Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 7114981, 11 pages
https://doi.org/10.1155/2021/7114981

Research Article

Robot Navigation in Crowd Based on Dual Social Attention Deep Reinforcement Learning

Hui Zeng, Rong Hu, Xiaohui Huang, and Zhiying Peng

Department of Information Engineering, East China Jiaotong University, Nanchang 330013, China

Correspondence should be addressed to Rong Hu; hrdyx971013@163.com

Received 31 May 2021; Revised 18 August 2021; Accepted 9 September 2021; Published 24 September 2021

Academic Editor: Martin Velasco Villa

Finding a feasible, collision-free path in line with social activities is an important and challenging task for robots working in dense crowds. In recent years, many studies have used deep reinforcement learning techniques to solve this problem. In particular, it is necessary to find an efficient path in a short time which often requires predicting the interaction with neighboring agents. However, as the crowd grows and the scene becomes more and more complex, researchers usually simplify the problem to a one-way human-robot interaction problem. But, in fact, we have to consider not only the interaction between humans and robots but also the influence of human-human interactions on the movement trajectory of the robot. Therefore, this article proposes a method based on deep reinforcement learning to enable the robot to avoid obstacles in the crowd and navigate smoothly from the starting point to the target point. We use a dual social attention mechanism to jointly model human-robot and human-human interaction. All sorts of experiments demonstrate that our model can make robots navigate in dense crowds more efficiently compared with other algorithms.

1. Introduction

In the context of the rapid development of technique, the robot has expanded from an isolated work environment to a shared social space for cooperation with humans. As a basic subject of robotics, mobile robot navigation has been extensively studied. In the old days, traditional mobile robot navigation usually regarded other agents as static obstacles and only judged the next action when they walked in front of the agent. Not only did this cause a lot of safety problems, but also the navigation efficiency was very low. In recent years, due to the growing aging of the population and the increasingly expensive labor force, more service robots have been developed to work in a social environment [1]. In order for robots to navigate more efficiently and normatively in dense crowds, researchers need to follow the principles of social interaction between humans [2–5].

One of the most basic abilities of a robot to navigate a crowd is to avoid collisions. Humans have an innate ability to observe and adjust their behavior through their own observations, so we can easily pass through people safely. However, robots need to perceive and predict human behavior, which involves more complex techniques such as human-computer interaction. Researchers have proposed some manual calculation or data-driven methods for robot trajectory prediction [6, 7] to obtain interactive information between humans and machines. But the collision-free robot navigation in crowds is still a daunting task.

Early methods usually divide prediction and planning into two steps. After predicting the future trajectories of other people, the method intends a complete safe path without collision for a robot. These models first learn human motion patterns to predict the motion of other people and then plan the robot's path accordingly [8, 9]. But, in dense crowds, the predicted trajectories of people will spread across the entire space, so seeking a safe path that does not collide with humans in the crowd is tough. In this case, the robot using the above method may be blocked in the crowd, which is the problem of robot freezing, and the navigation time will be infinitely extended. In response to this problem, the researchers plan to treat humans and robots as a whole joint planning path [10, 11].

Most of the current researches use deep learning to extract the characteristics of human-computer interaction and use reinforcement learning framework training strategies to make the robot interact with the crowd in a "trial-and-error" manner and obtain the greatest reward. That is, the robot learns from experience to understand and adapt to a crowd of people and encodes the interaction between the crowd and the robot in the navigation strategy. Recent work in this field [12, 13] has made significant progress, but the existing model still has some shortcomings: (a) most methods only consider the impact of a one-way person on the robot, while ignoring the fact that the interaction between people will also affect the behavior of the robot; (b) the feature extraction module is relatively simple, resulting in the value of the incoming reinforcement learning being not accurate enough to avoid collisions. As the number of people increases, the collision rate of robots during movement is higher; (c) the reward is not clear enough, and the robot fails to respond in a short time.

In order to solve the above problems, it is proposed that the dual social attention reinforcement learning model integrates the perceived state characteristics of humans, robots, and human-human interactions. Then this model obtains accurate values through dual attention modules; and this model can handle any number of agents and encode them into a fixed-length value. To make navigation more efficient, we improve the existing reward function. The contributions of this paper are as follows:

(i) We develop a dual social attention reinforcement learning algorithm to predict human dynamics and make the robot navigate the crowd efficiently.

(ii) We design a new reward function to make robot navigation more efficient.

(iii) We evaluate our proposed DSARL algorithm in a simulated environment and compare this algorithm with four other navigation algorithms. Compared with other models, the performance of this model is better.

The rest of this article is organized as follows. The Section 2 introduces the related work of this article. Then, the Section 3 explains the definition of the problem. The Section 4 describes the details of the model DSARL proposed in this article, followed by the simulation experiment in Section 5 and the final conclusion in Section 6.

2. Related Works

2.1. Traditional Methods. In order to navigate in a crowd in a manner that meets social requirements, robots need the ability to perceive, understand, and predict the behavior of surrounding pedestrians and adjust their actions in time. In the early work, the main idea to solve this problem was to predict the trajectory of other people first and then plan a safe road. Some researchers use hand-made methods. They build a discrete model [14] or a statistical model [15] to predict the trajectory of other pedestrians and then choose the next location in the space around the person. In order to enhance the social consciousness of robots in human-computer interaction, researchers have created social forces [16, 17] to enable robots to better understand crowd movements. The social power model they built combined with predictive information can detect and locate the behavior of people in the crowd and obtain an interactive solution for the robot's perception and navigation in the crowd. The social power model has been applied to robots in real life.

There are also some researchers who use data-driven methods to predict pedestrian trajectories. They build an LSTM [6, 18] model to learn general human motion and predict the future trajectories [19]. Using this model, the agent can predict the pedestrian's position on the road and adjust its direction accordingly to avoid the collision; that is, the agent considers the influence of the social neighborhood and the scene layout to adjust itself accordingly.

However, the behavior of people in dense crowds is more complicated and random, so the work of predicting pedestrian trajectories will be more difficult and computationally expensive. In the circumstances of human-perceived navigation of mobile robots that require safety and time efficiency, this method of predicting and planning is still challenging in practical applications.

2.2. Deep Reinforcement Learning Methods. At present, deep reinforcement learning methods are applied in many fields including the field of robot perception and navigation. Unlike the traditional methods, the deep reinforcement learning methods combine the tracking and prediction of pedestrian route planning robot navigation together. That is, the robot learns from the experience of pedestrian movement, understands the crowded scene, and encodes the human-computer interaction characteristics into the navigation strategy. The results in this field [12, 20, 21] have confirmed the superior performance of deep reinforcement learning in crowd perception navigation. These deep reinforcement learning methods first collect relevant data from the surrounding people to build a value network, train in a reinforcement learning framework, and finally map the information to the control commands of the robot.

According to the above model, we combined the social LSTM method [6] and the social force model [22] to design a dual social attention reinforcement learning model to catch importance of each neighbour on the robot and code group cooperative behavior into this model.

3. Problem Formulation

In the entire navigation task, we define the problem as a robot moving from the start point to the endpoint through n humans. We first need to perceive the information of the nearby environment and the state of the robot itself. We adopt the method of taking the robot as the center [12, 23] so that the robot is located at the origin, and the X-axis points to the target of the robot.

Suppose that, for each agent (robot or human), other agents can observe position $\mathbf{p} = [p_x, p_y]$, velocity $\mathbf{v} = [v_x, v_y]$, and radius r. For the robot, there are some states that other agents cannot observe, including the preferred speed v_{pref} and its distance to the target point $d_g = \|\mathbf{p} - \mathbf{p}_g\|_2$. $\mathbf{p}_g$ is the position of the target point, so the state vector of the robot is defined as $s = [d_g, v_{\text{pref}}, v_x, v_y, r]$. Similarly, the human state vector h_i also includes position, speed, and radius, as well as the distance $d_i = \|\mathbf{p} - \mathbf{p}_g\|_2$ from the robot to the current human i. Therefore, the human state vector $h_i = [p_x, p_y, v_x, v_y, r_i, r_i + r]$. The interaction between people is represented by a local map M_i [24], which encodes the existence and speed of the surrounding people by constructing a $L \times L \times 3$ map tensor with each person i as the center:

$$M_i(m, n, :) = \sum_{j \in N_i} \delta_{mn}\left[x_j - x_i,\ y_j - y_i\right] h'_j, \tag{1}$$

where $h_j t$ is a local state vector of human j, $\delta_{mn}[x_j - x_i, y_j - y_i]$ is an indicator function that equals 1 only if the relative position $(\Delta x, \Delta y)$ is located in the cell (m, n), and N_i is a series of nearby humans of the i^{th} person.

Then, at time t, the robot's state is $\mathbf{s}_t$, the human's state is $\mathbf{h}_t = [\mathbf{h}_t^1, \mathbf{h}_t^2, \mathbf{h}_t^3, \ldots, \mathbf{h}_t^n]$, and the human-human interaction is $\mathbf{M}_t$. Then we define the joint state of robot navigation as $\mathbf{j}_t = [s_t, h_t, M_t]$. We assume that the speed of the robot changes depending on the action command, and the action command is determined by the navigation strategy, that is, $\mathbf{v}_t = \mathbf{a}_t = \pi(\mathbf{j}_t)$.

According to the joint state and action at time t to obtain the corresponding reward $R_t(\mathbf{j}_t, \mathbf{a}_t)$, we set the following reward function:

$$R_t(j_t, a_t) = \begin{cases} -0.25, & \text{if } d_{\min} < 0, \\ -0.01 + \dfrac{d_{\min}}{20}, & \text{else if } d_{\min} < 0.2, \\ 1, & \text{else if } p_t = p_g, \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $d_{\min}$ is the minimum separation distance between the robot and the human in the time period $[t - \Delta t, t]$. We assume that the current position of the robot is set to $\mathbf{P}_{\text{robot}} = [p_{x1}, p_{y1}]$ in the time period $[t - \Delta t, t]$, and the radius of the robot is r_{robot}. The position of the human is $\mathbf{p}_{\text{human}} = [p_{x1}, p_{y1}]$ and the radius of the human is r_{human}; and $d_{\min}$ is defined as $d_{\min} = \|\mathbf{p}_{\text{robot}} - \mathbf{p}_{\text{human}}\|_2 - (r_{\text{robot}} + r_{\text{human}})$. As shown in Figure 1, when $d_{\min}$ is less than 0, the robot collides with the human. When $d_{\min}$ is located at $[0, 0.2)$, the robot is too close to the human, causing human discomfort. For the above two situations, we will punish the robot. If the robot successfully reaches the target point without collision, it will be rewarded.

We hope that the optimal strategy $\pi^*(\mathbf{j}_t)$ can maximize the reward:

$$\begin{aligned} \pi^*(\mathbf{j}_t) = &\operatorname*{argmax}_{a_t} R(\mathbf{j}_t, \mathbf{a}_t) \\ &+ \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{j}_{t+\Delta t}} P(\mathbf{j}_t, \mathbf{a}_t, \mathbf{j}_{t+\Delta t}) V^*(\mathbf{j}_{t+\Delta t}) d\mathbf{j}_{t+\Delta t}. \end{aligned} \tag{3}$$

Among them, $R(\mathbf{j}_t, \mathbf{a}_t)$ is the reward obtained at time t, and v_{pref} is the preferred speed of the robot. $\gamma \in (0, 1)$ is the discount factor, and the preferred speed is introduced as the standardized item in the discount factor. Δt is the decision interval between two actions, and $P(\mathbf{j}_t, \mathbf{a}_t, \mathbf{j}_{t+\Delta t})$ is the conversion probability from time t to Δt.

V^* is the optimal value of the joint state $\mathbf{j}_t$ at time t as follows:

$$V^*(\mathbf{j}_t) = \sum_{k=0}^{K} \gamma^{k \cdot \Delta t \cdot v_{\text{pref}}} \cdot R(\mathbf{j}_t, \mathbf{a}_t^*), \tag{4}$$

where K refers to the entire number of decisions that the robot will make from the current state at time t to the last state.

4. Approach

When a robot moves in a dense crowd, the robot needs to not only avoid every human that may collide with it but also avoid being affected by human-to-human social interaction when walking. Robot navigation in a crowd needs to adjust its trajectory according to the movements of other people around. Therefore, we designed a dual social attention reinforcement learning model with three major modules to complete the entire navigation task, as shown in Figure 2.

Moreover, because different humans have different influences on the current state of the robot's location, the model needs to count the corresponding importance and encrypt the socially compliant navigation of the neighborhood's collective influence. This model is mainly composed of the three following modules:

(i) Perception module: integrates the state of the robot, humans' states in the crowd, and the interaction between people

(ii) Attention module: makes use of the dual attention mechanism to assign weights to the fixed-length vectors obtained by the interactive module, suppress some influential factors, and obtain more accurate state-input vectors

(iii) Planning module: processes the vector obtained by the attention module by a multilayer perceptron (MLP) and then gets the final value

Next, we introduce the structure and formula of the following modules in detail. Since the states discussed below are all at time t, we have omitted the time index t.

4.1. Perception Module

4.1. Perception Module. In the crowd, everyone has an influence on the robot and is also influenced by the human around. In dense scenes, it is necessary to consider not only the impact of all people on the robot but also the impact of human interaction on the robot. The data is huge and difficult to handle, so we designed a pair of interactive modules as shown in Figure 3. First, it collects the current state of the robot, then observes the interaction state of each pair of people and the robot, and finally uses the local map to represent the interaction between people. The perception module integrates the above three states to form a new unprocessed state input.

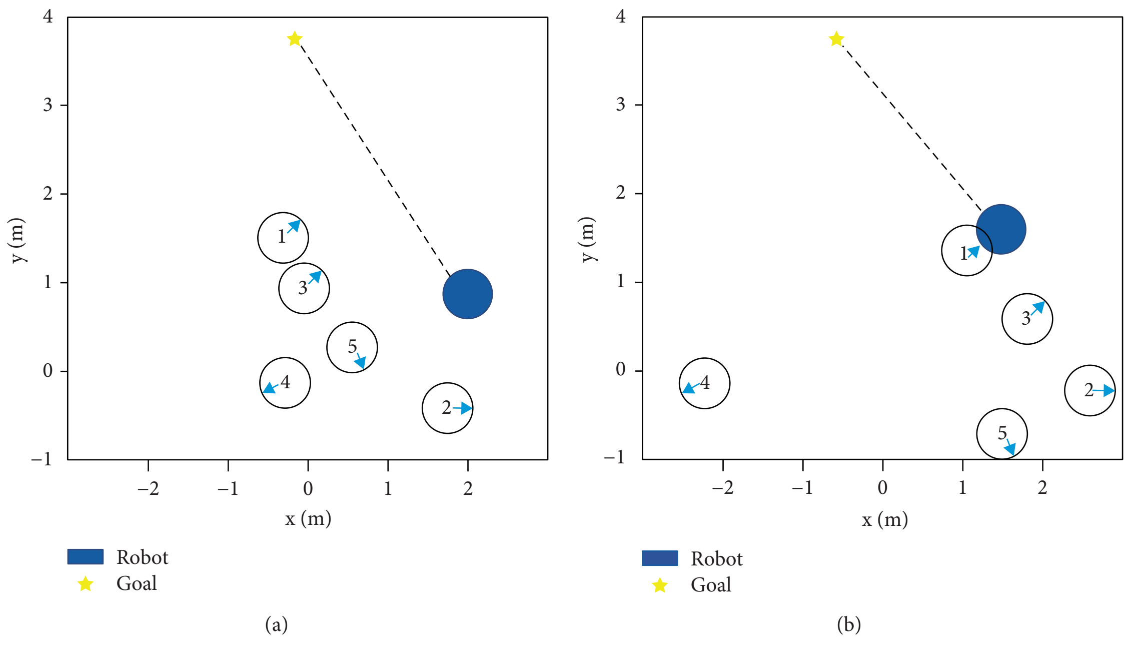


FIGURE 1: Collision between robot and human. When the robot collides with human 1, their minimum separation distance is less than 0.

The robot's state s, human's state h_i, and human-human interaction M_i are defined in the Problem Formulation section. After obtaining these three states, we use a multilayer perceptron to merge the robot's state s, the human's state h_i, and the human-human interaction state M_i into a fixed length vector e:

$$e_i = \phi_e(s, h_i, M_i; W_e), \tag{5}$$

where $\phi_e(\cdot)$ is an embedding function with ReLU activations and W_e denotes the embedding weights.

Then we return e_i to the subsequent MLP to get the paired interaction vector between human and robot:

$$p_i = \psi_p\left(e_i; W_p\right), \tag{6}$$

where $\psi_p(\cdot)$ is a fully connected layer with ReLU nonlinearity and W_p denotes the network weights.

4.2. Attention Module. When there are more humans in the environment, the state that the interaction module needs to handle becomes more complicated. So, in order to better process perception data, it is significant to construct a module that can handle any number of states. So, no matter how the number of humans increases or decreases, we can integrate all states into a fixed-length input vector through this model. At the same time, we need to correctly estimate how much influence each neighbor has on the robot during the movement, which can reflect whether this person will affect the next action of the robot. Therefore, we make use of the attention mechanism to assign different attention scores to different areas of the input data and focus on marking people who have a greater impact on the state of the robot. Chen et al. [24] obtained accurate state input by using the self-attention mechanism, so we propose a dual social attention module to catch the relative importance of each neighbor and the influence of the collective crowd on the robot as shown in Figure 4.

What we collect in the perception module is the state of the robot, the state of each person, and the state of interaction between people. In order to obtain a more accurate vector, we used the attention mechanism twice to process the data. Then our attention module must not only consider the impact of human-human interaction on the state of the robot but also emphasize the impact of the robot on the motion state and trajectory of the human body because the impact of the robot on the human body will further affect the state of the robot. Let us start with the first layer of the attention module.

First, all e_i obtained by the interactive module are pooled to obtain e_m, where e_m is the average value of all individual data:

$$e_m = \frac{1}{n}\sum_{i=1}^{n} e_i. \tag{7}$$

Then we pass e_m through a multilayer perceptron $\psi_\alpha(\cdot)$ to get the proportion of each vector p_i, which is the attention score α_i of the vector:

$$\alpha_i = \psi_\alpha(e_i, e_m; W_\alpha), \tag{8}$$

where $\psi_\alpha(\cdot)$ is an MLP and W_α denotes the weights.

After that, we assign each weight to the corresponding interaction vector p_i of the human and robot pair and then perform a weighted linear combination of all the pairs, that is, to standardize score α_i and perform a weighted summation of all p_i. The final vector is one of the representations of the overall crowd:

$$c = \sum_{i=1}^{n} \text{soft}\max(\alpha_i) p_i. \tag{9}$$

The above is the first-layer attention module. In order to obtain a more accurate crowd vector, we get p_m by pooling averagely p_i (human-robot interaction vector). We pass p_m and p_i through multilayer perceptron $\psi_\beta(\cdot)$ to obtain different attention scores β_i and obtain another vector d representing the crowd through a series of weighted summations:

$$\begin{aligned} p_m &= \frac{1}{n}\sum_{i=1}^{n} p_i, \\ \beta_i &= \psi_\beta\left(p_i, p_m; W_\beta\right), \\ d &= \sum_{i=1}^{n} \text{soft}\max(\beta_i) e_i, \end{aligned} \tag{10}$$

where $\psi_\beta(\cdot)$ is a multilayer perceptron with ReLU activation function and W_β denotes the weights.

Finally, vectors c and d are averaged to get the final vector representation of crowd k:

$$k = \left(\frac{c+d}{2}\right). \tag{11}$$

4.3. Planning Module. Based on the final vector k obtained by the attention module, we design this module to estimate value v:

$$v = f_v(s, k; W_v), \tag{12}$$

where function $f_v(\cdot)$ is a multilayer perceptron with ReLU activation function and W_v denotes the weight.

4.4. Value Network Training. The value network composed of the above three modules also needs to be trained using reinforcement learning algorithms. The value network can calculate the input state to obtain the value corresponding to the state. We use the temporal-difference method and experience replay mechanism to train the value network. Then the training process is shown in Algorithm 1.

From Step 1 to Step 3, we initialize the value network where the robot imitates the navigation strategy of the human expert to complete some demonstration experience. From Step 4 to Step 16, we use reinforcement learning to optimize the value network and the robot gains experience from the exploration.

5. Simulation

In this part, we conduct simulation experiments on five different models for evaluation. The specific content of the experiment is displayed in the next subsections, including the construction of the experimental environment, the setting of parameters, compared solutions, evaluation standards, and the analysis of specific experimental results.

5.1. Virtual Environment Construction and Parameter Setting. We install the Python-RVO2 library in the Python 2.7.0 environment to build a simulation environment for robots to navigate in the crowd. Simulated humans and robots are set up in this environment. The simulated human is controlled by ORCA [25], and the sampling parameters obey the Gaussian distribution so as to achieve the diversity of behavioral data. We used circle crossing scenes for people in both training and testing; that is, the positions of all humans are randomly arranged and represented by a circle with a radius of 4 m, and their destinations are on the opposite of the starting position. We randomly interfere with the X and Y coordinates of the person's starting position and target position. Different algorithms are input to the simulated robot to find a path in the crowd.

There are roughly four existing methods that can efficiently realize robot navigation in a crowd: ORCA, CADRL, LSTM-RL, and SARL. In order to make an even comparison, we ensure that the planning modules are the same in all methods. We train and test the above four models and our model in a simulated environment and get the results for comparison.

In order to strengthen the reliability of the experiment, we set up two states of humans and the robot: visible and invisible. When the robot is not visible, humans will only interact with humans and will not respond to the robots. When the robot is visible, humans will also be affected by the robot's actions in the process of social movement. The model which is trained for 10000 episodes is evaluated using 500 random test cases in both settings.

Our experiment runs on Ubuntu 16.04, and we configure the corresponding Python version and PyTorch. After meeting the hardware equipment for the experiments, we set the parameters required for the experiments as follows. In the simulator environment, the square width is set to 10, and the number of humans is set to 5. The radius of those circles of the simulated robot and human is set to 0.3 m, the priority speed of the robot and human is set to 1 m/s, and the number of humans is set to 5. The local map is a 4×4 grid centered at each human and the side length of each cell is 1 m. We set the batch size to 100 when we train the policy. In reinforcement training, the learning rate is set to 0.001 and the discount factor γ is 0.9. We assume that the robot has complete kinematics, and the action space includes 80 discrete actions. These actions consist of linear velocity and angular velocity. The linear velocity takes exponential values in $(0, v_{\text{pref}}]$; this allows the robot to take some of the more fine-grained operations when approaching the target; and the angular velocity takes 16 values uniformly in $[0, 2\pi)$.

5.2. Baselines. ORCA [25] is a classic distributed underlying algorithm. It is local navigation, and its navigation target is around the individual, allowing the individual to avoid other individual targets and obstacles that are close to him. Since ORCA is mutual, as long as both robots use ORCA, the two robots do not need to communicate, and the speed of the two is calculated in a distributed manner and there will be no collision.

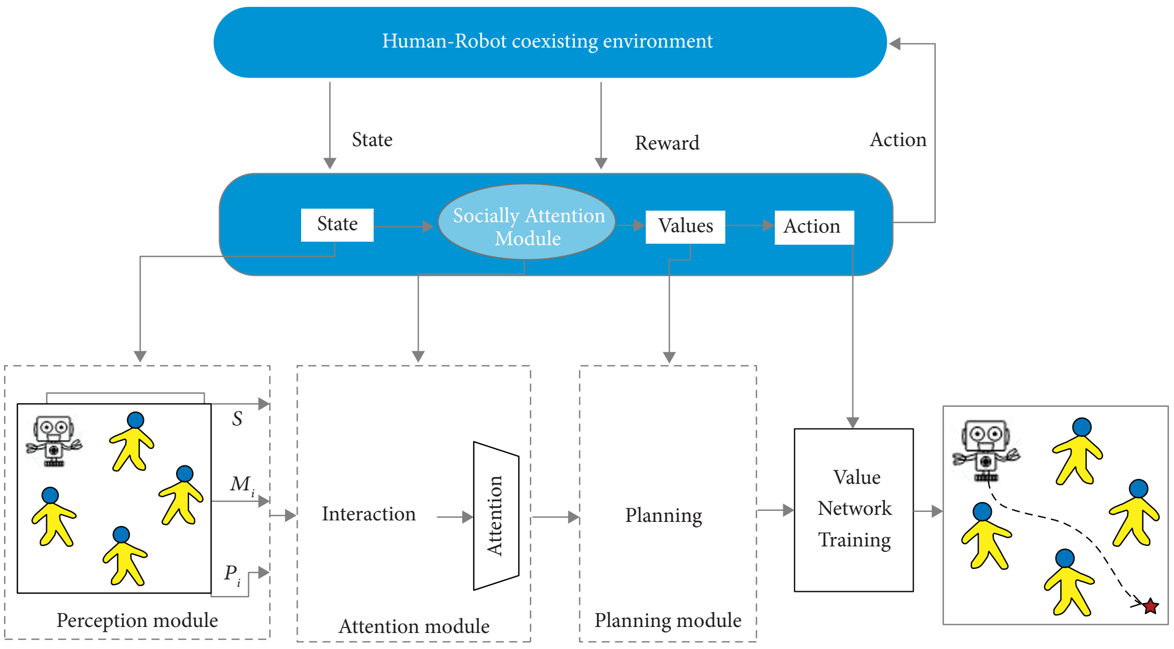


FIGURE 2: An overview of the deep reinforcement learning framework for robot navigation. This model consists of three modules. Then we use the reinforcement learning algorithm to train the model to complete the robot navigation.

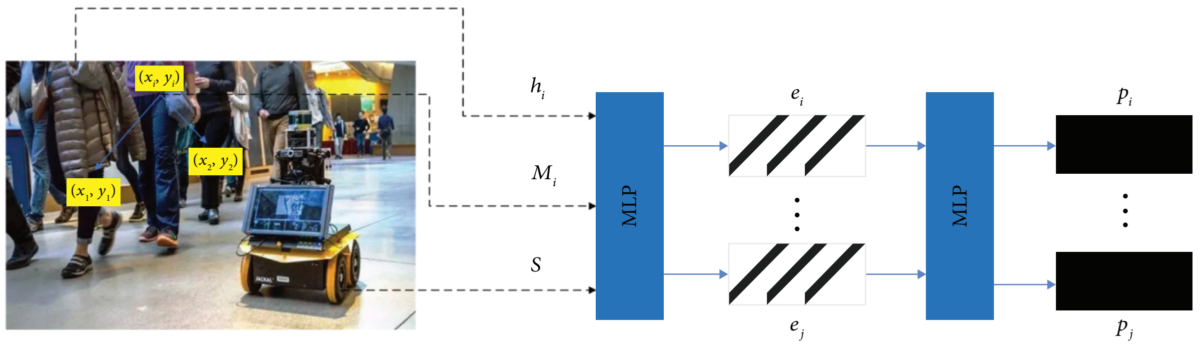


FIGURE 3: Description of our perception module. After sensing the state information of robot and each human i, we feed them into a multilayer perception to obtain lots of joint states.

ORCA can complete computing tasks when the number of robots in the scene is large. In solving the problem of n-body avoiding mutual collision, if this method is adopted, the computational complexity is $O(n)$. But the robots in this algorithm are independent and cannot communicate with each other. Therefore, when a single robot is navigating in a crowd, the robot that uses the ORCA algorithm for obstacle avoidance cannot observe all the parameters of the surrounding humans, and the robot can easily collide with humans during the movement. When the frequency of collisions increases, the robot's navigation time will also become longer. In addition, ORCA cannot consider the impact of human interaction on robots.

CADRL is an algorithm for robots to avoid collisions during navigation. It is simulated and trained through deep reinforcement learning. Specifically, this method develops a value network. The network of CADRL encodes the estimated time for the robot to reach the final position by observing the position and speed of the robot and its neighbors.

CADRL uses a deep reinforcement method on the basis of ORCA to process the obstacle avoidance navigation of the robot. CADRL first uses the ORCA algorithm to predict the pedestrian trajectory and then uses the deep V-learning algorithm to make the robot's next decision. It can collect the interaction information between the robot and the surrounding people within the effective time. Since CADRL obtains the interactive information of all agents by constructing a value network, this greatly reduces the computational consumption compared to ORCA.

LSTM-RL is another improved algorithm made on CADRL. The use of LSTM allows the algorithm to use any number of observations from other robots, instead of only using a fixed observation size. In other words, LSTM-RL supports LSTM to encode the states of all robots into a fixed-length joint vector and next enter the vector into the value

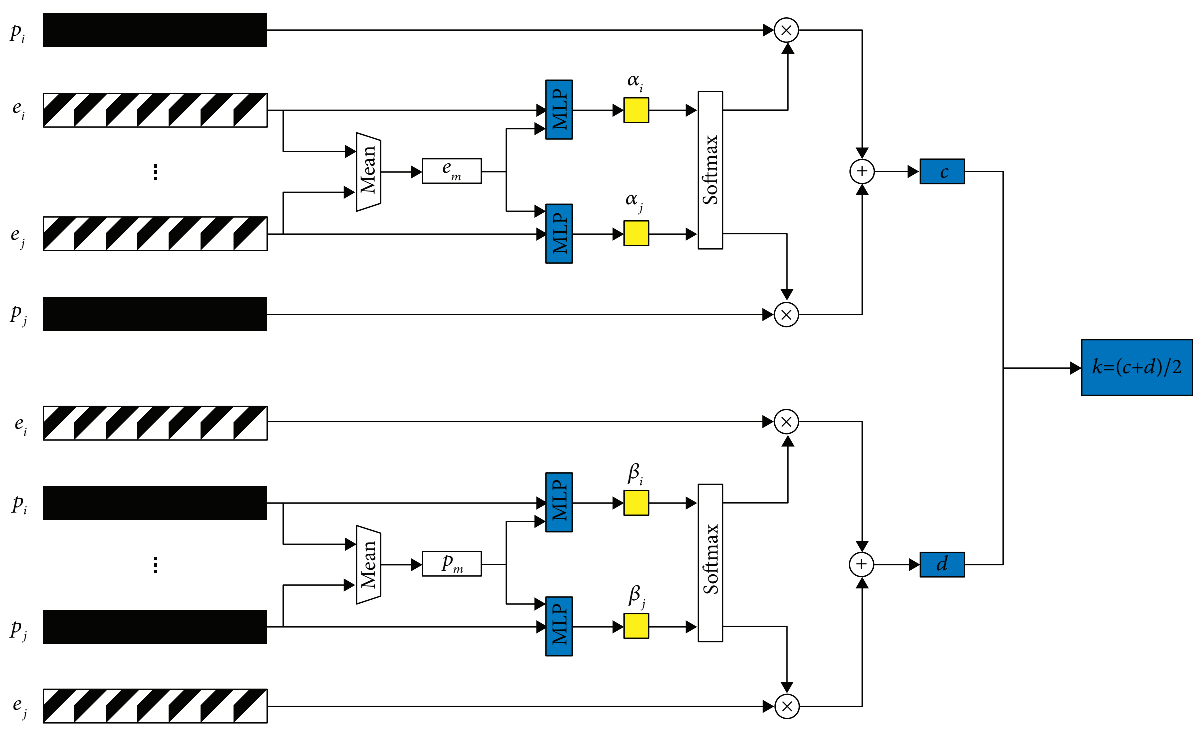


FIGURE 4: Architecture of attention module. We pool the different perceptual vectors and combine them with the attention scores obtained by the multilayer perceptron.

```
(1) Initialize experience replay memory $E$ with demonstration $D$
(2) Initialize value network $V$ with memory $E$
(3) Initialize target value network $V' \longleftarrow V$
(4) for $episode = 1, \ldots, M$ do
(5)     Initialize joint state $j_t = 0$
(6)     repeat
(7)         $a_t \longleftarrow \arg\max_{a_t \in A} R(\mathbf{j}_t, \mathbf{a}_t) + \gamma^{\Delta t \cdot v_{\text{pref}}} V(\mathbf{j}_{t+\Delta t})$
(8)         value $\longleftarrow R(\mathbf{j}_t, \mathbf{a}_t) + \gamma^{\Delta t \cdot v_{\text{pref}}} \cdot V'(\mathbf{j}_{t+\Delta t})$
(9)         state $\longleftarrow j_{t+\Delta t}$
(10)        Enrich experience $E \longleftarrow$ (state, value)
(11)        Optimize value network $V$ with experience $E$
(12)        Update value network $V$ by gradient descent
(13)    until terminal state $s_t$ or $t \geq t_{\max}$
(14)    Update target network $V' \longleftarrow V$
(15) end for
(16) return $V$
```

ALGORITHM 1: Value network training.

network to select the next operation. The inputs of CARDL and ORCA are fixed-size observation information. If a new agent approaches, the state input must be recalculated; and LSTM-RL has no such limitation, which greatly reduces the computational cost.

SARL integrates the collected state in the environment into a fixed vector, uses the pooling module to express it into a compact group representation, and finally navigates to the target place. Compared with the above three algorithms, SARL can make the robot pass the crowd completely without obstacles very accurately. The state information that the SARL algorithm can collect and integrate is the most abundant and extensive, including the state of the robot, the state of each human, and the impact of the interaction between humans on the robot. The self-attention mechanism is also used in SARL to emphasize those humans who have a great influence on the state of the robot.

The states information collected by our DSARL is also the most complete and accurate like SARL, and it takes more into account; and our DSARL uses dual attention to collect and emphasize the influence of crowd movement on robots and the influence of robot movement on the next step of human activities. The interaction is two-way, and the robot will also affect the change of the human trajectory during the movement, which in turn affects the robot's next decision; and DSARL gives extremely high rewards during the movement of the robot perfectly avoiding the crowd.

5.3. Evaluation Standard. We compare the performances of these five models from four aspects: success rate, obstacle rate, navigation time, and total reward.

(i) Success rate: the rate of safely reaching the target point without collision during robot navigation

(ii) Collision rate: the rate of collisions with other humans during robot navigation

(iii) Navigation time: the total time it takes for the robot to walk from the starting point to the target point

(iv) Total reward: accumulated rewards during robot navigation

5.4. Simulation Result Analysis

5.4.1. Quantitative Analysis. When the robot is set to be invisible in the experimental settings, the robot needs to predict the trajectory of all humans to avoid collisions. As shown in Table 1, we use success rate, collision rate, navigation time, and rewards to evaluate different algorithms.

As can be seen from Table 1, in an invisible environment setting, the success rate of ORCA is the lowest; and collision rate of ORCA is the highest. Because the robot in this case is independent, it cannot communicate with the surrounding humans and cannot accurately avoid the crowd reaching the destination safely. In all other algorithms that use reinforcement learning, CADRL has the worst performance. This is because only a single interaction pair is considered in the CADRL method, and other interactions are ignored. However, the navigation effect of CADRL is still much better than that of ORCA, and deep reinforcement learning has played a big role. The difference in navigation time between CADRL and LSTM-RL is not great, but the overall performance of LSTM-RL is still better than that of CADRL. This is because the LSTM algorithm can collect the state of any number of other robots instead of just using a fixed observation size. This makes the robot more accurately avoid moving crowds during navigation. Both SARL and DSARL have completed all test cases without timeouts and conflicts; and the total navigation time of SARL and DSARL is greatly reduced compared to other methods. The result demonstrates that the social attention mechanism is good at capturing the influence of crowd interaction. In addition, the success rate and collision rate of SARL and DSARL are almost the same, but the navigation time of our DSARL algorithm is shorter than that of SARL. The reward of DSARL is higher than that of SARL. This is because our DSARL has dual attention. It not only extracts the impact of crowd interaction on robots like SARL but also emphasizes that robots also affect crowd activities and thus affect the robot's navigation trajectory. In general, from the data point of view, our model is superior to other models in terms of navigation efficiency and rewards.

When the robot is set to be visible in the environment settings, the experimental data we got are shown in Table 2.

From Table 2, we can see that the navigation success rate of ORCA is almost one hundred percent, but the navigation time is relatively long and the reward is not high. That is because ORCA can only perceive the situation close to itself, without information about the global environment, so the robot does not collide with other individual targets and obstacles around itself when navigating. However, understanding and interacting with human behavior is necessary for robots to get high returns. The performance of CADRL when the robot is visible is stronger than that when the robot is not visible. The performance of LSTM-RL is stable and is always better than CADRL. In the visual situation, the robot can observe more information. Both SARL and DSARL have high success rates, but DSARL's navigation time and total rewards are still higher than those of SARL. Therefore, the navigation efficiency of DSARL will be better than that of SARL regardless of whether it is visible or not.

In general, regardless of whether the robot is visible or not, the results of reinforcement learning methods are similar. The performance of LSTM-RL is similar to that of SARL. The performance of ORCA in the visible environment is far better than that in the invisible environment. Our DSARL also improves the reward function part to obtain higher reward returns. In addition to the experimental data, we will also analyze the robot's trajectory changes under different algorithms in the next section.

5.5. Cases Study. We further analyze the effectiveness of the model in this section. As shown in Figure 5, we compare the paths formed by the robot using different navigation algorithms in an invisible environment. We use the same test case so that human trajectories using different algorithms are the same.

CADRL is going straight to the destination. When meeting a human in the center of space, CADRL does not make a big step to avoid or retreat but aggressively walks past the human. At a certain moment in the center of space, CADRL is very close to humans, and, in order to avoid collisions, CADRL takes a long time to navigate. In contrast, LSTM-RL turns sideways at the start and will largely avoid the crowd in 4.0 to 8.0 seconds. LSTM-RL navigation takes about the same time as CADRL.

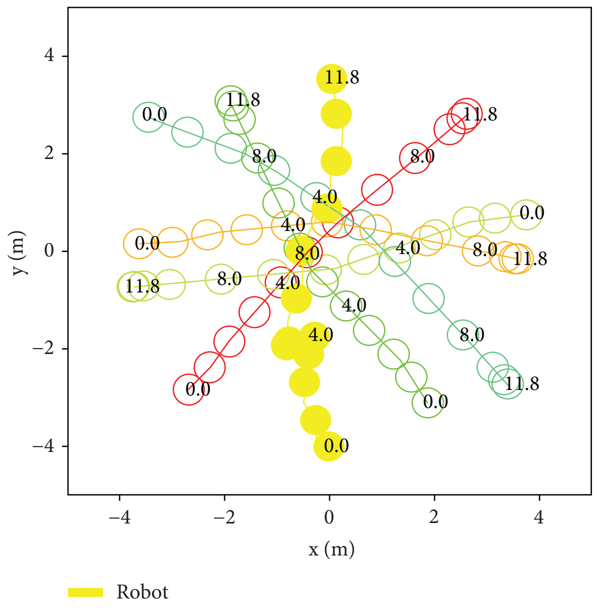
Since SARL hesitates when it starts, it travels a short distance in 0.0 to 4.0 seconds. But then SARL constructs a barrier-free shortcut through the center of the space to reach the target so that SARL can safely and quickly avoid humans. Our DSARL immediately identifies a safe path from the beginning; no matter at any moment, there is always a long distance between it and every human being. In 4.0 seconds, the robot has moved halfway. The human closest to the robot at this moment is a red circle, and the other humans are also far away at this moment. In 4.0 to 8.0 seconds, the robot chooses to go around a bend because it has to avoid more concentrated crowds, resulting in a shorter walking distance. In the last part

Table 1: Results in the invisible setting.

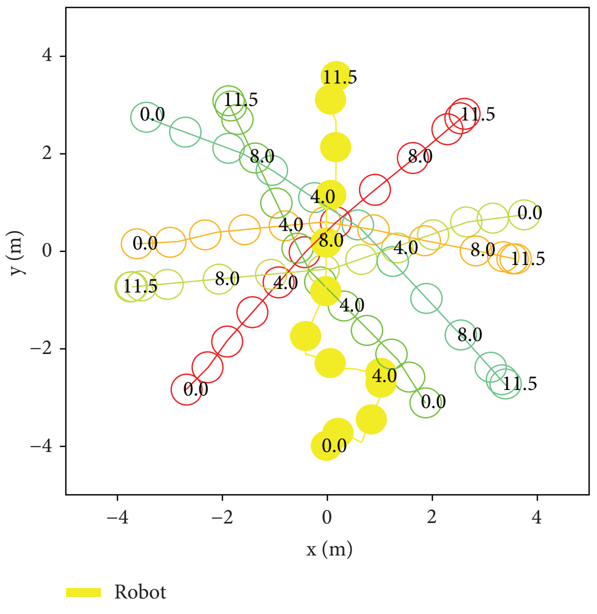
Methods	Success	Collison	Time (s)	Reward
ORCA	0.43	0.57	10.86	0.0483
CADRL	0.87	0.12	11.14	0.2511
LSTM-RL	0.97	0.02	11.92	0.2820
SARL	0.99	0.01	10.49	0.3351
DSARL (our)	1.00	0.00	10.39	0.3439

Table 2: Results in the visible setting.

Methods	Success	Collison	Time (s)	Reward
ORCA	1.00	0.00	12.19	0.2552
CADRL	0.92	0.08	10.76	0.2734
LSTM-RL	0.93	0.07	10.42	0.2854
SARL	0.99	0.01	10.34	0.3493
DSARL (our)	1.00	0.00	10.24	0.3505

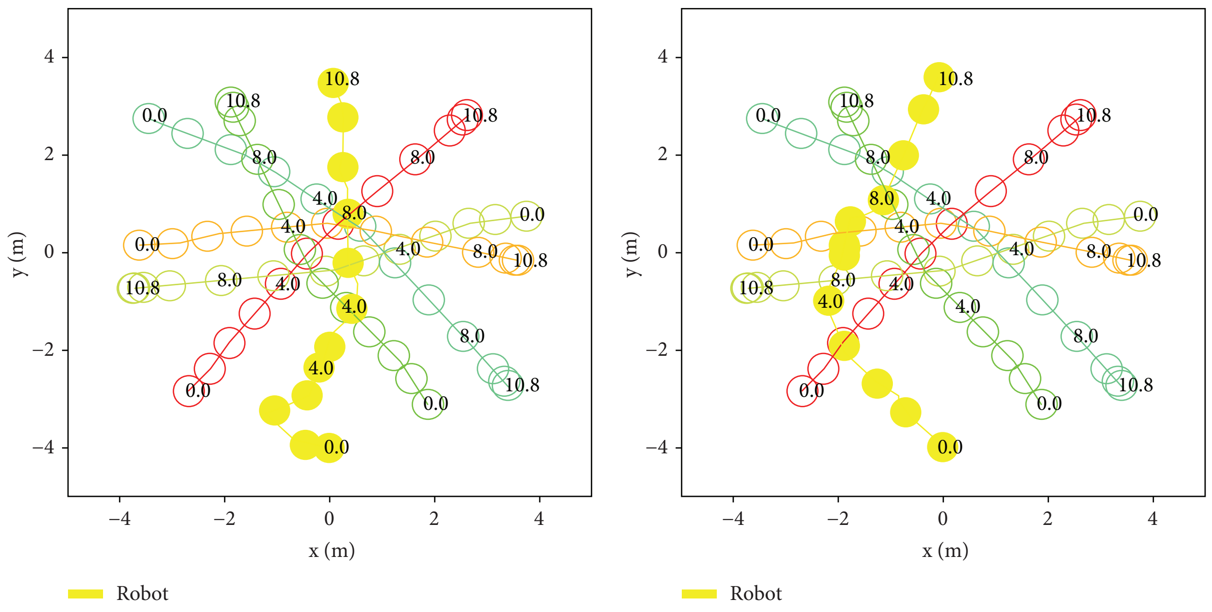


(a)



(b)

Figure 5: Continued.



(c) (d)

Figure 5: Trajectory comparison with other algorithms. These circles indicate the position of the robot or human at a certain moment. Yellow circle means the robot; other colours are human circle. (a) Trajectory path diagram using CADRL algorithm. (b) Trajectory path diagram using LSTM-RL algorithm. (c) Trajectory path diagram using SARL algorithm. (d) Trajectory path diagram using our DSARL algorithm.

of the journey, humans are hardly encountered and the robot reaches the end quickly and accurately.

In other words, the robot using the DSARL algorithm can navigate without hesitation and avoid the collision-prone central area and finally reach the destination safely. It can be clearly seen from the picture that DSARL can safely avoid all humans in a short time. Compared with SARL, robots that use the DSARL algorithm to navigate are more efficient and will not hesitate to wander on the road.

In general, the navigation success rate of the robot using our method is better than that of the robot using the SARL method, whether it is visible or not. In terms of navigation time, our method reduces the robot's navigation time by 0.97% on the basis of SARL. In terms of task rewards, our method improves by 2.6% compared to SARL. From the SARL trajectory in Figure 5, it is also obvious that the robot using SARL is hesitating when it first starts and does not know which direction to go in within 0.0 to 4.0 seconds. The robots using our method move forward clearly and decisively in navigation, without wasting time. In addition, robots that use SARL navigation are too close to humans within 4.0 to 8.0 seconds and are prone to collisions. However, robots that use our DSARL navigation will keep a certain distance from humans at any time.

6. Conclusions

In this article, we model all human-robot and human-human interactions to solve the problem of safe navigation of robots in crowds. We combine the state of people, the state of robots, and the interaction between people and aggregate all states into a compact group representation through a dual social attention enhancement module. Then the dual social attention reinforcement learning model is trained and tested in our reinforcement learning algorithm framework. Compared with the impact of human-human interaction on robot decision-making in the SARL algorithm, we consider that the robot's impact on the human trajectory during the movement will also affect the robot's next decision, so we add another attention module to use a dual attention to extract and emphasize the influence of crowd interaction on the robot trajectory and the influence of robot movement on the next human activities. The robot has an impact on the human trajectory during the movement, which will affect the robot's next decision; and we improved the reward function to strengthen the constraints on robot behavior. Experimental results show that our algorithm is superior to other algorithms in collision-free navigation success rate, time efficiency, and task achievement (total rewards given to the robot). In future research, we plan to further improve the algorithm and develop a new action space module to enable the robot to output continuous actions. We hope to further improve navigation efficiency.

Our simulation experiments verified the reliability and rationality of this method. We plan to further study our method in real scene in the future research. In the real scene, the robot may face some problems, such as errors and delays in obtaining position information and speed information, due to the gap between actual application and simulation. Therefore, we may need to fine-tune the parameters of the model with more data collected from real scene.

Data Availability

The data used to support the findings of this work are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by the Science and Technology Research Project of the Education Department of Jiangxi Province (no. 200604), the National Natural Science Foundation of China (no. 62062033), the Natural Science Foundation of Jiangxi Province (no. 20192ACBL21006), and Key Research & Development Plan of Jiangxi Province (no. 20203BBE53034).

References

[1] C. Z. Zhu and P. Chen, "Navigation for indoor robot: straight line movement via navigator," *Mathematical Problems in Engineering*, vol. 2018, Article ID 8419384, 10 pages, 2018.

[2] E. Repiso, A. Garrell, and A. Sanfeliu, "Adaptive side-by-side social robot navigation to approach and interact with people," *International Journal of Social Robotics*, vol. 12, no. 2, pp. 1–23, 2020.

[3] L. Tai, J. W. Zhang, M. Liu, and W. Burgard, "Socially compliant navigation through raw depth inputs with generative adversarial imitation learning," in *Proceedings-IEEE International Conference on Robotics and Automation*, pp. 1111–1117, Guangzhou, China, May 2018.

[4] H. Ponce, E. Moya-Albor, and J. Brieva, "A novel artificial organic control system for mobile robot navigation in assisted living using vision-and neural-based strategies," *Computational Intelligence and Neuroscience*, vol. 2018, pp. 1–16, Article ID 4189150, 2018.

[5] C. E. Tsai and J. Oh, "A generative approach for socially compliant navigation," in *Proceedings-IEEE International Conference on Robotics and Automation*, pp. 2160–2166, Paris, France, May 2020.

[6] A. Alahi, K. Goel, and V. Ramanathan, "Social LSTM: human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, Las Vegas, NV, USA, June 2016.

[7] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: a survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[8] A. Kanazawa, J. Kinugawa, and K. Kosuge, "Incremental learning of spatial-temporal features in human motion patterns with mixture model for planning motion of a collaborative robot in assembly lines," in *Proceedings of International Conference on Robotics and Automation*, pp. 7858–7864, Montreal, Canada, May 2019.

[9] M. Bennewitz, W. Burgard, and S. Thrun, "Learning motion patterns of persons for mobile service robots," in *Proceedings-IEEE International Conference on Robotics and Automation*, Washington, NJ, USA, May 2002.

[10] A. Cruz-Maya, "Target reaching behaviour for unfreezing the robot in a semi-static and crowded environment," 2020, https://arxiv.org/abs/2012.01206.

[11] P. Trautman and A. Krause, "Unfreezing the robot: navigation in dense, interacting crowds," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 797–803, Taipei, China, October 2010.

[12] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proceedings-IEEE International Conference on Robotics and Automation*, pp. 285–292, Singapore, June 2017.

[13] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine, "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *Proceedings-IEEE International Conference on Robotics and Automation*, pp. 5129–5136, Guangzhou, China, May 2018.

[14] G. Antonini, S. Venegas, J. Thiran, and M. Bierlaire, "A discrete choice pedestrian behavior model for pedestrian detection in visual tracking systems," *Advanced Concepts for Intelligent Vision Systems*, 2004.

[15] C. F. Wakim, S. Capperon, and J. Oksman, "A markovian model of pedestrian behavior," *Proceedings of the IEEE International Conference on Systems*, vol. 4, pp. 4028–4033, 2004.

[16] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: a social-force based approach with human awareness-navigation in crowded environments," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1688–1694, Tokyo, Japan, November 2013.

[17] Z. Yuan, R. Guo, S. Tang, B. He, L. Bian, and Y. Li, "Simulation of the separating crowd behavior in a T-shaped channel based on the social force model," *IEEE Access*, vol. 7, pp. 13668–13682, 2019.

[18] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pp. 1186–1194, Lake Tahoe, NV, USA, November 2018.

[19] A. Vemula, K. Muelling, and J. Oh, "Social attention: modeling attention in human crowds," in *Proceedings-IEEE International Conference on Robotics and Automation*, pp. 4601–4607, Guangzhou, China, May 2018.

[20] M. Brittain, X. Yang, and P. Wei, "A deep multi-agent reinforcement learning approach to autonomous separation assurance," 2020, https://arxiv.org/abs/2003.08353.

[21] G. Chen, S. Yao, J. Ma et al., "Distributed non-communicating multi-robot collision avoidance via map-based deep reinforcement learning," *Sensors*, vol. 20, no. 17, p. 4836, 2020.

[22] T. Kretz, J. Lohmiller, and P. Sukennik, "Some indications on how to calibrate the social force model of pedestrian dynamics," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2672, no. 20, pp. 228–238, 2018.

[23] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3052–3059, Madrid, Spain, October 2018.

[24] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proceedings of International Conference on Robotics and Automation*, pp. 6015–6022, Montreal, Canada, May 2019.

[25] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Systems*, pp. 1–10, 2019.