions.

3. Approach

In this section, we first introduce how to represent the problem of crowd navigation with deep reinforcement learning. Then, the spatio-temporal graph is used to model the Robot–Human interaction, and the local coarse-grained map is used to model the Human–Human interaction. Finally, our LM-SRNN neural network structure is derived according to the previous interactive method.

3.1. Problem Formulation

In this work, we view crowd navigation as a partial Markov decision process. s_t and a_t represent the state and action of the agent (robot or human) at time t, respectively. The state representation of the agent is similar to [13,25], each agent has an observable state that can be observed by other agents, including position (p_x, p_y), velocity (v_x, v_y), and radius r; each agent also has a hidden state that can only be observed by itself, including the target position (g_x, g_y), maximum speed v_{max}, and heading angle θ. Our task is to deal with the problem of a robot navigating in a scene with n people. The input state of the robot is defined as $S_t = \{s_0^t, s_1^t, \ldots, s_n^t\}$, s_0^t is the entire state of the robot, s_i^t $(\mathrm{i} = 1, 2, \ldots, n)$ is the observable human state. The action of the agent consists of the speed of the x-axis and the speed of the y-axis, $a_t = (v_x, v_y)$.

In each episode, the robot starts from the initial state S_0. At time t, the robot chooses an action a_t according to the policy $\pi(a_t|S_t)$, and then moves to the next state S_{t+1}. In return, the robot gets a reward r_t. At the same time, humans also choose their own actions according to their own strategies and move to the next state. The whole process may last the maximum length of an episode, or it may end early due to a collision or reaching the goal.

We follow the reward function formulation in [17], rewarding the robot for task achievement while penalizing the robot for collisions with humans or getting too close to humans. Additionally, in order to guide the robot, the behavior of approaching the goal is rewarded:

$$\begin{cases} -20, & \text{if } \mathrm{d}_{\min}^{\mathrm{t}} < 0 \\ 2.5(\mathrm{d}_{\min}^{\mathrm{t}} - 0.25), & \text{if } 0 < \mathrm{d}_{\min}^{\mathrm{t}} < 0.25 \\ 10, & \text{if } \mathrm{d}_{goal}^{\mathrm{t}} < \mathrm{r}_{robot} \\ 2(\mathrm{d}_{goal}^{t-1} - \mathrm{d}_{goal}^{t}), & \text{otherwise} \end{cases}, \tag{1}$$

where $d_{\min}^t$ is the minimum distance between the robot and the human at time t, and d_{goal}^t is the Euclidean distance between the robot position and the target position at time t. The total return at time t is defined as:

$$R_t = \sum_{k=0}^{\infty} \gamma^{(k)} r_{t+k}, \tag{2}$$

where $\gamma \in (0, 1)$ is the discount factor. The value function is defined as $V(S_t) = \mathbb{E}(R_t|S_t = S)$.

3.2. Modeling of Robot–Human Interactions

We model the spatio-temporal interactions between robot and humans through a decentralized spatio-temporal graph. Our spatio-temporal graph is expressed as $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$, where $\mathcal{V}$ represents the node of the agent's own state information, $\mathcal{E}_S$ represents the spatial relationship edges of different agents at the same time step, and $\mathcal{E}_T$ represents the temporal relationship edges of the same agent at adjacent times step. Figure 1a shows an example spatio-temporal graph of a robot interacting with n humans in a complex scene. Since each individual's behavior is only influenced by nearby humans, rather than all humans, we prune the spatiotemporal edges between humans and model them in a more concise and effective way in Section 3.3. Figure 1b shows the same spatio-temporal graph unfolded in

two steps. In the unfolded spatio-temporal graph, robot nodes at a given time t are connected with other human nodes to form undirected spatial edges; robot nodes at adjacent time steps are connected to form undirected temporal edges.

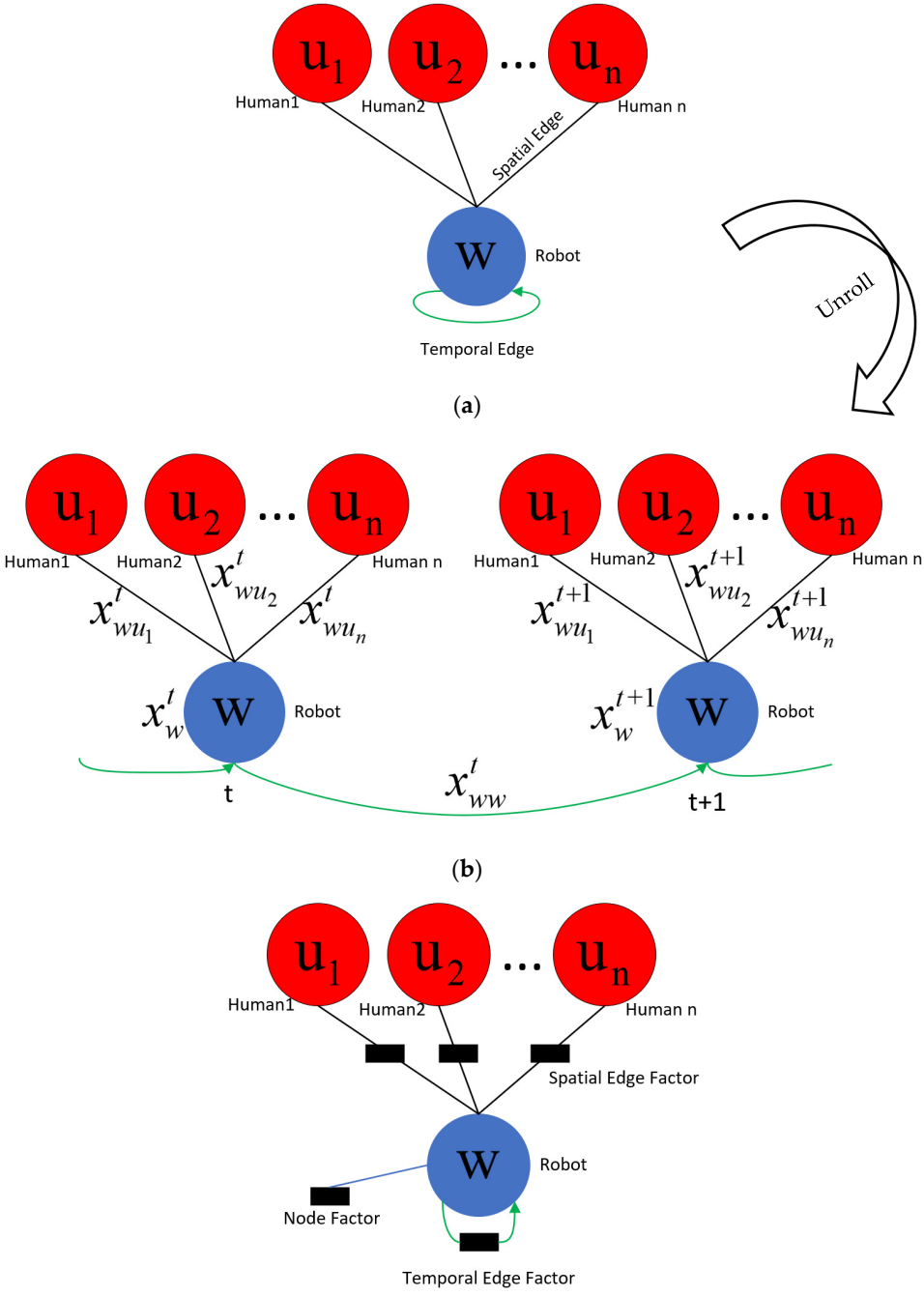

Figure 1. Spatio-temporal graph of robot interaction with humans during navigation. (**a**) A spatio-temporal graph capturing Robot–Human interactions. We use w to denote the robot node, and u_i to denote the i-th human node. (**b**) The spatio-temporal graph unfolded in two steps. At time step t, the spatial interaction feature between the robot and the *i*-th human is $x^t_{wu_i}$. The time feature of the robot is x^t_{ww}. The robot node feature is x^t_w. (**c**) The factor graph corresponding to the spatio-temporal graph.

In the interactions between robot and humans, node features can represent the state information of robot and humans, such as position, velocity, radius, target position, etc.

Edge features represent their relative direction and distance. The state of a node is affected by its own interactions with other nodes, forming a complex system. The interactions are usually parameterized by factor graphs, which can decompose complex problems in spatio-temporal graphs into simpler ones. As shown in Figure 1c, the one-way interactive robot navigation strategy is expressed as robot node factors, spatial edge factors, and temporal edge factors through a factor graph. The black rectangles in the figure represent factors with parameters that need to be learned.

More specifically, the $x_{\mathrm{w}}^{\mathrm{t}}$ is the robot state $[\mathrm{p_x}, \mathrm{p_y}, \mathrm{v_x}, \mathrm{v_y}, \mathrm{r}, \mathrm{g_x}, \mathrm{g_y}, \mathrm{v_{max}}, \theta]$, the feature $x_{\mathrm{ww}}^{\mathrm{t}}$ is the robot speed $(\mathrm{v_x}, \mathrm{v_y})$, and the feature $x_{\mathrm{wu}_i}^{\mathrm{t}}$ is represented by the relative positional relationship vector between the robot and the human $(p_x^i - p_x, p_y^i - p_y)$. Since all Robot–Human spatial edges are semantically similar, we let them share a factor, thereby reducing parameters.

3.3. Modeling of Human–Human Interactions

Everyone will have an impact on the robot's decision-making, but human behavior will be affected by other people. Therefore, it is extremely important to model the interaction among humans. If all the interactions among human are modeled in detail, the time complexity and space complexity of the algorithm will be very high, which is difficult to run in real-time in large-scale and complex scenes [26]. Inspired by study [14], we model interactions among humans through a local coarse-grained map.

As shown in Figure 2, for each human, we only consider other humans in the neighborhood of size N, and build an N * N * 3 local map L_i centered on each human i to encode the presence and speed of neighbors:

$$\begin{aligned} L_i(a,b) &= \sum_{j \in \mathcal{N}_i} \delta_{AB}[x_j - x_i, y_j - y_i][v_{xj}, v_{yj}, 1] \\ &= \begin{cases} \sum_{j \in \mathcal{N}_i} [v_{xj}, v_{yj}, 1], & if -A < x_j - x_i < A \; and \; -B < y_j - y_i < B, \\ -\infty, & otherwise \end{cases} \end{aligned} \tag{3}$$

where $[v_{xj}, v_{yj}, 1]$ is the local state vector of human j, $\delta_{AB}[x_j - x_i, y_j - y_i]$ is an indicator function, only when the relative position (Δx, Δy) is in the cell (A, B), the indicator function is equal to 1, and $\mathcal{N}_i$ is a group of adjacent humans around the i-th human.

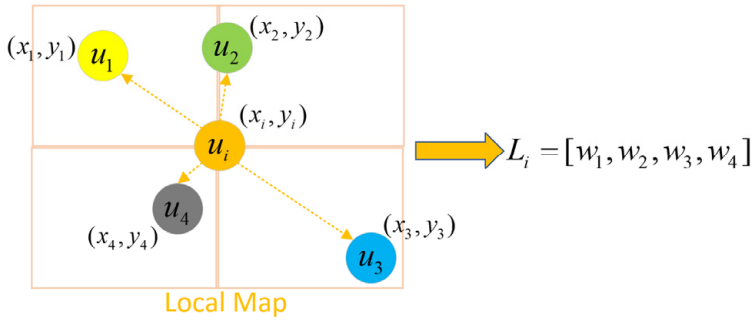

Figure 2. Local map representation of human i being influenced by other humans. where u_i represents a human, and (x_i, y_i) represents the position of human i.

3.4. Neural Network

According to study [24], the RNN network structure of Robot–Human interactions can be derived through the factor graph of the spatio-temporal graph. We process the interactions among human through an RNN network, so as to remember the previous interaction information. Our overall neural network structure is shown in Figure 3, R_N represents the robot node, R_S represents the spatial edge between the robot and the human, R_T represents the temporal edge of the robot trajectory change, and R_L is the node of Human–Human interactions around the robot.

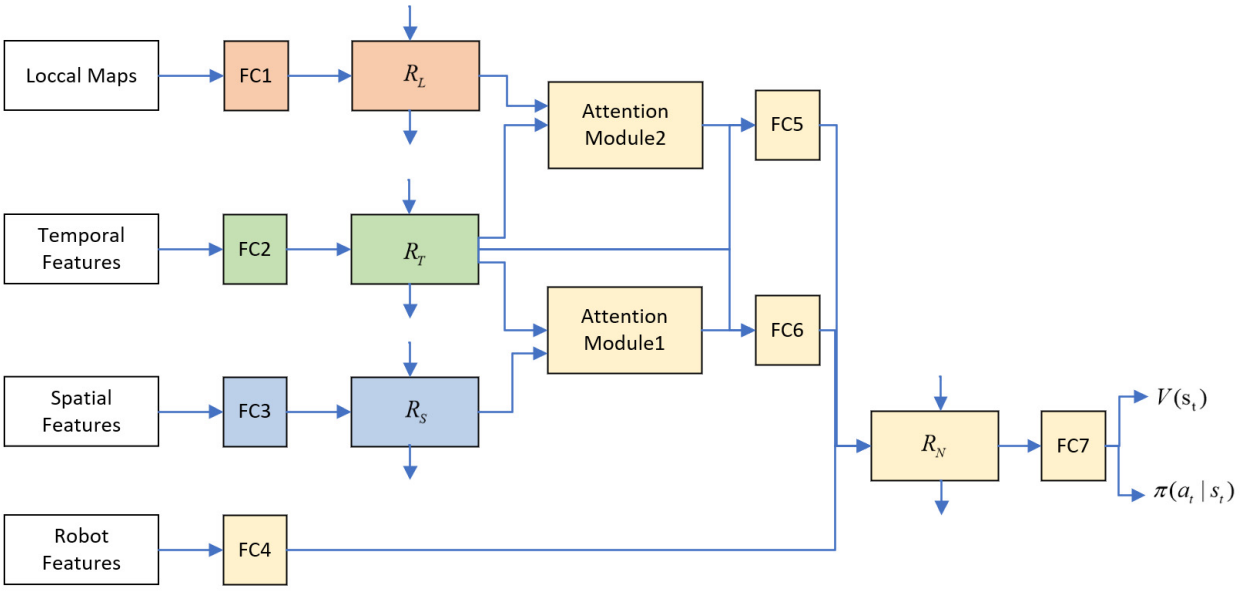

Figure 3. LM-SRNN neural network structure. FC* represents the fully connected layer.

Before the spatial features are provided to the spatial interaction RNN R_S, they will first be processed nonlinearly through a fully connected layer (FC3) to obtain the Robot–Human spatial interaction features at the current moment:

$$ES^{\mathrm{t}} = \psi_S(X^{\mathrm{t}}_{\mathrm{wu}}; W_S), \tag{4}$$

where $\psi_S(.)$ is a fully connected layer with ReLU activation and W_S is the network weight. $X^{\mathrm{t}}_{\mathrm{wu}}$ is the feature of spatial interaction between the robot and all other humans at time t, $X^{\mathrm{t}}_{\mathrm{wu}} = [x^{\mathrm{t}}_{\mathrm{wu}_1}, x^{\mathrm{t}}_{\mathrm{wu}_2}, \ldots, x^{\mathrm{t}}_{\mathrm{wu}_n}]^T$, where n is the total number of humans. Then the embedding vector ES^{t} is provided to the RNN unit to obtain the spatial interaction information with memory:

$$H^t_{wu} = RNN_S(H^{t-1}_{wu}, ES^t), \tag{5}$$

where $H^t_{wu} = [\mathrm{h}^{\mathrm{t}}_{\mathrm{wu}_1}, \mathrm{h}^{\mathrm{t}}_{\mathrm{wu}_2}, \ldots, \mathrm{h}^{\mathrm{t}}_{\mathrm{wu}_n}]$ is the hidden state of the RNN at time t for the spatial interaction between the robot and all humans.

Similar to R_S, the temporal features will also be processed nonlinearly (FC2) before being provided to the temporal interaction RNN R_T, and then processed by the RNN unit to obtain the robot's own trajectory variation:

$$h^t_{ww} = RNN_T(h^{t-1}_{ww}, ET^t), \quad ET^t = \psi_T(x^t_{ww}; W_T), \tag{6}$$

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

Compared with R_S and R_T, the Human–Human interaction RNN R_L contains information of all Human–Human interactions, which is very informative. First, we input the local coarse-grained map in Section 3.3 into a fully-connected layer (FC1), and then provide the results to RNN units to obtain Human–Human interaction information with memory:

$$H^t_L = RNN_L(H^{t-1}_L, LM^t), \ \mathrm{LM}^t = \psi_L(\mathrm{L}^t; W_L), \tag{7}$$

where $\mathrm{L}^t = [\mathrm{L}^t_1, \mathrm{L}^t_2, \ldots, \mathrm{L}^t_n]^T$ is the local coarse-grained map of all humans at time t, LM^t is the interaction data after nonlinear processing, and H^t_L is the hidden state of the RNN of Human–Human interaction features at time t.

To learn the relative importance of each human, we feed the output of the spatial interaction RNN and the temporal interaction RNN into an attention module. This module assigns attention weights to each spatial edge:

$$\alpha_S^t = softmax(\frac{n}{\sqrt{d}} H_{wu}^t W_1 (h_{ww}^t W_2)^T), \tag{8}$$

where d is the hyperparameter of attention size and W_1 and W_2 are the weight matrices of the linear transformation. The attention mechanism here is similar to the one in [27]. This attention module outputs a weighted sum of spatial edges:

$$HS_{attention}^t = (H_{wu}^t)^T \alpha_S^t, \tag{9}$$

To learn the relative importance of interactions among surrounding humans, we feed the output of the Human–Human interaction RNN and the temporal interaction RNN into another attention module. This module assigns attentional weights to each set of interaction features:

$$HL_{attention}^t = (H_L^t)^T \alpha_L^t, \tag{10}$$

The role of the robot node RNN R_N is to process all the previous information and determine the action and value of the robot at time t. First, $HS_{attention}^t$ and h_{ww}^t are embedded and connected (FC6), and the same processing is performed on $HL_{attention}^t$ and h_{ww}^t(FC5):

$$TS^t = \Phi_S(HS_{attention}^t, h_{ww}^t),\ \mathrm{TL}^t = \Phi_L(HL_{attention}^t, h_{ww}^t), \tag{11}$$

where $\Phi_S(.)$ and $\Phi_L(.)$ are the embedding function with ReLU activation. The result of the embedding is then connected to the robot node state, and the connection result is input to R_N:

$$h_w^t = RNN_N(h_w^{t-1}, [N^t, TS^t, TL^t]), \quad N^t = \psi_N(x_w^t; W_N), \tag{12}$$

where $\psi_N(.)$ is a fully connected layer with ReLU activation (FC4), and N^t is the result of a nonlinear transformation of the robot state.

Finally, h_w^t is fed into a fully connected layer (FC7) to obtain the value $V(S_t)$ and the policy $\pi(a_t|S_t)$. We train the entire model using the proximal policy optimization (PPO) algorithm proposed in [28], and the policy and value functions are continuously updated during the training process.

4. Experiments

4.1. Implementation Details

The local map of Human–Human interactions is a 4 × 4 grid centered on each human, and each cell has a side length of 1 m. The output dimensions of ψ_L, ψ_S, and ψ_T are (5, 64), (1, 64), and (5, 64), respectively. The dimensions of the hidden units of R_L, R_S, and R_T are (5, 256), (1, 256), and (5, 256). The output dimensions of Φ_S, Φ_L, and ψ_N are all (1, 64). The dimension of the final R_N hidden unit is (1, 192). Our strategy is implemented in PyTorch. The learning rate of reinforcement learning is 0.9, and the discount factor γ is set to 0.99. We train our policy on an NVIDIA GeForce RTX3080 for approximately 27,700 episodes.

We assume that after the robot takes an action at time t, it can always reach the target position at the next time t + 1, so the robot's position update is set to:

$$\begin{array}{l} p_x[t+1] = p_x[t] + v_x[t]\Delta t \\ p_y[t+1] = p_y[t] + v_y[t]\Delta t \end{array}, \tag{13}$$

4.2. Simulation Setup

Our simulation environment is similar to [14,17], where the robot navigates a scene with a radius of 6 m. Since there are too few dynamic humans in [14] and the static human group in [17] is indistinguishable from static obstacles, their experimental results cannot

reflect the performance of the algorithm in complex environments. In order to verify the performance of the algorithm in more complex environments, we increase the difficulty of the experiment. We added more dynamic humans in the environments. The simulated humans in the environment are controlled by ORCA, and the speed and radius of humans are random. The initial positions of all humans are randomly distributed on the circle, and their target position is on the other side of the circle. However, humans occasionally change their goals. In addition, we also added random perturbations to the start position and target position. Finally, when humans reach their goal, they do not remain stationary, but immediately proceed to a new target. These processes are to simulate the environment closer to the real environment.

We chose ORCA [5], SARL [14] and DSRNN [17] as the baseline methods, and our method is called LM-SRNN. When training was performed for SARL and DSRNN, we only modified the simulation environment, and other parameters are the same as in the original paper. In order to eliminate the performance gain brought by other factors, we remove the Human–Human interaction module in our model and implement an ablation model called ST-RNN.

To comprehensively evaluate our model, we set up three sets of simulation experiments: the first set is an experiment in which the robot is invisible, that is, the simulated human only responds to other humans and does not respond to the robot, and there are 10 dynamic humans in the environment. The second set is an experiment where the robot is visible, in which the robot and humans interact with each other, closer to reality. As in the first set of experiments, there are 10 dynamic humans in the environment. The third set is a high-density environment. There are 20 humans in the environment, including both dynamic humans and static humans. The ratio of dynamic humans to static humans is randomly determined, and all humans will not respond to the robot. In all three sets of experiments, each method is tested using 500 random examples.

4.3. Quantitative Comparison

4.3.1. Robot Invisibility

Navigation in an invisible environment is difficult because the robot needs to predict human behavioral trajectories in order to plan a safe path. In order to comprehensively evaluate the performance of the model, the test indicators include not only navigation success rate, navigation collision rate, and navigation time, but also discomfort frequency and minimum danger distance. Discomfort frequency refers to the percentage of the duration when the robot is too close to other humans to the total navigation time, which can reflect the frequency of the robot violating the comfort zone of other humans. The minimum distance refers to the minimum distance between the outer edge of the robot and the outer edge of other humans, which can reflect the degree of danger of the planned behavior. These two metrics can further evaluate the performance of navigation strategies in crowd navigation. Table 1 reports the experimental results in the invisible environment.

Table 1. Experimental results in an environment where the robot is invisible. "Success" is the success rate of robot navigation. "Collision" is the collision rate for robot navigation. "Time" is the time the robot navigates. "Discomfort" is the discomfort frequency (the percentage of the duration when the robot is too close to other humans to the total navigation time). "Min Distance" is the minimum danger distance between the robot and other humans. Bold indicates the best performing model on that metric.

Method	Success	Collision	Time	Discomfort	Min Distance
ORCA [5]	0.34	0.66	15.29	0.25	0.08
SARL [14]	0.89	0.11	16.65	0.05	0.14
DSRNN [17]	0.96	0.04	18.75	0.06	0.19
ST-RNN	0.95	0.05	18.95	0.06	0.18
LM-SRNN (Ours)	**0.99**	**0.01**	16.43	**0.02**	**0.20**

From the experimental results, it can be found that the model-based ORCA fails badly. The reason is that in an invisible environment, humans will not avoid robots, which does not meet ORCA's reciprocity assumption. In addition, ORCAs often violate the comfort zone of other humans, and the level of danger in planning behavior is high. The reason is that ORCA only considers the current state and makes short-sighted decisions. In contrast, our LM-SRNN is trained by reinforcement learning method, and the previous series of trajectories are memorized by RNN, so the decisions made are far-sighted.

Compared with learning-based SARL, our method has a significantly higher success rate and plans a less dangerous behavior. The reason is that although SARL considers the interaction among agents, it is initialized through ORCA, which inherits the shortcomings of ORCA, and its value network is not enough to provide a good state value estimate. In contrast, our LM-SRNN is trained from scratch and does not converge to a locally optimal policy. Furthermore, our model derives Robot–Human interactions through spatio-temporal graphs and models Human–Human interactions through local coarse-grained maps, which is more efficient than simple joint modeling in SARL.

Compared with DSRNN, our method has a higher success rate, significantly shorter navigation time, and significantly lower discomfort frequency. The reason is that our method considers the impact of Human–Human interactions on robot navigation, thereby planning a more reasonable path. For the difference in path planning, we will conduct a qualitative analysis in Section 4.4.

4.3.2. Robot Visible

To further validate the performance of our model, we compared it with the baseline approach in the robot visible setting. This is because the robot not only needs to understand human behavior, but also to plan a reasonable trajectory when interacting with humans. Table 2 reports the experimental results in the visible environment.

Table 2. Experimental results in the visible environment of the robot. Bold indicates the best performing model on that metric.

Method	Success	Collision	Time	Discomfort	Min Distance
ORCA [5]	0.87	0.13	14.32	0.26	0.07
SARL [14]	0.98	0.02	14.31	0.02	0.15
DSRNN [17]	0.99	0.01	12.01	0.01	0.21
ST-RNN (Ours)	0.99	0.01	12.87	0.02	0.19
LM-SRNN (Ours)	**1.00**	**0.00**	**11.97**	**0.00**	**0.23**

Unlike the invisible environment, the navigation success rate is greatly improved when ORCA is employed for navigation in the visible environment, which is due to the fact that this environment conforms to the reciprocity assumption of ORCA. However, since ORCA plans trajectories that are short-sighted and conservative, it still performs significantly worse than the reinforcement learning-based approach.

The performance of the reinforcement learning-based methods has been improved in the visible environment. The reason is that humans will also avoid the robot in the visible environment, and the difficulty of navigation is reduced. As with the invisible environment, our LM-SRNN still outperforms the other baseline methods.

4.3.3. High-Density Environment

To further verify the performance of our model in high-density crowd environments, we increase the number of humans from 10 to 20. Since there is a timeout in navigation in such a complex environment, we introduce the indicator "Timeout", which indicates the timeout rate of robot navigation. Table 3 shows the experimental results in a high-density environment.

Table 3. Experimental results in a high-density environment. "Timeout" is the timeout rate for robot navigation. Bold indicates the best performing model on that metric.

Method	Success	Collision	Timeout	Time	Min Distance
ORCA [5]	0.45	0.01	0.54	19.66	0.05
SARL [14]	0.35	0.05	0.60	29.24	0.12
DSRNN [17]	0.94	0.02	0.04	17.71	0.18
ST-RNN (Ours)	0.93	0.03	0.04	18.01	0.19
LM-SRNN (Ours)	**0.98**	**0.01**	**0.01**	**16.65**	**0.20**

As shown in Table 3, ORCA and SARL have higher timeout rates and longer navigation times in high-density environments. The reason is that the strategies adopted by ORCA and SARL initialized through ORCA are too conservative, and they will often fall into the state of robot freezing, resulting in long navigation time. Same as the robot-invisible environment and robot-visible environment, our LM-SRNN still outperforms DSRNN.

4.3.4. Model Effectiveness Analysis

Furthermore, we demonstrate the effectiveness of our LM-SRNN by comparing with the ablation model ST-RNN. As shown in Tables 1–3, compared with ST-RNN, our model shows higher success rate and shorter navigation time in all environments. This is because our LM-SRNN not only considers the influence of each person's behavior on the robot's decision-making, but also further considers the influence of each person's behavior by other people, so as to plan a more reasonable path.

4.4. Qualitative Comparison

To further study the performance of the model, we conduct a qualitative analysis. We compare the navigation paths of different methods in the invisible environment, as shown in Figure 4. For the convenience of observation, we visualize the radius of all agents as 0.3 m. ORCA will choose the shortest path, directly into the central congested area, and eventually collide with humans. Although SARL can avoid other humans in time and successfully reach the target, it inherits the shortcomings of ORCA and will also enter the central congested area. This can lead to planning dangerous behaviors that easily violate human comfort zones. Unlike SARL, DSRNN is able to avoid centrally congested areas, but when it encounters humans, it swerves violently to pass them. Due to the inappropriate timing of avoidance and the unreasonable path chosen, the final navigation time of DSRNN is very long. In contrast, our LM-SRNN is not only able to avoid centrally congested regions, but also plans a smooth intelligent path.

To take a closer look at our strategy, we analyzed how the robot was making a decision at a certain moment. As shown in Figure 5, due to the influence of human #2 and human #6, human #7 will suddenly move to the upper right. In this case, our LM-SRNN does not continue to move towards the target (the black dotted line in the figure). Instead, it moves upwards, so it does not violate the comfort zone of the human #7 and complies with social etiquette navigation norms.

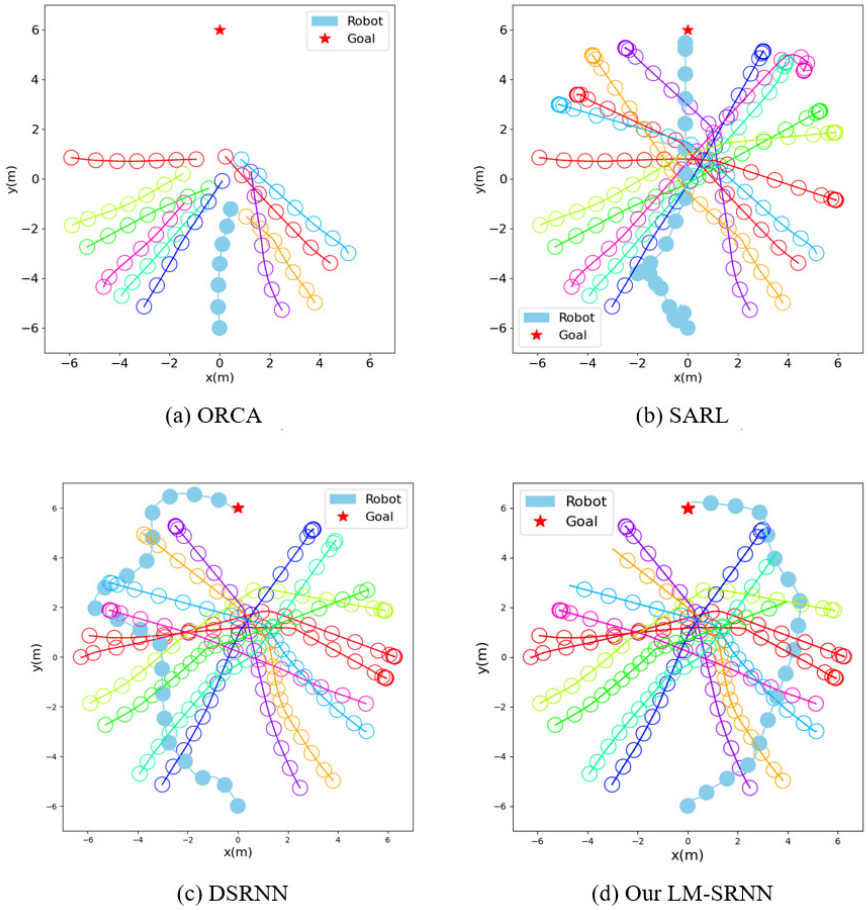

Figure 4. Comparison of trajectories in invisible environments. The blue solid circles in the figure represent robots, and the other hollow circles represent humans.

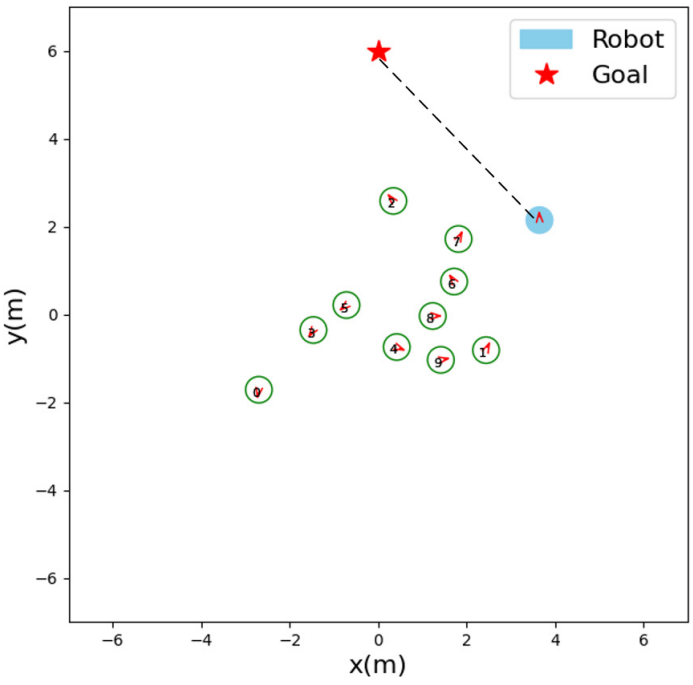

Figure 5. Screenshot of a moment in the navigation process. The red arrows indicate the direction of motion of the robot or human.

5. Conclusions

In this work, we propose a novel LM-SRNN neural network that combines spatio-temporal maps and crowd local interaction maps in robot navigation. We build Robot–Human interactions through spatio-temporal graphs, and Human–Human interactions through local coarse-grained maps. Experimental results show that our method performs better in terms of navigation success rate and time efficiency in denser environments than the baseline method.

Author Contributions: Conceptualization, Y.Z. and Z.F.; methodology, Y.Z.; software, Y.Z.; validation, Y.Z. and Z.F.; formal analysis, Y.Z.; investigation, Y.Z.; resources, Z.F.; data curation, Y.Z.; writing—original draft preparation, Y.Z.; writing—review and editing, Z.F.; visualization, Y.Z.; supervision, Z.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the China West Normal University Talent Fund (no. 17YC046).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Borenstein, J.; Koren, Y. Real-Time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments. In Proceedings of the IEEE International Conference on Robotics and Automation, Cincinnati, OH, USA, 13–18 May 1990; IEEE Computer Society Press: Washington, DC, USA, 1990; pp. 572–577.
2. Fox, D.; Burgard, W.; Thrun, S. The Dynamic Window Approach to Collision Avoidance. *IEEE Robot. Autom. Mag.* **1997**, *4*, 23–33. [CrossRef]
3. Kruse, T.; Pandey, A.K.; Alami, R.; Kirsch, A. Human-Aware Robot Navigation: A Survey. *Robot. Auton. Syst.* **2013**, *61*, 1726–1743. [CrossRef]
4. Gupta, A.; Johnson, J.; Fei-Fei, L.; Savarese, S.; Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, IEEE, Salt Lake City, UT, USA, 18–22 June 2018; pp. 2255–2264.
5. van den Berg, J.; Guy, S.J.; Lin, M.; Manocha, D. Reciprocal N-Body Collision Avoidance. In *Robotics Research*; Pradalier, C., Siegwart, R., Hirzinger, G., Eds.; Springer Tracts in Advanced Robotics; Springer: Berlin, Heidelberg, 2011; Volume 70, pp. 3–19. ISBN 978-3-642-19456-6.
6. Helbing, D.; Molnár, P. Social Force Model for Pedestrian Dynamics. *Phys. Rev. E* **1995**, *51*, 4282–4286. [CrossRef] [PubMed]
7. Kuderer, M.; Kretzschmar, H.; Sprunk, C.; Burgard, W. Feature-Based Prediction of Trajectories for Socially Compliant Navigation. In *Robotics: Science and Systems VIII*; MIT Press: Cambridge, MA, USA, 2013; pp. 193–200.
8. Kretzschmar, H.; Spies, M.; Sprunk, C.; Burgard, W. Socially Compliant Mobile Robot Navigation via Inverse Reinforcement Learning. *Int. J. Robot. Res.* **2016**, *35*, 1289–1307. [CrossRef]
9. Trautman, P.; Krause, A. Unfreezing the Robot: Navigation in Dense, Interacting Crowds. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Taipei, Taiwan, 18–22 October 2010; pp. 797–803.
10. Fan, T.; Cheng, X.; Pan, J.; Long, P.; Liu, W.; Yang, R.; Manocha, D. Getting Robots Unfrozen and Unlost in Dense Pedestrian Crowds. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1178–1185. [CrossRef]
11. Chen, Y.F.; Liu, M.; Everett, M.; How, J.P. Decentralized Non-Communicating Multiagent Collision Avoidance with Deep Rein-forcement Learning. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, Singapore, 29 May–3 June 2017; pp. 285–292.
12. Chen, Y.F.; Everett, M.; Liu, M.; How, J.P. Socially Aware Motion Planning with Deep Reinforcement Learning. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Vancouver, BC, Canada, 24–28 September 2017; pp. 1343–1350.
13. Chen, C.; Hu, S.; Nikdel, P.; Mori, G.; Savva, M. Relational Graph Learning for Crowd Navigation. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Las Vegas, NV, USA, 24 October–24 January 2020; pp. 10007–10013.
14. Chen, C.; Liu, Y.; Kreiss, S.; Alahi, A. Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), IEEE, Montreal, QC, Canada, 20–24 May 2019; pp. 6015–6022.

15. Everett, M.; Chen, Y.F.; How, J.P. Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Madrid, Spain, 1–5 October 2018; pp. 3052–3059.
16. Chen, Y.; Liu, C.; Shi, B.E.; Liu, M. Robot Navigation in Crowds by Graph Convolutional Networks With Attention Learned From Human Gaze. *IEEE Robot. Autom. Lett.* **2020**, *5*, 2754–2761. [CrossRef]
17. Liu, S.; Chang, P.; Liang, W.; Chakraborty, N.; Driggs-Campbell, K. Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May 2021–5 June 2021; pp. 3517–3524.
18. van den Berg, J.; Lin, M.; Manocha, D. Reciprocal Velocity Obstacles for Real-Time Multi-Agent Navigation. In Proceedings of the 2008 IEEE International Conference on Robotics and Automation, IEEE, Pasadena, CA, USA, 19–23 May 2008; pp. 1928–1935.
19. Fiorini, P.; Shiller, Z. Motion Planning in Dynamic Environments Using Velocity Obstacles. *Int. J. Robot. Res.* **1998**, *17*, 760–772. [CrossRef]
20. Long, P.; Liu, W.; Pan, J. Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation. *IEEE Robot. Autom. Lett.* **2017**, *2*, 656–663. [CrossRef]
21. Sadeghian, A.; Alahi, A.; Savarese, S. Tracking the Untrackable: Learning to Track Multiple Cues with Long-Term Dependencies. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, Venice, Italy, 22–29 October 2017; pp. 300–311.
22. Khodayar, M.; Wang, J. Spatio-Temporal Graph Deep Neural Network for Short-Term Wind Speed Forecasting. *IEEE Trans. Sustain. Energy* **2019**, *10*, 670–681. [CrossRef]
23. Yu, B.; Yin, H.; Zhu, Z. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 13–19 July 2018; pp. 3634–3640.
24. Jain, A.; Zamir, A.R.; Savarese, S.; Saxena, A. Structural-RNN: Deep Learning on Spatio-Temporal Graphs. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, Las Vegas, NV, USA, 27–30 June 2016; pp. 5308–5317.
25. Matsuzaki, S.; Hasegawa, Y. Learning Crowd-Aware Robot Navigation from Challenging Environments via Distributed Deep Reinforcement Learning. In Proceedings of the 2022 International Conference on Robotics and Automation (ICRA), IEEE, Phila-delphia, PA, USA, 23–27 May 2022; pp. 4730–4736.
26. Vemula, A.; Muelling, K.; Oh, J. Social Attention: Modeling Attention in Human Crowds. In Proceedings of the 2018 IEEE In-ternational Conference on Robotics and Automation (ICRA), IEEE, Brisbane, QLD, Australia, 21–25 May 2018; pp. 4601–4607.
27. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. In Proceedings of the Advances in Neural Information Processing Systems (NIPS), Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
28. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.