2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

# A Two-Stage Reinforcement Learning Approach for Robot Navigation in Long-range Indoor Dense Crowd Environments

Xing Hui Jing[1,†], Xin Xiong[1,†], Fu Hao Li[1], Tao Zhang[2], Long Zeng[1*]

***Abstract*— Safe and efficient mobility is vital for mobile robots navigating long-range indoor crowd environments, such as supermarkets, restaurants, and railway stations. Traditional path planning methods are challenged because of the high dynamics of pedestrians and constrained feasible regions. Existing long-range deep reinforcement learning (DRL) path planning methods often exhibit low success rates and driving speeds in long-range navigation tasks under crowded conditions. To overcome these issues, we propose a new two-stage DRL method, known as TSDRL, where the long-range navigation task is divided into subgoal generation (SG) and planning refinement (PR) stages. In the SG stage, the agent is trained to learn a decision-making policy to generate subgoals at each decision time to avoid dense crowds. In the PR stage, the agent learns a safer and more efficient planning policy based on each subgoal generated in the SG stage to improve the robot's movement safety and speed. Simulated experiments show that our method outperforms traditional and long-range DRL path planning methods in terms of safety, efficiency, generalization, and robustness. Furthermore, we evaluate our approach using the Turtlebot2 platform in a real-world setting, demonstrating that the robot can navigate safely and efficiently while avoiding dense crowds.**

***Index Terms*— deep reinforcement learning, robot navigation, path planning, two-stage navigation.**

## I. INTRODUCTION

RECENTLY, there has been rapid development and widespread use of mobile robots in indoor crowd environments, such as restaurants and supermarkets [1]. Due to the high dynamics of crowded pedestrians, there is an increased need for robots to ensure safety and flexibility. Path planning is therefore a crucial task in the autonomous navigation of robots, guiding them along a safe and collision-free path from start to goal. Traditional path planning methods for dynamic environments [2] are mostly optimized by handcrafted functions to obtain the optimal time and shortest path through partially known environments. However, these methods are prone to the " freezing robot problem" in dense crowd environments [3].

Deep Reinforcement Learning (DRL) has an inherent advantage of learning a reward-maximizing policy through trial and error interaction with environments, without the requirement of manual design of behaviors. Due to this advantage, it is now extensively utilized in complex environment navigation, particularly in crowd navigation [4]. However, learning an effective policy for long-range environments poses difficulties for the DRL model, due to the exponential growth of possible states and actions with increasing distance. To address this, most approaches implement a global planner to generate waypoints that guide DRL to learn a planning policy for collision avoidance. Existing methods, such as sub-sampling [5] and spatial horizon [6], define a subgoal selection policy using handcrafted functions.

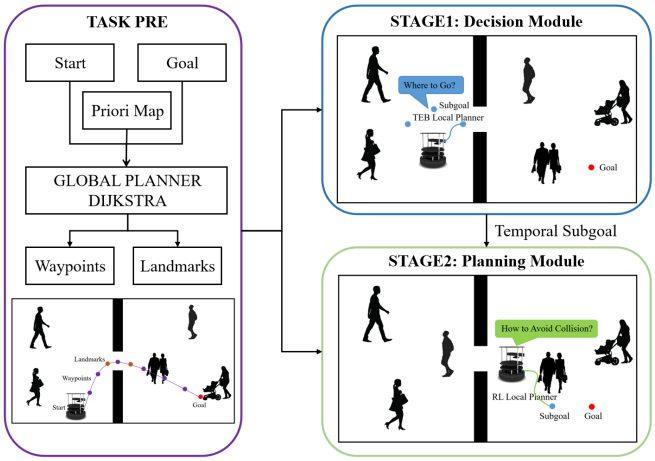



Fig. 1. Illustration of the two-stage DRL algorithm. In task pre, the global planner generates a global path on the priori map, and waypoints and landmarks are generated on the path. In stage 1, the robot decides where to go next for the next decision time. In stage 2, the robot learns how to avoid obstacles.

Although predefined handcrafted functions may be useful in long-range environments, they may face difficulties when applied in complex and constantly changing situations. A study [7] tackled this by utilizing DRL instead of handcrafted functions to learn a decision policy for selecting a subgoal, and coupling it with a traditional local planning method to avoid collisions. However, traditional local planning methods suffer from the "freezing robot problem" and low success rates in dense crowds. Therefore, we propose a two-stage DRL approach, as illustrated in Fig. 1. In the first stage (SG), the agent uses DRL to learn a subgoal selection policy based on a traditional local planning method. The aim is to avoid dense crowds as much as possible, thereby reducing the complexity of the local environment of the robot's movements, and minimizing the occurrence of the "freezing robot problem". In the second stage (PR), the agent uses DRL to learn a planning policy that replaces the traditional

† Equal contribution.

This research was funded by Shenzhen Major Science and Technology Project (KJZD20230923114900002, KJZD20230923115503007)

1 Department of Advanced Manufacturing, Shenzhen International Graduate School, Tsinghua University, Shenzhen 518055, China {jingxh21, xiong-x20, lfh23}@mails.tsinghua.edu.cn

2 Pudu Technology Ltd, Shenzhen 518055, China felixzhang@pudutech.com

∗ Corresponding author. zenglong@sz.tsinghua.edu.cn

local planning method in the SG stage, with an objective to improve robot safety in high dynamics pedestrian movement environments. The main contributions of this work are as follows:

- We propose a new two-stage DRL method in long-range dense crowd navigation, consisting of SG and PR stages.
- We conducted experiments in the simulator using various maps and pedestrian quantities to demonstrate that our approach achieves a higher success rate, increased and stable average driving speed, particularly in dense crowd environments. Additionally, our results show that the approach exhibits good generalization.
- We integrated this method into the Robot Operating System (ROS) and deployed it on the Turtlebot2 platform to demonstrate its effectiveness in a real-world, dense crowd environment.

## II. Related Work

We divide robot navigation methods in dense crowd into two categories: traditional and learning, and the learning methods are divided into agent-level and sensor-level according to whether to use raw sensor data directly.

### A. Traditional planning Methods

Traditional path planning methods can be categorized into two types: global planning and local planning. Global planning is suitable for static environments, where the environment is known in advance. On the other hand, local planning is required for dynamic obstacle environments. Local planning can be further divided into two categories: reaction-based and optimization-based methods. Reaction-based methods, such as RVO [8], have a limited time horizon and may not be effective for fast-moving obstacles. Optimization-based methods, such as TEB [9], rely on handcrafted functions that generate optimal control commands and may encounter the "freezing robot problem" in dense and complex crowd environments.

### B. Agent-level Navigation using DRL

Agent-level methods use the states of agents as part of observations, such as the robot's position and velocity. These methods are mainly utilized in multi-agent collision avoidance scenarios [10], [11]. They focus on analyzing the impact of human-robot and human-human interactions on robot planning, with the aim of improving both the safety and time efficiency of robots while avoiding pedestrian collisions in empty maps. Liu et al. [12] adopted an occupancy grid map or angular map as input, which helped avoid collisions with static obstacles in the environment. However, since learning-based planning approaches cannot guarantee absolute safety, Brito et al. [7] used DRL to learn a subgoal recommendation policy that can provide long-term guidance to the local planner. They also incorporate MPC to optimize the solution that satisfies the dynamic feasibility and collision avoidance constraints. Lodel et al. [13] and Brinkman et al. [14] expanded on the method proposed by [7], adapting it to navigate environments with static obstacles. Despite the potential of agent-level methods, they are challenging to apply in the real world due to the imperfect sensing capabilities of agents in practice.

### C. Sensor-level Navigation using DRL

Sensor-level methods use the raw sensor data as input and can be divided into end-to-end methods and two-stage methods depending on whether control commands are directly output.

*1) End-to-end Methods:* The end-to-end methods treat the robot navigation task as a black box, taking the raw sensor data directly as input and using the DRL algorithm to output control commands (e.g., linear and angular velocities). Long et al. [15] presented a multi-scenario, multi-stage training framework to learn a policy that enables the robot to perform well in large-scale robot systems and complex environments. Jin et al. [16] considered the robot's perspective and social safety to avoid crowd using 2D lidar only. Yao et al. [17] presented an obstacle avoidance policy closer to the real world using multiple pedestrians' motion strategies. Ryu et al. [18] considered the uncertainty caused by occlusion and proposed confidence-based navigation to maximize the local confidence of the robot by exploring uncertain regions. [19] based on the static danger zone and dynamic danger zone proposed by [20], considers semantic information about nearby obstacles such as children, elderly, and adults, and proves that semantic information and danger zone can significantly improve safety. The end-to-end methods rely on online real-time computation, are less generalizable, and are dependent on system configurations such as robot size and sensor accuracy. However, they can be easily deployed if the simulator is as realistic as possible.

*2) Two-stage Methods:* The two-stage methods divide the navigation task into two stages: where to go and how to go [4]. The two-stage methods can be divided into three categories depending on whether DRL is used in each stage. The first category uses DRL to decide where to go and traditional planning algorithms to decide how to go, which is currently widely used in visual navigation. In [21], a waypoint generator is learned using DRL and a traditional planning algorithm is used to decide how to go in an indoor static environment. The second one uses the handcrafted function to generate waypoints and uses the DRL to decide how to go, this method is mainly used in long-range navigation. Guldenring et al. [5] used a global planner to generate a global path and subsamples waypoints from the path. kastner et al. [6] proposed a spatial horizon approach for the robot's current position to take waypoints on the global path. The third one is using RL for the robot to learn where and how to go both in multi-robot system [22]. However, according to our knowledge, the third category has not yet been widely used in crowd robot navigation. Therefore, we propose a two-stage crowd robot navigation method that uses DRL for the robot to learn where to go and how to go.

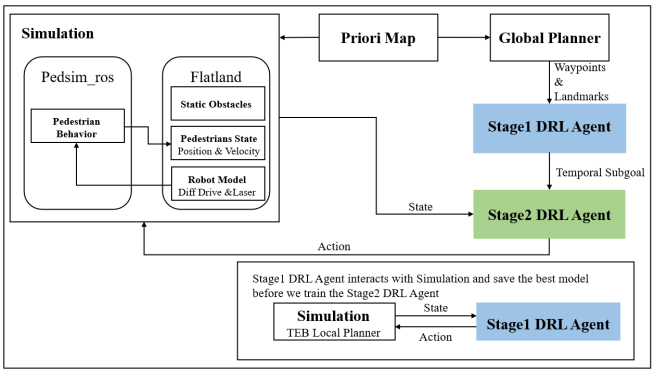



Fig. 2. The system is built based on the Flatland Simulator used in [5]. Waypoints and landmarks are generated on the global path. Stage1 learns the long-term decision policy based on observations and uses the TEB local planner to control the robot's motion.Stage2 outputs control commands directly based on observations.

## III. Methods

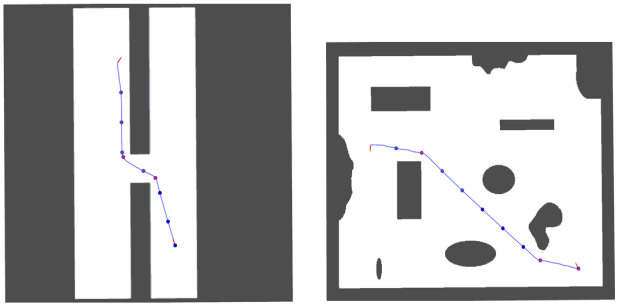
Real-world indoor environments are often large-scale and complex, with pedestrians and static obstacles of various shapes and sizes. Traditional optimization-based local planning algorithms struggle to handle unpredictable pedestrian movements, while DRL is myopic and not suitable for long-range scenarios [23]. Therefore, we propose a two-stage DRL approach. We first introduce the framework of the two-stage approach (Section III-A). We then describe the decision module (Section III-B) and planning module (Section III-C) in the system. Finally, we elaborate on the training process of the two-stage algorithm (Section III-D).

### A. System Design

As shown in Fig 2, for long-range indoor pedestrian navigation, we have designed an overall system architecture based on the main components of the simulation environment in [5], using Flatland as the simulator and the Pedsim_ros package to control pedestrian movements. The entire system is composed of four parts: simulator, global planner, decision module (Section III-B), and planning module (Section III-C).

*1) Simulation:* The simulation environment consists of three parts: robot model, pedestrian model, and a priori map. The robot and pedestrian are modeled as a circle of radius 0.3 m ($r_{robot} = r_{ped} = 0.3m$). The robot is differential-driven and has a 360-degree lidar, which has a maximum range of 12 meters and provides 360 distance values per scanning. The walking speed of the pedestrian is sampled from a Gaussian distribution $v_{ped} \sim N(0.6, 0.2)m/s$ which is similar to the real-world situation. The real-time walking speed is solved using the social force model (Pedsim_ros).

*2) Global planner:* Since Flatland is compatible with ROS, we use the global planner package in ROS Navigation Stack to generate a global path and extract waypoints and landmarks on the global path. As shown in Fig. 3, waypoints are subsampled by look-ahead distance ($dist_{ahead} = 3.0m$) on the global path, and the goal is considered as the last waypoint when the distance near the goal is smaller than the look-ahead distance. However, waypoints subsampled by a fixed distance are not equally important for the navigation task, so we also extract landmarks, which are part of the more important critical points. We choose a candidate point every 20 path points along the global path generated by Dijkstra. Subsequently, we calculate the angular change of the line between each candidate point. Any candidate point with an angular change greater than the threshold ($\phi_{lm} = \pi/4$) is considered a landmark.



(a) Easy map (b) Complex map

Fig. 3. Waypoints and landmarks generation. Waypoints and landmarks are represented by blue and purple points, respectively.

### B. SG Stage: Decision Module

The agent employs DRL to learn a subgoal selection policy, which is then passed to the TEB local planner to control the robot's motion accordingly. We can model the indoor crowd navigation task as a partially observable Markov decision process (POMDP) which can be described by a 6-tuple $(S, A, O, P, R, \gamma)$, where $S$ is the state space, $A$ is the action space, $O$ is the observation space, $P$ is the state-transition model, $R$ is the reward function, and $\gamma$ is the discount factor. The goal of indoor crowd navigation is for the robot to learn a collision-free policy that minimizes the time to goal. The policy needs to satisfy the time, collision avoidance, and terminal constraints and be optimal by maximizing the cumulative reward of the agent. Thus, we can define the optimal policy $\pi^*$ as follows:

$$\pi^* = \arg\max_{\pi} \quad \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R(s^t, a^t)\right] \tag{1}$$

$$s.t. \quad \forall(t) \in [0, T] \tag{1a}$$

$$\min(O_{lidar}) > r_{robot} \tag{1b}$$

$$\|p_r - p_g\| < r_{goal} \tag{1c}$$

where (1a) is a time constraint, the robot's running time cannot exceed the maximum time $T$, (1b) is a collision avoidance constraint, which ensures that the robot will not collide with static obstacles and pedestrians, and (1c) is a terminal constraint, the robot reaches the goal when the distance between the robot's position ($p_r$) and the goal's position ($p_g$) is less than the goal radius ($r_{goal}$).

*1) Observation Space:* The observation space consists of three parts $O^t = [O^t_{lidar}, O^t_{waypoints}, O^t_{landmarks}]$.The configuration of lidar, waypoints generation, and landmarks generation can be referred to in III-A and Fig. 3.

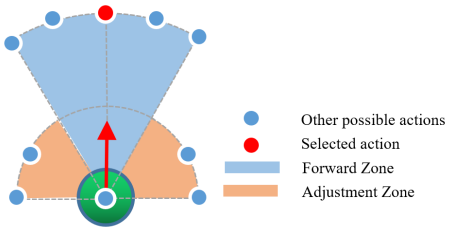



Fig. 4. Action Space.

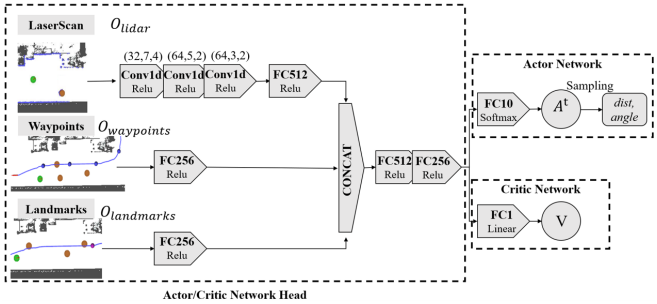



Fig. 5. Neural network architecture of Stage 1. (32,7,4)means that the layer convolves 32 one-dimensional filters with kernel size = 7, stride = 4.

The $O^t_{lidar}$consist of the 360 distance values of the lidar scanning$\left(O^t_{lidar} \in \mathbb{R}^{360}\right)$, $O^t_{waypoints}$ and $O^t_{landmarks}$ consist of the x and y coordinates $([x_i, y_i], i = 0, ..., n)$ of the n nearest points under the coordinate system of the robot's current position. We take $n_{waypoints} = 4$ and $n_{landmarks} = 1$ respectively$\left(O^t_{waypoints} \in \mathbb{R}^8, O^t_{landmarks} \in \mathbb{R}^2\right)$.

*2) Action Space:* The action space is a series of permissible subgoals$(a^t = [dist^t, angle^t])$ in the robot coordinate system within each decision time interval. As shown in Fig. 4, the action space is divided into the forward zone$\left(dist_{forward} = 2m, angle : \left[-\frac{\pi}{6}, \frac{\pi}{6}\right]\right)$ and adjustment zone$\left(dist_{adjustment} = 1m, angle : \left[-\frac{\pi}{2}, -\frac{\pi}{6}\right] \cup \left[\frac{\pi}{6}, \frac{\pi}{2}\right]\right)$. The actions in the forward zone make the robot move to goal quickly, and the actions in the adjustment zone adjust the robot's position when it encounters obstacles block. We discretize the action space into the 10 actions$[(0,0), (1,-\frac{\pi}{2}), (1,-\frac{\pi}{3}), (1,\frac{\pi}{3}), (1,\frac{\pi}{2}), (2,-\frac{\pi}{6}), (2,-\frac{\pi}{12}), (2,0), (2,\frac{\pi}{12}), (2,\frac{\pi}{6})]$.

*3) Reward Design:* The goal of stage 1 is to find the optimal subgoal selection policy for a certain decision time interval (e.g., 1.0s, we conduct an ablation study for different decision time in section IV-B). The policy is learned based on the TEB local planner that allows the robot to avoid obstacles and minimizes the time to reach the goal. We set the reward function for stage 1 as follows:

$$\begin{aligned} R^t =& R^t_{time} + R^t_{goal} + R^t_{collision} \\ &+ R^t_{landmarks} + R^t_{waypoints} + R^t_{ang} \end{aligned} \tag{2}$$

$$R^t_{time} = \omega_t \times \Delta t \tag{3}$$

$$R^t_{goal} = \begin{cases} r_g, & if\ \|p^t_r - p^t_g\| < r_{goal} \\ 0, & otherwise \end{cases} \tag{4}$$

$$R^t_{collision} = \begin{cases} r_{collision}, & if\ \min(O^t_{lidar}) \le r_{robot} \\ 0, & otherwise \end{cases} \tag{5}$$

$$R^t_{landmarks} = \omega_l \times (\|p^{t-1}_r - p^{t-1}_l\| - \|p^t_r - p^t_l\|) \tag{6}$$

$$R^t_{waypoints} = \begin{cases} \omega_w \times \|p^t_r - p^t_w\|, & if\ \|p^t_r - p^t_w\| > r_{wp} \\ 0, & otherwise \end{cases} \tag{7}$$

$$R^t_{ang} = \omega_a \times \frac{|ang^t_{sg} - ang^{t-1}_{sg}|}{\pi} \tag{8}$$

where (3) is a time reward ($\omega_t = -0.1$,$\Delta t$ is the decision time interval), which encourages the robot to reach the goal with an overall shorter time, (4) is a goal reward ($r_g = 10, r_{goal} = 1.0m$), which the robot gets when it reaches the goal, (5) is a collision reward ($r_{collision} = -10$), which the robot gets when it collides the obstacles, (6) is a landmarks reward ($\omega_l = 1.0$), which encourages the robot to approach landmarks, (7) is a waypoint reward ($\omega_w = -0.05$,$r_{wp} = 4.0m$), which is given as a negative reward when the robot is farther than $r_{wp}$ from the global path, and (8) is a subgoal's angle change reward ($\omega_a = -0.36$), which encourages the robot to make smoother decisions.

*4) Network Architecture:* We refer to similar network structures for indoor crowd navigation tasks [5], [15], [24] and experiment with various variants to finally obtain the network structure shown in Fig. 5. It consists of an actor network and a critic network. The inputs to the network are the lidar data, waypoints, and landmarks observed by the robot, and the output is a set of [dist, angle] sampling in the action space.

### C. PR Stage: Planning Module

The agent in stage 2 uses DRL to learn a control planning policy based on the subgoal output by the stage 1 agent to directly control the motion of the robot. Similarly, we model it as POMDP as described in III-B, and it needs to satisfy the task's goal and constraints.

*1) Observation Space:* The observation space consists of three parts $O^t = \left[O^t_{lidar}, O^t_{subgoal}, O^t_{goal}\right]$.The $O^t_{lidar}$ consist of the 360 distance values of the lidar scanning$\left(O^t_{lidar} \in \mathbb{R}^{360}\right)$, $O^t_{subgoal}$ and $O^t_{goal}$ consist of the x and y coordinates $([x, y])$ under the coordinate system of the robot's current position$\left(O^t_{subgoal}, O^t_{goal} \in \mathbb{R}^2\right)$.

*2) Action Space:* The action space is a series of permissible velocities, including translational and rotational velocity$(a^t = [v^t, w^t])$. Based on the requirements for the robot to operate in a real-world scenario, we define $v^t \in (0, v_{max}), w^t \in (-w_{max}, w_{max})$, where $v_{max} = w_{max} = 1.0$.We discrete the action space into 6 actions $[(0,-1.0), (1.0,0), (0,1.0), (1.0,0.5), (1.0,-0.5), (0,0)]$ as in [5].

*3) Reward Design:* The goal of stage 2 is to learn to directly control the robot's motion to avoid obstacles and

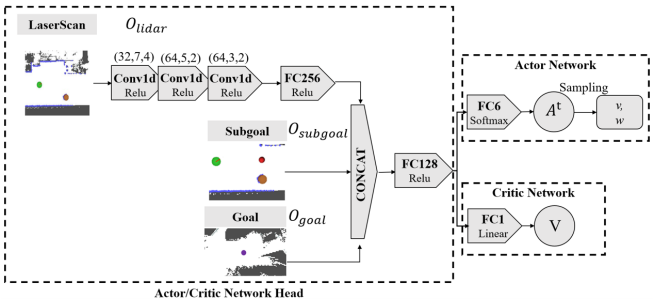



Fig. 6. Neural network architecture of Stage 2.

reach the goal in the shortest time. We set the reward function for the stage 2 as follows:

$$R^t = R^t_{subgoal} + R^t_{goal} + R^t_{collision} + R^t_{sz} + R^t_{vel} \tag{9}$$

$$R^t_{subgoal} = \omega_{sg} \times (\|p^{t-1}_r - p^{t-1}_{sg}\| - \|p^t_r - p^t_{sg}\|) \tag{10}$$

$$R^t_{goal} = \begin{cases} r_g, & if \ \|p^t_r - p^t_g\| \ < r_{goal} \\ 0, & otherwise \end{cases} \tag{11}$$

$$R^t_{collision} = \begin{cases} r_{collision}, & if \ \min(O^t_{lidar}) \le r_{robot} \\ 0, & otherwise \end{cases} \tag{12}$$

$$R^t_{sz} = \begin{cases} \omega_{sz} \times (r_{sz} - \min(O^t_{ped\_lidar})), & if \ \min(O^t_{ped\_lidar}) < r_{sz} \\ 0, & otherwise \end{cases} \tag{13}$$

$$R^t_{vel} = \begin{cases} \omega_{vel} \times |w^t|, & if \ |w_t| > 0.5 \\ 0, & otherwise \end{cases} \tag{14}$$

where (10) is a subgoal reward ($\omega_{sg} = 1.0$), which encourages the robot to approach the subgoal, (11) is a goal reward ($r_g = 10, r_{goal} = 0.3m$), which the robot gets when it reaches the goal, (12) is a collision reward ($r_{collision} = -10$), which the robot gets when it collides the obstacles, (13) is a static danger zone reward ($\omega_{sz} = 1.0, r_{sz} = 0.8m$), referring to [20], which gives a safety radius to pedestrians to improve the safety of planning, and (14) is a velocity reward ($\omega_{vel} = -0.1$), which is expected to make the robot run more smoothly.

*4) Network Architecture:* As shown in Fig. 6, it consists of actor and critic networks. The inputs to the network are the lidar data, subgoal, and goal observed by the robot, and the output is a set of $[v, w]$ sampling in the action space.

### D. Two-stage DRL-Navigation

In this paper, we use Proximal Policy Optimization (PPO) [25]to train our policies for both stages 1 and 2. We used the PPO-clip version to train our policies, so we directly adopted the PPO2 algorithm from open-source project stable-baselines [26]. The entire algorithm is shown in Algorithm 1. We first randomly initialize the policy and value function parameters for the first and second stages $\{\theta^\pi_1, \theta^V_1\}$,$\{\theta^\pi_2, \theta^V_2\}$, respectively. As shown in Fig. 5 and Fig. 6, $\theta^\pi$ and $\theta^V$ share the same parameter except for the final layer. Then, we collect data to update $\{\theta^\pi_1, \theta^V_1\}$ and save the model of the optimal policy $\pi^*_1$ at a fixed decision time. We get the subgoal under the current environment observation according to $\pi^*_1$ and collect the data to update $\{\theta^\pi_2, \theta^V_2\}$. In addition, in order to increase the training speed for fast convergence, we use curriculum learning in stage 1, and gradually increase the number of pedestrians for training.

**Algorithm 1** Two-stage DRL-Navigation Training.

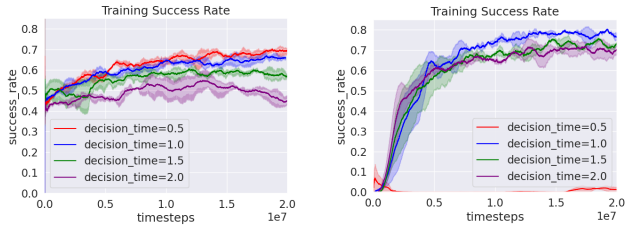
Initialize actor and critic network parameters for the first and second stages $\{\theta^\pi_1, \theta^V_1\}$,$\{\theta^\pi_2, \theta^V_2\}$.
// Stage 1:Decision Module
**for** $iteration = 1, 2, ...,$ **do**
  Initialize $D_1 \leftarrow \emptyset$
  **for** $t = 0, 1, ..., T_1$ **do**
    $(s_t, a_t, r_t, s_{t+1}, done) = Step_1(TEB(\pi_{\theta 1}(s_t)))$
    $s^1_t = O^1_t = [O^t_{lidar}, O^t_{waypoints}, O^t_{landmarks}]$
    $D_1 \leftarrow (s_t, a_t, r_t, s_{t+1}, done)$
  **end for**
  stage 1 PPO Training, update $\{\theta^\pi_1, \theta^V_1\}$
**end for**
Save the optimal policy ($\pi^*_{\theta 1}$) parameters $\{\theta^{\pi *}_1, \theta^{V *}_1\}$
// Stage 2: Planning Module
**for** $iteration = 1, 2, ...,$ **do**
  Initialize $D_2 \leftarrow \emptyset$
  **for** $t = 0, 1, ..., T_2$ **do**
    $(s_t, a_t, r_t, s_{t+1}, done) = Step_2(\pi_{\theta 2}(s_t))$
    $s^2_t = O^2_t = \left[O^t_{lidar}, \pi^*_{\theta 1}(s^1_t), O^t_{goal}\right]$
    $D_2 \leftarrow (s_t, a_t, r_t, s_{t+1}, done)$
  **end for**
  stage 2 PPO Training, update $\{\theta^\pi_2, \theta^V_2\}$
**end for**
Save the optimal policy ($\pi^*_{\theta 2}$) parameters $\{\theta^{\pi *}_2, \theta^{V *}_2\}$

## IV. EXPERIMENTS AND RESULTS

In this section, we first describe the experimental setup and the hyperparameters in different stages of training. Then, we provide an ablation study for different decision time. Finally, we compare our algorithm against traditional algorithm (DWA, TEB) and some state-of-the-art DRL methods (Drl-Baseline [5], Arena-Rosnav [6]) in long-range indoor navigation.

### A. Experimental Setup and Training

To train and evaluate our algorithm, we chose two navigation scenarios as shown in Fig. 3. We randomize the start and goal positions on the map in each navigation task and make the global path satisfying farther than 15m as a valid navigation task. The number of pedestrians in this task is set as a random number from $6 \sim 12$ and pedestrians' start and goal positions are randomly selected within the predefined area. We train stages 1 and 2 three times using random seeds for $2 \times 10^7$ timesteps each. In stage 1, we first train $5 \times 10^6$ timesteps in the environment without pedestrians, then increase the number of pedestrians to $6 \sim 12$. In stage 2 we directly train the algorithm in the environment with $6 \sim 12$ pedestrians. The training success rate of stages 1 and




(a) Stage 1 (b) Stage 2

Fig. 7. Training success rate results for different decision time. For each setup, we train 3 agents and plot the average success rate with timesteps.

TABLE I

HYPERPARAMETERS VALUE.

| Hyperparameter | Value | Hyperparameter | Value |
|---|---|---|---|
| $\gamma$ (stage 1:DT=1.0S) | 0.96 | $\gamma$ (stage 2) | 0.99 |
| n_steps (stage 1:DT=1.0S) | 512 | n_steps (stage 2) | 4096 |
| learning rate | $3 \times 10^{-3}$ | clip range | 0.2 |

2 with timesteps are shown in Fig. 7. The hyperparameters are summarized in Table I.

To compare our algorithm with other methods, we use four metrics, which are defined as follows:

- **Success Rate (SR)**: Percentage (%) of the robot in the total evaluation tasks that reach the goal without collision within the time limit.
- **Average Speed (AS)**: The average speed ($m/s$) of the robot for all successful evaluation tasks.
- **Average Path Ratio (APR)**: The average distance ratio of the robot's driving path to the global path for all successful evaluation tasks.

### B. Ablation Study

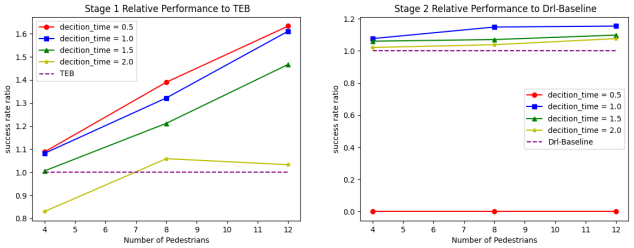
The decision time (DT) is an important factor affecting the policy. To evaluate the impact of DT, we conduct an ablation study on the performance of the algorithm in stages 1 and 2, respectively. We evaluate different decision time algorithms for 200 test runs on the easy map with 4, 8, and 12 numbers of pedestrians ($n_{peds} = 4, 8, 12$). The results in Table II show that the shorter decision time in stage 1 makes the robot have a higher success rate, higher speed, and shorter driving path. However, the policy in stage 2 falls into a local optimum dilemma due to the rapid change of the subgoal in a short time. Therefore, combining the evaluation results of different time in stages 1 and 2, we choose $DT = 1.0s$ as the criterion for our two-stage algorithm.

In addition, To demonstrate the advantages of the subgoal selection policy in stage 1 and the subgoal-based planning policy in stage 2, we compare the performance of the algorithms for different decision time in stage 1 with the TEB and stage 2 with the Drl-Baseline, respectively. The relative performance of the two stages is shown in Fig. 8. (a) proves that the subgoal selection policy can improve the robot's navigation success rate in dense crowd. (b) demonstrates that both the flexibility and stability of the subgoal selection policy are key to the stage 2 algorithm and the performance of the learning-based planner can be improved based on the subgoal selection policy.

TABLE II

EVALUATION ON DIFFERENT DECISION TIME IN STAGES 1 AND 2.

| Decision Time (s) | SR | AS | APR | SR | AS | APR | SR | AS | APR |
|---|---|---|---|---|---|---|---|---|---|
| Stage 1 | 4 peds | | | 8 peds | | | 12 peds | | |
| **0.5** | **93** | **0.551** | **1.183** | **82** | **0.497** | **1.234** | **73.5** | **0.448** | **1.243** |
| 1.0 | 92.5 | 0.509 | 1.210 | 78 | 0.446 | 1.257 | 72.5 | 0.407 | 1.299 |
| 1.5 | 86 | 0.420 | 1.229 | 71.5 | 0.407 | 1.276 | 66 | 0.373 | 1.313 |
| 2.0 | 71 | 0.388 | 1.258 | 62.5 | 0.366 | 1.273 | 46.5 | 0.368 | 1.326 |
| Stage 2 | 4 peds | | | 8 peds | | | 12 peds | | |
| 0.5 | 0 | – | – | 0 | – | – | 0 | – | – |
| **1.0** | **97.5** | **0.805** | **1.140** | **88.5** | **0.767** | **1.195** | **82** | **0.638** | **1.171** |
| 1.5 | 96 | 0.753 | 1.165 | 82.5 | 0.689 | 1.288 | 78 | 0.575 | 1.250 |
| 2.0 | 92.5 | 0.728 | 1.219 | 80 | 0.631 | 1.272 | 76.5 | 0.546 | 1.271 |



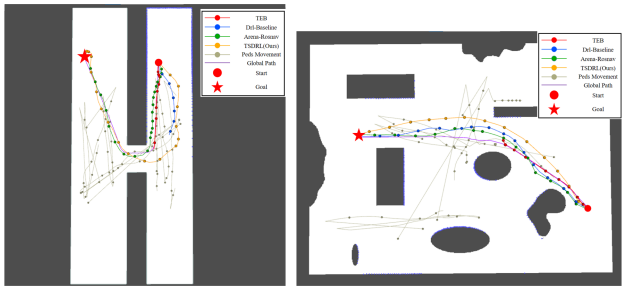

(a) Stage 1 (b) Stage 2

Fig. 8. Relative performance for different decision time in stages 1 and 2. In stage 1, we choose TEB as the baseline, which is equivalent to stage 1 without subgoal selection policy. In stage 2, we choose Drl-Baseline as the baseline, which is equivalent to subgoal selection policy defined by the handcrafted functions instead of the subgoal-based policy.

### C. Qualitative and Quantitative Evaluation

This section presents a qualitative and quantitative analysis of different algorithms. The qualitative analysis compares the trajectories of different algorithms executing in the same environment. The quantitative analysis compares the performance of different algorithms for different numbers of pedestrians and different complexity of maps.

*1) Qualitative Evaluation:* We evaluate on the easy map and the complex map with 8 pedestrians, respectively. We record the current positions of the robot and pedestrians every 2s and 4s, respectively. Fig. 9 shows that the robot with TEB planner fails to reach its goal on both maps, with Drl-Baseline planner fails on the easy map but succeeds on the complex map, and with Arena-Rosnav planner and our TSDRL planner succeeds on both maps. In addition, the trajectories show that the robot with TEB planner is more




(a) Easy Map (b) Complex Map

Fig. 9. Trajectories of different planners on different evaluation scenarios. In each map, we randomly place 8 pedestrians in a predefined area and select the global paths larger than 15m as the valid navigation task. In the easy map, pedestrians walk along a corridor or go from one corridor through a door to another. In the complex map, pedestrians walk in random horizontal or vertical directions within the map.

TABLE III

EVALUATION ON DIFFERENT NUMBERS OF PEDESTRIANS AND MAPS.

| Object | SR | AS | APR | SR | AS | APR | SR | AS | APR |
|---|---|---|---|---|---|---|---|---|---|
| Peds | 4 peds | | | 8 peds | | | 12 peds | | |
| DWA | 75.5 | 0.657 | 1.116 | 43.5 | 0.604 | 1.138 | 36.5 | 0.506 | 1.137 |
| TEB | 85.5 | 0.735 | 1.060 | 59 | 0.650 | 1.079 | 45 | 0.563 | 1.092 |
| Drl-Baseline [5] | 90.5 | 0.742 | 1.095 | 77 | 0.706 | 1.121 | 71 | 0.615 | 1.144 |
| Arena-Rosnav [6] | 94 | 0.784 | 1.067 | 83 | 0.721 | 1.085 | 76 | 0.622 | 1.122 |
| **TSDRL (Ours)** | **97.5** | **0.805** | **1.140** | **88.5** | **0.767** | **1.195** | **82** | **0.638** | **1.171** |
| Maps | Easy Map | | | Complex Map | | | Overall Average | | |
| DWA | 43.5 | 0.604 | 1.138 | 53.5 | 0.627 | 1.152 | 48.5 | 0.616 | 1.145 |
| TEB | 59 | 0.650 | 1.079 | 60 | 0.642 | 1.094 | 59.5 | 0.646 | 1.087 |
| Drl-Baseline [5] | 77 | 0.706 | 1.121 | 68 | 0.688 | 1.154 | 72.5 | 0.697 | 1.138 |
| Arena-Rosnav [6] | 83 | 0.721 | 1.085 | 78.5 | 0.713 | 1.139 | 80.75 | 0.717 | 1.112 |
| **TSDRL (Ours)** | **88.5** | **0.767** | **1.195** | **83.5** | **0.732** | **1.248** | **86** | **0.750** | **1.222** |

likely to collide because of moving along the global path without avoiding crowded areas, with Arena-Rosnav planner and Drl-Baseline planner have similar running time on the complex map, but with Arena-Rosnav planner has a shorter running path benefit from its spatial-horizon subgoal policy. We also find that the running time of the robot with Arena-Rosnav planner and Drl-Baseline planner is longer compared to our TSDRL planner, because both planners are unable to consciously avoid crowded areas, despite the fact that they are able to gain some foresight through the subgoal policy defined by handcrafted functions and trial-to-error training by DRL. The trajectories also show that the robot with TSDRL planner is more stable than other planners by comparing the distance the robot runs every 2s, has a shorter running time, but pays the cost of a longer driving path to actively avoid dense crowd.

*2) Quantitative Evaluation:* We evaluate each algorithm for 200 test runs on the easy map with 4, 8, and 12 numbers of pedestrians and on different maps with 8 numbers of pedestrians. The results in Table III show that, from $n_{peds} = 4$ to $n_{peds} = 12$, our TSDRL drops a little from 97.5% to 82% better than Arena-Rosnav from 94% to 76% and Drl-Baseline from 90.5% to 71% and TEB drops the most, from 86% to 45%. We find that all three learning-based planning algorithms outperform TEB and DWA in terms of success rate and average speed, especially in dense crowd. We can also conclude that our TSDRL has a higher success rate and a higher speed, but pays a longer driving path compared to Drl-Baseline and Arena-Rosnav Besides. This is because our TSDRL guides the robot to avoid dense areas of the map because of the stage 1 learning-based subgoal selection policy, which selects the subgoal every decision time according to the dynamic changes of the environment. For different maps, Table III shows that our TSDRL also has a higher success rate and higher average speed, but costs a longer driving path than the other four algorithms on both easy and complex maps.

### D. Generalization Assessment

To validate the adaptability of TSDRL to maps, we conducted generalization experiments. As shown in Fig. 10, we directly applied the TSDRL network model trained in scenes A and B to the mobile robot navigation tasks on maps A1, A2, B1, and B2.The evaluation experiment involved randomly placing 4 to 8 pedestrians in the map

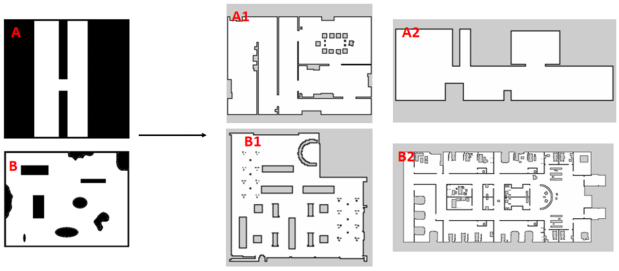



Fig. 10. Maps in the generalization evaluation experiment. After training solely on Map A with TSDRL, the model was directly applied to Maps A1 and A2, and after training solely on Map B, the model was directly applied to Maps B1 and B2.

TABLE IV

GENERALIZATION EVALUATION OF TSDRL.

| Training 2 Evaluation | SR | AS | APR |
|---|---|---|---|
| A 2 A1 | 67 | 0.683 | 1.152 |
| A 2 A2 | 86 | 0.727 | 1.136 |
| B 2 B1 | 62.5 | 0.642 | 1.215 |
| B 2 B2 | 49 | 0.574 | 1.179 |

and generating 200 valid evaluation tasks randomly. TableIV demonstrates that the TSDRL algorithm can still guarantee a high success rate of up to 86%. Even when generalizing from map B to the challenging map B2, our navigation success rate can be maintained at 49%. This fully confirms the outstanding planning effectiveness of the TSDRL algorithm even as maps change, showcasing its exceptional generalization capabilities.

### E. Real-world Scenarios

We deployed the TSDRL algorithm on the Turtlebot2 robot, comprising a Kobuki base, Realsense depth camera, mobile power supply, Xavier processor, display screen, and Rplidar A2 single-line lidar. Initially, we conducted mapping of real-world scenes, followed by offline training of the network model on a simulation platform, and finally, we loaded the trained network model onto the robot. During the actual testing in real-world scenarios, we conducted 10 sets of randomly generated evaluation tasks. Through these tests, we observed a planning success rate of 50% in online reasoning, with an average global path distance of 11.2m, average actual travel path distance of 12.8m, path ratio averaging 1.143, average travel time of 26.3s, and average travel speed of 0.487m/s. Comparative analysis with simulation results revealed a certain degree of decrease in success rate, higher path ratio, and increased average travel speed. This is attributed to the evaluation tasks in real environments involving short-distance dense crowd scenarios, leading to decreased success rate due to sensor and positioning errors in both simulation and real environments, while the increase in path ratio and average speed is due to the robot avoiding crowds. Fig. 11 illustrates one of the real-world testing scenarios.

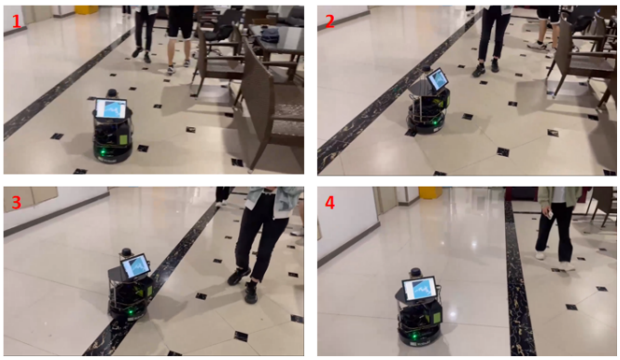



Fig. 11. Real-world planning scenario for TSDRL.

## V. CONCLUSIONS

In this paper, we propose a two-stage DRL approach that divides the long-range indoor crowd navigation task into temporal decision subgoal generation and subgoal-based learning-based planning. Firstly, we design an optimal decision time for the robot to learn a subgoal selection policy that balances long-term decisions and real-time changes. Subsequently, the traditional local planner is replaced by a DRL local planner to improve the low success rate under dynamic obstacles and prevent the robot from getting stuck in dense crowd environments. Our planner plans at a comfortable and safe distance from the crowd with a smooth trajectory. The experiment results demonstrated the outstanding performance of our two-stage approach, compared with other state-of-the-art in the complex map, long-range, and dense crowd navigation tasks in terms of success rate and average speed, demonstrating excellent performance in real-world scenarios as well.

## REFERENCES


[1] C. S. Chen, C. J. Lin, and C. C. Lai, "Non-contact service robot development in fast-food restaurants," *IEEE Access*, vol. 10, pp. 31 466–31 479, 2022.

[2] 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.

[3] A. J. Sathyamoorthy, U. Patel, T. Guan, and D. Manocha, "Frozone: Freezing-free, pedestrian-friendly navigation in human crowds," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4352–4359, 2020.

[4] L. C. Garaffa, M. Basso, A. A. Konzen, and E. P. de Freitas, "Reinforcement learning for mobile robotics exploration: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[5] R. Guldenring, M. Görner, N. Hendrich, N. J. Jacobsen, and J. Zhang, "Learning local planners for human-aware navigation in indoor environments," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 6053–6060.

[6] L. Kästner, T. Buiyan, L. Jiao, T. A. Le, X. Zhao, Z. Shen, and J. Lambrecht, "Arena-rosnav: Towards deployment of deep-reinforcement-learning-based obstacle avoidance into conventional autonomous navigation systems," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6456–6463.

[7] B. Brito, M. Everett, J. P. How, and J. Alonso-Mora, "Where to go next: learning a subgoal recommendation policy for navigation in dynamic environments," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4616–4623, 2021.

[8] 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*. Ieee, 2008, pp. 1928–1935.

[9] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *2015 european control conference (ECC)*. IEEE, 2015, pp. 3352–3357.

[10] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022.

[11] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 10 007–10 013.

[12] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dubé, "Robot navigation in crowded environments using deep reinforcement learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5671–5677.

[13] M. Lodel, B. Brito, A. Serra-Gómez, L. Ferranti, R. Babuška, and J. Alonso-Mora, "Where to look next: Learning viewpoint recommendations for informative trajectory planning," *arXiv preprint arXiv:2203.02381*, 2022.

[14] S. Brinkman, "Learning a guidance policy to navigate among dynamic agents in constrained environments with continual reinforcement learning," 2021.

[15] P. Long, T. Fan, 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)*. IEEE, 2018, pp. 6252–6259.

[16] 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 *2020 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2020, pp. 6979–6985.

[17] S. Yao, G. Chen, Q. Qiu, J. Ma, X. Chen, and J. Ji, "Crowd-aware robot navigation for pedestrians with multiple collision avoidance strategies via map-based deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 8144–8150.

[18] H. Ryu, M. Yoon, D. Park, and S.-E. Yoon, "Confidence-based robot navigation under sensor occlusion with deep reinforcement learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 8231–8237.

[19] L. Kästner, J. Li, Z. Shen, and J. Lambrecht, "Enhancing navigational safety in crowded environments using semantic-deep-reinforcement-learning-based navigation," *arXiv preprint arXiv:2109.11288*, 2021.

[20] S. S. Samsani and M. S. Muhammad, "Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5223–5230, 2021.

[21] F. Xia, C. Li, R. Martín-Martín, O. Litany, A. Toshev, and S. Savarese, "Relmogen: Integrating motion generation in reinforcement learning for mobile manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4583–4590.

[22] X. Cao, C. Sun, and M. Yan, "Target search control of auv in underwater environment with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 96 549–96 559, 2019.

[23] D. Dugas, J. Nieto, R. Siegwart, and J. J. Chung, "Navrep: Unsupervised representations for reinforcement learning of robot navigation in dynamic human environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 7829–7835.

[24] C. Pérez-D'Arpino, C. Liu, P. Goebel, R. Martín-Martín, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1140–1146.

[25] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[26] A. Hill, A. Raffin, M. Ernestus, A. Gleave, A. Kanervisto, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Stable baselines," https://github.com/hill-a/stable-baselines, 2018.