2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Learn to Navigate in Dynamic Environments with Normalized LiDAR Scans

Wei Zhu and Mitsuhiro Hayashibe

Abstract— **The latest robot navigation methods for dynamic environments assume that the states of obstacles, including their geometries and trajectories, are fully observable. While it's easy to obtain these states accurately in simulations, it's exceedingly challenging in the real world. Therefore, a viable alternative is to directly map raw sensor observations into robot actions. However, acquiring skills from high-dimensional raw observations demands massive neural networks and extended training periods. Furthermore, there are discrepancies between simulated and real environments that impede real-world implementations. To overcome these limitations, we propose a Learning framework for robot Navigation in Dynamic environments that uses sequential Normalized LiDAR (LNDNL) scans. We employ long-short-term memory (LSTM) to propagate historical environmental information from the sequential LiDAR observations. Additionally, we customize a LiDAR-integrated simulator to speed up sampling and normalize the geometry of real-world obstacles to match that of simulated objects, thereby bridging the sim-to-real gap. Our extensive comparisons with state-of-the-art baselines and real-world implementations demonstrate the potentials of learning to navigate in dynamic environments using raw sensor observations and sim-to-real transfer.**

I. INTRODUCTION

Service mobile robots have been making their presence felt in human societies. These robots are being deployed across various industries, including healthcare, hospitality, retail, and logistics, to improve efficiency, reduce costs, and enhance customer experiences [1]. However, safely and efficiently navigating service mobile robots in human-rich and highly dynamic environments remains an exceedingly difficult task.

Map-based (MB) navigation strategies have been widely utilized for mobile robots in stable environments [2], [3]. Various studies have employed high-resolution maps to search for global paths via efficient planning algorithms, such as A-Star (A*) [4] and rapidly-exploring random tree (RRT) [5]. Local motion planners, such as dynamic window approach (DWA) [6] and directional approach [7], are subsequently executed to track the paths and avoid obstacles. However, maintaining global maps can be time-consuming, especially in environments that undergo consistent changes. Furthermore, local motion planners are sensitive to moving obstacles, particularly in environments with a high human presence.

To reduce the time and effort required for map maintenance, map-free and end-to-end deep reinforcement learning (DRL) based navigation methods have become increasingly popular in the research literature [8]. However, early studies [9]–[17] were limited to relatively stable environments where obstacles were static, and testing configurations did not significantly deviate from the training settings. More recent studies have focused on using DRL to navigate agents in highly dynamic environments, but they assume that other moving agents' states, including size, shape, number, and speed, are fully known [18]–[25]. Obtaining these states with accuracy in real-world scenarios can be particularly challenging, which has limited the practical applications of these cutting-edge approaches. Alternatively, directly mapping continuous raw sensor observations to robot actions is an appealing navigation strategy that eliminates the need for assuming fully known environments [26]–[32]. Nevertheless, commonly used simulators which can generate raw sensor observations can not significantly accelerate running, such as Gazebo [33] and V-REP [34]. Moreover, sim-to-real transfer is a significant challenge for these approaches, as simulation settings cannot fully replicate real-world situations. Furthermore, implementing navigation policies that demand large networks on small mobile robots with limited computational resources presents a significant challenge.

Our study comprehensively addresses the aforementioned shortcomings by employing the following techniques. Firstly, we designed a simulator that is equipped with a LiDAR sensor, which significantly accelerates running during DRL training. Secondly, to simulate real-world scenarios, we assume that dynamic humans have a circle shape, and static obstacles are rectangles with variable sizes. Since the contours of real-world obstacles are notably different from the shapes of simulated objects, we normalized the collision margins of real-world obstacles as circles or rectangles with variable sizes. We accomplished this by employing clustering algorithms to localize and frame moving humans or static obstacles, and obtain their centroids and circumscribed cuboids in 3D space. We can then normalize obstacles as circles or rectangles on a 2D plane. Subsequently, we can re-generate 2D LiDAR scans according to the normalized obstacles. Thirdly, instead of using tens of continuous LiDAR scans [29] or high-dimensional depth images [32] which require colossal networks for decoding, we leverage long-short term memory (LSTM) to process ego-centric sequential LiDAR scans, which reduces the consumption of hardware resources and enables deployment on small mobile robots with limited computational resources. We name our approach Learn to Navigate in Dynamic environments with Normalized LiDAR

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

scans (LNDNL). In summary, our main contributions are outlined as follows.

- We have developed a specialized simulator that can efficiently generate LiDAR observations. This simulator allows for the incorporation of numerous objects in motion, each with their own distinct shapes.
- To enable seamless transfer of simulations to the real world, we ensure that collision margins of real-world obstacles are normalized. Our approach involves using clustering techniques to localize and frame obstacles from 3D LiDAR pointclouds. Additionally, we re-generate 2D LiDAR scans from the normalized obstacles to be consistent with simulated settings. We name these procedures as LiDAR scan normalization.
- We present a navigation framework that employs a combination of LSTM and DRL to translate the sequential normalized LiDAR observations into robot actions from an ego-centric perspective. This framework features lightweight networks, making it feasible for implementation on compact and onboard computers.
- To showcase the benefits of our approach, we compared it with state-of-the-art baselines. Additionally, we conducted extensive real experiments to highlight the potential of sim-to-real transfer.

The code of the whole project is publicly available at `https://github.com/zw199502/LSTM_EGO` and the video is shown at `https://youtu.be/Eiyp8V8EjWo`.

II. RELATED WORK

A. Robot Simulators

In order to obtain a significant number of samples for DRL training in an efficient and safe manner, it is essential to use simulators that incorporate dynamic models, rich scenarios, and sensors. Several simulators such as MuJoCo [35], its extensions such as DMC [36] and Robomimic [37], are available that can significantly accelerate the sampling process. However, generating ego-centric sensor observations using these simulators is a challenging task. On the other hand, simulators like Gazebo [33] and V-REP [34] are powerful physics-based simulators that integrate various sensors like LiDAR and RGB-D camera. However, these simulators are not able to significantly speed up the running process, which may result in several days of DRL training. In comparison, Isaac Gym [38] and iGibson [39] can quickly generate image and LiDAR observations and support parallel computations. However, the drawback of these simulators is that they require powerful hardware resources, making them limited to small mobile robots equipped with compact and onboard computers. In our study, we developed a specialized simulator that can efficiently generate LiDAR observations. Furthermore, this simulator has the capability to customize the quantity, shapes, and movements of obstacles.

B. Robot Navigation in Stable Environments

Map-based navigation techniques have found widespread application in various industrial scenarios, such as robotics logistics in hotels [40] and restaurants [41]. These methods plan robot motion using global maps, which can be time-consuming to maintain, particularly in consistently changing environments. In contrast, DRL-based navigation strategies can be map-free, as they are end-to-end approaches that directly map sensor observations to robot actions [9]–[12], [14]–[17]. However, they face challenges in terms of generalization and sim-to-real transfer. Moreover, both map-based non-learning and map-free DRL-based motion planners are sensitive to dynamic obstacles, which limits their usefulness in environments that are rich in human activity. Our work aims to address this limitation, particularly in scenarios that are densely populated with pedestrians, by developing approaches that can handle highly dynamic environments.

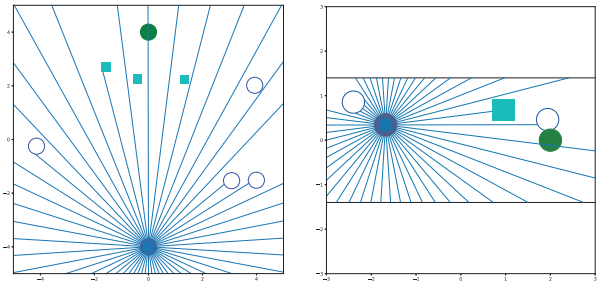
C. Robot Navigation in Dynamic Environments

One approach to navigate robots in dynamic environments is by estimating the current states of surrounding moving obstacles, including their position, shape, size, and speed, and predicting their future trajectories. This intuitive method enables the online optimization of a collision-free robot trajectory using model-based motion planners [42]. However, predicting trajectories is a complex task as it involves object tracking and intricate interactions between moving objects [43]. Additionally, modeling complex interactions is challenging, and the online optimization approaches that rely on complex models, such as model predictive control (MPC)-based motion planners [45], are time-consuming. Recently, imitation learning [44] is able to reproduce expert driving experience, yet a prohibitively large dataset is required, the navigation policy may be sub-optimal, and its generalizability is a challenge.

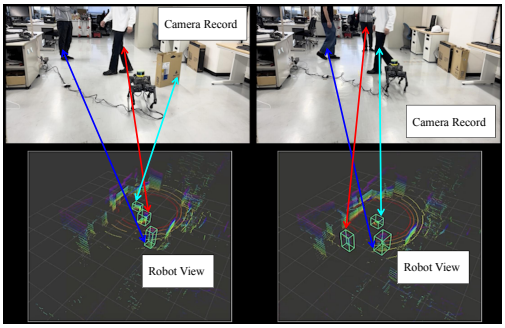
DRL-based motion planners in dynamic environments are becoming increasingly popular due to their ability to optimize navigation policies offline, eliminating the need for trajectory prediction through the sequential decision-making mechanism of reinforcement learning [46]. In fully observable environments, where the number, shape, size, and speed of obstacles are completely known, DRL-based navigation strategies can directly map observations to robot actions without relying on dynamic models or explicit trajectory prediction. The collision avoidance DRL (CADRL) algorithm [18] was a pioneering study that paired the robot with each moving object and defined a value function estimating the value of each individual pair. However, this pairing strategy neglected interactions among surrounding moving objects, which prompted further research to focus on including social interactions among dynamic agents. For instance, recurrent neural networks such as LSTM networks [20] were used to accumulate motion information from all pairs, which were ordered by the distance between the robot and each obstacle and then fed into LSTM networks. The output of LSTM networks is further set as a latent state vector for DRL-based value networks, named LSTM_RL. Recently, attention mechanisms [21] and graph convolutional networks (GCNs) [22] have been widely applied to capture social relationships among pedestrians. These two algorithms are

named socially aware (SA)RL and relational graph learning (RGL) respectively. Despite the success of these approaches in simulations, where fully observable states can be obtained quickly and easily, deploying them in the real world is challenging due to the difficulty in accurately estimating these states.

Consequently, employing DRL to directly map continuous raw sensor observations into robot actions is a highly effective alternative. Surrounding motion features can be derived from consecutive LiDAR scans [28], [29], [47], sequential images [32], [48], or multi-sensor observations [27]. In hybrid environments where dynamic humans and static obstacles such as walls and boxes coexist, a feasible strategy is to combine fully known human states and partially observable raw sensor data [49] as observations. Unfortunately, there are few open-source solutions available. Furthermore, complex neural networks are required to be embedded into policy networks to process high-dimensional observations, which necessitates physical robots to be equipped with a powerful computer, thereby limiting their deployment on small mobile robots with limited hardware resources. Moreover, simulated sensor observations cannot adequately replicate real-world scenarios, posing a challenge in sim-to-real transfer. Our study seeks to address these challenges by developing a lightweight network structure that processes sequential LiDAR observations. Additionally, we normalize real-world obstacles to match the simulated settings, allowing for successful sim-to-real transfer.



(a) Open Scenario without Margins (b) Narrow Scenario with Margins



(c) Real Environments

Fig. 1. Simulation and real environments. We represent dynamic humans as unfilled circles and static obstacles as rectangles. The green filled circle represents the goal and the LiDAR beams are emitted from the robot center. (a) illustrates an open scenarios without margins while (b) depicts a narrow space with margins. The real environments are shown in (c).

III. APPROACH

We begin by introducing a customized simulator that can generate LiDAR observations, incorporating a wide range of moving obstacles that vary in shape and size. We then outline a method for normalizing real-world obstacles to ensure consistency with the simulator's settings. Finally, we present an LSTM-DRL navigation algorithm that can map sequential LiDAR observations, taken from the robot's egocentric perspective, to appropriate robot actions.

A. Simulator

We aim to develop an end-to-end navigation algorithm that can directly translate raw LiDAR observations into robot motions. However, to achieve this objective, an efficient simulator is essential for generating training samples quickly. Unfortunately, the currently available simulators have some drawbacks; they either lack ego-centric sensors, are not efficient enough to accelerate running, or rely on expensive hardware. Therefore, we have designed a specialized simulator (shown in Fig. 1-a and b) that can swiftly generate LiDAR scans with minimal calculations. In our simulator, the robot emits 1800D beams to detect surrounding obstacles with shapes that can be circles, rectangles, and straight margins. We assume that the motions of dynamic obstacles are generated by the widely used optimal reciprocal collision avoidance (ORCA) rule [50], which simulates collision-free motions for multi-agents. To maintain the ORCA assumption, dynamic circle agents visualize static rectangle obstacles as bounding circles. In contrast, the robot perceives its surroundings according to their original outlines.

To generate each beam, we use a looping traversing algorithm. Initially, we calculate all the intersections between the beam and all the obstacles in the environment. Each beam is considered as a straight line, and we determine if it intersects with any circular obstacle. We disassemble each rectangle obstacle into four straight lines, and then calculate the possible intersections between the beam and all the straight lines of the rectangle. Finally, we select the intersection point that is nearest to the robot and represent the beam up to that point. To speed up the calculations, we use the C programming language. We test the computation time using CPU i7-10750H. Quantitatively, when there are 4 circles, 3 rectangles, and 4 margins in the environment, the calculation time is 5.8ms. Comparatively, when there is only one circle and one rectangle, the calculation time is reduced to 4.8ms. Furthermore, it only takes 3.7ms even the human number is up to 25, which indicates that our simulator is applicable for dense crowds. Because it is more complex to calculate the intersection point between a beam and a polygon than a circle, it will take longer calculation time with more rectangles.

B. Obstacle Normalization

The studies which directly utilize raw real-world sensor observations [28], [29] have the challenge of sim-to-real transfer because simulated configurations can not represent

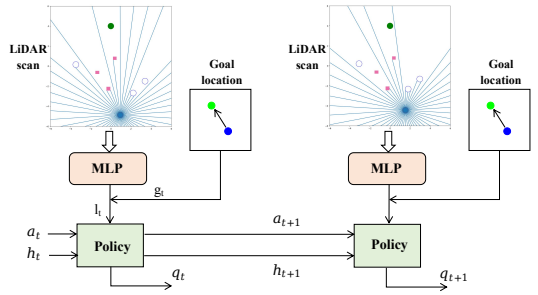


Fig. 2. Structure of LNDNL. We only require a multi-layer perceptron (MLP) to pre-process a single-frame LiDAR scan. The inputs of our policy networks include history information h_t, a processed LiDAR scan l_t, and the goal position g_t in the robot frame. The action-value function q_t is represented by a MLP with the inputs of h_t, l_t, g_t, and a_t.

all real-world scenarios. To reduce the sim-to-real gap, it is necessary to normalize real-world obstacles in a manner consistent with simulated ones, given that our simulator assumes obstacles as circles or rectangles. In this study, we leverage an adaptive clustering technique [51] to individually extract obstacles from the 3D LiDAR pointclouds captured in the real world as shown in 1-c. To achieve this, we first remove unnecessary 3D points in the sensor frame, such as the lower points, which are considered as ground, and the upper points, which are assumed to be the ceiling. We also eliminate points that are far from the sensor. Since our simulator generates beams in the world frame, we need to use simultaneous localization and mapping (SLAM) algorithms [52] to locate the robot. The remaining points are then used for clustering. We frame each cluster and normalize it as a circle or rectangle in the world frame as illustrated in Fig. 1-b. Subsequently, we re-generate LiDAR scans to reduce sim-to-real discrepancies. Note that we need additional identification techniques to differentiate circle and rectangle obstacles in the real world.

C. Algorithm Framework

To capture dynamic motion features, DRL-based policy networks generally utilize continuous raw sensor observations as input [28], [29]. However, this requires colossal CNNs to decode high-dimensional observations, especially when collected over a long time horizon [29]. Additionally, if the time horizon is short [28], extracting long-term motion features becomes challenging. To address these challenges, we introduce LSTM as a lightweight method to accumulate and propagate historical motion features (as illustrated in Fig. 2).

We formulate the robot navigation in dynamic environments using raw sensor observations as a partially observable Markov decision process (POMDP) represented by a tuple $(S, A, T, R, O, \Omega, \gamma)$. S is a set of states including history motion features, ego-centric goal position, and pre-processed observations. A denotes a set of actions composed by two orthogonal velocities of omnidirectional mobile robots. T represents a set of conditional transition probabilities between states. R stands for a reward function. O is a set of raw observations from sensors. Ω is a set of conditional observation probabilities. $\gamma \in [0, 1)$ represents a discount factor.

Network structure. At time t, we obtain the observation $o_t \in O$ from the LiDAR sensor. Subsequently, the raw high-dimensional observation is pre-processed by a MLP to yield a low-dimensional latent vector l_t. The state $s_t \in S$ includes three parts: history motion features h_t, goal position in the robot frame g_t, and l_t. We utilize an actor network to project s_t into next action $a_{t+1} \in A$. In addition, we map s_t and $a_t \in A$ into an action value function q_t by a critic network. The overall networks are represented as follows:

$$\begin{aligned} l_t &= f_\theta(o_t), \\ s_t &= \{l_t, h_t, g_t\}, \\ q_t &= f_\psi(s_t, a_t), \\ a_{t+1} &= f_\phi(s_t), \\ h_t &= f_\varphi(h_{t-1}, l_t, g_t). \end{aligned} \tag{1}$$

We leverage the twin delayed deep deterministic policy gradient (TD3) algorithm [53] to optimize the weights $\Phi = \langle \theta, \psi, \phi, \varphi \rangle$ in Eq. (1) simultaneously.

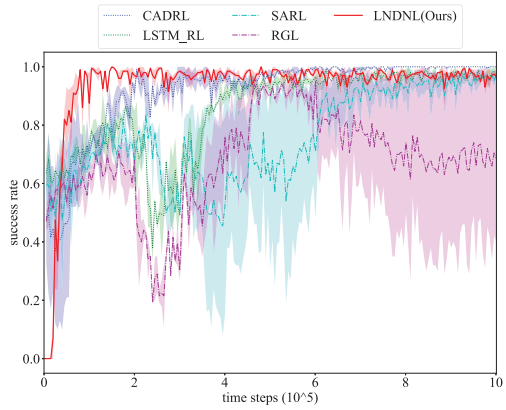
Action, observation, and reward function. The action space comprises two orthogonal velocities of omnidirectional mobile robots. We set their bounds as 1.0m/s in simulation. In addition, the velocities are continuous instead of discrete values in state-of-the-art baselines [18], [21], [22], [54]. The observation at each time step is a 1800D ego-centric LiDAR scan and each beam length ranges from $b_{\min}$ to $b_{\max}$. The navigation goal is to avoid obstacles and reach a target. Therefore, we define a reward function as follows:

$$r_t = \begin{cases} -r_c & \text{if } d_s < d_c, \\ 1.0 & \text{else if } g_t < g_r, \\ \omega_c \cdot (d_s - r_r - d_u) & \text{else if } d_s < d_u, \\ \omega_g \cdot (g_{t-1} - g_t) & \text{else.} \end{cases} \tag{2}$$

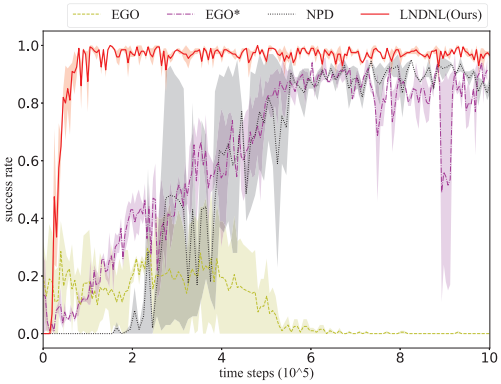
r_c is a positive constant less than 1. d_s denotes the length of the shortest one of all 1800D LiDAR beams. d_c defines a collision area. g_t represents the distance between the robot and the goal at time t. g_r is a constant that defines a goal-reaching area. r_r stands for the robot radius. d_u defines an uncomfortable area where the robot is close to obstacles. ω_c and ω_g weighs the discomfort and goal approaching, respectively.

IV. Experiments

On the one hand, we conducted two sets of simulation training for comprehensive comparison with state-of-the-art navigation methods. The first group, named Sim1, consisted of 9 baselines executed in a relatively simple scenario that involved only dynamic circle humans with a fixed number. The second group, named Sim2, consisted of 2 baselines that were tested in a more challenging situation that mixed dynamic circle humans with static rectangle obstacles whose number was changeable. On the other hand, we individually trained a model for our physical robot platform to validate



(a) Fully Observable States



(b) Partially Observable States

Fig. 3. Extensive simulation ablations with respect of learning efficiency. (a) The baselines, CADRL, LSTM_RL, SARL, and RGL assume fully observable states, including human number, size, shape, position, and velocity. (b) The states in the baselines, EGO, EGO*, and NPD are partially observable. In addition, EGO and EGO* directly use continuous LiDAR scans whereas NPD leverages sequential occupation maps.

the effectiveness of our navigation system, including SLAM algorithm, clustering method, LiDAR normalization, and our policy network.

A. Simulation Ablation

To conduct a comprehensive comparison with the state-of-the-art baselines described in the related work section, which includes 2 non-learning traditional navigation strategies – MB method [4] which assumes static obstacles and revised DWA [6] that relies on dynamic information, and 4 DRL-based policies – CADRL [18], LSTM_RL [20], SARL [21], and RGL [22], we designed a simple scenario that includes five moving humans in a 10×10m area, named Sim1. These DRL-based approaches assume that humans are circular in shape, their number remains constant during training, and their states, such as velocities, are fully observable. We assume that the maximum velocity of all agents is 1m/s to be consistent with the original settings of baselines. Furthermore, the robot is assumed to be invisible to humans to guarantee that our navigation policy plays the role of collision avoidance instead of that humans attempt to avoid the robot. Although open-source implementations of these

TABLE I

FINAL EVALUATION. 500 RANDOM TESTS ARE EXECUTED WITH THE BEST NEURAL NETWORKS SAVED DURING THE TRAINING.

Method	SR	NT	AT	NV
MB	0.42	15.85	**5.0e-4**	–
DWA[1]	0.95	11.94	2.6e-3	–
CADRL[1, 2]	0.80	12.40	0.035	**0.27**
LSTM_RL[1, 2]	0.97	10.95	0.067	0.87
SARL[1, 2]	0.98	10.75	0.060	1.0
RGL[1, 2]	0.97	11.22	0.067	0.38
EGO[2]	0.54	12.67	1.5e-3	1.9
EGO*[2]	0.94	11.20	1.5e-3	1.9
NPD	**0.99**	11.15	3.3e-3	106.3
LNDNL(ours)	**0.99**	**9.28**	**7.5e-4**	26.3

[1]Requiring fully known human information.
[2]Requiring imitation learning and positive samples.
SR: success rate; NT: navigation time (s)
AT: action time (s); NV: network variables (10^5)

baselines are available, there are few open-source projects that utilize raw sensor observations. To address this, we developed another baseline, called EGO, which was inspired by studies that map continuous LiDAR scans to robot movements [28], [29]. Additionally, we compared an extension of EGO, namely EGO* which decouples the robot movement and individually extracts the motion features of surrounding objects. Moreover, we compared another Navigation strategy in Pedestrian environments with a Dreamer-based (NPD) motion planner [55]. NPD firstly projects the robot and humans onto an occupation map, and then optimizes navigation policy from sequential occupation maps through a model-based DRL algorithm. We present the training process in Fig. 3 and provide a quantitative analysis in TABLE I.

The results presented in Fig. 3 show that our method exhibits significantly better learning efficiency than EGO, EGO*, and NPD, and is slightly superior to CADRL, LSTM_RL, SARL, and RGL. Notably, our method and NPD are not reliant on imitation learning or supervised samples, while other DRL-based approaches rely heavily on these prior experiences. In TABLE I, we demonstrate that our method achieves the highest success rate and shortest average navigation time among 500 random evaluations. We also introduce a new factor, action time, to measure real-time performance. Action time refers to how long it takes for the policy network to generate an action when provided with observations. CADRL, LSTM_RL, SARL, and RGL require tens of milliseconds of action time because they only have a value network and must inquire each state by repeatedly calculating the value network. However, the advantage of these four baselines is that their network variables are notably fewer due to their simpler observations. Despite our network being comparatively larger, it is still more lightweight than NPD. Additionally, the action time of our network outperforms EGO and EGO*, even though they have smaller networks. The advantages of a short action time and an acceptable network scale enable sim-to-real

TABLE II

VALIDATION OF MODEL GENERALIZATION ABILITY. 500 RANDOM TESTS ARE EXECUTED FOR EACH CASE.

Human number	5	6	7	8	9	10	10	10	15	5	6	7
Obstacle number	0	0	0	0	0	0	0	0	0	3	2	1
Distribution radius(m)	4	4	4	4	4	4	4	8	8	4	4	4
Maximum NT (s)	20	20	20	20	20	20	30	30	30	20	20	20
Success rate	1.0	0.988	0.958	0.940	0.932	0.852	0.948	0.982	0.960	0.972	0.960	0.970
Collision rate	0.0	0.008	0.032	0.024	0.020	0.046	0.046	0.012	0.030	0.024	0.032	0.020
Overtime rate	0.0	0.004	0.010	0.036	0.048	0.102	0.006	0.006	0.010	0.004	0.008	0.010
Average NT (s)	10.26	10.59	10.98	11.45	11.82	11.99	12.91	18.66	19.22	11.38	11.31	11.28

transfer on compact robots with limited hardware resources. Interestingly, the non-learning map-based method is sensitive to highly dynamic environments because it regards all objects as static obstacles. In the meanwhile, DWA's performance is inferior than ours even it assumes fully known environments. The advantages with respect of traditional methods indicate the potential of applying our end-to-end navigation strategy into highly dynamic environments.

B. Generalizability Validation

To test the generalization capability of our method, Sim2 presents more complex environments than Sim1. The motion area remains the same, but we increase the maximum obstacle number to seven (shown in Fig. 1-a), which includes a combination of dynamic human circles (with a maximum of four) and static obstacles (with a maximum of three), each with a side length ranging from 0.3 to 0.4m. Existing methods such as CADRL, LSTM_RL, SARL, and RGL assume that obstacles are fixed circles with a predetermined number, which hinders their performance. EGO, in particular, demonstrates notably inferior results. Thus, we chose EGO* and NPD as our baselines for Sim2. Our method achieved a feasible navigation policy within 100K steps, while the baselines required over 200K steps. Moreover, our method's success rate remained stable around 0.99, while the success rates of the baselines fluctuated between 0.8 to 0.95, indicating our method's superior generalizability to complex environments.

Additionally, our model trained in Sim2 can be generalized into other scenarios which have not been explored. More specifically, we first train our model in the Sim2 scenario, where the human number changes from 1 to 4 and the obstacle number varies from 1 to 3. We then test our model in the scenario where only humans exist in a circular motion area with a radius of 4m. The human number ranges from 5 to 10. Moreover, we also insert static obstacles to increase complexity. By default, we set the maximum navigation time as 20 seconds. Next, we distribute 10 and 15 humans in a larger motion area with the radius of 8m and the maximum navigation time is prolonged to 30 seconds. The result is shown in TABLE II. Although we train our model in Sim2, where the human number changes from 1 to 4 and the obstacle number ranges from 1 to 3, we are able to generalize our model into more complex scenarios. The human number can reach 10 in a small motion area. When we enlarge the motion area and maximum navigation time, the human number can be increased up to 15. Please note that it is still difficult to verify our model can be generalized into all kinds of scenarios. Therefore, we only partially indicate the generalization ability in TABLE II.

C. Real Implementation

We simplified the environment illustrated in Sim2 and individually trained a model for our physical robot platform to validate the effectiveness of our navigation system, including SLAM algorithm, clustering method, LiDAR normalization, and our policy network. More specifically, the motion area is narrowed to 6.0×2.8m, and a maximum of three obstacles are present because of the constraints of our physical robot platform. The sim-to-real transfer is shown in the attached videos, which indicate the potential of sim-to-real transfer with our end-to-end navigation algorithm.

V. DISCUSSION

Conclusion. This study presents an end-to-end navigation strategy that maps sequential LiDAR observations into robot actions through LSTM-DRL. To expedite training, we developed a simulator that can swiftly produce LiDAR scans and configure different types of obstacles that vary in number, shape, and size. We also standardized the obstacles in the real world to ensure consistency with simulated settings, facilitating sim-to-real transfer. This was accomplished using an adaptive clustering technique and SLAM algorithm to locate and frame obstacles from 3D LiDAR scans. Our method demonstrated superior learning efficiency, navigation time and success rate, and real-time performance through extensive simulation ablations. Moreover, sim-to-real transfer highlighted the potential of our approach to guide robots through complex real-world scenarios involving dynamic and static obstacles with variable shapes, sizes, and movements.

Limitations. Although our approach outperformed state-of-the-art baselines in various environments, our simulation scenarios require more improvements, such as considering grouped objects. In the real world, humans might be grouped with objects such as other humans, suitcases, and baby strollers. Moreover, the real-world implementations are limited in laboratory environments. Our future study will focus on more broad applications in our human societies. In addition, we will take sensor noise into consideration in simulation to be consistent with physical sensors.

REFERENCES

[1] G. A. Zachiotis, G. Andrikopoulos, R. Gomez, et al., "A survey on the application trends of home service robotics," in *IEEE international conference on Robotics and Biomimetics (ROBIO)*, 2018, pp. 1999-2006.

[2] J. Cheng, H. Cheng, Q. H. Meng, and H. Zhang, "Autonomous navigation by mobile robots in human environments: A survey," in *IEEE international conference on robotics and biomimetics (ROBIO)*, 2018, pp. 1981-1986.

[3] F. Rubio, F. Valero, and C. Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[4] C Wang, L Wang, J Qin, et al., "Path planning of automated guided vehicles based on improved A-Star algorithm," in *IEEE International Conference on Information and Automation*, 2015, pp. 2071-2076.

[5] I. Noreen, A. Khan, and Z. Habib, "Optimal path planning using RRT* based approaches: A survey and future directions," *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 11, 2016.

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

[7] J. Minguez and L. Montano, "Nearness diagram navigation (ND): A new real time collision avoidance approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2000, pp. 2094-2100.

[8] L. Dong, Z. He, C. Song, and C, Sun, "A review of mobile robot motion planning methods: from classical motion planning workflows to reinforcement learning-based architectures," *arXiv preprint arXiv:2108.13619*, 2021.

[9] Y. Zhu, R. Mottaghi, E. Kolve, et al., "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357-3364.

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

[11] M. Pfeiffer, S. Shukla, M. Turchetta, 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.

[12] 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.

[13] E. Wijmans, et al., "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," *arXiv preprint arXiv:1911.00357*, 2019.

[14] 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.

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

[16] 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.

[17] 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.

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

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

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

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

[22] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational graph learning for crowd navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10007-10013.

[23] 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.

[24] A. Francis, et al., "Long-range indoor navigation with PRM-RL," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1115-1134, 2020.

[25] X. Xiao, et al., "APPL: Adaptive planner parameter learning," *Robotics and Autonomous Systems*, vol. 154, no. 104132, 2022.

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

[27] 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 *International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5993-6000.

[28] T. Fan, P. Long, W. Liu, and Pan J, "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.

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

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

[31] W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework with high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962-4971, 2022.

[32] N. Yokoyama, Q. Luo, D. Batra, and S. Ha, "Benchmarking augmentation methods for learning robust navigation agents: the winning entry of the 2021 iGibson challenge," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 1748-1755.

[33] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2149-2154.

[34] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A versatile and scalable robot simulation framework," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1321-1326.

[35] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012, pp. 5026-5033.

[36] Y. Tassa, Y. Doron, A. Muldal, et al., "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[37] A. Mandlekar, D. Xu, J. Wong, et al., "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[38] V. Makoviychuk, L. Wawrzyniak, Y. Guo, et al., "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv preprint arXiv:2108.10470*, 2021.

[39] B. Shen, F. Xia, C. Li, et al., "iGibson 1.0: A simulation environment for interactive tasks in large realistic scenes," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7520-7527.

[40] J. Lopez, D. Perez, E. Zalama, and J. Bermejo, "Bellbot - a hotel assistant system using mobile robots," *International Journal of Advanced Robotic Systems*, vol. 10, no. 1, 2013.

[41] C. S. Chen, C. J. Lin, and C. C. Lai, "Non-contact service robot development in fast-food restaurants," *IEEE Access*, vol. 10, pp. 31466-31479, 2022.

[42] Y. Chen, F. Zhao, and Y. Lou, "Interactive model predictive control

for robot navigation in dense crowds," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 4, pp. 2289-2301, 2022.
[43] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5921-5928.
[44] M. Bansal, A. Krizhevsky, and A. Ogale, "Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst," *arXiv preprint arXiv:1812.03079*, 2018,
[45] A. S. Lafmejani and S. Berman, "Nonlinear MPC for collision-free and deadlock-free navigation of multiple nonholonomic mobile robots," *Robotics and Autonomous Systems*, vol. 141, no. 103774, 2021.
[46] R. S. Sutton and G. B. Andrew, "Reinforcement learning: An introduction," *MIT press*, 2018.
[47] H Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end sith AutoRL," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007-2014, 2019.'
[48] 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.
[49] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dube, "Robot navigation in crowded environments using deep reinforcement Learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 5671-5677.
[50] J. Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research*, Springer, Berlin, Heidelberg, 2011, pp. 3-19.
[51] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for 3D LiDAR-based human detection: experimental analysis of point cloud clustering and classification methods," *Autonomous Robots*, vol. 44, pp. 147-164, 2020.
[52] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems (RSS)*, 2014.
[53] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (PMLR)*, 2018.
[54] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10357-10377, 2021.
[55] W. Zhu and M. Hayashibe, "Autonomous navigation system in pedestrian scenarios using a Dreamer-based motion planner," *IEEE Robotics and Automation Letters*, vol. 8, no. 6, pp. 3836-3843, 2023.