Sampling Efficient Deep Reinforcement Learning for Dynamic Navigation with Raw Laser Scans

Wei Zhu and Mitsuhiro Hayashibe

***Abstract*—We present a Sampling Efficient deep reinforcement learning (DRL) framework for Dynamic Navigation (SEDN) directly using raw laser scans. To accelerate DRL training and simulate laser scans, we specially design a kinematics-based simulator. The navigation policy learned in this simulator can be directly transferred into a physics-based Gazebo simulator and real-world scenarios. Moreover, the policy acquired from a specific environment can be generalized into diverse environments that have never been explored. Because the robot motion is highly coupled with dynamic surroundings, we transform the center of the previous laser scans into the center of the current laser scan to individually extract surrounding motion features. To further enhance sampling efficiency, we integrate optimal reciprocal collision avoidance (ORCA) to generate auxiliary action alternatives. Various experiments against state-of-the-art baselines and sim-to-real implementations demonstrate that our approach has a high success rate of dynamic navigation, superior generalizability, and efficient sampling, with the videos shown on the website[1].**

***Index Terms*—Deep reinforcement learning, Dynamic navigation, efficient sampling, laser scans.**

I. Introduction

INDUSTRY 4.0 envisions that robots and humans can cooperate with each other in harmony. This trend promotes the research and development of autonomous driving vehicles and unmanned mobile platforms in human environments. Nevertheless, it is a challenge to create a fully autonomous navigation system in dynamic surroundings. For one thing, it is difficult to precisely detect and recognize moving objects, such as humans, pets, and vehicles. For another thing, accurately predicting their future states, such as position, orientation, and speed, is significantly challenging. Most non-learning navigation algorithms, such as path planning approaches [1] and optimal reciprocal collision avoidance (ORCA) [2], and deep reinforcement learning (DRL)-based navigation studies [3]–[8] require fully known information of dynamic objects. However, it is extraordinarily difficult to obtain this information in real environments because of complex and reciprocal motions of dynamic objects. Moreover, some DRL-based approaches assume that the number of moving objects must be consistent in training and testing scenarios, which restricts their general applications. In contrast, directly leveraging raw-sensor data can release this restriction [9]–[13]. Nevertheless, the raw-sensor data, such as visions and laser scans, exponentially enlarge the observation space of DRL. Consequently, it becomes challenging to obtain feasible navigation policies with colossal observations.

To learn a feasible and generalized navigation policy in diverse environments, millions of pieces of data are required. However, operating real robots to collect samples is impractical and time-consuming. Therefore, simulators are essential to generate training data quickly and safely. Nevertheless, commonly used commercial physics-based simulators lack vision and LiDAR sensors, such as MuJoCo. Moreover, some sensor-integrated simulators can only run in real time and are unable to accelerate data collection, such as Gazebo. Furthermore, random sampling during DRL training may take a long time and even fail in acquiring a feasible navigation policy owing to colossal exploration space, sparse reward, and so forth. In addition, sim-to-real transfer is a crucial issue for DRL-based navigation policies because of discrepancies between simulations and real-world scenarios. Although a plenty of studies have been successfully applied in simulations, real-world implementations are filled with challenges.

In this study, we aim to efficiently learn a navigation policy in a specially designed simulator consisting of dynamic objects and laser scans. Although the training environment has a constant human number, the policy learned from the specific environment can be directly generalized into other scenarios with different human numbers. Such a generalizability is associated with that our method does not rely on specific human states, such as position and speed, but extracts humans' motion features from consecutive laser scans. Additionally, the policy obtained from the specially developed simulator can be directly transferred into a physics-based Gazebo simulator and physical environments. In addition, we significantly improve sampling efficiency by defining ORCA-assisted action space. The main contributions of this study are summarized as follows:

- We create a kinematics-based simulator integrated with a laser sensor. The navigation policy learned from this simulator can be directly transferred into physics-based simulators and physical environments. Moreover, we can generalize the policy to various scenarios having significant differences.
- We utilize consecutive raw-sensor data to individually extract the latent motion features of surrounding objects with various numbers and shapes. Therefore, our end-to-end and straightforward strategy can release the assumption of fully known environments and be generalized into diverse environments.

The authors are with the Department of Robotics, Graduate School of Engineering, Tohoku University, 980-8579, Sendai, Japan. `zhu.wei.r5@dc.tohoku.ac.jp`

[1]https://youtu.be/wL3-diEXMes

- The sampling efficiency is significantly improved by bringing in ORCA-assisted action space.
- Various validations in diverse simulation environments are performed and sim-to-real implementations are realized to demonstrate the generalizability and practicality of the proposed navigation framework.

Our project is publicly accessible at https://github.com/zw199502/RLDynamicNav.

II. Related Works

Navigation in static and dynamic environments. Navigation in static environments has been maturely studied, whereas it is a challenge to design generalized motion planners in dynamic scenarios. For one thing, ROS-based navigation packages[2] can maturely deal with motion planning problems in static and unstructured environments. However, global or local maps are required to plan collision-free and goal-reaching trajectories. Conversely, end-to-end DRL frameworks that utilize raw-sensor data [14]–[21] are prevailing owning to their mapless property and promising representation and self-learning capabilities. For another thing, dealing with dynamic environments is a prohibitive challenge. In regulated dynamic scenarios with explicit motion features, rule-based trajectory-planning algorithms are safe and efficient [22]. However, it is extraordinarily difficult to develop a safe and generalized navigation policy in dynamic environments without obviously regulated patterns.

Non-learning based dynamic navigation. Two commonly used approaches, reciprocal velocity obstacle (RVO) [23] and optimal reciprocal collision avoidance (ORCA) [2], can safely generate an action for each moving object based on a reciprocal assumption. Trajectory-planning-based methods [1], [24], [25] can release this assumption by optimizing robot trajectories with fully known information of surrounding objects, such as their size, position, and velocity. However, both reciprocity and trajectory based motion planners rely heavily on fully known states of surrounding objects, which are cumbersome to accurately obtain in real-world scenarios.

Machine learning based dynamic navigation. Owing to the powerful detection and prediction capabilities of deep learning, researchers have focused on obtaining the size, position, and speed of moving objects at first. Subsequently, optimal navigation trajectories can be planned via this rich information [27], [28]. However, deep learning models may fail in other scenarios because of notable differences between unexplored and supervised datasets. Furthermore, planning algorithms may not be optimal with respect to safety, efficiency, and route distance. In contrast, DRL-based navigation frameworks can optimize and generalize navigation policies via trial-and-error. Among these frameworks, collision avoidance deep reinforcement learning (CADRL) [3], socially aware (SA)-CADRL [4], and GA3C-CADRL [5] pioneer DRL-based navigation in dynamic environments. Meanwhile, SARL [6], relational graph learning (RGL) [7], and decentralized structural (DS)-RNN [29] can acquire optimal navigation strategies by extracting latent relations between humans and robots through attention mechanism. Nevertheless, these DRL models assume that overall information, such as human size, shape, position, and speed, is prior knowledge during simulation training. However, this information is difficult to obtain in real-world scenarios because detecting and tracking humans, and estimating their current and future states is time-consuming and results in inaccuracy. Furthermore, the human number in training and evaluation environments should be consistent, which limits their generalizability in diverse human environments. Conversely, directly leveraging raw-sensor data can overcome these limitations. For instance, a generative adversarial imitation learning (GAIL) [30] DRL model is leveraged to clone the dynamic navigation policy generated from expert demonstrations with only raw RGB-D image observations [9]. However, owing to the generalizability restriction of imitation learning, only relatively simple real implementations are realized in this study. Similarly, another study [10] utilizes a depth camera to obtain surrounding point cloud information, which is further processed to extract surrounding motion features and optimize the navigation policy. Additionally, LiDAR sensors can perform in the same way as cameras [11], [12]. However, because of prohibitively high dimension of raw-sensor data, obtaining a feasible navigation policy may take a long training time, or even be impractical.

[2]http://wiki.ros.org/navigation

III. Approach

We firstly describe a specially designed environment that integrates laser scans for DRL training, then introduce the key elements of our DRL model, and finally illustrate how we improve the sampling efficiency of DRL training. Fig. 1 shows the overview of our approach.

A. Deep Reinforcement Learning

For the navigation task in environments with dynamic obstacles, the mobile robot needs to make decisions to avoid moving objects and reach a target according to surrounding dynamic information gathered by attached sensors such as LiDAR and camera. We formulate this task as a Markov decision process (MDP) defined by a tuple $< S, A, T, R, \gamma >$. S is the state space, including the target position in the robot frame and consecutive laser scans. A stands for the action space, which is composed of X and Y velocities of omni-directional mobile robots. T represents the state transition, that is the next state $\boldsymbol{s}'$ is generated by given current action $\boldsymbol{a}$ and state $\boldsymbol{s}$. Such a state transition is implied by a kinematics-based simulator specially developed in this study. R, with the actual value r, denotes the immediate reward after executing $\boldsymbol{a}$ in $\boldsymbol{s}$. We relate it to the distance from the robot to surrounding obstacles and the distance from the robot to a target point. γ is a discount factor. The goal of navigation task is to figure out a policy π to maximize the expectation of a long-term

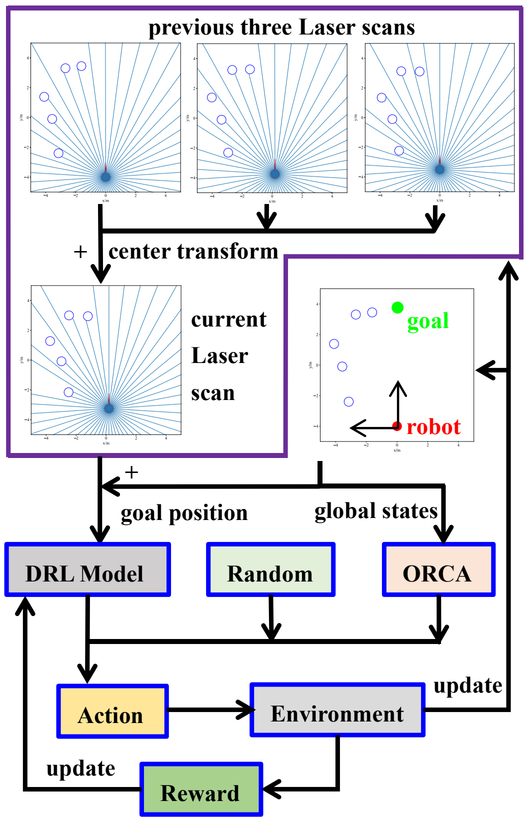


Fig. 1: Overview of our approach. First, the centers of the previous three laser scans are transformed into the center of the current laser scan. Next, these four laser scans and the goal positionin the robot frame are combined as the observations of the DRL model. The action has three options. One choice is generated from the DRL model. Meanwhile, the fully known states, including human sizes, positions, and speeds, are fed into ORCA to generate another action. In addition, a random action is integrated for broad exploration. The final action is selected from these three alternatives to balance exploration and exploitation.

cumulative return $V_\pi(\boldsymbol{s})$:

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} V_\pi(\boldsymbol{s}), \\ V_\pi(\boldsymbol{s}) &= E_\pi\left[\sum_{k=0}^{\infty}\gamma^k r_{t+k+1} | S_t = \boldsymbol{s}\right] \\ &= \sum_{\boldsymbol{a}\in A}\pi(\boldsymbol{a}|\boldsymbol{s})Q_\pi(\boldsymbol{s},\boldsymbol{a}), \end{aligned} \tag{1}$$

where Q is the action-value function. We utilize the Deep Q-learning networks (DQN) [33] algorithm to iteratively update the action-value function and optimize navigation policy:

$$\begin{aligned} Q(S_t, A_t) \leftarrow & Q(S_t, A_t) + \alpha[R_{t+1} + \\ & \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \end{aligned} \tag{2}$$

where α is a step size parameter.

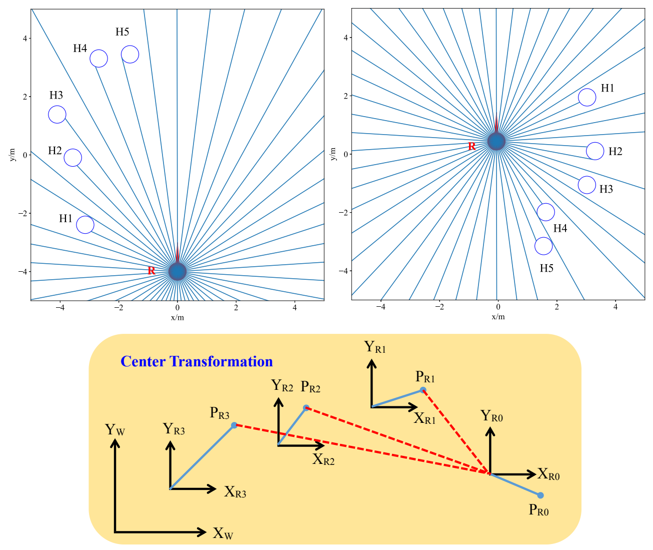


Fig. 2: Raw laser scans and center transformation in training environments. The left top frame displays the initial states and the right top frame shows the intermediate motion. There is a total of 1800 laser scan beams, and a few are shown in these two sub-figures. The bottom figure illustrates the center transformation of the previous three laser scans. The endpoints of previous laser scan beams are transformed into the frame of the current laser scan. Note that although the direction of the laser sensor is constant in training environments, it changes in the real world because the robot's initial pose is randomly set and the robot's motion heavily drifts. Therefore, we require further coordinate transformation in real-world scenarios.

B. Simulation Environment

Simulation plays a significant role in generating sufficient data for optimizing DRL-based policy on robots, especially mobile platforms with complex mechanisms and safety issues, such as quadruped [31] and quadrotor [32] robots. However, the commonly used physical simulation engines either lack sensors, such as MuJoCo, or cannot accelerate sampling, such as Gazebo. Therefore, we aim to create an environment that can simulate moving objects and laser scans, and significantly speed up data collection. Consequently, the navigation policy in dynamic scenarios can be straightforwardly and swiftly optimized in the laser sensor integrated and accelerated simulator.

Two frames of the simulation environment are shown in Fig. 2. Similar to the simplifications in existing studies [6], humans and the robot are represented by circles with variable radius. The moving objects are assumed to be omnidirectional agents with a maximum X/Y speed of $1m/s$. Although the human number is kept at 5, and humans' radius is fixed at $0.3m$ during training, the final learned policy can be generalized into environments with different obstacle numbers, inconsistent sizes and variable shapes of moving objects. Humans are randomly located around a circle with a radius of $4m$. The i-th human moves back and forth, from the position $(x_s^{(i)}, y_s^{(i)})$ and the position $(-x_s^{(i)}, -y_s^{(i)})$. Meanwhile, human motions are generated by ORCA. In addition, the robot is invisible to humans; otherwise, it will be difficult to differentiate whether our motion planner is effective or whether humans avoid the robot.

For state-of-the-art DRL-based crowd navigation algorithms [3], [4], [6]–[8], the states of all humans, including human number, size, position and speed, are assumed to be fully known in simulations. However, these states are derived from raw-sensor data such as RGB-D images and laser scans in the real world. The cumbersome processing of raw-sensor data may cause inaccurate human detection and state estimation due to sensor limitations, mutual blocking, and algorithm flaws. Furthermore, the human number in training and evaluation scenarios must be same because of the input dimension immutableness of neural networks. Consequently, the application of the learned RL policy is restricted in specific human scenarios. Conversely, using direct raw-sensor data as the input can effectively eliminate this restriction. Nevertheless, simulating raw-sensor data is difficult because of high dimensionality. We aim to update an 1800D, 360° laser scan with the robot's motion to match the Velodyne LiDAR sensor used in the real world. Although we only display the intersection between the laser scan and the circles shown in Fig. 2, any straight lines can also be included in the environment. Therefore, moving objects are not only limited to circles, but to all shapes. To accelerate calculation, we create a C language library to generate laser scans. We found that simulating one laser scan only took $2ms$, which was approximately 30 times faster than the case of using Python. Furthermore, the center transformation shown in Fig. 1 and Fig. 2 was also achieved with C language. The total time for processing laser scans in each simulation step was approximately $11ms$, which significantly accelerated DRL training.

Instead of using original laser scans, we transform the centers of previous laser scans to individually extract surrounding motion features, shown in Fig. 2. Specifically, we have four consecutive laser scans and their centers are different because the robot is moving. For each of the previous three laser scans, we can calculate the end point position of each beam in the world frame. Next, we transform these end points into the local frame of the current laser scan. Subsequently, we are able to obtain a new laser scan that has the same center as that of the current laser scan. Therefore, we can exclude the robot motion and only represent the human motion with the transformed laser scans. We then feed the transformed laser scans into our DRL model to individually extract the motion features of humans.

C. Elements of the DRL Model

States. The states s_t are composed of two parts $\boldsymbol{s}_t = [\boldsymbol{s}_t^l, \boldsymbol{s}_t^g]$: four center transformed laser scans $\boldsymbol{s}_t^l \in \Re^{7200}$ and the goal position in the robot frame $\boldsymbol{s}_t^g \in \Re^2$. We represent the goal position in the form of polar coordinates, $\boldsymbol{s}_t^g = [r_g, \theta_g]$. To extract the features of human motions, we utilize four consecutive laser scans with a $dt = 250ms$ time interval between two neighboring frames. In contrast to one pioneering study directly using original consecutive laser scans [11] and another one slightly processing original laser scans to guarantee that all scans start from a fixed direction [12], we transform the centers of previous laser scans into the center of the current laser sensor shown in Fig. 2. Consequently, we can decouple robot's motion and independently extract surrounding motion features.

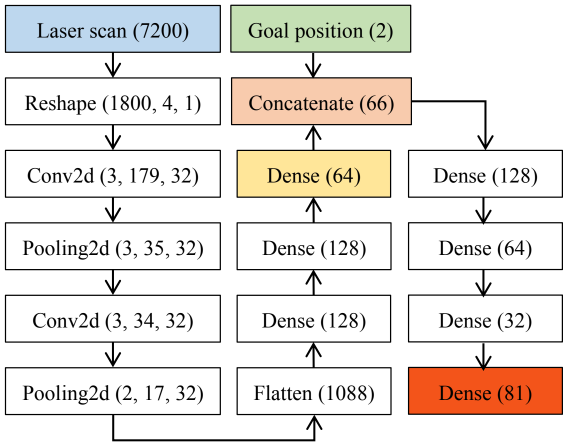


Fig. 3: Network structure. For each unit except the laser scan and the goal position, the left text represents the network operation, and the right tuple is the output dimension after operation.

Actions. We define the action space as a set of 81 discrete speed pairs. Because the robot motion is assumed to be omni-directional with a maximum X/Y speed of $1m/s$, we discretize the X/Y speed as $-1.0 + 0.25i$ with $i = 0, 1, ..., 8$. Pairing the X-and Y-speeds forms the final action space. Note that the holonomic assumption can be released by further planning the linear and angular velocities for differential mobile robots, which is verified in the experiment section.

Rewards. The two fundamental navigation goals in dynamic environments are collision-free and target-reaching; therefore, the reward function is defined by two parts:

$$R(\boldsymbol{s}_t) = R_l(\boldsymbol{s}_t^l) + R_g(\boldsymbol{s}_t^g), \tag{3}$$

where $R_l(\boldsymbol{s}_t^l)$ represents the collision penalty, and $R_g(\boldsymbol{s}_t^g)$ denotes the goal-reaching award. Let r be the robot radius, dt be the time of one simulation step, $d_{\min}$ be the minimum range of the current laser scan, and d_c be the comfortable threshold distance from the robot center to the human circle rim. Subsequently, $R_l(\boldsymbol{s}_t^l)$ is defined as:

$$R_l(\boldsymbol{s}_t^l) = \begin{cases} -1.0 & d_{\min} \le r \\ -0.5 \cdot dt \cdot (d_c - d_{\min}) & r < d_{\min} \le d_c \\ 0 & d_{\min} > d_c \end{cases}. \tag{4}$$

The second case in (4) enables the robot to maintain a safe distance from humans, thus avoiding possible collisions ahead. Let r_{t-1}^g be the last distance from the goal to the robot, and r_t^g be the present distance, we can define $R_g(\boldsymbol{s}_t^g)$ as:

$$R_g(\boldsymbol{s}_t^g) = \begin{cases} 1.0 & r_t^g \le r \\ 0.01 \cdot (r_{t-1}^g - r_t^g) & r_t^g > r \end{cases}. \tag{5}$$

The last case in (5) encourages the robot to get close to the goal, thus avoiding a possibly invalid exploration.

Network structure. We leverage the DQN algorithm [33] to model the dynamic navigation, with the network structure illustrated in Fig. 3. First, a 64-D feature vector, which can be regarded as a certain prediction for future human states, is extracted from a concatenated 7200-D laser scan through convolutional neural networks (CNNs). Next, the feature vector concatenates the goal position to create a new low-dimension state for another deep neural network. Finally, 81-D Q values that correspond to the aforementioned discrete action space are derived from the dimension reduced state. An action alternative, shown in Fig. 1, is selected from the index corresponding to the maximum Q value.

D. Improvement of Sampling Efficiency

Because of the prohibitively enormous state space and the relatively large action space, completely random sampling during DRL training may take a long time and even fail to obtain a feasible navigation policy. To improve sampling efficiency, ORCA is integrated to generate samples to fill the experience pool of the DQN model. More specifically, the robot knows all humans' global states and its own states, including the radius, position, and X/Y speed. Therefore, an alternative action can be derived by ORCA using this fully known information. However, collisions still occur although using the mature ORCA approach because the robot is invisible to humans. To further avoid collisions, a random disturbance is added to the action produced by ORCA, which yields the second alternative action to explore possibly better decisions. The third action is output from the DQN model for exploitation. The final action is selected from these three alternatives (shown in Fig. 1) at different probabilities to balance exploration and exploitation. Additionally, the probability of selecting the ORCA-generated action decreases, whereas the probability of choosing the action produced by our DRL model becomes larger during training. Moreover, the experience pool of the DQN model is initialized by ORCA. In addition, the neural networks are pre-trained at a relatively high learning rate by replaying the initial experiences to further increase sampling efficiency.

E. SEDN Algorithm

We summarize our Sampling Efficient deep reinforcement learning for Dynamic Navigation (SEDN) as Algorithm 1. In training, the maximum capacity of the experience pool EP_m is 10^5 for the off-policy RL framework. We first set the learning rate as $\alpha_1 = 0.001$ and pretrain the DRL model $\text{T}_1 = 2000$ times to initialize the deep neural networks. Then, we repeat the trial and error for $\text{EP}_m = 2 * 10^4$ episodes. After each episode, we train the model $\text{T}_2 = 20$ times with a smaller learning rate $\alpha_2 = 0.0001$. We decay the exploration factor from $\epsilon_s = 0.8$ to $\epsilon_e = 0.03$ within $\text{EP}_d = 1.6 * 10^4$ episodes to progressively balance exploration and exploitation.

Algorithm 1: SEDN algorithm

```
Prefill: Use ORCA to prefill the experience pool with
  a maximum capacity $\text{EX}_m$;
Pretrain: Pretrain the DRL model $\text{T}_1$ times with a
  learning rate $\alpha_1$;
$episode \leftarrow 0$;
while $episode < \text{EP}_m$ do
    Randomize the initial human positions;
    $done \leftarrow False$;
    while not done do
        if $\text{EP}_d < episode$ then
            $p_1 = \epsilon_s - (\epsilon_s - \epsilon_e)/\ \text{EP}_d * episode$;
        else
            $p_1 = \epsilon_e$;
        end
        $p_2 = Random(0, 1)$;
        if $p_2 < p_1$ then
            Generate action with ORCA;
            Add noise to the action;
        else
            Generate action with the DRL model;
        end
        Interact with the environment and update done;
    end
    if collision or target reaching then
        Add the episode data to the eperience pool;
    end
    Train the DRL model $\text{T}_2$ times with a learning rate
      $\alpha_2$;
    $episode \leftarrow episode + 1$;
end
```

IV. EXPERIMENTS

We first trained the DRL model in a specially designed simulator with laser scans. Subsequently, we compared our navigation framework with the EGO baseline that similarly utilized raw laser scans [12]. In addition, we revised our model to yield another 6 baselines to explain the motivations of our DRL motion. Moreover, 4 state-of-the-art approaches that assume fully known environments were reproduced to demonstrate our method is straightforward and can perform as well as even better than these methods with ideal assumptions. Meanwhile, the generalizability with respect to variable human numbers was verified by executing the trained model in different human scenarios. Besides, our method could deal with hybrid environments with static and dynamic obstacles. We further validated the transfer capability of our navigation framework by directly deploying the learned policy in a physics-based Gazebo simulator and various real-life scenarios without retraining and fine-tuning. The physics-based experimental environments are shown in Fig. 4.

A. Training

To verify the high sampling efficiency and superior navigation performance of our method, we reproduced a pioneering baseline – EGO [12], for comparison. Because the EGO algorithm is not publicly accessible, we reproduced it according to the aforementioned definitions in Section III. The state space of EGO is four consecutive laser scans without center transformation. EGO's action space is same

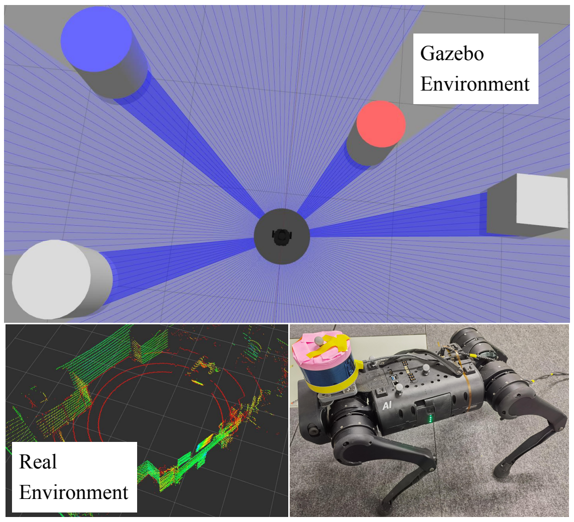


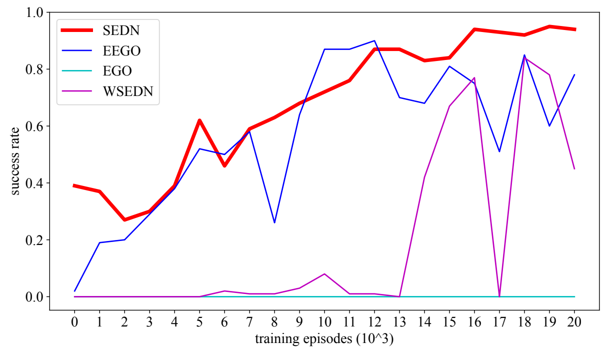
Fig. 4: Physics-based experiment environments. The top image displays the Gazebo environment and the bottom two figures show the real-world scenario. In Gazebo, surrounding motions are generated by ORCA. In the real world, we transform the 3-D LiDAR point cloud into a 2-D laser scan. We utilize external motion capture system to localize the robot because the quadruped robot's odometer drifts heavily.

as our method's. Because EGO does not have ORCA-assisted samples, its success rate shown in Fig. 5 is zero at all times. Subsequently, we enhanced EGO's sampling efficiency by integrating ORCA, namely EEGO. As shown in Fig. 5, EEGO's peak success rate is significantly increased to 90%. Additionally, we weakened our method by removing ORCA-assisted samples, namely WSEDN. As shown in Fig. 5, WSEDN's success rate is decreased to 84% and becomes notably unstable, which further indicates that ORCA-assisted samples can significantly improve learning efficiency. Moreover, EGO can rarely realize successful navigation. Comparatively, WSEDN is able to improve the success rate to 84%, which qualitatively demonstrates that the center transformation for laser scans can improve navigation performance.

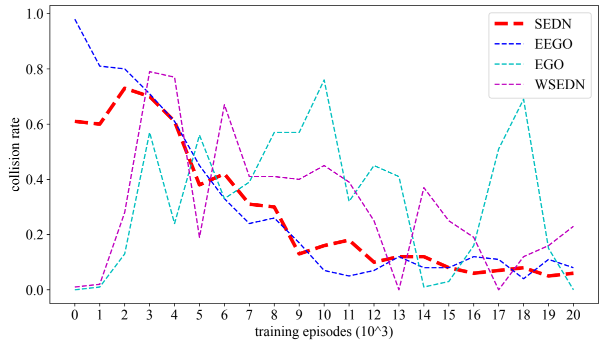
In summary, EEGO and SEDN, with maximum success rates of 90% and 95%, notably outperformed EGO-0% and WSEDN-84%, respectively. This qualitative and quantitative analysis verifies that the ORCA-assisted sampling is able to enhance learning efficiency. Although EEGO can progressively improve the success rate of dynamic navigation, its stable success rate is approximately 80%, while the one for SEDN is approximately 94%, which further validates that the center transformation can improve dynamic navigation.

B. *Model Revision*

Firstly, we removed the CNN shown in Fig. 3 to simplify our network structure, namely SEDN-Simplified. Secondly, we replaced our DQN framework with soft actor critic (SAC) – SEDN-SAC. Thirdly, we down-sampled the raw laser scan from 1800D to 60D and leveraged the deep deterministic policy gradient (DDPG) algorithm similar to an existing study [17]. We removed the CNN because of the low-dimension

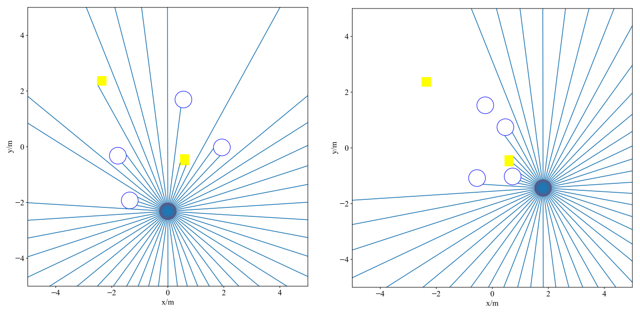


(a) Success rate



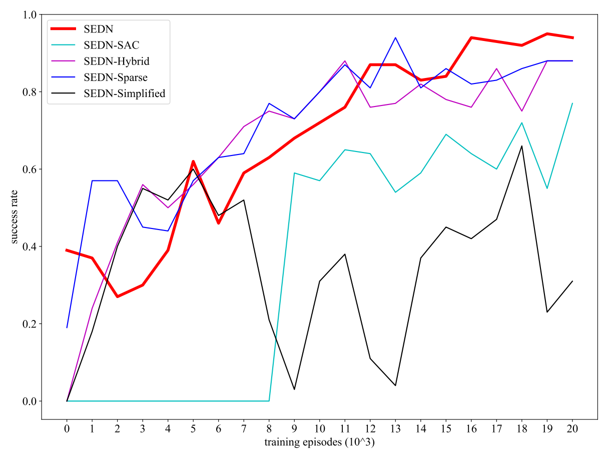
(b) Collision rate

Fig. 5: Training process. The result of each training episode is either successful navigation, collision, or running overtime. We illustrate the success rate in the top figure and the collision rate is shown in the bottom figure. We evaluated the model 100 times every 1000 training episodes.

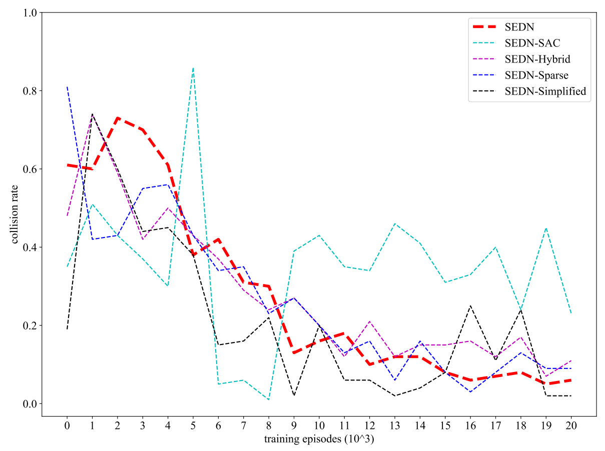


(a) Scene 1 (b) Scene 2

Fig. 6: Hybrid scenarios with both dynamic and static obstacles. The circles are dynamic objects while the yellow rectangles are static barriers with random positions. The side length of the rectangle varies from 0.3 to 0.4m during training.



(a) Success rate



(b) Collision rate

Fig. 7: Training process of revised SEDN baselines. We omit SEDN-DDPG-Simple and SEDN-SAC-Simple because their success rates are zero at all times.

observation. We name this baseline SEDN-DDPG-Simple. Based on SEDN-DDPG-Simple, we substitute DDPG with SAC to create the fourth baseline – SEDN-SAC-Simple. Our fifth and sixth baselines are related to environment settings. To explore the effect of reward definition, we removed the goal distance reward in (5) to generate more sparse reward – SEDN-Sparse. Finally, we constructed a hybrid environment with both static and dynamic obstacles shown in Fig. 6 – SEDN-Hybrid. The corresponding training processes of these six baselines are shown in Fig. 7.

As shown in Fig. 7, the performance of SEDN-Simplified is significantly degraded. We conclude that CNNs can extract latent and intrinsic features from high-dimension data, which motivates us to choose complex neural networks. Meanwhile, SEDN-SAC yields less efficient training because its peak success rate is 77% while the original method reaches 94%. We have to mention that the SAC algorithm is able to perform as well as the DQN approach via carefully tuning hyper-parameters and network structures. The motivation of selecting the DQN framework is that DQN has relatively simpler network structures and fewer hyper-parameters. SEDN-DDPG-Simple and SEDN-SAC-Simple result in zero success rate at all times. We think that dense sensor data are required to detect small moving objects. When the reward function becomes sparse, the success rate can still reach 88% but is slightly lower than that of the dense reward. We can imply that dense reward is able to improve navigation performance. Although we mix rectangular obstacles, our laser scan simulator is still able to detect surroundings. Moreover, the proposed method can deal with hybrid scenarios (4 moving objects and 2 static obstacles) with a high success rate (88%). Additionally, we found that most of the failure cases occurred when the robot collided with moving objects instead of static ones, which further indicates that our method is able to handle static barriers.

TABLE I: Comparison in terms of success rate and collision rate in 500 random tests. We omit SEDN-SAC-Simple and SEDN-DDPG-Simple because neither of them can achieve collision-free and target-reaching navigation.

Policy	Success Rate	Collision Rate
*ORCA [2]	0.43	0.57
EEGO [12]	0.83	0.08
*RGL [7]	0.83	**0.03**
*CADRL [3]	0.94	0.06
*SARL [6]	0.88	0.08
SEDN	**0.94**	0.06
SEDN-Simplified	0.66	0.24
SEDN-SAC	0.77	0.23
SEDN-Sparse	0.88	0.09
SEDN-Hybrid	0.88	0.11
*assume fully known environments		

C. Evaluation

We reproduced four open-source baselines, namely, ORCA [2], SARL [6], RGL [7], and CADRL [3]. These state-of-the-art baselines assume fully known information including human number, size, position, and speed. In addition, we implemented another baseline – EEGO, directly using laser scans [12] similar to our observation space. We implemented 500 tests for each baseline, with the comparison results shown in Table I. Our method quantitatively outperforms or behaves as well as these 5 baselines in terms of the success rate of dynamic navigation.

Although CADRL has the same success rate as our method shown in Table I, it relies heavily on prior knowledge of human number, size, position, and speed. In real worlds, it is a challenge to precisely obtain these human states owing to complex uncertainties. Moreover, CADRL, SARL, and RGL require that human number in training and evaluation environments should be consistent. The DRL models of CADRL, SARL, and RGL are required to be retrained when human number changes, which restricts the generalizability with respect of diverse human scenarios. Conversely, our method releases

TABLE II: Comparison with different policies deployed in various human scenarios. The two values in one unit represent the success and collision rates, respectively.

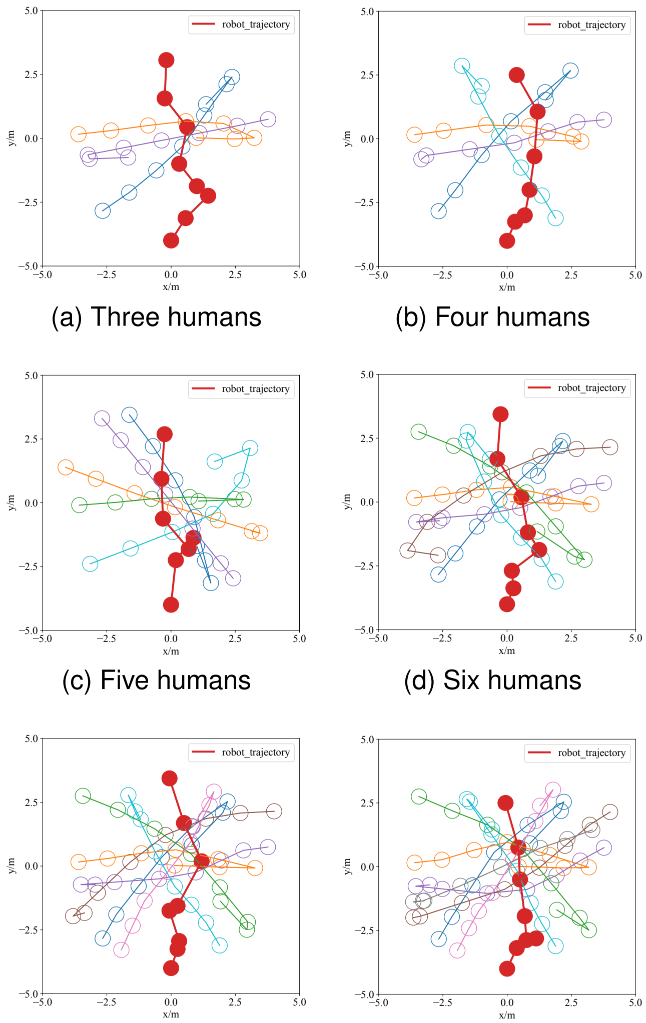
Policy	ORCA [2]	EEGO [12]	**SEDN**
human number	success rate / collision rate		
3	0.68/0.32	0.96/0.02	**0.97/0.02**
4	0.56/0.44	0.90/0.05	**0.97/0.03**
6	0.34/0.66	0.76/0.13	**0.92/0.07**
7	0.31/0.69	0.72/0.14	**0.88/0.10**
8	0.27/0.73	0.66/0.17	**0.82/0.14**

this ideal assumption by directly leveraging consecutive laser scans. We can straightforwardly extract surrounding motion features from raw-sensor data. Therefore, our DRL model can be directly generalized to variable scenarios with different human numbers without retraining or fine-tuning once it is optimized in a specific environment. Compared with the baselines requiring fully known environments, our approach is more straightforward because we directly project the raw-sensor data to the robot action, excluding cumbersome human detection, tracking, and speed estimation. We first train our DRL model in an environment with 5 humans. Subsequently, the learned policy can be directly deployed in environments with 3, 4, 6, 7, and 8 humans, which further demonstrates the straightforward advantage of our approach. The video is shown on our website[1]. The representative trajectories of humans and the robot are illustrated in Fig. 8. We also compare our method with another 2 baselines that are independent of human numbers, with the results illustrated in Table II. 500 validations were executed for each environment, and our method outperformed the baselines in terms of both the success and collision rate.

Moreover, we revised our model and environment settings to create another 6 baselines mentioned in Section IV-B, with the results shown in Table I. For one thing, when we changed our DRL model such as simplifying the network structure and choosing an advanced RL algorithm, the navigation performance was not improved. For another thing, when we dowm-sampled sensor data, we could rarely obtain a feasible navigation strategy. An interesting result is that the navigation behavior was not significantly degraded when we chose a sparse reward and hybrid environments. These exploratory results demonstrate that our original DRL-based navigation framework is superior and robust to environment variations.

D. Implementation in Physics based Environments

Because the maximum side speed of the omnidirectional quadrupedal robot (shown in Fig. 4) is $0.3m/s$, we reduce the action space from $\{-1.0 + 0.25i\}$ with $i = 0, 1, ..., 9$ to $\{-0.3 + 0.1j\}$ with $j = 0, 1, ..., 6$. In addition, because the motion area is a relatively small rectangle with a length of $3.4m$ and a width of $2.4m$, we reduce the human number from 5 to 3 and initialize these humans around a $2m$-radius



(a) Three humans (b) Four humans
(c) Five humans (d) Six humans
(e) Seven humans (f) Eight humans

Fig. 8: Generalization in various scenarios with different human numbers. The trajectories of humans and the robot are represented by successive circles with the same time interval. The motion of humans is generated using ORCA with the robot invisible. The red line represents the discrete trajectory of the robot and other lines are the trajectories of humans.

circle. Owing to the speed limitation of the robot, the human's maximum X/Y speed is correspondingly decreased to $0.3m/s$. Otherwise, the robot will not have enough time to avoid fast humans. We retrained our DRL-based navigation model because environment settings significantly changed. The success rate in 500 tests could still reach 96% despite the changes of environment settings, which demonstrates that our DRL-based navigation model can be generalized into diverse environments.

Although we optimize our navigation policy in a kinematics-based simulator, the simulated policy can be directly deployed in both a physics-based Gazebo simulator and the real world without any retraining or fine-tuning. Figs. 9 and 10 illustrate various Gazebo scenarios and real-world scenarios with diverse shapes, sizes, numbers, and motions of obstacles.

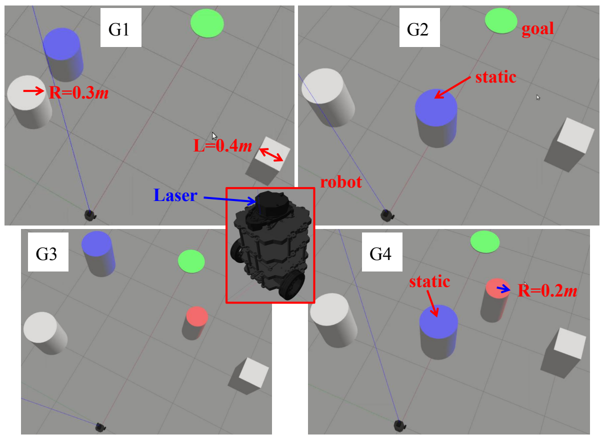


Fig. 9: Gazebo scenarios with the motion area 4×4m. G1 has three dynamic objects, whereas one obstacle is static in G2. Adding one smaller dynamic object into G1 and G2 yields G3 and G4, respectively.

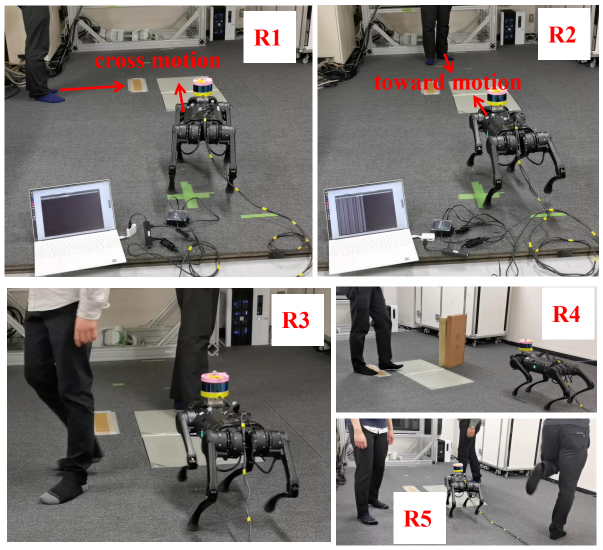


Fig. 10: Real environments with 2.4×3.4m accessible rectangle area. R1 and R2 have one human with a cross and toward motion, respectively. Two humans randomly walk in R3, while one dynamic human and one static obstacle are included in R4. R5 has three moving humans.

Moreover, we can replace omnidirectional mobile robots with differential wheeled robots. More specifically, we can calculate the desired waypoint according to the X/Y speeds output from our DRL-based navigation policy. Subsequently, we can plan robot's linear and angular velocities via the follow the carrot (FTC) algorithm[3] to direct the robot towards the waypoint.

In addition to the policy transfer from a kinematics-based simulator to another physics-based simulator, the sim-to-real transfer is implemented in various environments. In real scenarios, we deploy our simulated navigation policy on a quadruped robot, Unitree-A1[4] equipped with a Velodyne LiDAR sensor, as shown in Fig. 4. We found that the actual speed generated by the official controller was significantly different from that of the desired command. Therefore, we use the system identification technique to calibrate the desired speed command to be consistent with the actual speed. Although the motion and laser scan are 2D in the simulator whereas the locomotion and LiDAR data are 3D in the real world with complex uncertainties, the policy acquired from the simulator can be successfully implemented in physics-based environments, with the video shown on our website[1]. Note that we only test our method in a small area because of the limitation of the external motion capture system. Additionally, the humans' motions are slow due to the speed limitation of the quadruped robot. Although our method could behave well in certain scenarios, it also failed because our simulation training could not cover all patterns of human motions. Therefore, our future study will focus on improving the generalizability in the real world.

[3]http://wiki.ros.org/asr_ftc_local_planner
[4]https://www.unitree.com/products/a1/

V. Conclusions

We presented a sampling-efficient deep reinforcement learning framework for dynamic navigation with direct raw laser scans. A kinematics-based simulator with a high running speed was specially developed to simulate laser scans and accelerate DRL training. The dynamic navigation policy optimized in this simulator can be directly deployed in a physics-based Gazebo simulator and real-world scenarios. Sufficient transfer implementations demonstrate the effectiveness of our specially designed simulator and the sim-to-real transfer capability of our DRL-based navigation framework. Our end-to-end navigation strategy is straightforward because we directly utilized raw consecutive laser scans. Additionally, we transformed the center of previous laser scans into the center of the current laser sensor to individually extract surrounding motion features. Comparison results demonstrated that center transformation could improve dynamic navigation. To further increase sampling efficiency and quickly acquire a superior navigation policy, we integrated ORCA to generate assistive samples for model training. Sufficient simulation comparisons demonstrated that our approach could achieve faster learning, better navigation performance, and superior generalizability with respect to diverse environments.

Although we achieved some sim-to-real cases, the success rate was not high enough for real industrial applications. One reason is that our simulation can not cover all motion patterns of dynamic obstacles. Therefore, one of our future efforts is to improve sim-to-real performance. Besides, our future study will focus on the dynamic navigation using nonholonomic mobile robots because of their universality in the human society. In addition, we need to further improve the generalizability with respect to hybrid dynamic environments having humans, vehicles, and more irregular obstacles.

References

[1] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902-911, 2009.
[2] J. Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research*, Springer, Berlin, Heidelberg, 2011, pp. 3-19.

[3] Y. Chen, M. Liu, M. Everett, and J. How, “Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 285-292.

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

[5] M. Everett, Y. F. Chen, and J. How, “Motion Planning among dynamic, decision-making agents with deep reinforcement learning,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3052-3059.

[6] 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)*, 2019, pp. 6015-6022.

[7] 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)*, 2020, pp. 10007-10013.

[8] S. Samsani and M. 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.

[9] L. Tai, J. Zhang, M. Liu, and W. Burgard, “Socially compliant navigation through raw depth inputs with generative adversarial imitation learning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1111-1117.

[10] J. Choi, K. Park, M. Kim, and S. Seok, “Deep reinforcement learning of navigation in a complex and crowded environment with a limited field of view,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5993-6000.

[11] T. Fan, P. Long, W. Liu, and J. Pan, “Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios,” *The International Journal of Robotics Research*, vol. 39, no. 7, pp. 856-892, 2020.

[12] 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)*, 2020, pp. 6979-6985.

[13] 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)*, 2021, pp. 7829-7835.

[14] Y. Zhu, et al., “Target-driven visual navigation in indoor scenes using deep reinforcement learning,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357-3364.

[15] L. Tai, G. Paolo, and M. Liu, “Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31-36.

[16] M. Pfeiffer, et al., “Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 4423-4430, 2018.

[17] J. Jesus, J. Bottega, M. Cuadros and D. Gamarra, “Deep Deterministic Policy Gradient for Navigation of Mobile Robots in Simulated Environments,” in *International Conference on Advanced Robotics (ICAR)*, 2019, pp. 362-367.

[18] H. Shi, L. Shi, M. Xu, and K. Hwang, “End-to-end navigation strategy with deep reinforcement learning for mobile robots,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 4, pp. 2393-2402, 2020.

[19] E. Marchesini and A. Farinelli, “Discrete deep reinforcement learning for mapless navigation,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 10688-10694.

[20] G. Kahn, P. Abbeel, and S. Levine, “BADGR: An autonomous self-supervised learning-based navigation system,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1312-1319, 2021.

[21] Y. Zhu, Z. Wang, C. Chen and D. Dong, “Rule-based reinforcement learning for efficient robot navigation with space reduction,” *IEEE/ASME Transactions on Mechatronics*, vol. 27, no. 2, pp. 846-857, 2022.

[22] A. Bolu and O. Korcak, “Path planning for multiple mobile robots in smart warehouse,” in *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, 2019, pp. 144-150.

[23] J. Berg, M. Lin, and D. Manocha, “Reciprocal velocity obstacles for real-time multi-agent navigation,” in *2008 IEEE International Conference on Robotics and Automation (ICRA)*, 2008, pp. 1928-1935.

[24] J. Guzzi, A. Giusti, L. Gambardella, G. Theraulaz, and G. Caro, “Human-friendly robot navigation in dynamic environments,” in *2013 IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 423-430.

[25] H. Dong, C. Weng, C. Guo, H. Yu and I. Chen, “Real-time avoidance strategy of dynamic obstacles via half model-free detection and tracking with 2D lidar for mobile robots,” *IEEE/ASME Transactions on Mechatronics*, vol. 26, no. 4, pp. 2215-2225, 2021.

[26] S. Mitsch, K. Ghorbal, D. Vogelbacher, and A. Platzer, “Formal verification of obstacle avoidance and navigation of ground robots,” *The International Journal of Robotics Research*, vo. 36, no. 12, pp. 1312-1340, 2017.

[27] C. I. Mavrogiannis, V. Blukis, and R. A. Knepper, “Socially competent navigation planning by deep learning of multi-agent path topologies,” in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 6817-6824.

[28] A. Mateusa, D. Ribeiroa, P. Miraldoab, and J. C. Nascimentoa, “Efficient and robust pedestrian detection using deep learning for human-aware navigation,” *Robotics and Autonomous Systems*, vol. 113, pp. 23-37, 2019.

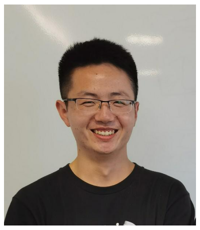
[29] S. Liu, P. Chang, W. Liang, N. Chakraborty, and K. Driggs-Campbell, “Decentralized structural-RNN for robot crowd navigation with deep reinforcement learning,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 3517-3524.

[30] J. Ho and S. Ermon, “Generative adversarial imitation learning,” in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 4565-4573.

[31] J. Hwangbo, et al., “Learning agile and dynamic motor skills for legged robots,” *Science Robotics*, vol. 4, no. 26, 2019.

[32] K. Kang, S. Belkhale, G. Kahn, P. Abbeel, and S. Levine, “Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight,” in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 6008-6014.

[33] V. Mnih, et al, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, pp. 529-533, 2015.



Wei Zhu received the bachelor’s degree in intelligent science and technology from the School of Computer Science and Control Engineering, and the master’s degree in Control Science and Engineering from the School of Artificial Intelligence, Nankai University, Tianjin, China, in 2017 and 2020 respectively. He is currently pursuing the Ph.D. degree with the Department of Robotics, Tohoku University, Sendai, Japan.

His research interests include deep reinforcement learning, snake-like robots, wheeled bipedal robots, and autonomous navigation.

Mitsuhiro Hayashibe (Senior Member, IEEE) received the B.S. degree from the Tokyo Institute of Technology, in 1999, and the M.S. and Ph.D. degrees from the Graduate School of Engineering, The University of Tokyo, in 2001 and 2005, respectively. From 2001 to 2006, he was an Assistant Professor with the Department of Medicine, Jikei University School of Medicine.

In 2007, he was a Postdoctoral Fellow with the Institut National de Recherche en Informatique et en Automatique (INRIA) and the Laboratoire d?Informatique, de Robotique et de Microelectronique de Montpellier (LIRMM), CNRS/University of Montpellier, France. Since 2008, he has been a tenured Research Scientist with INRIA and the University of Montpellier. Since 2012, he has been a Visiting Researcher with the RIKEN Brain Science Institute and TOYOTA Collaboration Center. Since 2017, he has been a Professor with the Department of Robotics, Tohoku University, and founder of the Neuro-Robotics Laboratory. He is currently a Senior Member of the IEEE Engineering in Medicine and Biology Society. He was a recipient of the 15th Annual Delsys Prize 2017 for Innovation in Electromyography from De Luca Foundation, USA. He serves as the Technical Activity Board Member for the International Foundation of Robotics Research (IFRR).