Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

SARL*: Deep Reinforcement Learning based Human-Aware Navigation for Mobile Robot in Indoor Environments

Keyu Li, Yangxin Xu, Jiankun Wang and Max Q.-H. Meng*, *Fellow, IEEE*
Department of Electronic Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong SAR, China
{*kyli, yxxu, jkwang1992*}*@link.cuhk.edu.hk, max.meng@cuhk.edu.hk*

***Abstract*— In a human-robot coexisting environment, reaching the goal position safely and efficiently is essential for a mobile service robot. In this paper, we present an advanced version of the Socially Attentive Reinforcement Learning (SARL) algorithm, namely SARL*, to achieve human-aware navigation in indoor environments. Recently, deep reinforcement learning has achieved great success in generating human-aware navigation policies. However, there exist some limitations in the real-world implementations: the learned navigation policies are limited to certain distances associated with the training process, and the simplification of the environment neglects obstacles other than humans. In this work, we improve the state-of-the-art SARL algorithm by introducing a dynamic local goal setting mechanism and a map-based safe action space to tackle the above problems. Real-world experimental results demonstrate that the proposed algorithm outperforms the original SARL algorithm in both time cost and path length in the human-aware navigation tasks in the indoor environment.**

***Index Terms*— mobile robot, human-aware navigation, deep reinforcement learning, collision avoidance, indoor navigation**

I. Introduction

As a fundamental topic in robotics, mobile robot navigation has been extensively studied. In the past, robots were mainly deployed in human-free industrial environments, such as the Automatic Guided Vehicles (AGV) [1]. Recently, due to the rising labor costs and aging population, more service robots are developed to work in human-robot coexisting environments, in fields such as autonomous vehicle [2], smart wheelchairs [3], luggage collection robot [4] and in countless others.

Collision avoidance is a necessary capability for mobile robots to navigate in crowded scenarios such as airports and shopping malls. As human beings, we have the inborn ability to adjust our behavior by observing others, so we can easily pass through the crowds. However, collision-free navigation in a highly dynamic and crowded scenario remains a daunting task for mobile robots. Collision avoidance modules in the traditional navigation framework usually treat dynamic obstacles as static, such as the Dynamic Window Approach (DWA) [5], or just focus on the next step of action according to certain interaction rules, such as Reciprocal Velocity Obstacle (RVO) [6] and Optimal Reciprocal Collision Avoidance (ORCA) [7]. Since these methods prevent collisions by passive reaction and often use some manually defined functions to guarantee safety, they are found to cause the robot's movement to be unnatural, short-sighted and unsafe.

In order to navigate in a socially compliant manner in the crowd, a robot needs the ability to perceive, understand and anticipate the behavior of the surrounding pedestrians. The combination of human motion prediction and motion planning of the robot remains a challenge in the human-aware navigation task. One idea is to plan after prediction, i.e., to determine a safe path after predicting other people's future trajectories. Some hand-crafted models (e.g., constant velocity model [8], discrete choice model [9][10], social force model and its variants [11][12]) and data-driven methods (e.g., social LSTM [13], social GAN [14]) have been proposed to encode the intentions of pedestrians to forecast their future trajectories. However, the computational cost and the reliability of pedestrian trajectory prediction are often affected by the high stochasticity of the crowd behavior. In the context of human-aware navigation for mobile robot that requires both security and time efficiency, such planning-after-prediction methods are still challenging for practical applications.

Another stream of methods for human-aware navigation are based on Deep Reinforcement Learning (DRL), which integrate the prediction of human motion into the decision-making process, i.e., the robot learns from experiences to understand the crowded scenarios and encode the crowd-robot interaction in the navigation policy. Recent works in this field [15][16][17][18] have demonstrated superior ability to generate crowd-aware navigation policies in simulation, among which the Socially Attentive Reinforcement Learning (SARL) algorithm achieves the state-of-the-art performance. However, there exist some limitations for the real-world implementations: a) the learned navigation policy is limited to certain distances associated with the training process, causing

*Corresponding author.

This work is partially supported by Hong Kong ITC ITSP Tier2 grant #ITS/105/18FP and Hong Kong ITC MRP grant #MRP/011/18 to Max Q.-H. Meng.

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

the robot to be short-sighted and make detours when the navigation goal is far away; and b) the representation of the environment is simplified to only include the robot and the crowd, which would result in the inability for the robot to avoid obstacles other than humans in the environment. These shortcomings would limit the navigation performance of the robot in the real-world implementations.

Based on the SARL model specified in [18], we present a novel algorithm, namely SARL*, to tackle the aforementioned problems. The main contributions of our work are three-fold:

- We address the first problem by introducing a global path planning module and designing a dynamic local goal setting mechanism;
- A dynamic safe action space with a map-based action screening mechanism is proposed to tackle the second problem; and
- We build a human-aware navigation system based on the Robot Operating System (ROS), which can be applicable to mobile robots that operate in human-robot coexisting environments.

The rest of this paper is organized as follows. Section II provides the formulation of the human-aware navigation problem. Section III introduces the details of the proposed SARL* algorithm. The system setup and real-world experiments are presented in Section IV, before we draw some conclusions in Section V.

II. Problem Formulation

Suppose there is a robot and n humans in a 2D workspace (X-Y plane), each agent can be simplified as a circle. We assume that the position, velocity and radius of each agent can be observed by other agents, which are denoted as $\boldsymbol{p} = [p_x, p_y]$, $\boldsymbol{v} = [v_x, v_y]$ and r, respectively. The orientation θ, goal position $\boldsymbol{g} = [g_x, g_y]$ and preferred speed v_{pref} cannot be observed by other agents. It is assumed that humans do not avoid the robot during the navigation. The state of the robot at time t can be defined as $\boldsymbol{S}_t = [p_x, p_y, v_x, v_y, \theta, g_x, g_y, r, v_{pref}]$, and the observable state of the i-th human at time t can be defined as $\boldsymbol{O}_t^i = [p_x^i, p_y^i, v_x^i, v_y^i, r^i, v_{pref}^i]$. A robot-centric frame defined in [15] is adopted to make the state representation more general, where the origin is set at the current position of the robot $\boldsymbol{p}_t$, and X-axis points at the goal position $\boldsymbol{g}$. Let d_g denote the distance from $\boldsymbol{p}_t$ to $\boldsymbol{g}$, and d^i denote the distance from $\boldsymbol{p}_t$ to the position of the i-th human $\boldsymbol{p}_t^i$. After transformation, the states $\boldsymbol{S}_t$,$\boldsymbol{O}_t^i$ in the robot-centric frame become:

$$\begin{aligned} \boldsymbol{S}_t &= [d_g, v_{pref}, v_x, v_y, r], \\ \boldsymbol{O}_t^i &= [d^i, p_x^i, p_y^i, v_x^i, v_y^i, r^i, r^i + r]. \end{aligned} \tag{1}$$

The joint state of all $(n+1)$ agents at time t can be obtained by concatenating the state of the robot with the observable states of humans as $\boldsymbol{J}_t = [\boldsymbol{S}_t, \boldsymbol{O}_t^1, \boldsymbol{O}_t^2, \ldots, \boldsymbol{O}_t^n]$. Assume that the robot can change its velocity immediately according to the action command at time t which is determined by the navigation policy: $\boldsymbol{v}_t = \boldsymbol{a}_t = \boldsymbol{\pi}(\boldsymbol{J}_t)$. The corresponding reward at time t is denoted as $R(\boldsymbol{J}_t, \boldsymbol{a}_t)$. We follow the definition of reward function in [15], where d_{min} is the shortest separation distance between the robot and humans within the decision interval Δt, and d_c represents the minimum comfortable distance that humans can tolerate:

$$R(\boldsymbol{J}_t, \boldsymbol{a}_t) = \begin{cases} -0.25, & \text{if } d_{min} < 0; \\ 0.5 * (d_{min} - d_c), & \text{if } 0 < d_{min} < d_c; \\ 1, & \text{if } d_g = 0; \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Equation (2) indicates that actions that lead the robot to the goal will be awarded while actions that may cause collision or discomfort to pedestrians will be penalized.

With an optimal navigation policy $\boldsymbol{\pi}^\star(\boldsymbol{J}_t)$, the optimal value of the joint state $\boldsymbol{J}_t$ at time t can be formulated as:

$$V^\star(\boldsymbol{J}_t) = \sum_{i=0}^{K} \gamma^{i \cdot \Delta t \cdot v_{pref}} \cdot R(\boldsymbol{J}_t, \boldsymbol{a}_t^\star), \tag{3}$$

where K is the total number of decision steps from the state at time t to the final state, Δt is the decision interval between two actions, and $\gamma \in (0, 1)$ is a discount factor in which the preferred speed v_{pref} is introduced as a normalization parameter for numerical reasons [15]. The optimal policy is formulated by maximizing the cumulative reward as the following:

$$\begin{aligned} \boldsymbol{\pi}^\star(\boldsymbol{J}_t) =& \underset{\boldsymbol{a}_t \in \boldsymbol{A}}{\arg\max}\, R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot \\ & \int_{\boldsymbol{J}_{t+\Delta t}} P(\boldsymbol{J}_{t+\Delta t} \mid \boldsymbol{J}_t, \boldsymbol{a}_t) \cdot V^\star(\boldsymbol{J}_{t+\Delta t})\, \mathrm{d}\boldsymbol{J}_{t+\Delta t}, \end{aligned} \tag{4}$$

where $\boldsymbol{A}$ is the action space (i.e., the set of velocities that can be achieved), $P(\boldsymbol{J}_{t+\Delta t} \mid \boldsymbol{J}_t, \boldsymbol{a}_t)$ is the transition probability from $\boldsymbol{J}_t$ to $\boldsymbol{J}_{t+\Delta t}$ when action $\boldsymbol{a}_t$ is carried out. This probability describes the uncertainty of the next joint state due to the unknown crowd behavior.

Considering the fact that the intentions of pedestrians are difficult to predict, and that the duration Δt is very small, it is acceptable to assume that the pedestrians move with a constant velocity during the time interval $[t, t + \Delta t]$. Therefore, by using a constant velocity model, we can estimate the next states of humans and approximate the next joint state: $\boldsymbol{J}_{t+\Delta t} \leftarrow \text{propagate}(\boldsymbol{J}_t, \Delta t, \boldsymbol{a}_t)$. By this simplification, the calculation of the optimal policy in (4) is modified as:

$$\boldsymbol{\pi}^\star(\boldsymbol{J}_t) = \underset{\boldsymbol{a}_t \in \boldsymbol{A}}{\arg\max}\, R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot V^\star(\boldsymbol{J}_{t+\Delta t}). \tag{5}$$

III. Algorithms

State-of-the-art algorithms such as CADRL [15] and SARL [18] have proposed different deep neural networks to approximate the optimal value of the next state $V^{\star}(\boldsymbol{J}_{t+\Delta t})$ in (5). Their navigation strategies can be considered as the same, which is outlined in Algorithm 1. With a pre-trained value network N, a robot plans its path to the goal by continuously selecting actions that maximize the cumulative reward (line 4-6). Note that the optimal value function $V^{\star}$ is estimated by the output of the value network N (line 6).

Algorithm 1: DRL-based navigation strategy (e.g., SARL [18])

```
1 Initialize the action space $\boldsymbol{A}$;
2 Load the pre-trained value network $N$;
3 Set goal position $\boldsymbol{g}$;
4 while Goal not reached do
5     Update the joint state of the robot and the crowd $\boldsymbol{J}_t$;
6     $\boldsymbol{a}_t \leftarrow \arg\max_{\boldsymbol{a}_t \in \boldsymbol{A}} R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot N(\boldsymbol{J}_{t+\Delta t})$;
7 end
8 $\boldsymbol{a}_t \leftarrow$ Stop();
```

Although such methods have achieved great performance in simulation, there exist some limitations in practice. The learned navigation policy of the robot is limited to certain distances associated with the RL training process, which will cause the robot to be short-sighted and make detours when the goal position is far away. Besides, the designed strategy can only handle the environment without obstacles other than humans. These shortcomings will limit the performance of the robot in the real-world navigation tasks.

This paper proposes an advanced version of the SARL algorithm for practical application in the crowded indoor environment, namely SARL*, as outlined in Algorithm 2.

Algorithm 2: SARL*

```
Initialize the action space $\boldsymbol{A}$;
Load the pre-trained value network $N$;
Build the 2D occupancy grid map of the environment $map$;
Set global goal position $\boldsymbol{g}_{global}$;
while global goal not reached do
    Update the joint state of the robot and the crowd $\boldsymbol{J}_t$;
    $GlobalPlan \leftarrow$ Dijkstra($\boldsymbol{p}_t, \boldsymbol{g}_{global}, map$);
    $\boldsymbol{g}_{local} \leftarrow$ FindLocalGoal($\boldsymbol{p}_t, GlobalPlan, d$);
    $\boldsymbol{a}_t \leftarrow \arg\max_{\boldsymbol{a}_t \in \boldsymbol{A}_s} R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot N(\boldsymbol{J}_{t+\Delta t})$,
    where $\boldsymbol{g} \leftarrow \boldsymbol{g}_{local}$,
        $\boldsymbol{A}_s \leftarrow$ FindSafeActionSpace($\boldsymbol{p}_t, \boldsymbol{A}, map$);
end
$\boldsymbol{a}_t \leftarrow$ Stop();
```

A. Value Network

In order to predict the optimal value of the next state, we adopt the deep neural network architecture presented in the SARL algorithm [18], which uses a socially attentive pooling module to capture the relative importance of each human in the crowd, as shown in Fig. 1.

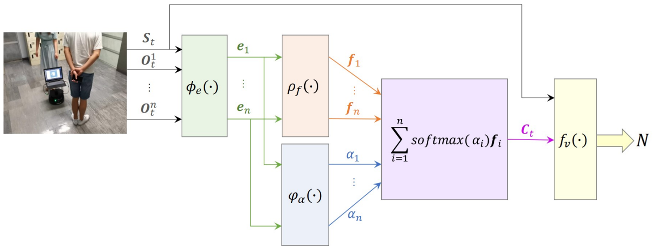


Fig. 1. The deep value network is comprised of multiple MLPs with ReLU nonlinearities, which inputs the joint state of the robot and humans and outputs an estimation of the optimal value function.

Firstly, a multi-layer perceptron (MLP) $\phi_e(\cdot)$ with ReLU nonlinearities learns the embedding weight of each element in each human-robot state pair to generate n fixed length vectors. Then the embedding vectors are input to a following MLP $\rho_f(\cdot)$ to obtain the pairwise human-robot interaction features. In parallel, the embedding vectors are fed to another MLP $\varphi_\alpha(\cdot)$ to get the attention scores of each human, which can represent their relative importance to the robot. A linear combination of the interaction features weighted by the attention scores is subsequently obtained as the crowd feature, as shown in (9). Finally, a MLP $f_v(\cdot)$ calculates the value of the crowd-robot interaction feature as an estimation of the optimal value function $V^{\star}(\boldsymbol{J}_t)$. The following equations present the formulation of each module.

$$\boldsymbol{e}_i = \phi_e(\boldsymbol{S}_t, \boldsymbol{O}_t^i), i = 1, ..., n \tag{6}$$

$$\boldsymbol{f}_i = \rho_f(\boldsymbol{e}_i), i = 1, ..., n \tag{7}$$

$$\boldsymbol{\alpha}_i = \varphi_\alpha(\boldsymbol{e}_i), i = 1, ..., n \tag{8}$$

$$\boldsymbol{C}_t = \sum_{i=1}^{n} \text{softmax}(\alpha_i) f_i \tag{9}$$

$$N(\boldsymbol{J}_t) = f_v(\boldsymbol{S}_t, \boldsymbol{C}_t) \tag{10}$$

The training procedure of the value network is outlined in Algorithm 3. From step 1 to 3, the value network is initialized where the robot mimics the navigation policy of human experts to finish some demonstration experiences. From step 4 to 18, the value network is optimized with reinforcement learning, where the robot gains experiences from exploration. Step 7 uses a ε-greedy policy to choose actions, i.e., randomly choose an action with the probability of ε and choose the optimal action with the probability of $1 - \varepsilon$.

In implementation, the value network is trained with PyTorch [19] in a simulation environment built with the OpenAI Gym library [20]. The sizes of hidden layers in $\phi_e(\cdot)$, $\rho_f(\cdot)$, $\varphi_\alpha(\cdot)$, and $f_v(\cdot)$ are set to (150, 100), (100, 50), (100, 100) and (150, 100, 100), respectively. r is set to $0.3m$, v_{pref} is set to $1m/s$, Δt is set to $0.25s$, d_c is set to $0.2m$, and t_{max} is

Algorithm 3: Value Network Training

```
Initialize experience replay memory $E$ with demonstration;
Initialize the value network $N$ with memory $E$;
Initialize the target value network $N' \leftarrow N$;
for $episode = 1, \ldots, M$ do
    Initialize the joint state $\boldsymbol{J}_{t=0}$ randomly;
    repeat
        $\boldsymbol{a}_t \leftarrow \begin{cases} \text{RandomAction}(), & \varepsilon \\ \arg\max_{\boldsymbol{a}_t} R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot N(\boldsymbol{J}_{t+\Delta t}), & 1-\varepsilon \end{cases}$;
        $value \leftarrow R(\boldsymbol{J}_t, \boldsymbol{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \cdot N'(\boldsymbol{J}_{t+\Delta t})$;
        $state \leftarrow \boldsymbol{J}_{t+\Delta t}$;
        Enrich experience $E \leftarrow (state, value)$;
        Optimize value network $N$ with experience $E$;
        Update time and state $t \leftarrow t + \Delta t, \boldsymbol{J}_t \leftarrow \boldsymbol{J}_{t+\Delta t}$;
    until final state is reached $\boldsymbol{J}_t = \boldsymbol{J}_{end}$ or timeout $t > t_{max}$;
    if $episode \mid TargetUpdateInterval$ then
        Update the target value network $N' \leftarrow N$
    end
end
return $N$
```

set to $25s$. Each agent navigates from a random position on a circle with a radius of $4m$ to a goal position on the opposite of the circle. 5 simulated pedestrians navigates with the Optimal Reciprocal Collision Avoidance (ORCA) policy [7]. For initialization, the robot first finishes $3,000$ demonstration episodes with the ORCA policy, and the model is trained 50 epochs with the gradient descent method (learning rate: 0.01). Then the robot completes $20,000$ episodes where ε linearly decays from 0.5 to 0.1 in the first $4,000$ episodes and remains unchanged thereafter. γ is set to 0.9. The value network is optimized after each episode using minibatches of size 100 and a learning rate of 0.001. The target value network is updated every 50 episodes. The entire training process took about 21 hours on an i5-6300HQ CPU, and the rate of the robot successfully reaching the goal without collision reaches 99% on the 500 test cases with the same simulation settings.

B. Dynamic Local Goal Setting

The traditional path planning problem typically includes a global planner that uses a priori information of the environment to find the best route and a local planner that adjusts the global plan to avoid dynamic obstacles. This motivates us to introduce a global planner module and design a dynamic local goal setting mechanism (line 7-8 in Algorithm 2) to achieve human-aware navigation with high distance efficiency. Given a 2D occupancy grid map of the environment, a global path with minimum cost is generated with the Dijkstra algorithm [21] between the current position of the robot $\boldsymbol{p}_t$ and the global goal $\boldsymbol{g}_{global}$. Then all the waypoints on the global path are traversed to find the nearest waypoint in the circle centered at $\boldsymbol{p}_t$ with a diameter of d, which is set as the dynamic local goal $\boldsymbol{g}_{local}$. As illustrated in Fig. 2, the local goal position is dynamically changed since the global path is calculated based on the current position of the robot. At each decision, the goal position $\boldsymbol{g}$ specified in the joint state $\boldsymbol{J}_t$ is updated to the current local goal until the robot reaches the global goal position.

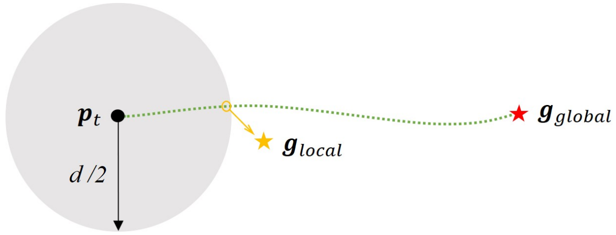


Fig. 2. Illustration of the dynamic local goal setting mechanism. At time t, the local goal position is set at the nearest waypoint on the global path plan in the circle centered at the robot position with a diameter of d.

C. Safe Action Space

A map-based safe action space is introduced in line 9 in Algorithm 2 to screen out dangerous actions from the initial action space. At the beginning of the algorithm, the action space is initialized based on the kinematic constraints of the robot. An action can be described by the linear and angular velocity of the robot, namely v and ω, where the positive value of v indicates the forward direction, and the positive value of ω indicates the clockwise direction. The action space can be formed as a combination of all achievable values of the (v, ω) pair:

$$\boldsymbol{A} = \{(v, \omega) \mid v \in X, \omega \in Y\} \tag{11}$$

Inspired by the basic idea of the Dynamic Window Approach (DWA) algorithm [5], we propose a map-based velocity screening mechanism to form a safe action space, which enables the robot to avoid known obstacles in the indoor environment. The safe action space $\boldsymbol{A}_s$ is constructed in each decision step according to the current position of the robot, the 2D occupancy grid map of the environment and the initial action space: $\boldsymbol{A}_s \leftarrow \text{FindSafeActionSpace}(\boldsymbol{p}_t, \boldsymbol{A}, map)$. For each velocity in the action space, a forward simulation is performed to see whether the robot would collide with obstacles in the map if the velocity is applied for time t_{sim}, as shown in Fig. 3. The velocities that lead to collision in the forward simulation are considered unsafe and are excluded from the action space. The screened velocities constitute the safe action space, from which the action command is selected based on (5). Through a number of experiments, we found that it can offer the best performance when the forward simulation time t_{sim} is set to $2\Delta t$.

For robots with unicycle kinematic constrains (i.e., the angular velocity is limited in a certain range), if there is no safe velocity left after the screening, the robot will be ordered to rotate in place to find an opportunity to escape.

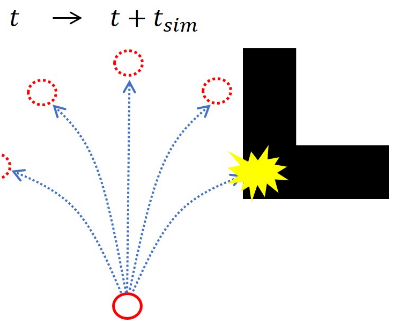


Fig. 3. The map-based velocity screening mechanism. The black block represents the occupied area in the map. Velocities that lead to collision in the forward simulation of t_{sim} are excluded from the action space.

IV. Experiments

In this section, we compare the proposed SARL* algorithm with the original SARL algorithm, where the time cost and the path length are selected as the metrics. Note that the original SARL algorithm is not able to avoid collision with obstacles other than humans in the environment, and in our implementation, a map-based action screening module is added as described in Section III for safety concerns.

A. Experiment Setup

In order to evaluate the proposed SARL* algorithm in the real-world experiments, we build a mobile robot navigation system as shown in Fig. 4. A TurtleBot 2 robot with a Hokuyo UTM-30LX 2D lidar serves as the experimental platform, and the action commands for the robot is generated by a Lenovo ideapad 700-15ISK laptop installed with Ubuntu 16.04 and ROS Kinetic [22]. A lobby with a roughly rectangular shape ($19m\times14m$) is used as the indoor experimental environment. The map of the environment, the location of the robot, and the position and velocity of the pedestrians are obtained using the ROS packages *gmapping* [23], *amcl* [24] and *leg_detector* [25], respectively.

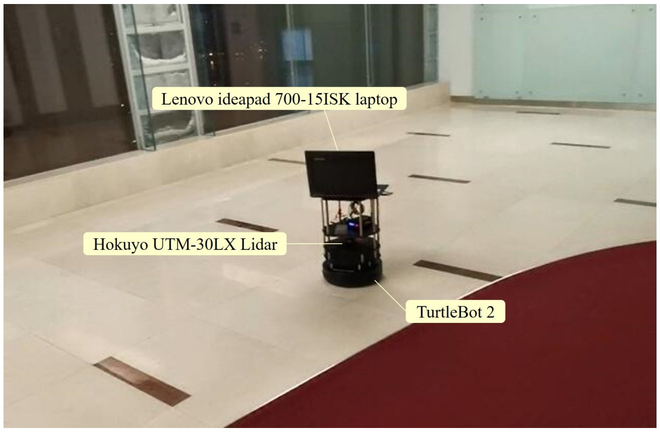


Fig. 4. The mobile robot navigation system consisting of a Turtlebot 2 robot, a Hokuyo UTM-30LX lidar sensor and a laptop in the indoor experimental environment.

Based on the hardware limitations, the experimental parameters are configured as follows. The radius of the robot and humans is set to $0.3m$, the preferred velocity of the robot is set to $0.5m/s$, the decision interval of the robot is set to $0.25s$, and the navigation time limit is set to $120s$. The minimum comfortable distance for human is set to $0.5m$ to increase tolerance for the delay of pedestrian detection and tracking. The parameter d in the dynamic local goal setting mechanism is set to $5m$. The action space is initialized by (11) where X and Y are given by (12) and (13), respectively.

$$X = \left\{ v_{pref} \cdot \frac{e^{i/5}-1}{e-1} \;\middle|\; i = 0,1,2,3,4,5 \right\} \tag{12}$$

$$Y = \left\{ -\frac{\pi}{4} + \frac{\pi}{12} \cdot i \;\middle|\; i = 0,1,2,3,4,5,6 \right\} \tag{13}$$

B. Navigation Without Human

Firstly, we compare our method with the baseline in a navigation experiment without human to see their sensitivity to distances. The navigation time with different goal settings is presented in Table I. When the distance to goal is within $5m$, the navigation time of the two methods is similar, but as the distance increases, the navigation time of the baseline increases quickly, while that of our proposed SARL* algorithm increases smoothly and almost proportionally with the goal distance. When the goal is more than $11m$ away, the robot fails to navigate to the goal with the baseline method, but can successfully reach the goal with high time efficiency using the SARL* algorithm.

Fig. 5 illustrates the trajectories generated by the two methods as the navigation distance increases from $6m$ to $12m$ for a detailed analysis. As we can see in Fig. 5 (a), when using the baseline algorithm, the robot makes a number of detours at the beginning of navigation, and only takes a shortcut until it gets close enough to the goal. As the goal distance increases, the robot spends more time making detours, which strongly affects its time and distance efficiency in navigation. When the goal is more than $12m$ away from the robot, it becomes "blind" and cannot generate a policy to navigate to the goal. In contrast, with the proposed SARL* algorithm, the robot moves almost straight towards the goal when there is no human in the workspace, as shown in Fig. 5 (b). This result validates that by dynamically setting local goals, the path planning ability of the robot is not affected by navigation distance, and the robot can move toward the goal with higher time and distance efficiency.

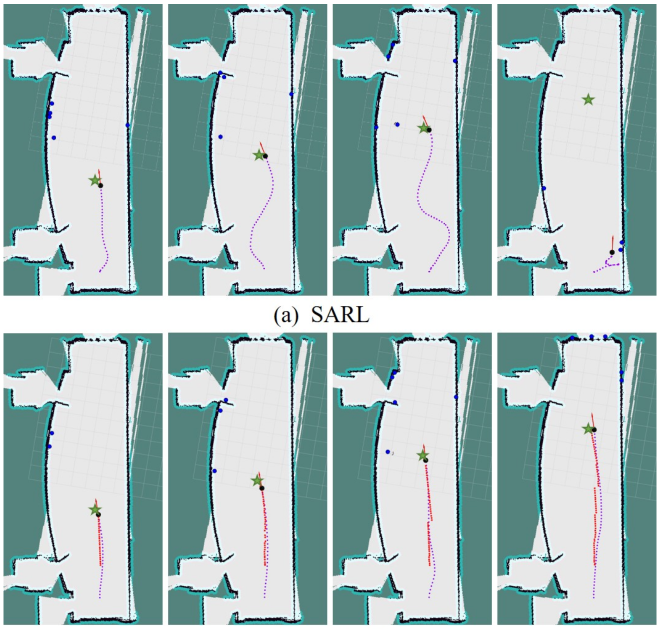
C. Navigation in a Human-Robot Coexisting Scenario

In order to further evaluate the effectiveness of our proposed method, we conduct another set of experiments in a human-robot coexisting scenario, where the the robot is obstructed by a pedestrian for n times ($n = 0 \sim 3$) during the

TABLE I

NAVIGATION TIME OF THE TWO METHODS IN AN ENVIRONMENT WITHOUT HUMAN

Navigation time (s)		Goal distance (m)												
		3	4	5	6	7	8	9	10	11	12	13	14	15
Methods	SARL	7.00	9.25	11.50	15.50	20.00	19.75	22.00	29.00	37.50	∞	∞	∞	∞
	SARL*	7.00	**9.00**	11.50	**13.75**	**15.50**	**17.50**	**19.75**	**21.75**	**23.75**	**26.00**	**28.25**	**30.00**	**32.50**


(a) SARL

(b) SARL*

Fig. 5. The trajectories generated by the SARL algorithm and our proposed SARL* algorithm with the goal distance of 6m, 8m, 10m, 12m in an environment without human are presented in (a) and (b), respectively. The green star marks the goal position, the black circle is the final position of the robot, and the purple dots show the trajectory of the robot. The red dots in (b) mark the dynamic local goals set by the SARL* algorithm during the navigation.

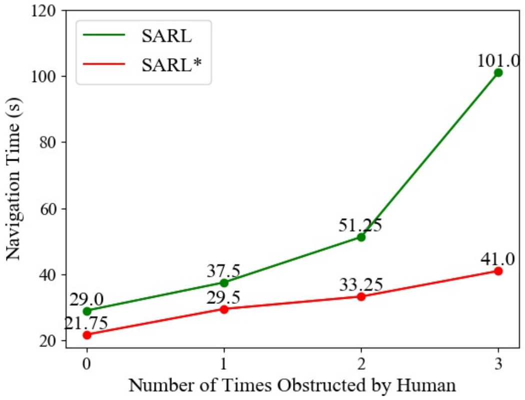


Fig. 6. The navigation time of the SARL algorithm and our proposed SARL* algorithm when the robot is obstructed by human for 0 to 3 times. As the number of human interferences increases, the navigation time of the SARL algorithm rises significantly, while that of the SARL* algorithm only witnesses a slight increase.

navigation process. The distance between the starting position and the goal position is set as $10m$. From the two navigation time curves in Fig. 6, we can see that when the number of times the robot is obstructed by human increases, the navigation time of our method is less affected compared to the baseline, which indicates that our method is more stable to human interference in long-distance navigation than the baseline.

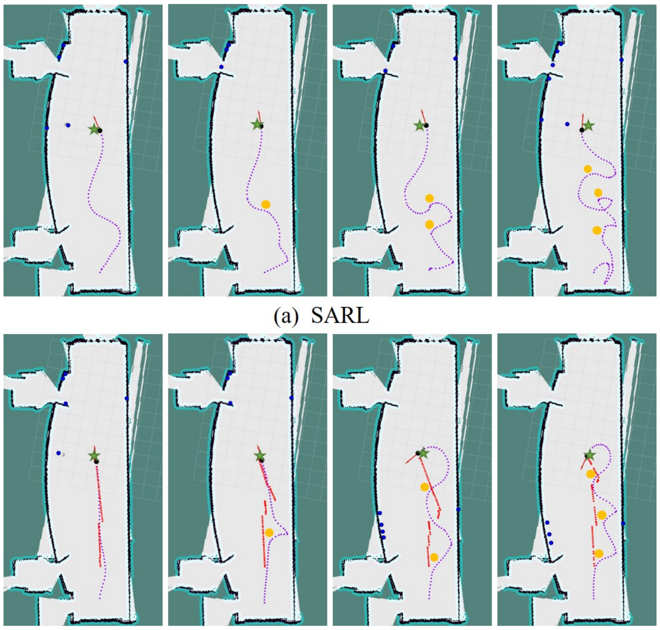
For a clear comparison and discussion, we take a close look at the trajectories generated by the two methods when the robot is obstructed by the pedestrian for 0 to 3 times, which are presented in Fig. 7. As shown in Fig. 7 (a) and (b), both methods demonstrate strong flexibility in the avoidance behavior. The difference lies in that SARL usually turns a large angle or even heads to the opposite direction of the goal to avoid the human, causing the robot to take a number of detours and end up with a long navigation time. In contrast, SARL* prefers the velocities close to the direction of the goal when turning to avoid collision, which makes the robot reach the goal quickly. The results demonstrate that the dynamic local goal setting introduced in the proposed SARL* algorithm can effectively increase the time and distance efficiency in the human-aware navigation for mobile robots.

D. Navigation in the Dense Crowd

In addition to the experiments above, we also carried out navigation experiments in the dense crowd in order to verify the effectiveness of our proposed method in more realistic scenarios. The crowd-robot interaction scenarios include vertical crossing and center crossing, aiming at simulating the crowded scenarios in public places such as airports and train stations. The video demonstrations are available at `https://youtu.be/izrrctuUd-g` and `https://www.bilibili.com/video/av74520445`.

V. CONCLUSIONS

In this work, based on the SARL algorithm, we present the SARL* algorithm to tackle the problems in the real indoor implementation of human-aware navigation for mobile robots. A dynamic local goal setting mechanism and a map-based safe action screening mechanism are proposed to


(a) SARL

(b) SARL*

Fig. 7. The trajectories generated by the SARL algorithm and our proposed SARL* algorithm when the robot is obstructed 0 to 3 times by human with the goal distance of 10m are presented in (a) and (b), respectively. The yellow circles mark the positions where the robot is obstructed by the human. Both methods avoid pedestrians flexibly, but the original SARL algorithm takes a lot of detours, while our method navigates with higher distance and time efficiency.

achieve efficient navigation in the human-robot coexisting environment. Real-world experiment results reveal that the proposed SARL* algorithm outperforms the original SARL in both time cost and path length in the human-aware navigation in indoor environments, and achieves convincing performance in the dense crowd. Future research should consider about dynamic obstacles other than humans in the complex environment, and improve the speed and accuracy of pedestrian detection and tracking algorithms.

Acknowledgment

We really appreciate the members in our lab for their kind help and cooperation in the experiments. We also would like to thank Changan, the first author of the SARL algorithm, for his discussion and encouragement.

References

[1] R. C. Arkin and R. R. Murphy, "Autonomous navigation in a manufacturing environment," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 4, pp. 445–454, 1990.

[2] A. Ferdowsi, U. Challita, W. Saad, and N. B. Mandayam, "Robust deep reinforcement learning for security and safety in autonomous vehicle systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 307–312, IEEE, 2018.

[3] Y. Morales, A. Watanabe, F. Ferreri, J. Even, K. Shinozawa, and N. Hagita, "Passenger discomfort map for autonomous navigation in a robotic wheelchair," *Robotics and Autonomous Systems*, vol. 103, pp. 13–26, 2018.

[4] J. Wang and M. Q.-H. Meng, "Socially compliant path planning for robotic autonomous luggage trolley collection at airports," *Sensors*, vol. 19, no. 12, p. 2759, 2019.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *2008 IEEE International Conference on Robotics and Automation*, pp. 1928–1935, IEEE, 2008.

[7] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*, pp. 3–19, Springer, 2011.

[8] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, IEEE, 2009.

[9] G. Antonini, S. Venegas, J.-P. Thiran, and M. Bierlaire, "A discrete choice pedestrian behavior model for pedestrian detection in visual tracking systems," in *Advanced Concepts for Intelligent Vision Systems*, no. CONF, 2004.

[10] G. Antonini, M. Bierlaire, and M. Weber, "Discrete choice models of pedestrian walking behavior," *Transportation Research Part B: Methodological*, vol. 40, no. 8, pp. 667–687, 2006.

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

[12] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with gaussian processes," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pp. 1229–1234, IEEE, 2009.

[13] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971, 2016.

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

[15] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 285–292, IEEE, 2017.

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

[17] P. Long, T. Fanl, X. Liao, W. Liu, H. Zhang, and J. Pan, "Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6252–6259, IEEE, 2018.

[18] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," *arXiv preprint arXiv:1809.08835*, 2018.

[19] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

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

[21] S. Skiena, "Dijkstra's algorithm," in *Implementing discrete mathematics: combinatorics and graph theory with mathematica*, pp. 225–227, Addison-Wesley Reading, MA, 1990.

[22] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009.

[23] B. Gerkey, "Gmapping ros package." `http://wiki.ros.org/gmapping/`, 2015. Accessed: 2019-08-20.

[24] B. Gerkey, "Amcl ros package." `http://wiki.ros.org/amcl/`, 2015. Accessed: 2019-08-20.

[25] C. Pantofaru, "Leg_detector ros package." `http://wiki.ros.org/leg_detector/`, 2011. Accessed: 2019-08-20.