Foresight Social-aware Reinforcement Learning for Robot Navigation

Yanying Zhou
Institute for Numerical Simulation
University of Bonn
Bonn, Germany
zhou@ins.uni-bonn.de

Shijie Li
Institute for Computer Sciences
University of Bonn
Bonn, Germany
lsj@uni-bonn.de

Jochen Garcke
Institute for Numerical Simulation
University of Bonn
and Fraunhofer SCAI
Bonn, Germany
garcke@ins.uni-bonn.de

***Abstract*—When robots handle navigation tasks while avoiding collisions, they perform in crowded and complex environments not as good as in stable and homogeneous environments. This often results in a low success rate and poor efficiency. Therefore, we propose a novel Foresight Social-aware Reinforcement Learning (FSRL) framework for mobile robots to achieve collision-free navigation. Compared to previous learning-based methods, our approach is foresighted. It not only considers the current human-robot interaction to avoid an immediate collision, but also estimates upcoming social interactions to still keep distance in the future. Furthermore, an efficiency constraint is introduced in our approach that significantly reduces navigation time. Comparative experiments are performed to verify the effectiveness and efficiency of our proposed method under more realistic and challenging simulated environments.**

***Index Terms*—Robot navigation, Deep reinforcement learning, Foresight obstacle avoidance, Human-robot interaction**

I. Introduction

With the uptake of artificial intelligence, robots play a more and more important role in human life, for example as delivery or service robots. Instead of working in a restricted space, these robots share their working space with humans and have frequent interactions, while they are expected to arrive at their destinations without a negative impact on people.

As robot navigation can be formulated as a reinforcement learning (RL) task, recently, more and more works apply RL algorithms to utilize the strong decision-making ability of RL [1]. Meanwhile, deep learning, with its excellent representation ability, has achieved great success in many areas, like computer vision and natural language processing. Hence, some works [2]–[7] combine deep learning and RL for social-aware robot navigation. In these approaches, an effective policy is learned that implicitly models the complex interactions and cooperation of agents. Specifically, a deep neural network is adopted to approximate the value function and the optimal action is then chosen based on this value function. Although learning-based methods outperform non-learning methods, these methods are still limited when mobile robots navigate in more realistic complex environments.

A main issue in previous robot navigation works is that they oversimplify the environment by only considering the avoidance of dynamic objects. In this scenario, the robot usually learns a simple policy that achieves the goal by bypassing all dynamic objects. First, it is problematic that these previous methods can behave shortsighted. They make the decision only according to the current interactions and ignore possible future situations, which limits their navigation ability in complex crowds. Additionally, note that previous methods only implicitly penalize inefficient policies. Hence, these approaches often lack the capability to navigate through complex crowds and are in particular prone to bypass them inefficiently by taking excessive detours to avoid collisions. In more realistic scenarios where standing objects exist, these problems deteriorate further, as illustrated in Fig. 1.

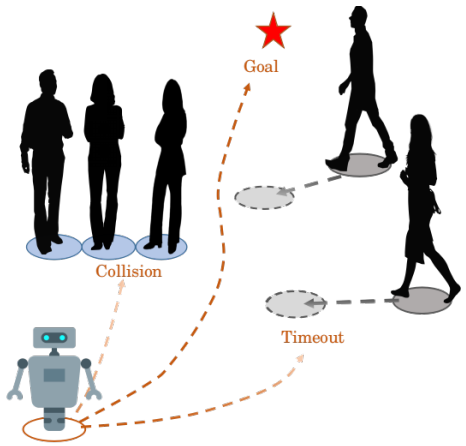


Fig. 1: In a realistic scenario, both standing (blue) and dynamic (grey) objects exist. Unlike oversimplified scenarios in previous methods where only dynamic objects exist, traps or blind spots will form more frequently and not disappear over time. These will trap the robot in the crowd more easily.

Another problem in many existing works is the assumption that the robots have holonomic kinematics. It eases the robot navigation task as the robots can freely move in any direction at any timestep. Holonomic robots can perform large-angle rotation actions to access their destination easily with non-smooth navigation paths. However, most robots in daily life have nonholonomic kinematics [8], such as service and delivery robots. It requires that the robots can only move smoothly,

This work was supported by the China Scholarship Council (CSC).

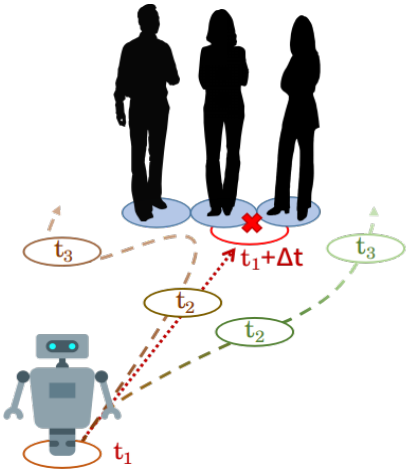


Fig. 2: Previous methods (brown) only detect the collision at time t_2 hence turn around sharply to avoid a collision, which is shortsighted. In contrast, our method (green) can forecast potential collision at time t_1 within Δt (red) hence can take action in advance to avoid collision smoothly, which is foresighted. This is more important for nonholonomic robots as they cannot get out of traps easily.

which intensifies the above problems. For example, for trapped non-holonomic robots, it is hard to get out of their trap.

Based on the above limitations, we present a novel deep reinforcement learning (DRL) approach for robot navigation with collision avoidance. We enrich our RL method with the ability to forecast future potential interactions based on the current states. In other words, the robots are more foresighted, which is shown in Fig. 2. The robot can make a decision combining both current and predicted interactions, thereby reacting earlier in advance of potential collisions. Hence, no matter the complexity of environment, it is feasible for the robot to move smoothly between successive timesteps. In addition, our approach explicitly takes standing objects into account. Specifically, unlike previous methods that ignore the different kinematics between objects, we apply different restrictions on obstacles according to their movement states. Our method can detect future potential unsafety and take corresponding actions in advance, which can be seen in Fig. 2. Moreover, our method employs a constraint on the navigation time, which enables the learning of a more efficient strategy.

As our method can forecast future collisions (foresighted), we name it as Foresight Social-Aware Reinforcement Learning (FSRL) algorithm. In summary, our contributions are:

- Our method can forecast potential collisions in the future and can take actions in advance to avoid these collisions, which improves the quality of navigation.
- Our method explicitly uses a constraint on the navigation time, hence the learned strategy is more efficient.
- We perform experiments in complex environments with dynamic and standing humans.
- Together, we introduce a foresighted navigation method that achieves the best performance in different environments, which proves its effectiveness and robustness.

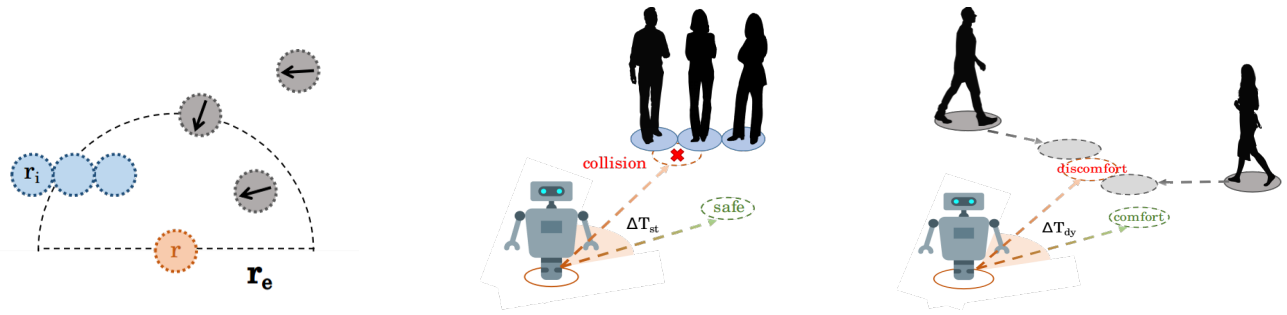
II. RELATED WORK

Robot navigation in pedestrian-rich environments is complicated since human behavior is diverse and stochastic depending on human intent and social interactions. Earlier works on robot navigation used *reaction-based methods* [9]–[14] that choose the optimal action according to one-step interaction rules. Social Force Model [9]–[11] models the interactions as 'forces', such as attractive forces and repulsive forces, to drive the agent to reach the goal while avoiding obstacles. Optimal Reciprocal Collision Avoidance (ORCA) [12] and Reciprocal Velocity Obstacle (RVO) [13] achieve collision-free navigation under reciprocal assumption. Interacting Gaussian Process (IGP) [14] presents an interaction potential function to couple independent Gaussian Processes. However, these methods are limited by their hand-crafted social functions that can only capture simple interactions and are hard to be extended to crowded situations. Also, they are prone to lack foresight and have unnatural behaviors [7].

Trajectory-based methods [15]–[17] plan a proper path for robots by predicting other agents' trajectories from the large-scale datasets. [16] combines Gaussian Processes with Rapidly-exploring Random Trees to generate probabilistic safe paths. [17] presents a spatially local interaction function to predict the joint human motion. Although trajectory-based methods are long-sighted, they are computationally expensive and time-consuming by explicitly accounting evolution of joint paths, especially when there are large teams. In addition, since robots take overly conservative strategies, robots have less planable navigation space and thus are more likely to face the freezing robot problem [4].

Compared to these non-learning-based methods, *learning-based methods* are more computationally efficient and can perform much better. DRL methods [2]–[7] explore a computationally efficient interaction rule by pre-training a value function. The optimal actions can then be obtained based on the currently observed states. The robot can thus avoid collisions while planning the trajectory. In SA-CADRL [5] the optimal action is chosen through a maxmin operation. SARL [3] jointly models the human-robot interactions as well as human-human interactions based on CADRL [4]. [6] proposes a framework by combining ego-safety and social-safety in mapless navigation.

However, current methods do not perform well in realistic environments. It is because they oversimplify the environment and robotics kinematics in robot navigation which are far from realistic scenarios. This leads the robots are usually shortsighted in crowded or movement restricted environments, while lacking the ability to be aware of potential collisions. In contrast, our method estimates not only the current interactions but also predicts future potential situations. Hence, it can navigate through complex crowds smoothly and balance safety and efficiency. Furthermore, our method is more efficient by explicitly introducing constraints on the navigation time.



(a) Effective range r_e (b) Static situation (c) Dynamic situation

Fig. 3: (a) An illustration of the effective range r_e of the robot. The blue and grey circles respectively represent the standing and dynamic humans with radius r^i, where arrows indicate the moving direction. (b) By detecting potential collisions with standing objects in ΔT_{st}, the robot can take action in advance to avoid collisions. This is achieved by the reward component R_{st}. (c) According to the comfortable distance r_c within ΔT_{dy}, the robot chooses to detour the crowd instead of going through it, which would annoy humans. This is achieved by the reward component R_{dy}.

III. APPROACH

A. Problem Formulation

We formulate social robot navigation in 2D space as a sequential decision making problem in a RL framework, which is a Partially Observable Markov Decision Process (POMDP). At each time step t, for each agent (robot or human), we denote its state and action as $\mathbf{s}_t$ and $\mathbf{a}_t$, respectively. The state $\mathbf{s}_t$ can be divided into observed and hidden parts, that is $\mathbf{s}_t = [\tilde{\mathbf{s}}_t,\ \hat{\mathbf{s}}_t]$. Here, the observable part is $\tilde{\mathbf{s}}_t = [p_x, p_y, v_x, v_y, r] \in \mathbb{R}^5$, where $\mathbf{p} = [p_x, p_y]$, $\mathbf{v} = [v_x, v_y]$ and r are the agent's position, velocity, and radius, respectively. The hidden part is $\hat{\mathbf{s}}_t = [g_x, g_y, v_{pref}, \theta] \in \mathbb{R}^4$, where $\mathbf{p}_g = [g_x, g_y]$, v_{pref} and θ are the goal position, preferred speed, and orientation angle, respectively. We denote all humans' observed state as $\tilde{\mathbf{s}}_t^H = [\tilde{\mathbf{s}}_t^1, \tilde{\mathbf{s}}_t^2, \ldots, \tilde{\mathbf{s}}_t^n]$ at time step t. For simplicity, we use $\mathbf{s}_t$ to denote the robot state. Thus, the joint state observed by the robot is defined as $\mathbf{S}_t = [\mathbf{s}_t, \tilde{\mathbf{s}}_t^H]$.

Considering that the robot has unicycle nonholonomic kinematics, the action $\mathbf{a}_t$ can be denoted by linear and angular velocities, $\mathbf{a}_t = [v_t, \omega_t] \in \mathbb{R}^2$. Following the RL methodology [1], the robot will receive a reward R_t when performing an action to judge the quality of its decision. The goal is to find an optimal policy $\pi^* : \mathbf{S}_t \mapsto \mathbf{a}_t$ that assigns each state an action to maximize the expected future cumulative reward of the actions taken till the goal is reached. A value network is designed to encode an estimate of the optimal value function:

$$V^*(\mathbf{S}_t) = \mathbb{E}\left[\sum_{t'=t}^{T} \gamma^{t' \cdot v_{pref}} R\left(\mathbf{S}_{t'}, \pi^*(\mathbf{S}_{t'})\right)\right], \tag{1}$$

where $\gamma \in (0, 1)$ is a discount factor, and $R(\mathbf{S}_t, \mathbf{a}_t)$ is the received reward at time t. The optimal policy $\pi^*(\mathbf{S}_t)$ can be determined after computing the value function $V^*(\mathbf{S}_t)$:

$$\begin{aligned}\pi^*(\mathbf{S}_t) = \underset{\mathbf{a}_t}{\operatorname{argmax}}\, R(\mathbf{S}_t, \mathbf{a}_t) + \\ \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{S}_{t+\Delta t}} P(\mathbf{S}_t, \mathbf{a}_t, \mathbf{S}_{t+\Delta t}) V^*(\mathbf{S}_{t+\Delta t})\, d\mathbf{S}_{t+\Delta t},\end{aligned} \tag{2}$$

where Δt is the time step. $P(\mathbf{S}_t, \mathbf{a}_t, \mathbf{S}_{t+\Delta t})$ denotes the transition probability between t and $t + \Delta t$.

B. Parametrization

We set the robot as the center of the coordinate system with its first-person perspective as x-axis pointing towards the goal. Based on this transformation, the states of the robot and humans can be represented by:

$$\begin{aligned} s_t &= [d_g, v_{pref}, v_t, \omega_t, r], \\ \tilde{s}_t^i &= [p_x^i, p_y^i, v_x^i, v_y^i, r^i, d^i, r^i + r], \end{aligned} \tag{3}$$

where $d_g = \|\mathbf{p} - \mathbf{p}_g\|_2$ and $d^i = \|\mathbf{p} - \mathbf{p}^i\|_2$ respectively represent the distance between the robot and its goal and the distance between the robot and the human i.

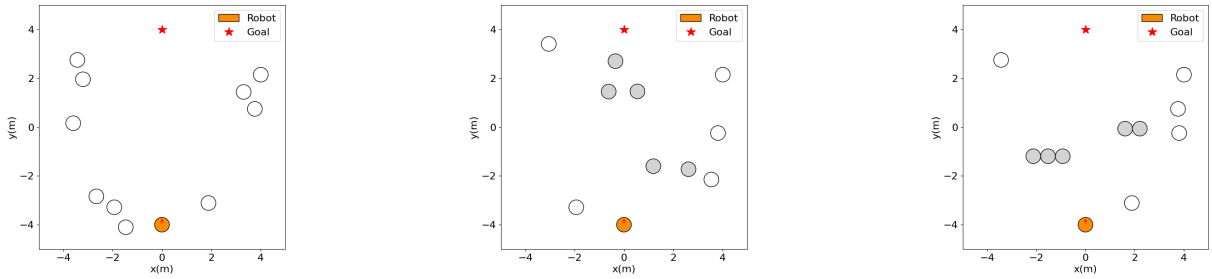
C. Reward Function

In previous methods [3], [4], the decision at each time step is only determined using the current state, which is formulated by the reward function $R_c(\mathbf{S}_t, \mathbf{a}_t)$:

$$R_c(\mathbf{S}_t, \mathbf{a}_t) = \begin{cases} -0.25 & \text{if } d_t < 0 \\ 0.25(-0.1 + d_t/2) & \text{else if } d_t < r_c \\ 1 & \text{else if } p = p_g \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

where $d_t = min\{d^i - r - r^i\}$ denotes the closest distance from the robot to the other humans at time t. The comfortable distance $r_c = 0.2m$ is defined as the minimum comfort distance between the humans and the robot.

One reason for the poor performance of previous methods in complex environments is that they are easily trapped, mainly since they cannot detect potential collisions in the future. This deteriorates further when more realistic nonholonomic kinematics are adopted as sharp movements are then forbidden. Thus, we propose a foresighted method, which means to take actions in advance. Since in nonholonomic kinematics the robot's rotation angle is limited, its movement in a short time period are smooth and can be seen as linear. Future situations then can be estimated from the current state. Based



(a) Env.1 (b) Env.2 (c) Env.3

Fig. 4: Simulation environments. Hollow circles are dynamic objects, whereas solid grey circles are static standing agents. (a) Env.1 has 10 dynamic objects. (b) Env.2 has 5 random standing objects and others are dynamic. (c) Env.3 has two group barriers of 2 and 3 standing objects and others are also dynamic. Compared to the previous environment, robots are more easily trapped and hard to detour in our latter two environments.

on this, we introduce a foresight penalty R_f that penalizes potential collisions in the future. Furthermore, in a realistic environment, both dynamic and standing objects exist, which we in contrast to previous methods distinguish. Our method classifies them based on their velocities and treats them differently. Hence an improved strategy can be learned.

The foresight penalty R_f is now formulated as:

$$R_f(\mathbf{S}_t, \mathbf{a}_t) = R_{dy}(s_t^{jn}, \mathbf{a}_t) + R_{st}(s_t^{jn}, \mathbf{a}_t), \tag{5}$$

where R_{dy} and R_{st} are responsible for dynamic and standing objects, respectively, and are introduced in the following.

As far objects have little influence on the current decisions, we only take the objects within the effective range $r_e = 2m$ into account, shown in Fig. 3(a). For standing objects in r_e, the robot can only choose to bypass them as they will not disappear over time. If the robot detects that there are potential collisions with standing objects in $\Delta T_{st} = 2s$ according to the current decision, a penalty is applied. This penalty is proportional to the number of potential collisions with standing objects, which reveals a crowding level, and is formulated as:

$$R_{st}(\mathbf{S}_t, \mathbf{a}_t) = -\alpha * \frac{N_{col}}{N_{static}}, \tag{6}$$

where $\alpha = 0.15$. N_{col} is the number of detected potential collisions between standing objects (in r_e) and the robot in ΔT_{st}. N_{static} is the number of standing humans that are in the effective range r_e of the robot. By this reward penalty, potential collision are encouraged to be avoided, while as an effect of the foresight built into this penalty the robot also shall take actions in advance to avoid sharp movements, hence making the navigation more smooth, see Fig. 3(b).

As for dynamic objects, they easily gather into clusters and thereby cause collisions. Different from the above scenario, these clusters will disappear as time goes by. Thus the robot needs to determine whether to wait and then go through or to bypass clusters. Further, in a real-world application, the robot and humans are expected to not influence each other. Therefore, apart from navigating to the goal successfully, we also care about the perceived quality of navigation. Since dynamic humans would move to achieve a comfortable distance, an additional penalty will be applied when the robot disturbs the people (their distance is less than the comfortable distance). This can be formulated as:

$$R_{dy}(\mathbf{S}_t, \mathbf{a}_t) = \beta * (d_{\Delta T_{dy}} - r_c), \tag{7}$$

where $\beta = 0.5$, $r_c = 0.2m$. $d_{\Delta T_{dy}}$ is the minimum distance between the dynamic humans and the robot after $\Delta T_{dy} = 1s$, which is half of ΔT_{st} in R_{st} as dynamic objects move and thus this time window needs to be shorter. By this reward penalty, the robot shall avoid aggressive navigation, in other words, it avoids disturbing people frequently, see Fig. 3(c).

Apart from the success rate of navigation, we also explicitly model efficiency in the reward, which is often neglected in previous methods. To achieve this, we add a constraint on the navigation time and encourage efficient strategies by:

$$R_t(\mathbf{S}_t, \mathbf{a}_t) = \begin{cases} -0.1 * \frac{t}{t_{limit}} & \text{if } p = p_g \\ -0.2 & \text{else if } t >= t_{limit}, \end{cases} \tag{8}$$

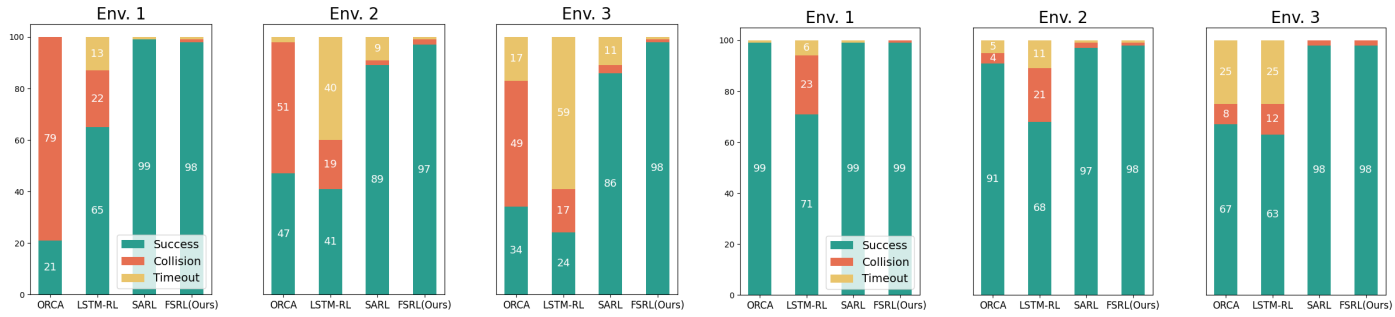
where t_{limit} is the maximum navigation time, which is set as $t_{limit} = 25s$ in our experiments. By adding this term, detours will be discouraged.

Together, the whole reward function R is defined as:

$$R(\mathbf{S}_t, \mathbf{a}_t) = R_c(\mathbf{S}_t, \mathbf{a}_t) + R_f(\mathbf{S}_t, \mathbf{a}_t) + R_t(\mathbf{S}_t, \mathbf{a}_t). \tag{9}$$

D. Training the Policy

We train the policy using the proposed reward via a deep V-learning algorithm [3]. The training process can be divided into two stages. First, the policy is initialized via imitation learning by collecting 3000 episodes demonstrations based on the ORCA [12] policy, which is trained using the Adam optimizer [18] with 50 epochs and a learning rate of 0.01. Then, the policy is refined based on the proposed reward functions with the learning rate set to 0.0001. In addition, the discount factor γ is 0.9. In a greedy fashion, the exploration rate decays linearly from 0.5 to 0.1 in the first 4000 episodes and remains at 0.1 for another 6000 episodes. We assume that the robot has nonholonomic kinematics, which is a constraint on the rotation angle. The robot's velocity is exponentially



(a) Invisible Setting

(b) Visible Setting

Fig. 5: Quantitative evaluation among ORCA [12], LSTM-RL [2], SARL [3] and FSRL(Ours) methods in three environments: (a) The three pictures on the left are with invisible setting; (b) The three pictures on the right are with visible setting.

discretized into 5 speeds between (0, V_{pref}] and 10 headings spaced between $[-\pi/8, \pi/8]$.

IV. EXPERIMENTS

A. Simulation Setup

We build the simulation environment using Gym [19] and RVO [13] as in [3]. There is one robot and 10 dynamic/standing humans in the environment. Our goal is to learn an optimal strategy to make the robot navigate to the destination efficiently. In each experiment, the start and goal position of the robot are set to $(0, -4)$ and $(0, 4)$, respectively. Previous works adopt a circle-crossing where all humans are dynamic and randomly placed on a circle with their destinations on the opposite side. Further, all humans act following the ORCA policy [12]. As mentioned, this scenario is oversimplified and objects can be easily bypassed, which is far from a realistic environment. Hence, one can not demonstrate the navigation ability of algorithms comprehensively.

To better show the navigation ability, we test all methods on *three increasingly challenging environments*, shown in Fig.4.

- **Env. 1** is the same as in previous works where only dynamic agents exist.
- **Env. 2** contains 5 dynamic and 5 standing agents. The positions of the standing agents are distributed randomly in the whole environment. We can see that traps and blind areas are more easily formed in this environment.
- **Env. 3** is an extreme version of Env. 2. The only difference is that the standing agents form two fixed traps and blind areas that will not disappear over time.

In addition, we perform experiments in two settings: one sets the robot **invisible** to humans, which means that humans do not react to the robot's behavior, such as giving way; the other is the **visible** setting, where the mutual human-robot interactions increase the uncertainty of environments. In either setting, the robot needs a more foresighted policy.

We compare our method with three state-of-the-art methods: ORCA [12] is the typical reaction-based method and LSTM-RL [2], SARL [3] are DRL methods. Specifically, for the three environments and the two settings for each environment (6 environments in total), we train the unicycle rotation-constrained robot with all four algorithms separately and then obtain the corresponding test results. To reduce the influence of randomness, we conduct 3 experiments for each method on each environment and report the average performance.

B. Metrics

For each environment, we obtain the quantitative evaluation after 500 random test episodes based on the trained models.

There are five metrics used in the quantitative evaluation:

(1) "Succ.(%)": the rate of success cases where the robot reaches its goal without a collision in the maximum allowed time t_{limit}.
(2) "Coll.(%)": the rate of collision cases where the robot collides with other agents.
(3) "Timeout(%)": the rate of timeout cases where the robot neither collides with others nor reaches its goal in the maximum allowed time t_{limit}.
(4) "Nav. Time": average navigation time of the above success cases in seconds.
(5) "Disc. (%)": average frequency of duration that the robot has distances to any other agents less than comfortable distance (0.2m).

C. Comparison with State-of-the-art Methods

We show the comparison of performance between FSRL (ours) and state-of-the-art methods in Fig. 5 and Tab. I, II under invisible and visible settings. We can easily find that FSRL achieves the best performance on almost all scenarios.

Fig. 5 shows the success, collision and timeout rates. No matter if in the invisible or visible setting, FSRL achieves the highest success rates and the lowest rates of collision and timeout. Since learning-based methods, such as LSTM-RL or SARL, choose the optimal action only based on the current states, they are usually short-sighted. In particular, a nonholonomic robot, i.e. with limited rotations, cannot rotate at a wide angle when meeting obstacles, so it needs longer navigation time, easier gets trapped, or even fails. As shown in Fig. 5, there also are more timeouts or collisions. As expected, FSRL performs the best as it is foresighted and not only

considers the current state but also estimates future situations. The nonholonomic robot thus can take proper actions in advance to move smoothly and safely.

Table I: "Nav. Time" quantitative results in 3 environments

Methods	Nav Time (invisible)				Nav Time (visible)			
	Env.1	Env.2	Env.3	AVG	Env.1	Env.2	Env.3	AVG
ORCA [12]	12.49	11.98	12.04	12.14	11.99	10.97	10.74	11.23
LSTM-RL [2]	15.52	14.12	12.81	14.15	11.01	10.96	12.32	11.29
SARL [3]	12.25	11.39	11.15	11.60	9.85	10.33	10.01	10.06
FSRL(*ours*)	10.90	10.92	11.69	11.04	11.08	10.32	11.60	11.00

Table II: "Disc. (%)" quantitative results in 3 environments.

Methods	Disc(%) (invisible)				Disc(%) (visible)			
	Env.1	Env.2	Env.3	AVG	Env.1	Env.2	Env.3	AVG
ORCA [12]	0.39	0.38	0.43	0.40	0.41	0.41	0.45	0.42
LSTM-RL [2]	0.07	0.28	0.16	0.17	0.11	0.35	0.39	0.20
SARL [3]	0.01	0.06	0.06	0.04	0.09	0.11	0.10	0.10
FSRL(*ours*)	0.01	0.07	0.04	0.04	0.06	0.08	0.07	0.07

In the **invisible** setting, as anticipated, the ORCA method seriously fails due to the violated reciprocal assumption. However, it is interesting that the ORCA robot has much better performance under Env. 2, 3 than under Env. 1. This is because non-learning-based methods, like ORCA, are more sensitive to dynamic objects. Hence, fewer dynamic objects will lead to better performance.

In contrast, there is a significant performance drop for learning-based methods (LSTM-RL, SARL) when Env. 1 is transformed into Env. 2, 3. This is due to them having difficulties to learn a good representation for a complex environment where both standing and dynamic objects exist, compared to a simpler environment where either standing or dynamic objects exist. It demonstrates that previous learning-based methods are usually suitable for simple scenarios and do not perform well in a more complex environment. In comparison, the LSTM-RL method is more conservative, which leads to timeout cases increasing sharply and also to longer navigation times in success cases (Tab. I) . In contrast, FSRL has the shortest navigation time and the lowest discomfort rate. These results demonstrate the foresightedness of FSRL, allowing for a better balance between safety and efficiency.

In the visible setting, although the robot more easily navigates through the crowd due to the aid by the give-way behaviors of humans, it needs to understand more about interactions with humans to face the increasing uncertainty. For example, there is an increased likelihood that humans with give-way behavior will enter the comfort zone of the robot. In addition, due to the mutual human-robot interaction, the robot is prone to take risky behaviors to achieve better navigation. Thus, unlike in the invisible case, the performance of all methods improves except for the discomfort rate.

For LSTM-RL and ORCA methods, they are shortsighted and even fail to take standing agents into account in Env. 2,3. They thus navigate unsafely as well as inefficiently, as seen by a low success rate, a longer navigation time, and a high discomfort rate. As shown in Fig. 5, FSRL performs as well as the SARL method. Compared with SARL, although FSRL takes longer (less than 1s on average) in terms of the navigation time, it decreases the discomfort rate by 3% by moving the robot more safely (Tab. II).

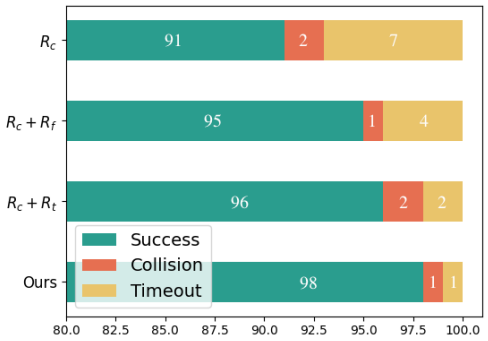


Fig. 6: Average quantitative results under three environments of ablation experiments. (Invisible setting)

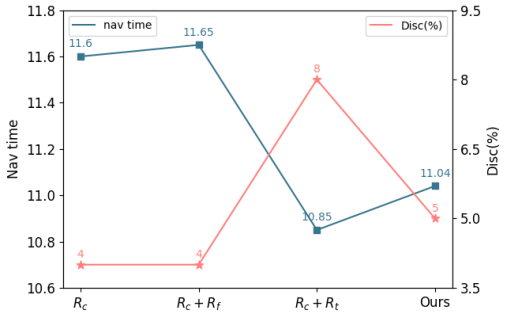
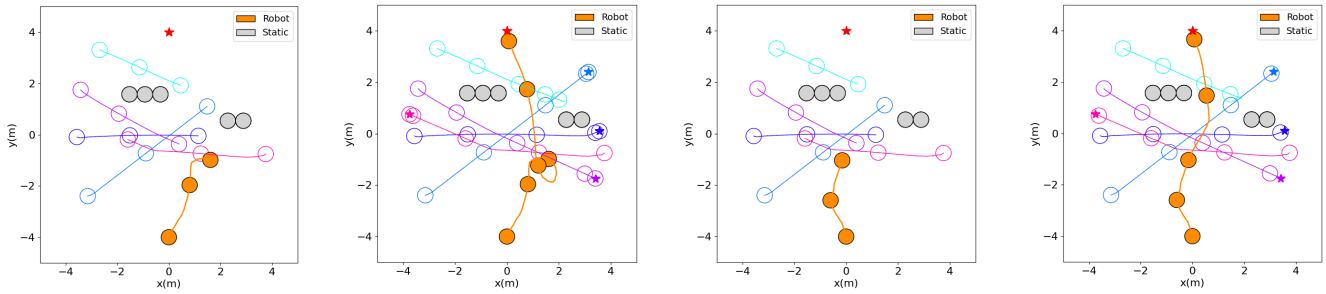


Fig. 7: Average "Nav. Time" and "Disc(%)" results under three environments of ablation experiments. (Invisible setting)

We can see the FSRL metrics in the visible cases are as well as in the invisible cases. In an overall comparison to previous methods, FSRL is more effective, efficient, and robust.

D. Ablation Study

We show the impact of each component of our reward R in Fig. 6 and 7, where R_c is the reward that only considers the current state, Eq.(4), R_f is the foresight penalty, Eq. (5), and R_t is the efficiency constraint, Eq. (8). For clarity, we take the average of the quantitative results under three environments for comparison. In Fig. 6, we can observe that both foresight penalty (R_f) and efficiency constraints (R_t) can improve the success rate (Succ.) a lot. Furthermore, for the efficiency constraint (R_t), the improvement mainly comes from shortening the navigation time shown in Fig. 7. In other words, some timeout cases are avoided due to higher efficiency, as can be seen by the reduced navigation time. However, this improvement is at the cost of aggressive navigation, resulting in a much higher discomfort rate (Disc.), which means the robot will disturb people intensively to save navigation time. In contrast, the foresight penalty (R_f)



(a) SARL (t = 6) (b) SARL (t = 14.25) (c) FSRL (t = 6) (d) FSRL (t = 11.50)

Fig. 8: Qualitative results. We find our method (FSRL) can make robot (orange) navigate more smoothly and efficiently in a complex environment compared to SARL [3] where standing (grey solid) and dynamic (other colors) objects exist.

can improve the success rate (Succ.) with less disturbance by caring more about a comfortable distance. By combing both foresight penalty (R_f) and efficiency constraints (R_t), our method can achieve the best success rate (Succ.), while balancing efficiency and human feeling (less disturbance) well. This demonstrates the effectiveness of our design.

E. Qualitative Evalutaion

We show qualitative results for Env. 2 (invisible setting) in Fig. 8. At time $t = 6$, SARL turns around sharply close to people. It is because SARL makes decisions only according to the current state, it hence neglects potential collisions in the future. Furthermore, it also results in a longer navigation time (14.25s) to arrive at the goal. As a comparison, the navigation trajectory of FSRL is much more smooth. This is due to FSRL's capability to forecast potential collisions in the future and to take corresponding actions in advance. The navigation time is also shorter ($t = 11.50$), which shows the efficiency of FSRL. Hence FSRL learns a better strategy that is more effective and efficient.

V. CONCLUSIONS

In this work, we propose a novel Foresight Social-Aware Reinforcement Learning (FSRL) framework for robotics navigation. Our method can estimate potential collisions in the future and hence takes action in advance to avoid collisions. In contrast, previous methods only make the decision according to the current state thus perform not well when the environment becomes complex. Furthermore, an efficiency constraint is introduced in our work that reduces navigation time a lot. We conduct experiments on three increasingly challenging environments and our method achieves the best performance regarding both effectiveness and efficiency.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT Press, 2018.

[2] 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.

[3] 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.

[4] 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.

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

[6] J. Jin, N. M. Nguyen, N. Sakib, D. Graves, H. Yao, and M. Jagersand, "Mapless navigation among dynamics with social-safety-awareness: a reinforcement learning approach from 2d laser scans," in *IEEE Internat. Conf. on Robotics and Automation (ICRA)*, 2020, pp. 6979–6985.

[7] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *IEEE Internatational Conference on Robotics and Automation (ICRA)*, 2021, pp. 3517–3524.

[8] X. Gao, R. Gao, P. Liang, Q. Zhang, R. Deng, and W. Zhu, "A hybrid tracking control strategy for nonholonomic wheeled mobile robot incorporating deep reinforcement learning approach," *IEEE Access*, vol. 9, pp. 15 592–15 602, 2021.

[9] G. Ferrer, A. Garrell, and A. Sanfeliu, "Social-aware robot navigation in urban environments," in *European Conference on Mobile Robots*, 2013, pp. 331–336.

[10] G. Ferrer, A. G. Zulueta, F. H. Cotarelo, and A. Sanfeliu, "Robot social-aware navigation framework to accompany people walking side-by-side," *Autonomous robots*, vol. 41, no. 4, pp. 775–793, 2017.

[11] X.-T. Truong and T. D. Ngo, "Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model," *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 4, pp. 1743–1760, 2017.

[12] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Springer Berlin Heidelberg, 2011, pp. 3–19.

[13] J. van den Berg, Ming 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.

[14] 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.

[15] K. Li, M. Shan, K. Narula, S. Worrall, and E. Nebot, "Socially aware crowd navigation with multimodal pedestrian trajectory prediction for autonomous vehicles," in *IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–8.

[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] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1685–1692.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[19] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.