2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

SCAN: Socially-Aware Navigation Using Monte Carlo Tree Search

Jeongwoo Oh[1,3*], Jaeseok Heo[1,3*], Junseo Lee[2], Gunmin Lee[1],
Minjae Kang[1], Jeongho Park[1], and Songhwai Oh[1,2,3]

Abstract— **Designing a socially-aware navigation method for crowded environments has become a critical issue in robotics. In order to perform navigation in a crowded environment without causing discomfort to nearby pedestrians, it is necessary to design a global planner that is able to consider both human-robot interaction (HRI) and prediction of future states. In this paper, we propose a socially-aware global planner called SCAN, which is a global planner that generates appropriate local goals considering HRI and prediction of future states. Our method simulates future states considering the effects of the robot's actions on the future intentions of pedestrians using Monte Carlo tree search (MCTS), which estimates the quality of local goals. For fast simulation, we execute pedestrian motion prediction using Y-net and future state simulation using MCTS in parallel. Neural networks are only used in Y-net and not in MCTS, which enables fast simulation and prediction of a long horizon of future states. We evaluate the proposed method based on the proposed socially-aware navigation metric using realistic pedestrian simulation and real-world experiments. The results show that the proposed method outperforms existing methods significantly, indicating the importance of considering human-robot interaction for socially-aware navigation.**

I. Introduction

The application of autonomous robots in crowded environments has infiltrated our daily lives [1], [2]. In order to make robots assist humans in such environments, robots must be able to navigate in a socially-friendly manner, such that nearby pedestrians do not feel discomfort. However, developing such navigation methods is a challenging task due to two reasons. First, as pedestrians are dynamic obstacles, conventional static obstacle avoidance methods have shortcomings when navigating in crowd, as the environment constantly changes [3]. Second, pedestrians are reactive obstacles meaning the future intentions of humans can be influenced by the current action of a robot [4], [5]. Due to these reasons, human-robot interaction (HRI) must be considered for socially-aware navigation.

To compensate for the above difficulties, previous methods have treated this task as either decoupled prediction and planning or coupled prediction and planning [1]. Traditional approaches consider decoupled prediction and planning, predicting the future trajectories of pedestrians before finding possible paths for the robot to traverse [3], [6], [7]. In such cases, it is challenging to consider HRI, and it can lead to the reciprocal dance problem where the human and robot repeatedly select incompatible actions and can not pass each other [8]. On the other hand, coupled prediction and planning methods consider the effects of HRI during the path planning process. Early works have designed the reactive movements of pedestrians and planned the best action considering HRI [9], [10]. Recent methods have proposed learning-based approaches using deep learning to embed pedestrian motion and to find optimal actions simultaneously in a crowded environment [11]–[18]. Although learning-based methods have shown promising results for reaching local goals with short travel distances, they are limited to a low-level controller, which can not be used alone when the robot is placed in large environments since they only consider a short horizon of future steps. Due to this fact, these social navigation methods must be used with a global planner, which finds an efficient local goal for reaching the global goal considering HRI.

Traditional global planners for static environments [19]–[21] are inadequate for socially-aware navigation as they are vulnerable to environmental changes [22]. To resolve this problem, dynamic global planners have been proposed, which consider the nearby spatial and temporal information along with the global path to guide the robot towards the global path [23], [24]. Since these methods do not consider HRI, they can lead the robot to potentially socially-undesirable or hazardous situations.

To alleviate the aforementioned problems, it is necessary to develop a global planner, which can generate appropriate local goals considering a long horizon of future states and HRI. One way to solve the problem is to simulate future states using Monte Carlo tree search (MCTS) to estimate the quality of local goals. Pedestrian trajectory prediction and the effect of the robot's movement on pedestrian motion must be considered for future state simulation. Eiffert *et al.* has proposed MCTS-GRNN, a socially-aware local planner that uses a recurrent neural network to simulate the next state. Using this model, single-step simulation is performed for estimating the next state. The value of the next state is calculated using the distance between the goal position and the robot's position in the next state, and the uncertainty of the predicted robot's position. Since the computational cost of neural networks is expensive, it is challenging for MCTS-GRNN to simulate a long horizon of future states in real-time.

In this paper, we propose SCAN: a real-time global planner for socially-aware navigation. We first execute a traditional path planner for generating local goal candidates toward the global goal. For each local goal candidate, we estimate its efficiency and social-awareness using MCTS to simulate the future states of both the robot and surrounding pedestrians considering their interactive behavior. Finally, we select an appropriate local goal candidate by its value, which considers both path efficiency and social awareness. Unlike previous global planners, the proposed method considers HRI and is better suited for long-distance social navigation tasks. In SCAN, we only use a neural network, namely Y-net [25], for global pedestrian trajectory prediction. Since the simulation for MCTS in our method and global pedestrian trajectory prediction are performed in parallel, the algorithm is able to be executed in real-time. Thus, we are able to predict a long horizon of future states and find promising local goals, which are path efficient, socially acceptable, and less vulnerable to potential dangers.

In summary, the contributions of this paper are as follows:

- To the best of our knowledge, we are the first to

*Equal Contribution

[1]Department of Electrical and Computer Enginnering and ASRI, Seoul National University, Seoul, Korea (e-mail: jeongwoo.oh, jaeseok.heo, gunmin.lee, minjae.kang, jeongho.park@rllab.snu.ac.kr; songhwai@snu.ac.kr), [2]Graduate School of Artificial Intelligence (GSAI) and ASRI, Seoul National University, Seoul, Korea (e-mail: junseo.lee@rllab.snu.ac.kr), [3]Sequor Robotics, Inc., Seoul, Korea. (Corresponding author: Songhwai Oh.)

This work was supported by Institue of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2019-0-01190, [SW Star Lab] Robot Learning: Efficient, Safe, and Socially-Acceptable Machine Learning).

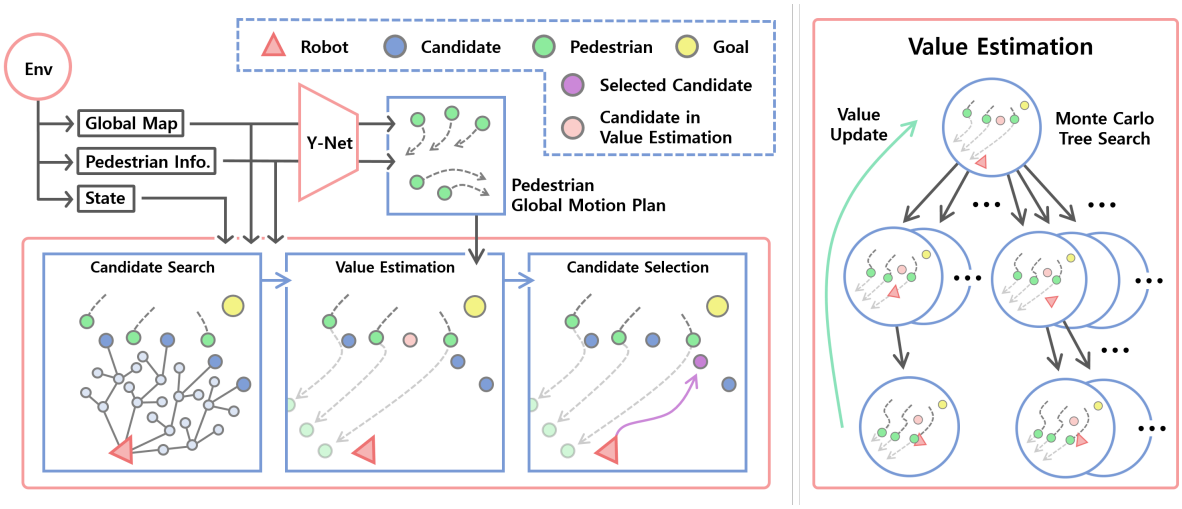

Fig. 1: **Overview of the proposed method, SCAN** SCAN outputs a local goal that is safe and high-valued for reaching the global goal efficiently and safely while considering social-awareness. First, we sample a diverse set of local goal candidates. Then for each candidate, we estimate the value and cost of each local goal candidate by executing MCTS. Finally, we select the local goal with the best value satisfying a given cost threshold.

introduce a socially-aware global planner using MCTS considering both HRI and long horizon future states for generating appropriate local goals.
- We propose a realistic pedestrian simulator and an evaluation metric for comparing various socially-aware navigation methods.
- We show the effectiveness of the proposed method in simulation and real-world experiments with a physical robot.

II. Background

A. Pedestrian Motion Modeling

We propose to divide pedestrian motion modeling hierarchically into global and local motion modeling. In the case of global motion modeling, it designs the path assuming pedestrians move without considering environmental changes. Mangalam *et. al* [25] have proposed Y-net, which predicts and generates pedestrian trajectories considering the map information. This model combines encoding features of the segmented map and past movement history and then predicts the future trajectory using a heatmap on the map.

Local motion modeling designs and controls how pedestrians interact with walls and obstacles they encounter while moving along a path toward a goal in the presence of dynamic obstacles such as robots and other pedestrians. Helbing *et al.* [9] have proposed the social force model to design the local movement of pedestrians using four terms: 1) attractive force toward the goal; 2) repulsive force from the wall; 3) repulsive force from a moving person; 4) random variations of the movement. If each person moves independently following this model, we can obtain realistic pedestrian local motions. Reynolds [26] has proposed Boids, a group behavior where groups avoid each other and each individual moves to cohere within the group. Each objects moves considering three components: the direction of the group movement, the direction toward the center of the associated group, and the direction of moving away from other group.

In this paper, Y-net is used for modeling global motion, while the social force model and Boids are used for modeling local motion to synthesize realistic pedestrian motions.

B. Monte Carlo Tree Search

Monte Carlo tree search (MCTS) is a search method for selecting an optimal sequence of decisions by simulating rollouts via random action sampling and using the result to build a search tree. It is one of the popular methods along with reinforcement learning for solving problems that can be formulated as a Markov decision process (MDP). MCTS uses a tree policy, which balances exploration and exploitation. The tree policy selects actions leading to future states with the best expected reward sum. On top of that, the tree policy considers the number of state-action visitation counts when selecting the next action for better exploration.

MCTS can be divided in to four stages: selection, expansion, simulation, and backpropagation. In the selection stage, the algorithm starts from the root node and selects successive children nodes until it reaches a leaf node. For selecting the successive children node, the tree policy considers both the visitation count and the expected reward summation of the children node for balancing exploration and exploitation. Once the algorithm reaches the leaf node, the expansion stage is executed. If the visitation count of the current leaf node exceeds a certain threshold, the search tree is expanded by adding new leaf nodes. After the expansion stage, the algorithm selects a new leaf node. Next, the simulation stage is executed where the future states are simulated starting from the current leaf node until the terminal state is reached or the time limit is exceeded. Finally, the backpropagation stage is executed, in which the information in the nodes on the path from the root node to the current leaf node is updated using the simulated results. Upon conducting the tree search, the larger the tree gets, the more promising state nodes and action edges are found. In the proposed method, we have used MCTS for measuring the quality of each local candidate by simulating the future.

III. Socially-Aware Navigation Using Monte Carlo Tree Search

In this section, we introduce *SCAN*, the proposed navigation method. We assume that the robot is placed in a crowded environment where pedestrians move randomly. It is assumed that a map of the environment and the (global) goal are given. This path planner aims to find the most appropriate local goals towards the global goal, which balances efficiency and social acceptance for reaching the global goal. The

proposed method is divided into candidate generation, value estimation, and candidate selection. Figure 1 shows the overview of SCAN.

A. Local Goal Candidate Search

The proposed method begins by finding a set of local goal candidates. To execute this process, it is necessary to generate appropriate candidates by planning various paths from the robot's current position to the global goal. In order to acquire local goal candidates, we draw multiple optimal paths toward the global goal. The number of paths drawn is identical to the number of local goal candidates. Creating the optimal path proceeds serially, and when the optimal path is created, high costs are given to regions around the planned path to prevent two different paths from overlapping one another. As a result, multiple non-overlapping paths can be obtained. These paths are generated using CARRT* [21], which requires a cost function, a root, and a goal. The root and goal are the robot's current position and the global goal, respectively. The cost function contains two components, costs extracted from a costmap and pedestrians. The costmap is composed of the global map cost, static obstacle cost, and the cost by other paths. The global map cost is the predefined cost near obstacles such as walls, pillars, and stairs. The static obstacle cost is the cost from surrounding static obstacles detected in real-time. In both simulation and real-world experiments, we use 2D Hokuyo LiDAR sensor data and set the static obstacle cost high in regions with high point cloud density. High costs are given where previously generated paths exist to generate diverse non-overlapping paths. For cost extracted from pedestrians, high costs are given to nearby pedestrians. Considering the continuity of cost and the fact that a higher cost should be received in the presence of a nearby obstacle, the cost function was designed as $Me^{-\beta d}$, where M is the maximum value of this cost function, β is a decaying hyperparameter, and d is the distance from the cost source or pedestrians to the robot. If one location has multiple sources of cost, the maximum cost among the sources is used as the cost of the location.

B. Candidate Value Estimation via MCTS

The next part of *SCAN* is value estimation. Each local goal candidate's value V is calculated using (1). V is used directly for selecting the local goal among candidates. V_t is a value obtained through MCTS, which represents the quality of the local goal candidate considering social awareness. V_l is a value inversely proportional to the distance from the local goal candidate to the previous local goal. V_l is added to adjust the unstable movement of the low-level controller when the local goal fluctuates. V_g is a value inversely proportional to the distance toward the global goal. Since our goal is to reach the global goal as efficiently as possible, we should assign large V_g to the candidate with a shorter distance towards the global goal. The hyperparameters λ_t, λ_l, λ_g control the influence of each term.

$$V = \lambda_t V_t + \lambda_l V_l + \lambda_g V_g \tag{1}$$

Let S be the state space of the robot, which contains position of a robot and pedestrians, and the map information. A is a finite quantized action space of the robot. $r : S \times A \to \mathbb{R}$ and $c : S \times A \to \mathbb{R}$ are the reward and cost functions, respectively. The reward function represents driving efficiency, while the cost function reflects the safety and social acceptance of driving. $V_r : S \to \mathbb{R}$ is the value function which can be computed as $V_r(s) := \mathbb{E}_\pi[\sum_{t=0}^{T} r(s_t, a_t)\gamma^t | s_0 = s, s_t \in S, a_t \in A \ \forall t]$. $V_c : S \to \mathbb{R}$ is the cost value function, which can be computed as $V_c(s) := \mathbb{E}_\pi[\sum_{t=0}^{T} c(s_t, a_t)\gamma^t | s_0 = s, s_t \in S, a_t \in A \ \forall t]$. $Q_r : S \times A \to \mathbb{R}$ is the expected state-action value function at (s, a), where $Q_r(s, a) := r(s, a) + V_r(s')$ and s' is the returned state acquired from executing action a at state s. $Q_c : S \times A \to \mathbb{R}$ is the expected state-action cost value function at (s, a), where $Q_c(s, a) := r(s, a) + V_c(s')$. $N(s)$ is the number of visitation to state s, and $N(s, a)$ is the number of visitation of (s, a), and γ is a discount factor.

The selection stage in MCTS proceeds weighted sampling with weight of $w(s, a)$, which is similar to the sampling weight defined in [27], until a leaf node is found.

The definition of the weight can be written as:

$$w(s, a) := \exp\left(Q_r(s, a) - \lambda Q_c(s, a) + \frac{\alpha}{1 + N(s, a)}\right), \tag{2}$$

where α is the hyperparameter for exploration and λ is the hyperparameter, which leverages driving efficiency, safety, and social-awareness. If the number of visitations of the leaf node exceeds a certain threshold, the expansion stage runs at the leaf node.

In the simulation stage, we evaluate the leaf node. The reward, cost, and simulation step rules must be defined for simulation. The definition of the reward and cost can be written as:

$$\begin{aligned} &r(s_t, a_t) = \omega_f r_f, \\ &r_f = -\Delta dist(x_r^t, x_c), \\ &c(s_t, a_t) = \omega_s c_s + \omega_p c_p, \\ &c_s = M_s \exp(-\beta_s dist(x_r^t, O^t)), \\ &c_p = M_p \exp(-\beta_p dist(x_r^t, P^t)), \end{aligned} \tag{3}$$

where ω_f, ω_s, and ω_p are the weight hyperparameters of forward reward, static obstacle cost, and pedestrian cost, respectively. M_s and M_p are the maximum values of the static obstacle cost and pedestrian cost, respectively. β_s and β_p are the decaying hyperparameters of static obstacle cost and pedestrian cost, respectively. x_r^t is the position of the robot at time t, and x_c is the position of the local goal candidate. $dist(x, y)$ is the Euclidean distance between two points x and y. $dist(x, Y) = \min_{y \in Y} dist(x, y)$, where Y is a set of points. $O^t = \{o_1, o_2, ..., o_n\}$ is the set of obstacles at time t and o_i is a polygon of wall or static obstacle. $P^t = \{p_1, p_2, ..., p_m\}$ is the set of pedestrians at time t, where p_j is the position of the jth pedestrian.

Next, we define the simulation step rules. For each simulation step, the robot moves according to the given action. The pedestrians follow the path prediction result obtained through Y-net [25] running in parallel to MCTS. The pedestrians also attempts to avoid the robot by following the rule of the social force model [9].

We assume that the low-level controller has a greedy policy considering only a short horizon of future states. Thus, in order to select an action at each step of the simulation, we use softmax sampling considering one-step lookahead: (4).

$$p(a|s) = \frac{\exp((r(s, a) - \lambda c(s, a))/\tau)}{\sum_{a' \in A} \exp((r(s, a') - \lambda c(s, a'))/\tau)}, \tag{4}$$

where τ is the temperature hyperparameter. The sampling probability of this sampling method is proportional to $r - \lambda c$. We execute the simulation until a terminal state is reached or the depth of the simulated state reaches the maximum simulation depth D, where D is a sufficiently large value. From the simulation stage, we are able to calculate the simulated expected discounted reward summation $z_r :=$

$\sum_{t=d}^{T} r(s_t, a_t)\gamma^{t-d}$, and the discounted cost summation $z_c := \sum_{t=d}^{T} c(s_t, a_t)\gamma^{t-d}$. a_t is the action sampled at the simulated node at depth t from the root node, s_t is the simulated state at depth t, d is the depth of the leaf node, and T is the depth of the node where the simulation terminates. If $T < D$, the simulation terminates prematurely either because the robot collides with an obstacle or the robot reaches the goal. If a collision occurs between the robot and the environment, we assume the robot receives a cost of $C_{max} := M_s + M_p$ over an infinite time horizon. Therefore, $C_{max}\gamma^{T-d}/(1-\gamma)$ is added to z_c. If $T = D$, it means that the simulation is terminated by timeout. In this case, we calculate $\bar{u} = (dist(x_r^0, x_c) - dist(x_r^T, x_c))/T$, which is the average change of the distance between the robot and the local goal candidate for each timestep, where x_r^0 and x_r^T are the position of the robot in s_0 and s_T, respectively. Then, we estimate the remaining time step $T' = dist(x_r^T, x_c)/\bar{u}$ for reaching the local goal candidate. If $\bar{u}$ is not positive, we set T' to $+\infty$. Then, we add the value $\bar{r}$ and the cost $\bar{c}$ along the time horizon of T', where $\bar{r} := \bar{u}$ and $\bar{c} := \frac{z_c(1-\gamma)}{1-\gamma^{T-d}}$. In summary, we obtain z_r and z_c shown in (5).

$$z_r \leftarrow \begin{cases} z_r + \frac{\bar{r}\gamma^{T-d}(1-\gamma^{T'})}{1-\gamma} & \text{if timeout and } \bar{d} > 0 \\ z_r + \frac{\bar{r}\gamma^{T-d}}{1-\gamma} & \text{if timeout and } \bar{d} \leq 0 \\ z_r & \text{otherwise} \end{cases}$$

$$z_c \leftarrow \begin{cases} z_c + \frac{\bar{c}\gamma^{T-d}(1-\gamma^{T'})}{1-\gamma} & \text{if timeout and } \bar{d} > 0 \\ z_c + \frac{\bar{c}\gamma^{T-d}}{1-\gamma} & \text{if timeout and } \bar{d} \leq 0 \\ z_c + \frac{C_{max}\gamma^{T-d}}{1-\gamma} & \text{if collision} \\ z_c & \text{otherwise} \end{cases} \tag{5}$$

Finally, the V_r-value, V_c-value, and sampling weights are updated in the backup stage following the parent nodes from the leaf node to the root node. The sampling weights are updated as in (2), while $V_r(s)$ and $V_c(s)$ are updated as shown in (6), where s' is the next state of state s where action a is executed.

$$V_r(s) \leftarrow \begin{cases} \frac{N(s)V_r(s)+z_r}{N(s)+1} & \text{if } s \text{ is a leaf node} \\ \frac{\sum_{a\in A} w(s,a)\big(r(s,a)+\gamma V_r(s')\big)}{\sum_{a\in A} w(s,a)} & \text{otherwise} \end{cases}$$

$$V_c(s) \leftarrow \begin{cases} \frac{N(s)V_c(s)+z_c}{N(s)+1} & \text{if } s \text{ is a leaf node} \\ \frac{\sum_{a\in A} w(s,a)\big(c(s,a)+\gamma V_c(s')\big)}{\sum_{a\in A} w(s,a)} & \text{otherwise} \end{cases} \tag{6}$$

C. Candidate Selection and Control

The last part of our method is to pick a local goal. If the cost value of the candidate exceeds a certain threshold, it is considered hazardous and excluded from selection. We select the one with the best value as the local goal among the safe remaining candidates. If all candidates are considered as dangerous, we deem the current state dangerous, and an emergency stop signal is transmitted to the low-level controller to stop the robot.

IV. Experimental Setup

In this section, we evaluate the proposed method in realistic pedestrian scenarios. We have tested the proposed method to check whether or not it can perform socially-aware navigation when placed in a random crowded environment. In the remainder of this section, we explain the details for setting up the simulation environment, the metric for evaluating the performance, and the baselines used for comparison. The experimental results can be found in Section V.

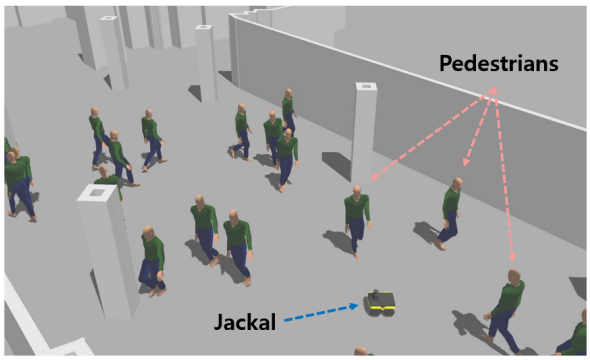

Fig. 2: **Gazebo simulator with realistic pedestrian motion**

A. Simulation Environment Setup

We have conducted our experiments in the Gazebo simulator [28], a commonly used simulator for testing simple robotic tasks. We have implemented a crowd navigation environment as shown in Figure 2, where each pedestrian tries to follow its global trajectory generated by Y-Net [25]. Also, the local motion of a pedestrian is modeled using the social force model [9]. To simulate pedestrians moving in groups, we have used the Boids model [26].

In this experiment, the robot has to navigate to a global goal while avoiding collisions with obstacles. We have conducted our experiments with various numbers of pedestrians and goal distances in 100 different episodes. We divide the difficulty by the distance to the goal. The distance from the initial point to the global goal in 'easy', 'medium', and 'hard' difficulties are $(20-30\ m)$, $(30-40\ m)$, and $(40-50\ m)$, respectively. For all difficulties, we have evaluated two different pedestrian densities: 'sparse' and 'dense', where the numbers of pedestrians are 15 and 30, respectively.

B. Evaluation Metric

For evaluation, we present a new socially-aware navigation score metric (**SANS**), which is composed of the following:

- **Comfortability rate (CR)**: the percentage of episodes where the robot successfully reaches the goal while not approaching a pedestrian or obstacle within a socially uncomfortable distance, which is set to $0.3\ m$.
- **Speed score (SP)**: the score considering the average speed (v) of the robot. The average speed has the lower bound of $0.5\ m/s$ and upper bound of $1.5\ m/s$. The score is calculated by normalizing the speed.
- **Path efficiency (PE)**: the length of the distance toward the global goal from the initial position of the robot divided by the total traveled distance.
- **Safety score (SF)**: the metric for measuring safety performance when in potentially dangerous situations, which is measured as: $1 - \min\{(a_1 + 2a_{0.5})/a_{1.5}, 1\}$, where a_d is the number of timesteps when the robot is within $d\ m$ from an obstacle.
- **Stablity score (ST)**: the number of timesteps where an obstacle is within $1.5\ m$ from the robot divided by the episode length.

SANS has been devised to reflect success, safety, and social acceptance. Comfortability rate **(CR)** is the measure of success. The episode terminates and is considered a failure when the robot approaches surrounding pedestrians

or obstacles within the socially uncomfortable distance. The speed score **(SP)** is the metric for measuring speed, which incentivizes the robot to move at the speed of regular pedestrian walking speed. Path efficiency **(PE)** is the metric for measuring the efficiency of the robot's path. Safety score **(SF)** is the metric to check whether the robot drives safely in potentially dangerous situations. This metric is important as such situations can be inevitable when the robot navigates in a crowded environment. Therefore, **SF** is the most important metric for measuring **SANS** alongside **CR**. The stability score **ST** is the metric for measuring how often the robot enters potentially dangerous states. **SANS** is calculated by: $\textbf{SANS} := \textbf{CR} \times (10 \times \textbf{SP} + 10 \times \textbf{PE} + 50 \times \textbf{SF} + 30 \times \textbf{ST})$.

C. Implementation Details

We have set the weights λ_t, λ_l, and λ_g for calculating the value of the local goal candidates as 1.0, 0.5, and 0.1, respectively. For reward and cost parameters, $\omega_f, \omega_s, \omega_p$ are each set to 2.0, 0.05, and 0.3, while β_s, β_p are each set to 0.05, 1.4. The maximum static and pedestrian costs, M_s and M_p are each set to 0.8 and 1.0, respectively. The cost threshold for selecting the final local goal is set to 1.5. Finally, the exploration parameter α and weighted sampling parameter λ are set to 1.0, and 0.2, respectively.

In the local goal candidate search stage, we execute CARRT* for 0.2 seconds and aquire the local goal candidates. In the candidate value estimation stage, MCTS is executed for another 0.2 seconds. Therefore, SCAN runs within 0.4 seconds for real-time operation.

For the low-level controller for reaching local goals, we have used a policy trained using TRC [29], a safe reinforcement learning method.

D. Baselines

For comparison, the following methods are used:

- **MCTS-CV**: This method is a spinoff of the proposed method, which uses a constant speed model in MCTS without considering HRI. The constant speed model assumes that the velocity of all pedestrians is $1.5\ m/s$.
- **CARRT* [21]**: We use CARRT* to plan a path to the global goal and try to follow the planned path.
- **CARRT*-Rep**: We replan the path to the global goal using CARRT* at a fixed time interval of $0.4\ seconds$.
- **Sample-based MPC**: For each local goal candidate, we sample random trajectories and select the one with the highest value within a given cost threshold. We use the value of this trajectory as the value estimate.
- **TRC [29]**: This is a safe RL method which can navigate while avoiding static and dynamic obstacles. We use this low-level controller without a global planner.
- **SARL [18]**: This method models pairwise interactions between the human and a robot with a self-attention mechanism for navigating in crowded environments. We use this method for the low-level controller without a global planner.
- **CADRL [16]**: This method is a reinforcement learning method with social norm violation constraints. Similar to SARL, we use this low-level controller without a global planner.

V. Results

A. Simulation Experiments

The simulation results are shown in Table I. SCAN shows the best results for both **CR** and **SANS** in all cases except for the sparse case with easy difficulty. The **CR** and **SANS** are improved by 1.17 times and 1.12 times, respectively, compared to the second best performing method in the dense case with easy difficulty. In the case of dense case with medium and hard difficulty, the proposed algorithm shows improvement of 1.46 times and 1.23 times in **CR**, and 1.34 times and 1.16 times in **SANS**, respectively, when compared to second best performing method. In the case of sparse density case, the proposed algorithm shows the best performance, except for the **SANS** in the easy case, with performance increase varying from 1.06 times to 1.41 times, when compared to second best performing method. Since SCAN simulates future states considering HRI using MCTS to find appropriate local goals, the robot can avoid potentially dangerous regions and reach the global goal efficiently and safely while considering social-awareness. MCTS-CV shows worse performance than SCAN. Since MCTS-CV does not consider HRI, it can make wrong predictions about the future states of pedestrians, leading to degraded performance. CARRT* shows better **SANS** compared to SCAN in easy difficulty with sparse pedestrian density as there are not many obstacles and the travel distance is short. However, the performance degrades when the difficulty increases or the pedestrian density is high. From this, we can see the importance of using a global planner that considers dynamic environmental changes. CARRT*-Rep assumes the surrounding obstacles are static during the replanning process and does not consider the future states, making the agent vulnerable to possible hazards, leading to worse performance than SCAN. MPC shows inferior performance compared SCAN. MPC samples random trajectories instead of using tree search to estimate the value of local goal candidates. Therefore, it cannot properly estimate the value of local candidates when an insufficient number of trajectories are sampled. Finally, TRC, SARL, and CADRL have all showed poor performance. From the results, we can see the importance of using a socially-aware global planner.

Density	Method	Easy		Medium		Hard	
		CR	SCANS	CR	SCANS	CR	SCANS
Sparse	**SCAN**	**0.83**	56.63	**0.80**	**52.82**	**0.76**	**49.76**
	MCTS-CV	0.67	30.21	0.66	40.90	0.50	31.35
	CARRT* [21]	0.78	**58.20**	0.65	45.78	0.54	36.18
	CARRT*-Rep	0.74	43.44	0.70	41.33	0.52	36.20
	MPC	0.69	51.37	0.63	45.25	0.43	28.99
	TRC [29]	0.73	52.08	0.59	37.59	0.48	31.58
	SARL [18]	0.75	56.12	0.54	37.80	0.50	36.51
	CADRL [16]	0.39	23.32	0.32	18.97	0.26	16.30
Dense	**SCAN**	**0.77**	**50.96**	**0.73**	**45.72**	**0.48**	**29.62**
	MCTS-CV	0.60	37.41	0.44	31.19	0.27	17.95
	CARRT* [21]	0.66	45.69	0.50	33.64	0.26	16.79
	CARRT*-Rep	0.56	37.85	0.44	25.08	0.34	22.14
	MPC	0.64	45.24	0.47	30.32	0.36	18.37
	TRC [29]	0.55	35.25	0.42	26.87	0.39	25.43
	SARL [18]	0.46	30.61	0.48	34.05	0.36	24.24
	CADRL [16]	0.31	16.80	0.15	8.95	0.08	4.64

TABLE I: **Global planner comparison results.** The table shows the performance of global planners on the realistic pedestrian simulator. The maximum values of CR and SANS are 1 and 100, respectively.

Method	CR (1)	SANS (100)	SP (10)	PE (10)	SF (50)	ST (30)
SCAN	**0.48**	**29.62**	4.30	8.87	24.12	24.43
MCTS-CV	0.27	17.96	8.68	9.12	23.67	25.03
CARRT* [21]	0.26	16.79	**10.00**	9.54	17.80	**27.24**
CARRT*-Rep	0.34	22.14	**10.00**	9.41	19.12	26.60
MPC	0.36	18.37	**10.00**	9.34	22.57	9.12
TRC [29]	0.39	25.43	**10.00**	**9.60**	20.11	25.51
SARL [18]	0.36	24.24	7.63	7.75	**26.62**	25.34
CADRL [16]	0.08	4.64	6.82	7.26	20.40	23.53

TABLE II: **Detailed score results.** The table shows the results in dense pedestrian scenarios with hard difficulty. The number inside each parenthesis represents the maximum score for its driving score evaluation standards.

Method	Avg. Depth	No. of Steps	Time (μs)
SCAN	6.46	45,828	0.67
MCTS-GRNN [11]	2.06	895	450

TABLE III: **Computational speed analysis results.** The comparison of the usage of neural network in the environment with high density and hard difficulty environment.

Method	CR (1)	SANS (100)	SP (10)	PE (10)	SF (50)	ST (30)
SCAN	**0.9**	**71.57**	4.20	8.83	**40.61**	**25.89**
MCTS-CV	0.7	52.29	4.24	8.70	37.14	24.63
CARRT*-Rep [21]	0.6	45.06	4.39	**9.06**	38.57	25.48
MPC	0.4	31.00	**4.44**	8.70	38.55	23.41

TABLE IV: **Results of real-world experiments.** Identical to Table II, the number inside each parenthesis represents the maximum score for its driving score evaluation standards.

B. Detailed Score Analysis

The detailed scores for calculating SANS in hard difficulty with a dense pedestrian environment are shown in Table II. SCAN shows the best **CR** and **SANS** compared to baseline methods. It can be seen that SCAN can proceed through the environment safely, considering social-awareness. The proposed method shows the highest **SF** among the baseline algorithms, except for SARL. SARL shows higher **SF** score due to the fact that the algorithm controls the agent to take a long detour when the agent is confronted by pedestrians. On the other hand, the proposed method shows higher **SANS** score, as the algorithm controls the agent to move slowly when the agent is near pedestrians. This movement induces the pedestrians to move away from the agent, allowing stable movements of the agent.

C. Computational Speed Analysis

We have compared the simulation inference time of the proposed method and MCTS-GRNN [11]. Due to its heavy computation time, MCTS-GRNN slows down the simulation. Since it is difficult to conduct fair comparison, we have only used this method for computational speed analysis. We have measured the average MCTS tree depth, the number of simulation steps executed within the time limit of 0.2 seconds, and the inference time of a single simulation step. For MCTS-GRNN, we have replaced the simulation time of each step with the inference time of passing the state information through the network. Experiments are conducted with Intel® Core™ i7-11700K CPU and NVIDIA GeForce RTX 3080 Ti 12GB GPU. The results are shown in Table III. The single step inference time of the proposed method has shown to be 672 times faster than MCTS-GRNN. In addition, the proposed method has simulated around 51.2 times more simulation steps within the given time limit. Despite the total computation time, which grows exponentially proportional to the MCTS tree depth, the average tree depth of the proposed method has shown to be 3.14 times of MCTS-GRNN. Therefore, SCAN can consider a longer horizon of future states considering HRI compared to methods using neural networks.

D. Real-World Experiments

The objective of real-world experiments is to navigate the Jackal robot [30] safely toward a given point in a crowded environment. The Jackal robot has a AMD Ryzen 7 5800H CPU and a NVIDIA GeForce RTX 3060 Laptop GPU. We use 2D Hokuyo LiDAR for obstacle detection, and ZED 2i Camera for pedestrian detection. Due to the safety issues, we have recruited five volunteers for this experiment, where each volunteer has two or three roles. For fair comparison, we have regulated the movement of pedestrians to follow a fixed scenario shown as Fig. 3, which is composed of the following: pedestrians moving perpendicular to the robot, pedestrians moving towards the robot while facing it, and pedestrians standing still. Also, the scenario includes pedestrians moving in groups. We have conducted the experiment 10 times for each method. We execute the emergency stop command when an obstacle is within $0.3\ m$ from the robot to prevent hazardous situations. The results are shown in Table IV.

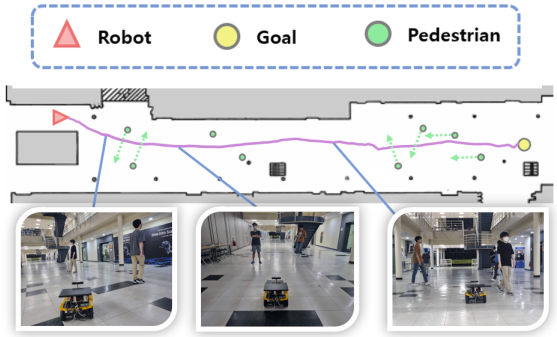

Fig. 3: **Real-world experiment scenario.** The robot navigates to the goal in a real-world environment, where pedestrians follow their respective trajectories.

SCAN has shown the best **CR** and **SANS**. SCAN has shown to successfully find local goals, which consider both safety and social-awareness. Due to the capability of finding appropriate local goals, SCAN is able to travel through the environment in a social-friendly manner when bypassing nearby pedestrians, which is reflected in its high **SF** and **CR**. MCTS-CV has used the motion of pedestrians using constant velocity, which can differ from the actual movement of pedestrians, resulting in worse performance compared to SCAN. CARRT*-Rep assumes that the surrounding obstacles and pedestrians are all static when replanning the path. Due to this fact, CARRT*-Rep planned hazardous paths when passing near by pedestrians, which move perpendicular to the robot. MPC shows the worst performance among the baselines. Since MPC uses random trajectory sampling instead of using a tree search to estimate the value of local goals, it cannot find appropriate local goals when trajectories are not sampled enough for proper value estimation. Therefore, it tends to select sub-optimal local goals and has shown poor performance when pedestrians move towards the robot while facing it.

VI. Conclusions

In this paper, we have proposed SCAN, a socially-aware real-time global planner, which generates appropriate local goals for reaching the global goal considering both HRI and prediction of future states. SCAN performs pedestrian global motion planning and future state simulation using Monte Carlo tree search, considering the influence of the robot's actions on the future motion of pedestrians. The global pedestrian motion is simulated using Y-net and executed in parallel to future state simulation. Due to this, fast simulation has been possible, enabling real-time prediction of a long horizon of future states. To evaluate our method, we have introduced a simulator containing realistic pedestrian motions and a new metric for evaluating the degree of socially-awareness. The proposed method has shown the best performance in both simulation and real-world experiments, which demonstrates the effectiveness of using a socially-aware global planner for selecting promising local goals for reaching the global goal.

REFERENCES

[1] C. I. Mavrogiannis, F. Baldini, A. Wang, D. Zhao, P. Trautman, A. Steinfeld, and J. Oh, "Core challenges of social robot navigation: A survey," *CoRR*, vol. abs/2103.05668, 2021.

[2] C.-E. Tsai and J. Oh, "A generative approach for socially compliant navigation," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2020.

[3] B. D. Ziebart, N. Ratliff, G. Gallagher, C. Mertz, K. Peterson, J. A. Bagnell, M. Hebert, A. K. Dey, and S. Srinivasa, "Planning-based prediction for pedestrians," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[4] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer, "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2019.

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

[6] H. Kretzschmar, M. Spies, C. Sprunk, and W. Burgard, "Socially compliant mobile robot navigation via inverse reinforcement learning," *International Journal of Robotics Research*, vol. 35, no. 11, pp. 1289–1307, 2016.

[7] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: the case for cooperation," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2013.

[8] F. Feurtey, "Simulating the collision avoidance behavior of pedestrians," *Master's thesis, University of Tokyo, Department of Electronic Engineering*, 2000.

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

[10] J. v. d. Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, 2011, pp. 3–19.

[11] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, "Path planning in dynamic environments using generative rnns and monte carlo tree search," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2020.

[12] M. Everett, Y. F. Chen, and J. P. How, "Motion planning among dynamic, decision-making agents with deep reinforcement learning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and System*, 2018.

[13] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2021.

[14] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017.

[15] G. Monaci, M. Aractingi, and T. Silander, "Dipcan: Distilling privileged information for crowd-aware navigation," in *Proc. of Robotics: Science and Systems*, 2022.

[16] Y. F. Chen, M. Liu, M. Everett, and J. P. How, "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017.

[17] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020.

[18] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. of the International Conference on Robotics and Automation*, 2019.

[19] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] J. Suh, J. Gong, and S. Oh, "Fast sampling-based cost-aware path planning with nonmyopic extensions using cross entropy," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1313–1326, 2017.

[22] J. P. van den Berg, "Path planning in dynamic environments," Ph.D. dissertation, Utrecht University, 2007.

[23] D. Mehta, G. Ferrer, and E. Olson, "Autonomous navigation in dynamic social environments using multi-policy decision making," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[24] B. Wang, Z. Liu, Q. Li, and A. Prorok, "Mobile robot path planning in dynamic environments through globally guided reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6932–6939, 2020.

[25] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2021.

[26] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proc. of the Conference on Computer Graphics and Interactive Techniques*, 1987.

[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[28] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[29] D. Kim and S. Oh, "Trc: Trust region conditional value at risk for safe reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2621–2628, 2022.

[30] "Jackal ugv - small weatherproof robot - clearpath," Dec 2020. [Online]. Available: https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/