博 士 学 位 論 文
Doctoral Thesis

論文題目

Thesis Title

Prior Knowledge-free Robot Navigation

in Dynamic Environments

through Deep Reinforcement Learning

東 北 大 学 大 学 院 工 学 研 究 科

Graduate School of Engineering,

TOHOKU UNIVERSITY

専攻/Department: Robotics

学籍番号/ ID No: C0TD9122

氏 名 /Name: Wei ZHU

TOHOKU UNIVERSITY

Graduate School of Engineering

Prior Knowledge-free Robot Navigation in Dynamic Environments through Deep Reinforcement Learning

(深層強化学習を用いた動的環境下における事前知識不要なロボットナビゲーションに関する研究)

A dissertation submitted for the degree of
Doctor of Philosophy (Engineering)

Department of Robotics

by

Wei ZHU

July 7, 2023

Prior Knowledge-free Robot Navigation in Dynamic Environments through Deep Reinforcement Learning

Wei Zhu

Abstract

The advancement in robotics and autonomous systems has led to an increase in the demand for intelligent agents that can navigate through dynamic environments safely and efficiently. However, developing a collision-free navigation system for autonomous agents that can operate in dynamic environments is a challenging task. The system needs to be able to perceive its environment, make decisions based on sensory input and history of actions taken, and adjust its behavior in response to changes in the environment. Traditional methods for collision avoidance rely on global maps, fully known obstacle states, and accurate state predictions, which may not be practical for dynamic environments. Deep reinforcement learning (DRL) has emerged as a promising solution to address this challenge. DRL combines deep neural networks with reinforcement learning to learn collision-free policies by interacting with the environment. The system learns from its experience by receiving feedback in the form of rewards or penalties and updates its policy accordingly. However, it is time-consuming to learn from exploratory interactions, especially when observations are high-dimensional raw sensor data, definitions of rewards or penalties are sparse, and navigation targets are positioned in a broad scope. In addition, complex observations may require colossal neural networks, which requires powerful hardware and possibly results in prolonged even unsuccessful training. Because of the limited training samples collected from simulation environments, it is prohibitively challenging to transfer pre-trained navigation policies into other environments which have not been explored. Moreover, the ideal assumptions in simulation, such as fully known obstacle states including shape, size, number, position, and speed, pose challenges for real-world implementation.

The final objective of this thesis is to enhance the collision avoidance capability of real-world mobile robots in dynamic environments using learning efficient DRL frameworks without the ideal assumptions or heavy neural networks. To progressively accomplish this objective, we developed four DRL models step by step to learn collision-free navigation policies in different scenarios. In addition, we employed conventional controllers, domain randomization, system identification, and localization algorithms to cater to practical applications. To evaluate the proposed approaches, a set of experiments was conducted on a variety of simulated environments and real scenarios with dynamic obstacles. The experiments demonstrate that the proposed approaches outperform traditional methods in terms of collision avoidance and navigation efficiency. Moreover, the proposed navigation systems are shown to adapt effectively to changes in the environment, making it suitable for practical applica-

tions. More specifically, the dissertation comprises four main bodies that are ordered based on the complexity of the task at hand.

First, we present a Hierarchical DRL-based (HDRL) navigation structure which is able to learn to avoid obstacles from a single-frame LiDAR scan and can adapt to unknown static environment configurations in Chapter 2. Our hierarchical framework has prominent sampling efficiency and sim-to-real transfer ability for fast and safe navigation. Specifically, the low-level DRL policy enables the robot to move toward the target position and keep a safe distance to obstacles simultaneously. The high-level DRL policy is supplemented to further enhance the navigation safety. For both policies, we select a waypoint located on a traversable path from the robot to the ultimate goal as the sub-goal to reduce the state space and avoid sparse reward. Moreover, the path is generated based on a local map and existing map-based path planners, which can significantly improve the sampling efficiency, safety, and generalization ability of the proposed DRL framework. Additionally, a target-directed representation for the action space can be derived based on the sub-goal to improve the motion efficiency and reduce the action space. Finally, we deploy the navigation strategy on a wheel-legged biped robot and a quadruped robot to show the potential of our method to navigate robots in unknown real environments.

In Chapter 3, the task becomes more challenging by mixing static and dynamic obstacles. We present a Sampling Efficient DRL framework for Dynamic Navigation (SEDN) directly using multi-frame LiDAR scans. To accelerate DRL training and simulate LiDAR scans, we specially designed 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 where we utilize a quadruped robot mounted with a LiDAR sensor. Moreover, because of the rich information from consecutive LiDAR scans and prominent representation capability of deep learning, 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 LiDAR scans into the center of the current LiDAR scan to individually extract surrounding motion features. To further enhance sampling efficiency, we integrate optimal reciprocal collision avoidance (ORCA) to generate auxiliary action alternatives. Our method has been extensively tested through numerous ablations and real-world implementations, which have unequivocally demonstrated its remarkable learning efficiency, superior generalization ability, and strong adaptability in real-world scenarios.

While our study in Chapter 3 presents a straightforward end-to-end navigation solution, its practical applicability in real-world situations is limited because the margins of obstacles in reality differ significantly from those used in our simulations. Moreover, the robot depends on external localization systems, which further restricts its practicality. Additionally, the research in Chapter 3 requires imitation learning and supervised samples, which may result in sub-optimal navigation policies. Therefore, to enable realistic and practical applications in human society, our third research in Chapter 4 integrates human detection and robot localization modules to form an autonomous navigation system. Furthermore, we have implemented a model-based and sampling-efficient DRL framework to develop a navigation policy solely from image observations, without relying on supervised samples. Specifically,

we create a collision-free Navigation system in diverse Pedestrian scenarios using a Dreamer-based motion planner (NPD). Our DRL framework can completely learn from zero experience via a model-based DRL with the advantage of efficient sampling. The robot and humans are first projected onto an occupancy map, which is subsequently decoded into a low-dimensional latent state. Moreover, we leverage recurrent neural networks to accumulate environment information from sequential occupancy maps. In addition, a predictive dynamic model in the latent space is jointly created to dream effective interaction episodes in the long-term horizon, which can efficiently optimize the navigation policy. Additionally, we leverage the techniques of system identification, domain randomization, clustering, and LiDAR SLAM for practical deployment. Our approach has been rigorously compared with state-of-the-art baselines, with results consistently showing superior performance. Furthermore, our extensive real-world experiments provide compelling evidence for the efficacy of our method in modeling complex, reciprocal human relations and successfully navigating robots among pedestrians

Finally, in Chapter 5, we further optimize the navigation strategy which offers several advantages that stem from the combination of the studies in Chapter 3 and 4. The robot can Learn to Navigate in Dynamic environments using Normalized LiDAR (LNDNL) scans. Specifically, the DRL-based navigation model enables fast and stable learning, lightweight computation, and strong adaptability. Instead of relying on consecutive laser scans, we leverage long short-term memory (LSTM) to extract surrounding motion features from egocentric and sequential LiDAR scans. Because we replace the image observation utilized in Chapter 4, our neural networks in Chapter 5 become significantly compact and lightweight, which can notably reduce training time and usage of hardware resources. Additionally, we employ a model-free DRL framework to handle deviated human motion models, rather than the previously used Dreamer algorithm in Chapter 4. Instead of directly using raw LiDAR scans in real-world scenarios mentioned in Chapter 3, we firstly obtain obstacle positions by an improved clustering algorithm derived from the study in Chapter 4. Accordingly, we are able to normalize the obstacles as circles and rectangles. Subsequently, we can re-generate LiDAR scans from the normalized obstacles, which enhances the generalizability of sim-to-real implementations. We have extensively evaluated our combined navigation strategy through simulation ablations and practical implementations, which demonstrate its notable improvement over the methods mentioned in Chapter 3 and 4.

The findings of this research contribute to the field of end-to-end robot navigation in dynamic environments by providing a robust and adaptive navigation system for autonomous agents. The proposed approach has the potential to be applied to various domains, including autonomous driving, robotics, and unmanned aerial vehicles. The proposed approach can also be extended to other related problems, such as multi-agent collision avoidance, which is an important topic in robotics and autonomous systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Autonomous navigation systems have gained significant attention and popularity recently, as they offer multiple advantages, including enhanced efficiency, safety, and intelligence. The development of autonomous navigation systems can be traced back to the 1980s when researchers began exploring the use of artificial intelligence and machine learning algorithms to enable autonomous robot navigation. Since then, there has been significant progress in the development of autonomous navigation systems, including the use of advanced sensors, including LiDAR, cameras, and radar, as well as the development of advanced algorithms that enable real-time decision making [1]. As shown in Figure 1.1, the promising potential of these systems have encouraged their widespread use in various sectors [2] such as indoors [3], open fields [4], busy roads [5], and high-speed ways [5]. The aging population, labor shortages, and the need for non-contact services during the pandemic have further fueled the research and development of autonomous mobile robots.

Although significant strides have been made in the advancement of autonomous navigation systems, there are still numerous obstacles that must be surmounted. One of the most pressing challenges is the ability of these systems to function effectively in complex and dynamic environments. Accomplishing this necessitates integrating a diverse array of sensors and creating sophisticated algorithms capable of managing the intricacies of real-world settings. Designing navigation systems for socially aware robots, in particular, is an exceptionally intricate undertaking. It involves addressing a wide variety of issues, including mapping and localization, human detection and behavior analysis, social norms, and decision-making and planning [6].

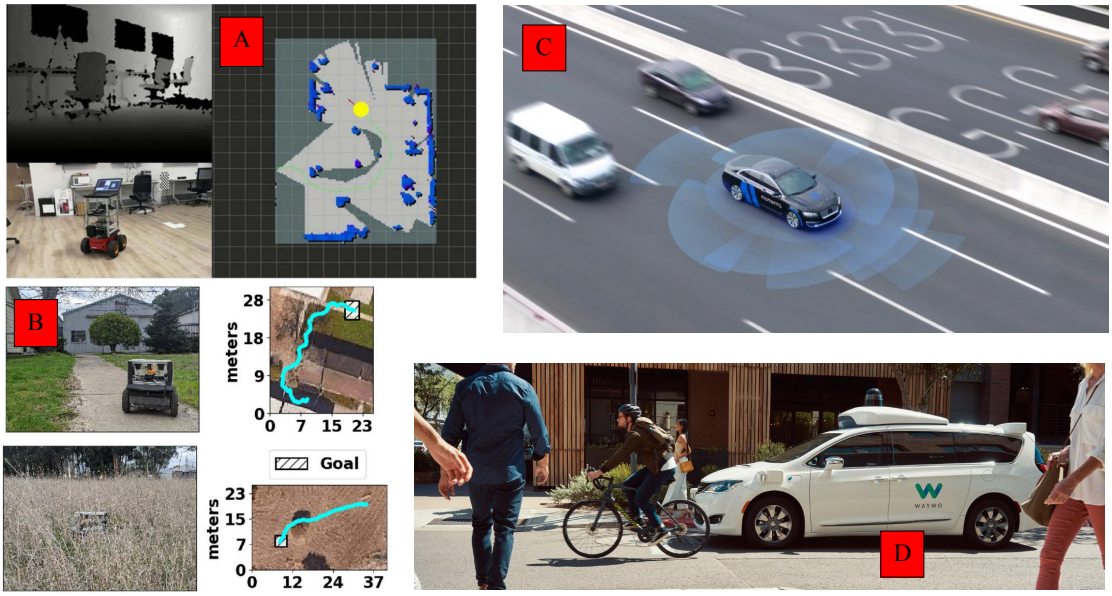


Figure 1.1: Practical application scenarios of mobile agents. A and B are authorized by IEEE [3, 4]. C and D are from Momenta and Waymo websites, respectively.

Various types of algorithms, such as map-based [7, 8], trajectory-based [9, 10], and DRL-based [11, 12], have been extensively developed to generate navigation strategies for both small service mobile robots and large unmanned vehicles in relatively stable or highly dynamic environments. These algorithms aim to ensure safe, efficient, and comfortable operation of these mobile agents in a range of environments, including human societies and the wilderness. This dissertation aims to explore the benefits of these navigation algorithms through a comprehensive analysis. By combining these algorithms, this study seeks to uncover their full potential and enhance their effectiveness.

1.2 Motivation and Objectives

Map-based [7, 8] robot navigation strategies enable autonomous collision-free and target-reaching motion planning in unstructured environments. By using maps to guide their movements (shown in Figure 1.2-A), robots can achieve higher levels of precision and accuracy in their operations. Moreover, map-based navigation systems can be scaled up or down to accommodate different unstructured environments, making them adaptable to a wide range of applications. However, map-based robot navigation systems rely heavily on the accuracy and completeness of maps, which may not always be available or up-to-date. In addition, map-based navigation systems may struggle in environments that are constantly changing or where unexpected obstacles arise.

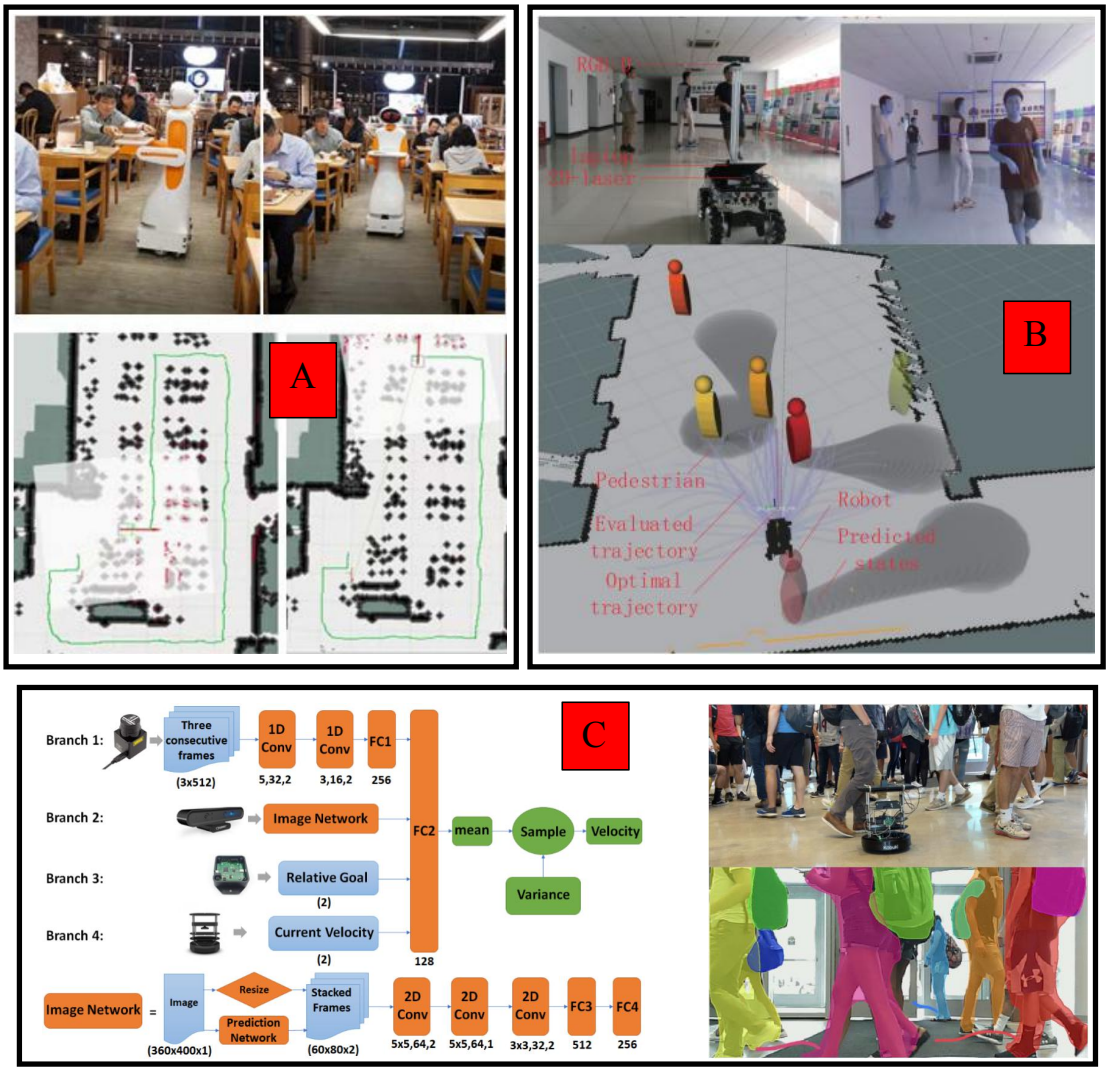


Figure 1.2: Types of navigation strategies. A, B, and C, authorized by IEEE [13–15], represent the map-based, trajectory-based, and DRL-based navigation strategies, respectively.

As an alternative, trajectory-based navigation strategies [9,10] can be map-free and quickly adapt to constantly changing obstacles. Trajectory planning takes into account the surrounding obstacles and plans the path accordingly (shown in Figure 1.2-B), which helps in avoiding collisions and ensuring safety. With a pre-planned trajectory, the robot's movement becomes more predictable, and this can be advantageous in scenarios where accuracy and repeatability are crucial. Moreover, trajectory-based navigation can be adapted to various types of robots and environments. Nevertheless, trajectory-based navigation relies heavily on accurate sensor data to create a precise path, and any errors or inaccuracies in the sensor data can lead to navigation failures. Additionally, if the estimation of the obstacle trajectory is incorrect, it could lead to a fatal crash and cause the robot's planned trajectory to deviate from its desired direction. Furthermore, detecting and tracking mov-

ing obstacles requires complicated techniques, which rely on powerful hardware and degrade real-time performance. At the meanwhile, online optimization of robot trajectory is time consuming, which further results in response delay. Fixed optimization methods may also cause frozen motions especially in the environments with dense moving objects. Moreover, short-sighted planning horizon used for trajectory optimization may bring about unnatural motions.

Unlike manually programmed robots, such as model predictive motion planners commonly used in trajectory-based navigation strategies, DRL-based navigation approaches [11,12] enable robots to learn and make decisions on their own in an end-to-end manner (shown in Figure 1.2-C), making them more flexible in responding to changes in the environment. Meanwhile, interactive learning mechanism enables robots to learn from their mistakes and improve their performance over time, making them more efficient at their tasks. With the significant development of big models such as ChatGPT that everlastingly improve their own generalizability and accuracy, we can envision that a generalized and practical DRL-based navigation model will be come into being. One significant advantage of DRL-based navigation methods is their model-free nature. This is particularly useful when dynamic models are highly complex and difficult to obtain with accuracy. However, DRL algorithms can sometimes overfit to a specific environment, which means that they may not perform well in new or unfamiliar environments. Moreover, the performance of robots trained with deep reinforcement learning heavily depends on the quality and quantity of data used during training.

Motivated by the advantages and disadvantages of the existing studies mentioned above, we developed four navigation algorithms to handle different scenarios. The objective of this thesis is to improve the ability of mobile robots to avoid collisions and reach targets using DRL. The goal is achieved by introducing different navigation frameworks in specific scenarios. To make the proposed frameworks more practical and applicable in real-world scenarios, several additional techniques are employed. These include conventional motion planners, domain randomization, system identification, and localization algorithms. These techniques are used to improve the performance of the DRL algorithms and to ensure that the robots are able to operate successfully in different environments and conditions. To be more precise and specific, we have outlined our detailed objectives as follows:

- Our motion planning method adopts a hierarchical DRL framework and leverages map-based navigation techniques. By integrating these approaches, we aim to enhance navigation performance in unknown environments. Specifically, our method aims to improve generalizability, learning efficiency, and

optimality, enabling it to adapt to a wide range of scenarios and improve navigation efficiency.

- In environments that are constantly changing, we are able to make navigation decisions based on continuous LiDAR scans without the need for maps or explicit dynamic models. To enhance the learning efficiency of our complex neural networks, we have integrated a trajectory-based navigation algorithm that aids in sampling during training. As a result, our approach allows for more effective mapping of LiDAR scans to navigation decisions, even in highly dynamic environments.

- To enhance the transferability of our approach from simulation to the real world, we represent the surrounding objects as circles or rectangles in a pixel image. Then, we employ a model-based DRL algorithm to optimize the navigation policy from sequential image observations, thereby improving the learning efficiency and freeing our approach from dependence on supervised samples.

- The networks responsible for image processing are intricate and their ability to plan effectively is compromised when dealing with large motion areas. Moreover, continuous LiDAR scans neglect long-term environment interactions. Our final objective is to develop a lightweight neural networks that can capture long-term surrounding variations and efficiently optimize navigation policies in highly dynamic environments without relying on maps, supervised samples, or dynamic models.

1.3 State of the Art

We have undertaken a comprehensive study of a navigation system that encompasses perception, prediction, decision-making, planning, and robot control. Our primary objective is to design navigation strategies based on DRL that are effective in dynamic environments. To achieve this, we have integrated traditional motion planners, cutting-edge machine learning techniques, and approaches for environment perception from raw sensor data. This enables us to not only learn viable navigation strategies but also implement them in real-world scenarios with maximum efficiency.

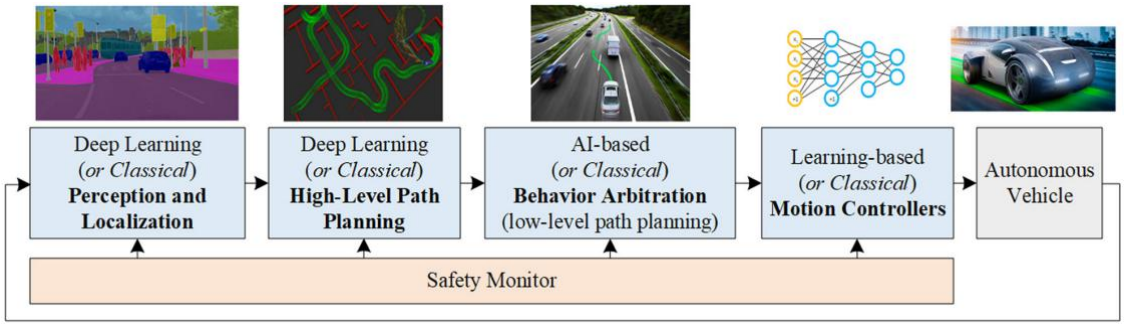


Figure 1.3: Structure of a complete autonomous navigation system. This figure is authorized by Journal of Field Robotics [16].

1.3.1 Navigation System

A complete navigation system shown in Figure 1.3 integrates perception, decision, planning, and control modules, which are broadly researched and developed for autonomous driving vehicles [16–18]. However, these modules are studied separately in both industry and academia. Although there are several mature and open-source solutions, such as Apollo[1] and Autoware[2], publicly accessible navigation systems among pedestrians are rare in the community of service robots. In this thesis, we construct a complete navigation system, in which the perception and control modules are derived from mature methods and decision and planning are achieved by novel DRL-based motion planners.

1.3.2 Map-based Navigation

Traditional non-learning map-based methods [19, 20] require a pre-built global map for path planning (shown in Figure 1.2-A), which can be cumbersome to maintain maps when the environment frequently changes. Moreover, if the environment is large the final goal is far from the robot, searching global paths will be prolonged. Once the global map is known, various path planning techniques can generate a global path from the initial robot position to the final goal, such as the A-star algorithm [21] and the rapidly-exploring random tree (RRT) method [22]. Alternatively, a local planner can be used to generate velocity commands and handle certain disturbances, such as moving, removing, or adding obstacles. One commonly used strategy, the dynamic window approach (DWA) and its variations are widely used for obstacle avoidance [23, 24]. However, global planners are unable to generate optimal paths because they search for a feasible path from the map using a heuristic

[1]https://github.com/chrislgarry/Apollo-11
[2]https://github.com/autowarefoundation/autoware

method. Additionally, local planners require careful tuning for cost weights involving collision, path following, motion speed, etc. For certain simple situations, path tracking methods can be used to follow the path without collision concerns, provided that the environment remains static and the global path avoids all obstacles [25,26]. The methods mentioned above assume that a global map is already known, which is impractical when the environment frequently changes. One alternative is to update the map in real-time while navigating in partially-known surroundings [27]. This approach solves the problem of time-consuming offline map building but comes at the expense of degraded safety and optimality of navigation.

1.3.3 Trajectory-based Navigation

Trajectory-based navigation strategies have the ability to produce collision-free and optimal trajectories based on current and future states of surrounding obstacles (shown in Figure 1.2-B), whether static or moving [28]. Two commonly used approaches that operate on a one-step trajectory planning basis are reciprocal velocity obstacle (RVO) [29] and optimal reciprocal collision avoidance (ORCA) [30], which can generate safe actions for each moving agent, based on a reciprocal assumption. However, since these methods only consider one-step motion, they tend to result in short-sighted and unnatural behaviors. On the other hand, multi-step trajectory planning methods [31–33] online optimize the robot's trajectory over a long horizon by predicting the trajectories of surrounding obstacles. However, these methods heavily rely on the accurate and complete knowledge of the states of surrounding objects and their future states, which is often difficult to obtain in real-world scenarios. In addition, online optimization is time-consuming, which may degrade real-time performance.

1.3.4 DRL-based Navigation

Deep reinforcement learning. Deep reinforcement learning (DRL) is a subfield of machine learning that combines deep learning and reinforcement learning techniques to enable machines to learn and make decisions in complex and dynamic environments. In recent years, there have been significant advancements in the field of DRL, leading to state-of-the-art performance on many challenging tasks. One of the major breakthroughs in DRL has been the development of the Deep Q-Network (DQN) algorithm [34]. DQN uses a deep neural network to approximate the Q-value function, which estimates the expected future reward for each possible action in a

given state. This algorithm has been successful in playing a variety of Atari games at a superhuman level. Our first two studies in Chapter 2 and 3 also leverage the DQN framework.

Another major advancement in DRL has been the development of policy gradient algorithms. These algorithms directly optimize the policy function, which maps states to actions, using gradient descent. One popular algorithm in this category is the deep deterministic policy gradient (DDPG) algorithm [35], which has been shown to achieve state-of-the-art performance on a range of complex environments, including robotics and game-playing tasks. The DRL frameworks used in our last two studies in Chapter 4 and 5 are founded on the actor-critic network structure initially introduced in the DDPG algorithm.

Furthermore, there has been a growing interest in DRL research on model-based methods, which learn a model of the environment and then use it to plan actions [36]. This approach can be more sample-efficient than model-free methods, which directly learn the optimal policy without explicitly modeling the environment. Model-based algorithms such as Model Predictive Control (MPC) and World Models have shown promising results in various domains. Our study in Chapter 4 leverages a model-based DRL approach with image observations, namely Dreamer. The Dreamer algorithm is a reinforcement learning agent that addresses long-horizon tasks from images purely by latent imagination [37], which yields a large number of achievements in simulated environments, such as Atari games and MuJoCo robots [37–40]. Conversely, we focus more on real implementations of collision-free and socially aware robot navigation by leveraging the key idea of the Dreamer algorithm. A map is created to represent complex scenarios with variable human numbers and random initial states. A dynamic model with a map as a unique observation is learned to represent social relationships among humans. The learned model can facilitate learning complex behaviors, thereby enabling the robot to learn an optimal navigation policy without any prior experience.

Navigation in stable environments. In stable environments, DRL-based navigation policies can generate collision-free and target-reaching trajectories in both simulations and real-world scenarios that resemble the training environment. For example, a pioneering study [41] trained a policy in simulation, where an abundance of visual data enabled optimization of the navigation strategy (shown in Figure 1.4-A). The resulting policy was then directly deployed in a similar real-world setting. Similarly, another state-of-the-art approach [42] successfully transferred a learned policy to novel scenarios by approximating the reward function using a linear combination of learned features. In addition to vision-based DRL navigation policies that are

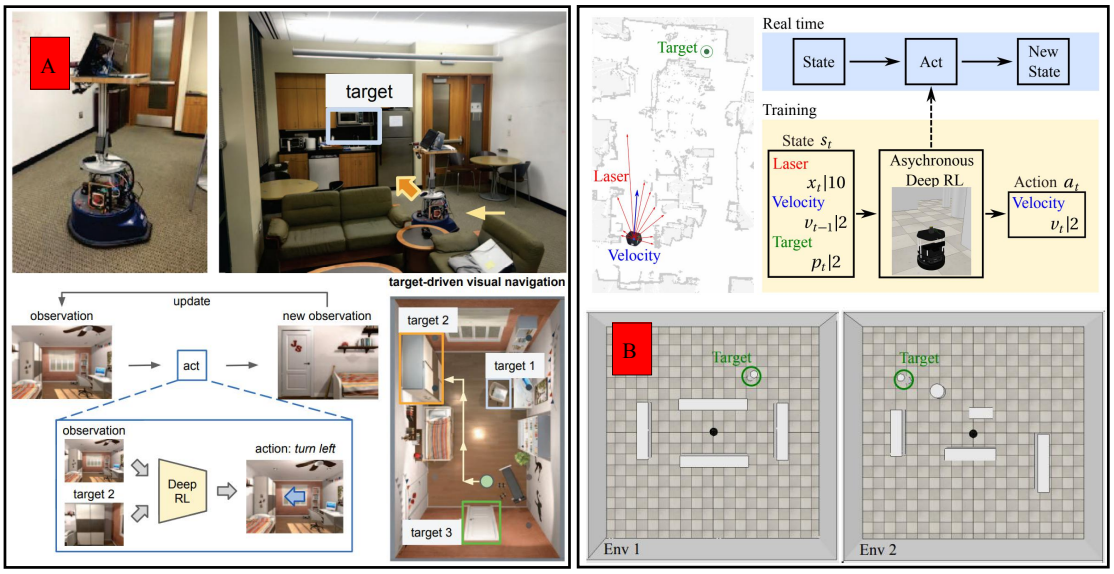


Figure 1.4: Navigation in stable environments. The observation of A is raw images whereas B's observation is LiDAR scans. A and B are authorized by IEEE [41, 43].

sensitive to light intensity, LiDAR sensors (shown in Figure 1.4-B) are also widely used to gather environmental information [43–45]. However, the complexity of environments poses a challenge for efficient sampling and learning, leading to a need for improved efficiency. To address this issue, some studies aim to improve sampling efficiency through demonstration learning [46] or model-based DRL [47]. Nevertheless, the generalization capability of model-based DRL and demonstration learning is limited due to its reliance on specific environments and supervised datasets. Furthermore, transferring DRL-based navigation frameworks from simulation to the real world remains a significant challenge because real-world uncertainties cannot be accurately represented in simulation.

Navigation in dynamic environments. With the rapid development of deep learning, researchers and engineers are focusing on learning-based methods, wherein DRL-based navigation algorithms are attractive because of their promising representation and optimization capabilities [48–60]. Collision avoidance DRL (CADRL) [48] is a pioneering study in the use of DRL for social navigation. However, its value function neglects the social relationships among pedestrians, as it only considers the robot's full state and one pedestrian's observable state. LSTM_RL [49] improves upon CADRL by leveraging LSTM to represent pairs of the robot's state and all pedestrians' states, but its ability to capture reciprocal relationships is limited since pairs are ordered by distance and then fed into LSTM networks (shown in Figure 1.5-A). Socially aware RL (SARL) [53] and relational graph learning (RGL) [54] represent state-of-the-art extensions to CADRL and LSTM_RL by using self-attention mechanisms and graph convolutional networks (shown in Figure 1.5-B), respec-

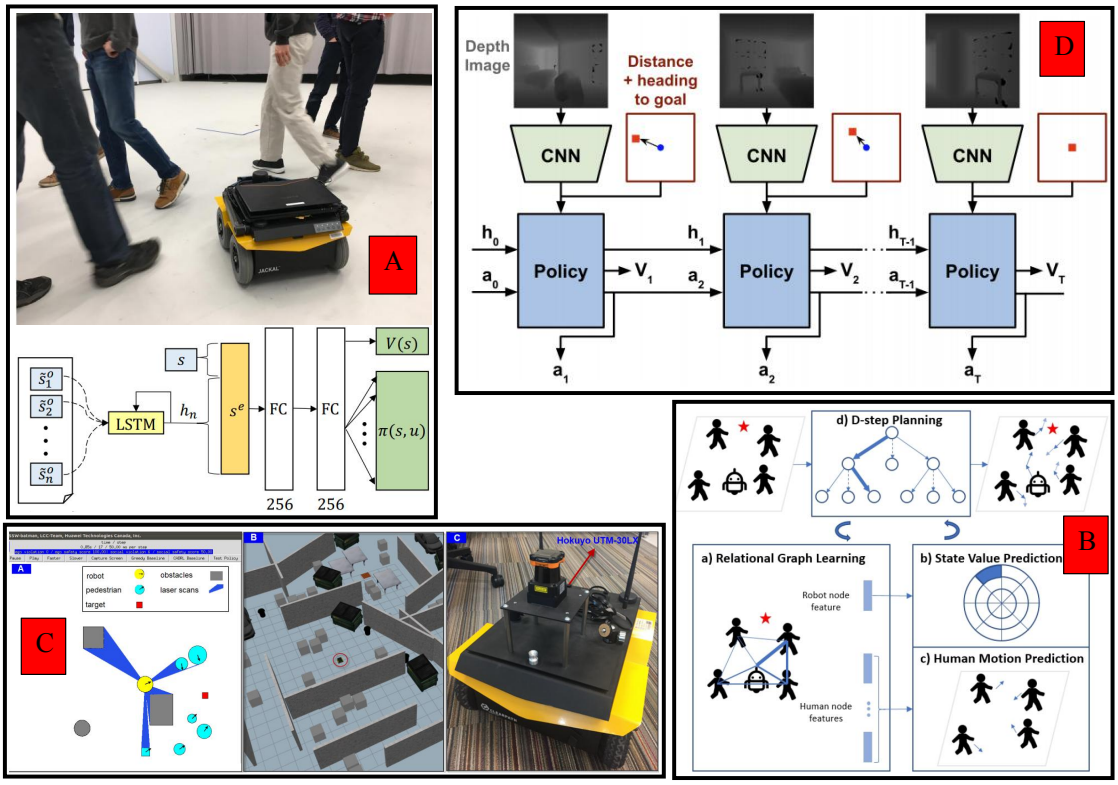


Figure 1.5: Navigation in dynamic environments. The observation of A and B is humans' fully known states including position, speed, and size whereas C and D directly map raw sensor observations such as images and LiDAR scans into navigation actions. A-D are authorized by IEEE [49, 54, 55, 61].

tively, to capture interactions and reason about relations between agents. However, these methods assume fully known pedestrian information and require massive positive datasets for learning, leading to degraded performance in real-time settings. EGO [55] (shown in Figure 1.5-C) and LSTM_EGO [61] (shown in Figure 1.5-D) offer a more direct mapping of raw sensor data to navigation actions, but open-source solutions for replicating their results are scarce. Moreover, it is time-consuming for training because of high-dimensional observations. Because simulation can not represent all real-world scenarios, sim-to-real transfer of these approaches is challenging.

1.3.5 Summary of State-of-the-art Navigation Strategies

Table 1.1 summarizes the pros and cons of state-of-the-art navigation strategies. Our goal is to comprehensively leverage the advantages of existing navigation techniques and diminish their downsides.

Table 1.1: Summary of state-of-the-art navigation strategies

Strategy	Pros	Cons
Map-based	accurate, explainable, repeatable, robust.	sensitive to dynamic obstacles, time-consuming to maintain maps and search global paths
Trajectory-based	adaptive to dynamic environments, explainable.	sensitive to state estimation and prediction, time-consuming for online optimization.
DRL-based, static environments	map-free, model-free offline optimization.	limited generalizability, low learning efficiency, sim-to-real gap.
DRL-based, dynamic environments, fully observable	map-free, model-free, offline optimization, handle social interactions.	ideal assumption, sim-to-real gap, rely on imitation learning.
DRL-based, dynamic environments, partially observable	fully end-to-end, little prior knowledge, handle social interactions.	limited adaptability, sim-to-real gap, low learning efficiency.

1.4 Approaches and Contributions

This thesis presents our findings on the research and development of robot navigation in dynamic environments using DRL. Our study consists of four progressive investigations, each with unique approaches and significant contributions, which are described below.

1.4.1 Navigation in Static Environments with a Single-frame LiDAR Scan

We present a hierarchical DRL (HDRL) framework with prominent sampling efficiency and sim-to-real transfer ability for fast and safe navigation. On the one hand, the low-level DRL policy enables the robot to move toward the target position and keep a safe distance to obstacles simultaneously. On the other hand, the high-level DRL policy is supplemented to further enhance the navigation safety. Both the lev-

els leverage a single-frame LiDAR scan to detect surrounding obstacles. To reduce the state space and avoid sparse reward, we select a waypoint located on the path from the robot to the ultimate goal as the sub-goal. Moreover, the path is generated based on either a local or a global map, which can significantly improve the sampling efficiency, safety, and generalization ability of the proposed DRL framework. Additionally, a target-directed representation for the action space can be derived based on the sub-goal to improve the motion efficiency and reduce the action space. In order to demonstrate the eminent sampling efficiency, motion performance, obstacle avoidance, and generalization ability of the proposed framework, we implement sufficient comparisons with the non-learning navigation methods and DRL-based baselines.

The main contributions of this work are: (1) creating a hierarchical DRL framework that can comprehensively consider fast and safe navigation; (2) introducing a sub-goal which is able to improve the sampling efficiency and generalization ability for the DRL model; (3) deploying the simulated policy on physical robots in the real world; (4) making our project publicly accessible.

1.4.2 Navigation in Dynamic Environments with Multi-frame LiDAR Scans

We present a Sampling Efficient DRL framework for Dynamic Navigation (SEDN) directly using raw LiDAR scans. To accelerate DRL training and simulate LiDAR 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 LiDAR scans into the center of the current LiDAR 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.

The main contributions of this study are summarized as follows:

- We create a kinematics-based simulator integrated with a LiDAR 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 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.

1.4.3 Navigation in Social Environments with Sequential Occupation Maps

We propose a Navigation system in diverse Pedestrian scenarios using a Dreamer-based (NPD) motion planner for collision-free and target reaching tasks. Our RL framework can completely learn from zero experience via a model-based DRL which can efficiently optimize navigation policy from sequential occupation maps. More specifically, the robot and humans are first projected onto an occupation map, which is subsequently decoded into low-dimensional latent state. A predictive dynamic model in the latent space is jointly created to efficiently optimize the navigation policy. Additionally, we leverage the techniques of system identification, domain randomization, clustering and LiDAR SLAM for practical deployment. Simulation ablations and real implementations demonstrate that our motion planner outperforms state-of-the-art methods, and that the navigation system can be physically implemented in the real world.

The contributions of this study are summarized as follows.

- A complete and publicly accessible autonomous navigation system among pedestrians is developed. We precisely obtain the robot pose using a LiDAR SLAM algorithm, and extract humans via a clustering approach. Moreover, we plan robot motion using a model-based DRL framework to avoid pedestrians and reach a target with a high success rate and navigation efficiency.

- We propose a Dreamer-based motion planning algorithm that can efficiently obtain an optimal motion planner and be generalized to arbitrary human number, variable human speed, and complex human relationships.

- We reproduce several state-of-the-art algorithms for more comprehensive ablation and ensure they are open-sourced. Additionally, sufficient sim-to-real experiments are implemented using domain randomization and system identification techniques.

1.4.4 Navigation in Social Environments with Sequential LiDAR Scans

We develop an efficient, versatile, and reliable DRL framework that enables the robot to Learn to Navigation in Dynamic environments using Normalized LiDAR (LNDNL) scans. Our approach utilizes LSTM to accumulate and propagate long-term surrounding motion features from sequential LiDAR scans taken from the ego-centric perspective. To expedite the training process, we have developed a simulator that can efficiently generate LiDAR scans and configure obstacles using limited hardware resources. Unlike other approaches that rely on continuous raw sensor observations and require large convolutional neural networks (CNNs) to decode, we leverage a lightweight LSTM unit to extract sequential motion features over a long time horizon. To minimize the sim-to-real discrepancies, we have normalized the shapes of real-world obstacles to make them consistent with the limited shapes in our simulator. Moreover, we have conducted extensive simulations and real-world implementations to demonstrate the advantages of our navigation strategy, including improved learning efficiency, versatility, repeatability, and practicality.

The contributions of this study are summarized 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 point-clouds. Additionally, we re-generate 2D LiDAR scans from the normalized obstacles to be consistent with simulated settings.

- We present a navigation framework that employs a combination of LSTM and DRL to translate sequential 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.

1.5 Thesis Organization

Our thesis is composed of four parts, as depicted in Figure 1.6. These parts are organized according to the level of task complexity and step-by-step improvements of navigation performance and algorithm framework.

Chapter 2 presents a preliminary investigation into robot navigation in static environments with initially unknown configurations, utilizing a DRL-based navigation policy in both simulated and real-world scenarios. To improve learning efficiency, we leverage existing path planners and local maps to generate traversable sub-goals.

To deal with dynamic environments without relying on maps, Chapter 3 explores a more complex navigation setting where obstacles are highly dynamic and observations are consecutive LiDAR scans. In contrast to the static environment assumed in Chapter 2, objects in this setting are consistently moving. Moreover, the study in Chapter 3 does not rely on global or local maps. To efficiently obtain an effective navigation policy, we leverage supervised samples and imitation learning to initialize neural networks.

Chapter 4 introduces an advanced DRL framework capable of efficiently optimizing navigation policy without prior knowledge such as the supervised samples and imitation learning required in Chapter 3. The observations in Chapter 4 are sequential occupation maps rather than continuous LiDAR scans whose time horizon is short. However, the massive neural networks used to auto-encode images result in prolonged learning iterations and degraded repeatability. Moreover, the occupation map is limited in relatively small motion areas because we are unable to infinitely enlarge image resolution when taking the observation dimension and network scale into consideration.

To address these limitations, Chapter 5 proposes a lightweight, efficient, versatile,

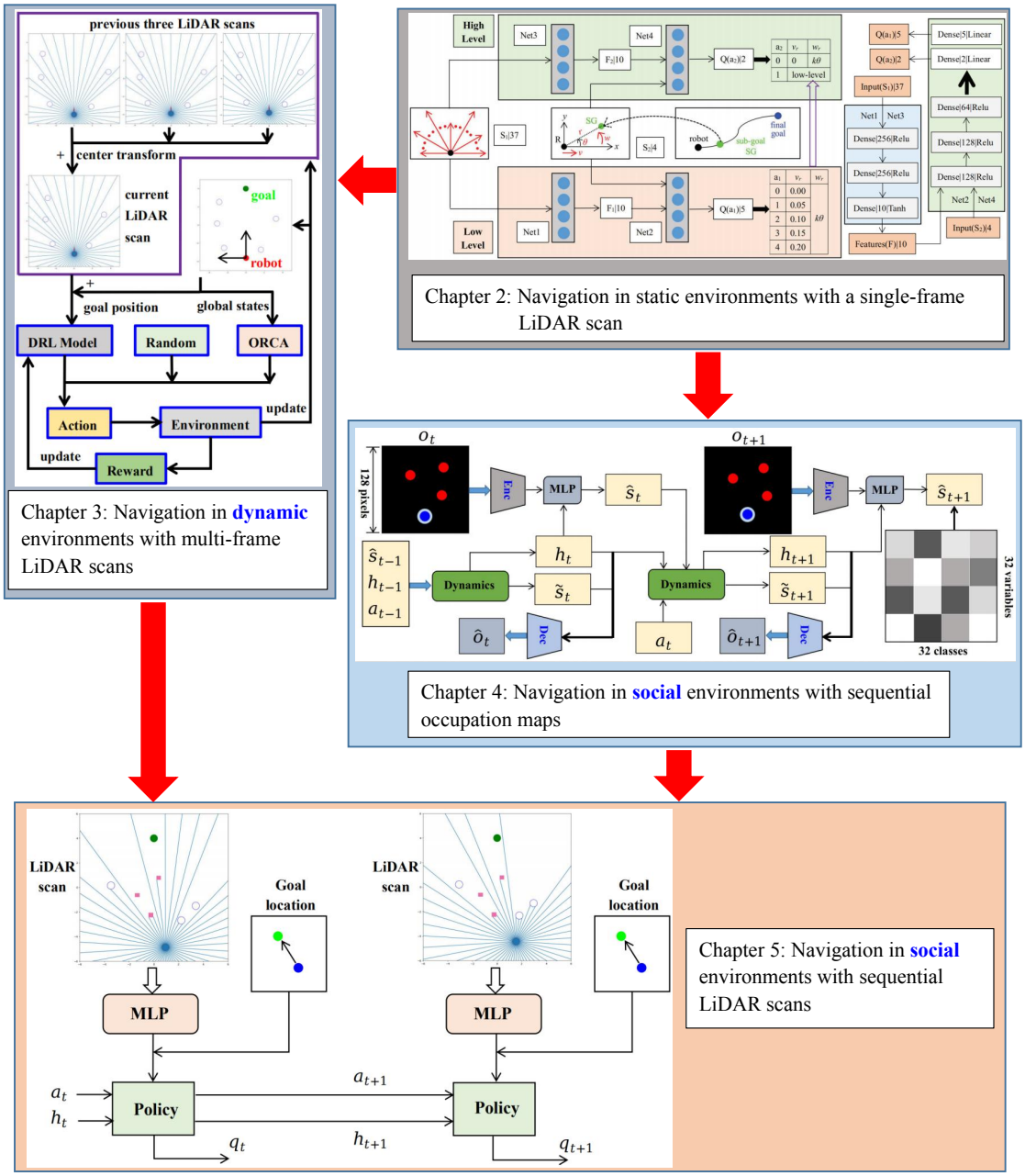


Figure 1.6: Thesis organization. Chapter 2 serves as the foundation for Chapters 3 and 4. The studies in Chapter 3 and 4 concentrate on the navigation of robots in environments that are highly dynamic, whereas the the research in Chapter 2 represents a fundamental experiment in deploying a DRL-based navigation policy in static environments with unknown initial configurations. By leveraging the strengths of studies in Chapter 3 and 4, our final work in Chapter 5 represents a significant advancement and results in a further improved outcome.

and repeatable DRL-based navigation strategy for highly dynamic environments by combining the strengths of the studies in Chapter 3 and 4. We leverage LSTM to deal with long-term LiDAR observations. In Chapter 3, the obstacle shapes in the real world are notably different from those in simulation where we assume obstacles to be circles or rectangles. To deal with this sim-to-real discrepancy, we normalize

Table 1.2: Summary of the studies from Chapter 2 to 5

Chapter	Task	Observations	Limitations
2	Navigation in static environment	Single-frame LiDAR scan	Static obstacles, existing path planner.
3	Navigation in dynamic environment	Multi-frame LiDAR scans	Imitation learning, short observations, sim-to-real gap.
4	Navigation in social environment	Sequential occupation maps	Heavy networks, limited motion area.
5	Navigation in social environment	Sequential LiDAR scans	–

the shapes of real-world obstacles to be consistent with simulated objects.

In summary, we list the task of each chapter, as well as its observations and limitations in Table 1.2. Finally, Chapter 6 summarizes the findings of the previous chapters and outlines possible future research directions for DRL-based robot navigation. Several videos demonstrating the achieved results have been published during this work and are listed at the end of the thesis. Furthermore, we make our projects accessible to the public by releasing them as open-source and providing links to the code alongside our video list.

Chapter 2

Navigation in Static Environments with a Single-Frame LiDAR Scan

This chapter incorporates material from the following publication:

W. Zhu and M. Hayashibe, "A hierarchical deep reinforcement learning framework wth high efficiency and generalization for fast and safe navigation," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 5, pp. 4962-4971, 2023, doi: 10.1109/TIE.2022.3190850.

2.1 Introduction

For autonomous navigation issues, mobile robots are required to get close to the target and keep safe distance to obstacles. To this end, either the traditional non-learning map-based methods, such as dynamic window approach (DWA) [23], or the state-of-the-art deep reinforcement learning (DRL) frameworks [3, 43, 44, 55], need comprehensive evaluation criteria that can balance the importance of obstacle avoidance and target reaching. It is however challenging to adjust the weights of the factors used in these criteria. Therefore, we propose a hierarchical DRL-based motion planner that can comprehensively take these criteria into consideration and achieve fast and safe navigation. Moreover, the motion planner acquired by the DRL framework is able to generate a novel and optimal navigation policy by interacting with environments whereas the traditional non-learning approaches are inclined to make mobile robots move along the pre-established trajectories which may be non-optimal with respect of path length and motion efficiency.

The representation for the action space, state space, and immediate reward function of DRL frameworks can make a significant difference to the sampling efficiency and generalization ability. For the DRL-based navigation, the action space is generally composed by linear and angular velocities [43, 44]. However, randomly generating either continuous or discrete velocities during the DRL training process may degrade the sampling and learning efficiency. Furthermore, the navigation environments are generally complicated because of the diversified locations, sizes, and shapes of obstacles, the time-varying robot states, and the different goal positions. Therefore, it is especially difficult to represent the state space and reward function for DRL frameworks. For instance, the data from either LiDAR or vision sensors used to detect obstacles have high dimensions and broad ranges [55]. When the final goal is far from the mobile robot, the reward function would be extremely sparse, which makes it prohibitively difficult to fully explore the environment. Furthermore, the complex obstacle distribution between the robot and the final goal makes it challenging to reasonably and comprehensively define the reward function. Even though the well-learned navigation policy can be successfully deployed on the same or similar environments as the training scenarios, it is tremendously difficult to generalize the policy into more variant environments.

To improve the sampling efficiency and generalization ability of the DRL model, we select a waypoint (namely sub-goal) which is close to the robot and located on a feasible path from the robot to the final goal. The definitions of DRL elements are based on the sub-goal rather the final goal. More specifically, the path is planned using either a fully-known global map or a real-time local map that is updated and maintained with the simultaneous localization and mapping (SLAM) algorithm [62]. Since the mobile robot explores the environment during the DRL training, it is reasonable and natural to update a map for path generation, thus to select an accessible waypoint for reducing the representation complexity of DRL elements. Since we only consider the neighboring surroundings around the robot and encourage the robot to move toward an accessible sub-goal, the generalization ability of the DRL framework is significantly improved. Based on the sub-goal, we define a target-directed representation for the action space that can further improve the motion efficiency for navigation and the sampling efficiency for the DRL model.

It is generally challenging for sim-to-real transfer even though the DRL-based navigation methods have achieved some real applications [43, 44, 55] because simulation environments are far different from real scenarios. The real mobile robots used to validate the proposed DRL framework are a wheeled bipedal robot [63] and a quadruped robot, whose dynamics are completely different from the dynamics of the differential mobile robot used in the simulation. In order to achieve the sim-to-

real transfer, we add noises to the action during the training to learn a policy that can deal with uncertainties and fine-tune the acceleration for both the simulated and real mobile robots because of their dynamic differences. Additionally, we also add mild noises to the action during validations in case of the frozen motion in certain extreme situations.

The main contributions of this work are: (1) creating a hierarchical DRL framework that can comprehensively consider fast and safe navigation; (2) introducing a subgoal which is able to improve the sampling efficiency and generalization ability for the DRL model; (3) deploying the simulated policy on physical robots in the real world; (4) making our project open-sourced on the website[1].

2.2 Related Work

There is a large body of work on navigation for wheeled mobile robots, using either LiDAR or vision sensors [64, 65]. For traditional non-learning map-based methods [19, 20], a pre-built global map is required for path planning. However, mapping is generally cumbersome when environments frequently change. When the global map is already known, a plenty of path planning literature can either produce a global path from the initial robot position to the final goal such as the A-star algorithm [21] and the rapidly-exploring random tree (RRT) method [22], or yield a local planner that can generate velocity commands and handle certain disturbances such as moving, removing or adding obstacles. Among the local planners, the dynamic window approach (DWA) as well as its variations is widely leveraged for obstacle avoidance [23, 24]. Nevertheless, the commonly-used global planners are unable to generate optimal paths because they search a feasible path from the map via a heuristic method. Moreover, the local planners need careful tuning for the coupled factors involving the cost weights of collision, path following, motion speed, etc.. For certain simple situations, if the environment does not change and the global path avoids all obstacles, some path tracking methods can be directly used to follow the path without the concern of collision [25, 26]. The aforementioned methods assume that a global map is already known, which is impractical when frequently changing the environment. One alternative is to simultaneously update a map in real time and navigate in the partially-known surroundings [27], which solves the problem of off-line time-consuming map building process at the expense of degrading the safety and optimality of navigation. Another solution is completely free of maps. In [28], a motion planner can generate collision-free action by the current and predicted

[1] http://wiki.ros.org/move_base

motion states of surrounding obstacles. For dynamic environments, the optimal reciprocal collision approach (ORCA) [30] and its extensions [66, 67] can ensure safe navigation but rely heavily on the overall information of the surroundings.

Recently, deep learning, especially DRL-based approaches have been widely applied for map-free navigation because of the powerful representation capability of deep neural networks and the strong exploration, interaction and self-learning ability of reinforcement learning (RL). In [4], a deep neural network structure is used to predict future relevant events, such as collision and terrain properties. The off-line gathered data with LiDAR and IMU sensors are required to update the deep neural networks. Similar collision prediction method is leveraged for safe navigation in [68]. However, collecting supervised data is arduous and it is challenging to generalize the prediction model in completely different scenarios because the off-line data are generally gathered from specific environments.

To improve the generalization ability and avoid manually collecting supervised data, researchers are focusing on autonomous interaction with environments and self-learning approaches, especially on RL frameworks. As a pioneering study [41], the DRL training is firstly implemented in simulation, where a great deal of vision data can be gathered to optimize the navigation strategy, then the simulated policy is directly deployed in the real world with similar settings as the simulation. A similar study [42] successfully transfers the learned policy into novel scenarios by approximating the reward function with a linear combination of learned features. Besides the vision-based DRL navigation policies which are sensitive to light intensity, LiDAR sensors are also broadly leveraged to gather environment information [43–45]. One crucial problem for these DRL frameworks is the inefficient sampling and learning because of the complexity of environments. Therefore, some studies are aimed for improving the sampling efficiency. For instance, prior demonstrations [46] are commonly leveraged to improve sampling efficiency and model-based RL frameworks [47] are practical to accelerate learning. However, the generalization capability is deteriorated because of the dependency on specific environments. Furthermore, it is still challenging to transfer the DRL-based navigation frameworks from simulation to the real world because the uncertainties in real scenarios can not be accurately represented in simulation. Moreover, most of real applications utilize wheeled mobile robots with stable motion properties [41, 43, 44]. Some sim-to-real applications are implemented on humanoid robots with omni-directional motion ability [69, 70]. However, it is rare to deploy the DRL-based navigation policy on wheel bipedal robots with unstable motion characteristics [63]. In this work, we achieved the sim-to-real transfer on a physical wheeled bipedal robot with unstable motion and a quadruped robot.

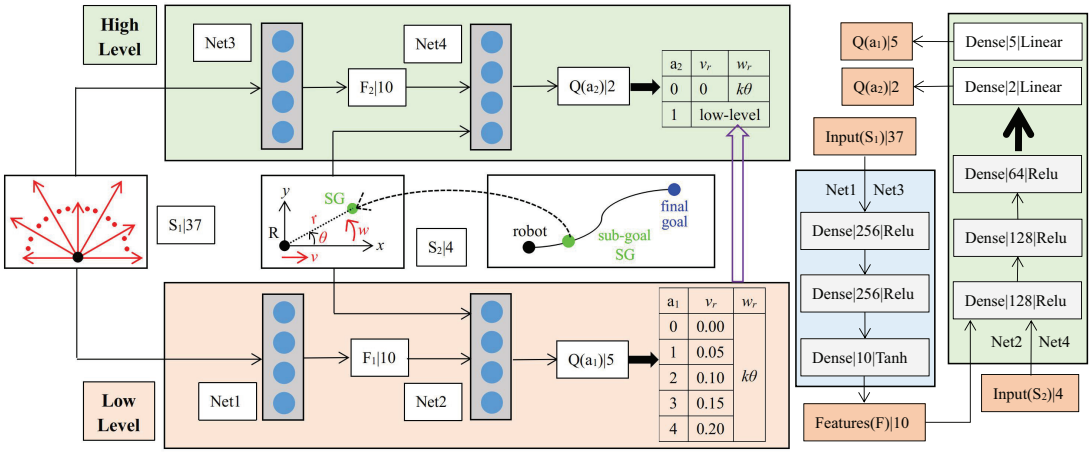


Figure 2.1: Hierarchical DRL framework. The low-level DRL policy is used for fast motion while the high-level DRL strategy is aimed at safe obstacle avoidance. Both the low- and high- level DRL policies have the same deep neural network structure and share the identical state input including a 37-dimension laser scan, robot's linear and angular velocities (v and w), and the position of the sub-goal in the robot frame (r and θ). Differently, the low-level DRL policy outputs five specific actions while the high-level DRL policy produces two abstract choices, one of which projects to the low-level DRL policy.

2.3 Hierarchical DRL Framework

We are aimed to propose a deep reinforcement learning approach with high training efficiency and generalization ability in respect to different surroundings and robot platforms for fast and safe navigation in complex environments. To this end, a two-layer DRL framework shown in Figure 2.1 is constructed, wherein the low-level DRL policy is responsible for producing quick motion while the high-level DRL policy is supplemented to further improve safety of obstacle avoidance. Because we choose the sub-goal which is a waypoint located on the path from the robot to the final goal, the observation space of RL is significantly narrowed. Moreover, the path is generated via traditional global path planning approaches which take consideration of both obstacles and the final goal position, the sampling space is therefore further reduced. Thanks to the introduction of the sub-goal, the training efficiency of our DRL framework is far higher than pure DRL methods. Besides the reduction of observation space, the scope of exploratory action is narrowed because the deep neural networks only output discrete linear velocity while the angular velocity is proportional to the orientation of the sub-goal in the robot frame. Additionally, the proposed DRL framework has great generalization ability not only for various environments but also for different robot platforms for three reasons: (1) a sub-goal located on a feasible path is selected to represent the DRL elements; (2) one part of observation is a high dimensional laser scan whose features are extracted via deep

neural networks; (3) the actions are generalized linear and angular velocities which are further transformed into actuator commands.

2.3.1 Problem Formulation

For the navigation task in environments with dense obstacles, the mobile robot needs to make decisions to avoid obstacles and reach targets according to the states of the robot as well as the surrounding information gathered by attached sensors. 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 states of the robot, surrounding obstacles, and the target position in the robot frame. A stands for the action space which is composed of the linear and angular velocities of the mobile robot. T represents the state transition, that is the next state $\boldsymbol{s}'$ is generated by the current action $\boldsymbol{a}$ and state $\boldsymbol{s}$. Such a state transition is implied by a physical simulator Gazebo. R, with the actual value r, denotes the immediate reward after executing $\boldsymbol{a}$ in $\boldsymbol{s}$, which is related to the distance from the robot to its surrounding obstacles and the distance from the robot to the target point. γ is the discount factor that indicates the changing reward in the time domain. The goal of this task is to acquire a policy π to maximize the expectation of the long-term cumulative reward $V_\pi(\boldsymbol{s})$:

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} V_\pi(\boldsymbol{s}), \\ V_\pi(\boldsymbol{s}) &= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = \boldsymbol{s} \right] \\ &= \sum_{\boldsymbol{a} \in A} \pi(\boldsymbol{a}|\boldsymbol{s}) Q_\pi(\boldsymbol{s}, \boldsymbol{a}), \end{aligned} \tag{2.3.1}$$

where Q is the action-value function. We leverage Deep Q-learning networks (DQNs) to formulate this optimization problem [34].

2.3.2 Low-level DRL

State space. Unlike the existing researches which use sparse laser scan as one part of state space, such as ref. [43] utilizes a 10-dimensional laser scan and ref. [44] selects a 13-dimensional laser scan, we choose a dense 37-dimensional laser scan $\boldsymbol{s}_l$ to extensively perceive surroundings. Moreover, instead of directly using this scan like ref. [43, 44], we pre-process it with deep neural networks, then 10 abstract features are extracted so as to generalize surroundings. For the feature extraction, we utilize

deep neural networks to map the relatively high-dimension raw laser scan to a low-dimension feature vector which can indicate the abstract obstacle features with the possibly authentic collision information. In fact, we can increase the dimensions of the raw laser scan with more details at the expense of more complex neural networks and longer training time. Moreover, rather than directly reaching the final goal, the robot is required to move close to the waypoint (namely sub-goal) one by one and gradually arrive at the ultimate goal location. Consequently, we define another part of the state space as the distance r from the sub-goal to the robot and the sub-goal orientation θ relative to the moving direction of the robot. In this case, the state space can be tremendously narrowed. Finally, we regard the current linear and angular velocities v and w of the robot as the third part of state because they are likely to predict future states of the robot. Therefore, the ultimate state $\boldsymbol{s}$ is defined as follows:

$$\boldsymbol{s} = \{\boldsymbol{s}_l | 37, r, \theta, v, w\} \in S. \tag{2.3.2}$$

To reduce the state space thus to improve training efficiency of DRL, we add certain constraints for these state elements. Firstly, the robot only needs to take close obstacles into consideration for local motion planning, so the maximum range of each laser scan beam is set as $l_{\max}$. Secondly, the maximum distance is defined as $r_{\max}$ with details introduced in Section 2.3.4 and the orientation of the sub-goal θ is free on the whole 2D plane. Finally, the ranges of v and w are limited according to the definition of the action space.

Action space. For the DRL-based navigation, the actions are generally either continuous or discrete linear and angular velocities of mobile robots [43, 44]. Randomly generating these two velocities during the trial and error process may cause inefficient and invalid sampling, thereby degrading the learning efficiency of DRL models. To improve the sampling efficiency, we propose a target-directed representation form for the actions, which is defined by two parts. (1) The angular velocity w is unrelated to the action space of DRL but proportional to the sub-goal orientation θ, which is represented as follows:

$$w = k\theta, \tag{2.3.3}$$

with k being a positive control gain. Such a traditional P controller can exclude one action variable from the DRL model, thereby exponentially reducing the exploratory action space for the DRL framework. Moreover, the motion orientation of the mobile robot is directed to the sub-goal, thus to significantly improve the motion efficiency and avoid certain invalid angular velocities. (2) The action space of RL only consists

of five non-negative discrete linear velocities which is defined as follows:

$$a_l \in \{0, 0.05, 0.1, 0.15, 0.2\} m/s \in A_l. \tag{2.3.4}$$

We assume the robot does not move backward and the maximum translation speed is $0.2m/s$ for the mobile robots we use in simulation and real worlds. Please note that we will decrease the maximum translation speed for the wheeled bipedal robot for stable motion and balance keeping. In this work, we define the action as discrete choices. Please note that continuous action can be beneficial for more smooth motion but an action network is required such as the deterministic deep policy gradient (DDPG) algorithm [71] and the soft-actor-critic (SAC) framework [72]. The discrete action is commonly used in the value-based reinforcement learning frameworks such as the DQN algorithm we utilize in this work. Although the discrete action may cause less smooth motion, the network framework is more simple. We choose the discrete action space only for simplicity. We think the continuous action can also work for the navigation task.

Reward function. The navigation objective is to simultaneously avoid obstacles and move close to the target point. Let r_o be the minimum distance from the robot to its surrounding obstacles which is measured by the laser sensor. Let r_t be the current distance from the robot to the sub-goal and r_{t-1} be the last distance. Let r_c be the collision distance. Then, we define the reward function R_l as below:

$$R_l = R_{lc} + R_{lg} + R_{la} \tag{2.3.5}$$

$$R_{lc} = \begin{cases} -200 & r_o < r_c & \text{(2.3.6a)} \\ -0.05 r_c / r_o & r_c \leqslant r_o \leqslant l_{\max} & \text{(2.3.6b)} \\ 0 & r_o > l_{\max} & \text{(2.3.6c)} \end{cases}$$

$$R_{lg} = \begin{cases} 0 & r_c > r_o & \text{(2.3.7a)} \\ 180(r_{t-1} - r_t) & r_c \leqslant r_o & \text{(2.3.7b)} \end{cases}$$

$$R_{la} = \begin{cases} 0 & a_l = 0.1, 0.15, 0.2 & \text{(2.3.8a)} \\ -4(0.1 - a_l) & a_l = 0, 0.05 & \text{(2.3.8b)} \end{cases}$$

R_{lc} is the low-level collision reward, R_{lg} represents the goal distance reward, and R_{la} stands for the action reward. On the one hand, if the robot collides with obstacles, a large punishment with -200 will be given. Furthermore, we also mildly punish the behavior which leads to approaching obstacles. On the other hand, a positive reward will be obtained if the robot gets closer to the sub goal. To avoid rotation in place or slow motion, we discourage small linear velocities. To summarize, the

definition of reward function can simultaneously encourage the robot to move close to the sub-goal as fast as possible and keep safe distance from obstacles.

Neural networks. Instead of directly using the 37-dimensional laser scan, we firstly process it with a three-layer neural network framework displayed in Figure 2.1 to extract a low-dimensional feature vector which can potentially describe the information about the free and occupied space. Then, we combine this feature vector with other four state elements in (2.3.2) to construct a new input for another three-layer neural network framework shown in Figure 2.1, which finally outputs five Q values corresponding to the five discrete linear speeds. During the training, the discount factor γ is 0.99, the learning rate α is set as 0.0001, and the batch size is 64, respectively.

2.3.3 High-level DRL

When we evaluated the pre-learned low-level policy, we found that the robot tended to collide with obstacles if the orientation of the sub-goal θ is far from zero. We think such a situation is caused by the long turning arc length corresponding to the large θ. For safer navigation, we replenish a high-level DRL framework that can learn a policy to avoid such collisions while keeping the motion efficiency of the low-level DRL framework.

State space, neural networks and action space. The description of the high-level DRL framework for the state space and neural networks is identical to that of the low-level framework while the definition of action space is different. The low-level policy is learned in advance and then regarded as one possible action for the high-level DRL policy. Moreover, we individually select the first element from the low-level DRL action space (2.3.4) as the other element to further improve the safety of obstacle avoidance. Therefore, the action space of the high-level DRL framework is summarized as below:

$$a_h \in \{0, 1\} \in A_h. \tag{2.3.9}$$

If $a_h = 0$, the linear velocity is zero. When $a_h = 1$, the linear velocity is generated by the pre-learned low-level policy. In both cases, the angular velocity is same as (2.3.3). Although the case of $a_h = 0$ may degrade the motion efficiency because the robot only rotates in place, it enables the robot to avoid the collisions caused by certain large turning arc length.

Reward function. Because the high-level policy is aimed for safer navigation while keeping the motion efficiency of the low-level policy, we therefore define the reward

function as below:

$$R_h = R_{hc} + R_{ha} \tag{2.3.10}$$

$$R_{hc} = \begin{cases} -200 & r_o < r_c \quad (2.3.11a) \\ -1.2r_c/r_o & r_c \leqslant r_o \leqslant l_{\max} \quad (2.3.11b) \\ 0 & r_o > l_{\max} \quad (2.3.11c) \end{cases}$$

$$R_{ha} = \begin{cases} 0 & a_h = 1, r_c \leqslant r_o \quad (2.3.12a) \\ -r_o & a_h = 0, r_c \leqslant r_o \quad (2.3.12b) \end{cases}$$

R_{hc} is the high-level collision reward while R_{ha} represents the action reward. Same as the R_l defined in (2.3.5), we heavily punish the behavior which causes collisions. But differently, we magnify the factor from 0.05 to 1.2 so as to further make the robot get far from obstacles. The action reward in (2.3.12) is used for avoiding possible long-time rotation in place and frozen motion, thereby maintaining the motion efficiency of the low-level policy.

2.3.4 Selection of Sub-goal

One of the key contributions in this work is the selection of sub-goal. Most of navigation studies with DRL are final-goal directed [43,44,55], that is the state space S and the immediate reward R are related to the final goal, which causes several serious flaws. Firstly, if the final goal is far from the robot, it will be extremely difficult to thoroughly explore all possible state space because covering the whole state space generally requires prohibitively long interaction with the environment. Secondly, it is challenging to reasonably define the reward function for two main reasons. (1) The large exploration space may cause exceptionally sparse reward distribution, thereby seriously influencing the sampling efficiency. (2) Most of studies directly define the reward function based on the straight line length from the robot to the final goal while ignoring the possible effects of the obstacles between the robot and the final goal. We also found that the robot tended to collide with obstacles when using such kind of reward functions. Thirdly, directly using final goal may potentially increase the representation complexity of the action space A because the policy networks, which generally project states into actions, are required for actor-critic (AC) based RL frameworks. In this case, the sampling efficiency is further deteriorated.

To effectively solve these problems, we select a sub-goal which is close to the robot rather than directly using the final goal. Instead of discarding past laser scans like how most of studies do [43, 44, 55], we utilize these historic laser data to update

a grid occupation map with the SLAM algorithm [62]. Then we can generate a global path from the robot to the final goal using global path planners. The path is generally non-optimal because of the non-fully explored surroundings but feasible for the local utilization. Therefore, we can select a waypoint which is close to the robot from this path at the beginning of each training iteration. Through moving close to the waypoint one by one, the robot can successfully reach the final target. Because both the SLAM algorithm and the path planning method are general for diversified environments, the proposed hierarchical DRL framework can be directly applied in the new scenarios which are never explored ahead during the training. Besides, the waypoint significantly narrows the state space and avoids the sparse reward, which enables fast learning for the DRL model.

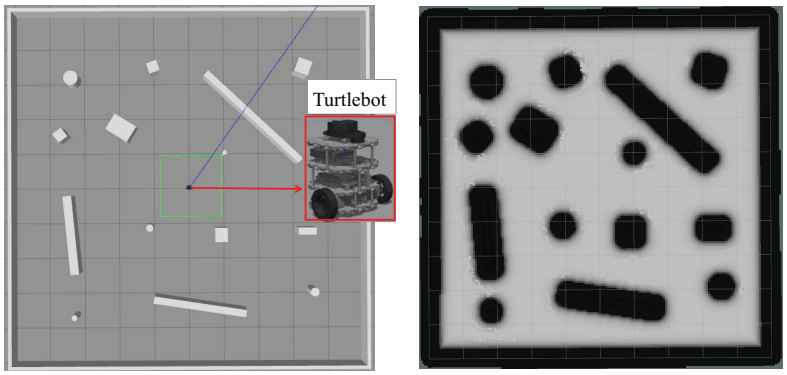
2.4 Experiments

To validate the predominant sampling efficiency, fast motion and eminent success rate of target reaching, and high generalization in respect of different robot platforms and surroundings for the proposed hierarchical DRL framework, we respectively implemented corresponding experiments and compared with existing approaches, with the overall video shown on the website[2].

2.4.1 Auxiliary Implementation Tools

To avoid frequently updating and resetting a non-fully-known global map during the training process, we assume that the global map is already off-line built utilizing the SLAM algorithm with the ROS gmapping package[3], and then we can on-line plan a global path with the ROS A-star package[4] and select a waypoint close to the robot as the sub-goal for improving the sampling efficiency. After training, we testified that the learned policy could be successfully deployed as well in the environments without off-line pre-built global maps. For instance, all real implementations are free of off-line global maps, thus our method can be quickly deployed in various real worlds without pre-building global maps by manual operations. In simulation, we obtain the odometer information with Gazebo-ROS APIs while it is collected with the internal wheel odometry when operating the wheel bipedal robot, and the external motion capture system for the quadruped robot, respectively. For the wheel

[2]https://youtu.be/S95BQDEiTE0
[3]http://wiki.ros.org/gmapping
[4]http://wiki.ros.org/nav_core



(a) Gazebo visualization (b) Occupation map

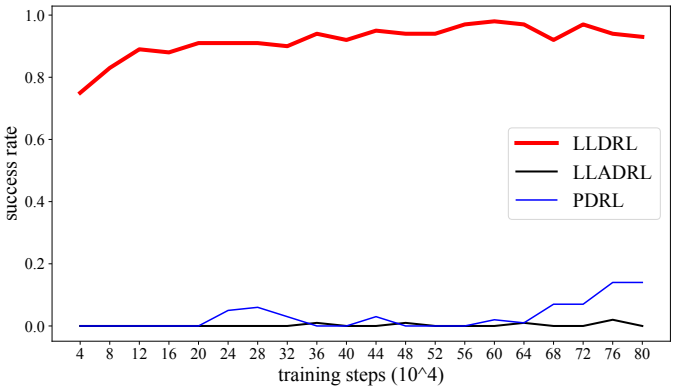
Figure 2.2: Training environment. The Turtlebot mobile robot moves at 10Hz control frequency in a rectangle with 10×10m size. The laser sensor, with 10Hz scanning frequency, is installed on the top of Turtlebot. The left figure visualizes the Gazebo simulation environment while the right figure displays the occupation map pre-built with the SLAM algorithm.

odometry, we leverage the ROS extended Kalman filter (EKF) package[5] to reduce the drift by fusing the IMU data and the wheel odometer information. For the traditional non-learning based comparison method, we utilize the ROS move_base package[1]..

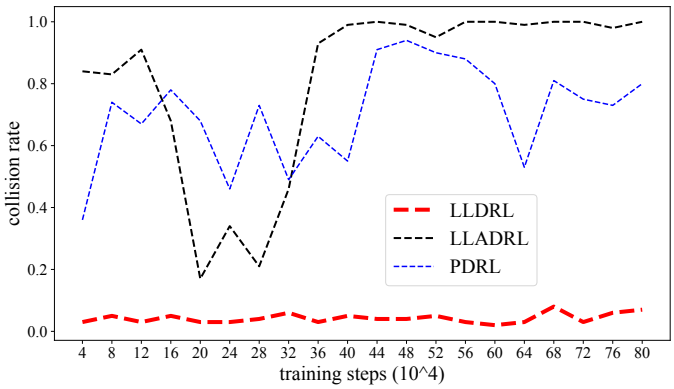
2.4.2 Training Environment Settings

We use Gazebo as the physical simulator. The size, shape, and location of obstacles are randomized in a 10×10m enclosed rectangle shown in Figure 2.2. At the beginning of each training iteration, the robot pose and the final goal position are randomly chosen from the free space of the global map to explore the environment as thoroughly as possible. The laser sensor, with 901 beams ranging from $-\pi/2$ to $\pi/2$ rad relative to the translation direction of the robot, is attached on the top of the robot. All 901 laser beams are used for updating the minimum distance (r_o) from the robot to its surrounding obstacles and building a global map with the SLAM algorithm. 37 evenly spaced laser beams are selected for composing the state space of the DRL model. We assume that the global map is already off-line built with the SLAM algorithm, and then we can on-line plan a global path and select a waypoint close to the robot as the sub-goal for improving the sampling efficiency. The control frequency is 10Hz which is the maximum sampling frequency of the real LiDAR sensor we use in the real world. The high-level and the low-level DQN policies are executed at 10Hz. Empirically, we update the sub-goal every 2 seconds. Moreover,

[5]http://wiki.ros.org/robot_localization



(a) Success rate



(b) Collision rate

Figure 2.3: The comparison of the training efficiency. 100 tests are executed every 40 thousand training steps. The test results include successful navigation, collision and time out that is the robot does not reach the final goal within a given time. The top figure shows the success rate while the bottom figure illustrates the collision rate.

the global path is also updated every 2 seconds because we only partially know the global environment.

For better comparison with baselines, we train all DRL frameworks 800 thousand steps. Because the control frequency is 10Hz, the theoretical time scale is about 22 hours. However, the real training time is around 7 days because the simulation is paused after each step for updating the deep neural networks. We found that updating policy cost tens of milliseconds while pausing and restarting the Gazebo took prohibitively long time (about half second), which is the major reason of the time-consuming training. Besides, unlike Mujoco, the Gazebo simulation can not be accelerated even runs with more time than the theoretical time because of the computer hardware limitations, which is another reason of the long-time training.

Table 2.1: The comparisons of the success rate of target reaching and the motion efficiency. The average goal reaching time used to imply the motion efficiency is calculated only from the tests where all five methods succeed in the target reaching.

	HDRL	LLDRL	MB	CDRL	DDRL
success rate (%)	**100**	93	**100**	70	87
average time (s)	**28.0**	29.7	34.4	117.2	65.4

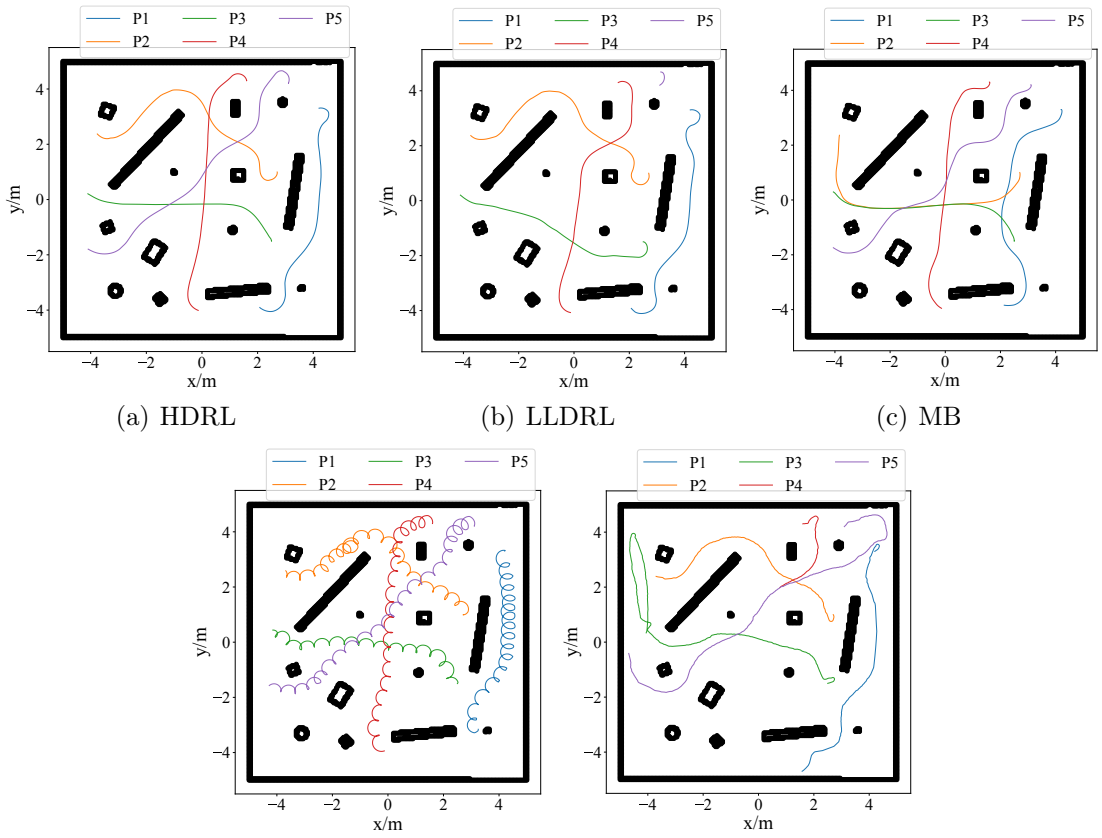
Table 2.2: Average speed (unit: cm/s) of each path for the proposed method and the move_base approach.

	P1	P2	P3	P4	P5	Average
HDRL	**18.7**	**17.6**	**17.6**	**18.8**	**18.4**	**18.2**
MB	15.8	13.5	12.4	16.6	13.4	14.3

2.4.3 Simulation Evaluations

Firstly, to validate the prominent sampling efficiency of the proposed low-level DRL (LLDRL) framework, we implemented a pure DRL (PDRL) baseline as the comparison. PDRL directly uses the final goal to represent the state space and the reward function. Moreover, to verify the significant effect of the action definition of our method, we introduced another DRL-based baseline which is similar to LLDRL except that the angular velocity is set as a discrete set, namely LLADRL, rather than the equation (2.3.3) utilized in the proposed approach. Because the goal is far from the robot, PDRL can not directly utilize the angular velocity defined in (2.3.3). Instead, the angular velocity is discretized as $-\pi/2 + \pi/6 \cdot i$ rad/s with $i = 0, 1, .., 6$. The linear velocity choice is same as the action space (2.3.4). Therefore, the action choices of PDRL are 35. The training result is shown in Figure 2.3, where LLDRL only needs around 120 thousand training steps to reach a stable and high success rate of navigation while PDRL can only achieve 14% success rate even after 800 thousand training steps. Meanwhile, the LLADRL method yields the worst result, only with 2% success rate.

Secondly, we demonstrated that the proposed hierarchical DRL (HDRL) framework could encourage fast motion and boost the success rate of target reaching by comparing with the traditional move_base[1]. (MB) navigation method, another two DRL-based baselines namely CDRL and DDRL respectively, and LLDRL. CDRL originates from [43] with continuous linear and angular velocities while DDRL is derived from [44] where the angular velocity is discrete and the linear velocity keeps constant. We tested these five frameworks 100 times in the same environment as



(a) HDRL (b) LLDRL (c) MB

(d) CDRL (e) DDRL

Figure 2.4: Path visualization. The paths of LLDRL are similar to those of HDRL but the path five of LLDRL enters the fatal collision area. The paths of the baseline CDRL illustrate the inefficient motion of rotation in place while the paths of the baseline DDRL display the vibrant jitter which degrades the motion efficiency.

the training one shown in Figure 2.2(a). The comparison results are shown in Table 2.1, where HDRL achieved the highest success rate while taking the shortest average motion time. During the testing, we found that the robot tended to rotate in place when using the baseline CDRL, which caused the lowest success rate and took the longest average time for successful target reaching. The phenomenon of rotating in place also happened in [43]. We think that it is because of the action space of 2-dimensional continuous velocity that makes it difficult to quickly acquire a stable policy. The baseline DDRL achieved relatively higher success rate and shorter motion time due to the relatively simple action space, but it brought about frequent jitter on the down side, which degraded the motion efficiency and the target reaching performance. Additionally, the success rate was increased to 100% from 93% when using HDRL, which demonstrated that the high-level DRL policy did play a significant role for performance improvement. Even though MB achieved 100% target reaching, its motion efficiency was obviously lower than HDRL. We think that the move_base method excessively emphasizes the collision avoidance and path tracking at the expense of motion efficiency while our method acquires a novel and optimal

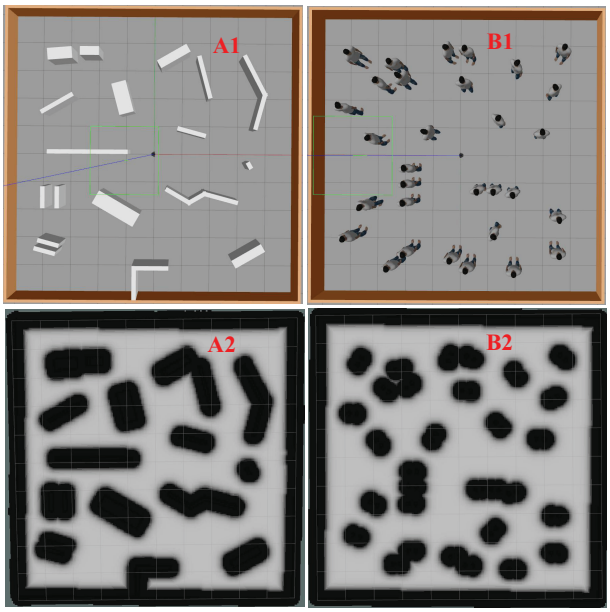
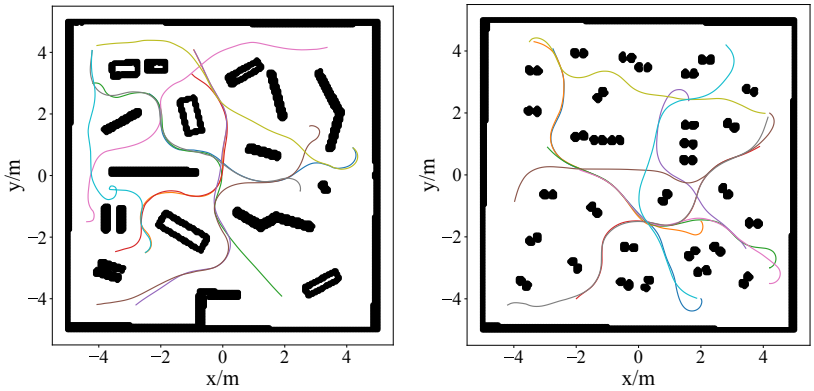


Figure 2.5: New environments. Neither the two environments are explored with the proposed DRL framework. A1 and B1 display Gazebo scenarios while A2 and B2 are corresponding occupation maps built with the SLAM algorithm. Compared with the training environment shown in Figure 2.2, the obstacles in these two environments are much more dense and exhibit more complex topological structure, such as the obstacle cluster.

motion policy during the training and learning process. For further visualizing comparisons, we selected five paths for each framework shown in Figure 2.4. The paths in Figure 2.4(d) displayed the phenomenon of rotation in space and the paths in Figure 2.4(e) were less smooth than those paths in Figure 2.4(a)-2.4(c). The initial turning arc length of the paths in Figure 2.4(b) was generally longer than that of the paths in Figure 2.4(a), which caused one failure shown in Figure 2.4(b). The average motion speed of these five paths for HDRL and MB is shown in Table 2.2, which demonstrated that our method could generate faster motion than the traditional move_base approach even though both achieved smooth and safe navigation at the similar level.

Thirdly, the generalization ability in respect to various environments were proved by deploying the proposed hierarchical DRL policy in two completely diverse and complex scenarios (shown in Figure 2.5) which had never been used for the DRL training before. We implemented 100 tests for each environment. The success rate of target reaching in the environment A was 99% and B reached 100%. We visualized 10 paths for each scenario displayed in Figure 2.6. At the beginning of some paths, the curvature radius was very small in order to quickly avoid collision and keep motion efficiency because the initial orientation of the sub-goal deviated hugely from the moving direction of the robot. However, we still found that the



(a) Paths in environment A (b) Paths in environment B

Figure 2.6: Successful navigation paths in two novel worlds totally different from the training environment.

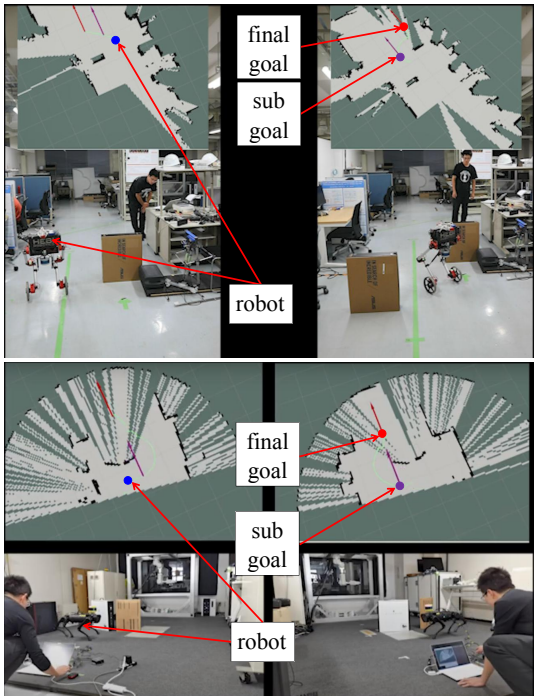


Figure 2.7: Sim-to-real transfer in diverse scenarios with different robot platforms. The localization for the wheeled bipedal robot is based on the wheel odometer and the Kalman filter while the quadruped robot relies on the external motion capture because the embedded odometer drifts heavily.

robot collides with the obstacle at the very beginning in the environment A. We think the initial location of the robot is so close to obstacles that there is no enough time and distance space for collision avoidance.

2.4.4 Sim-to-Real Transfer

Finally, we implemented the hierarchical DRL framework on a physical wheeled bipedal robot and a quadruped robot to validate the capability of sim-to-real transfer and generalization ability in respect to different robot platforms shown in Figure 2.7. The wheeled bipedal robot is developed by HEBI robotics[6] and we installed a Velodyne LiDAR sensor under the robot chassis. We use the quadruped robot from unitree[7], with the LiDAR sensor attached on the robot head. Because the dynamics of the bipedal robot [73] is completely different from that of the wheeled mobile robot used in simulation, we separately trained a hierarchical DRL framework. More specifically, we still utilized the wheeled mobile robot for the training in simulation. Differently, the action space of the low-level DRL framework was reduced to three choices (0.0, 0.05, 0.1)m/s since the small linear velocity is beneficial for the balancing control of the bipedal robot. Moreover, we supplemented acceleration limitations for the linear and angular velocities because of the motion constraints of the wheeled bipedal robot. For the quadruped robot with the stable motion property, we directly deployed the simulated policy in the real world without any fine-tuning. As shown in Figure 2.7, we tested the simulated DRL policy in diverse real scenarios with different robot platforms, where the global maps were initially unknown and simultaneously updated with the SLAM algorithm in real time. Thanks to the partially-known global map, a rough path from the robot to the final goal was planned and updated on-line. Based on this path, a sub-goal close to the robot was selected for the proposed DRL policy. Additionally, we performed another five practical experiments in more realistic scenarios to further demonstrate the generalization ability in the physical world, with the video shown on the website[8].

2.5 Conclusions

We presented a hierarchical deep reinforcement learning framework for navigation issues, with the low-level DRL policy being aimed for fast and safe navigation, and the high-level DRL strategy for further enhancing safety. The proposed framework demonstrated the eminent advantages in terms of high sampling efficiency. Moreover, the framework was able to achieve fast and safe navigation in diverse environments because we directly utilized the sub-goal close to the robot rather

[6]https://www.hebirobotics.com/
[7]https://www.unitree.com/
[8]https://youtu.be/FSm-EamibPI

than the ultimate goal and proposed a target-directed representation for the action space based on the sub-goal. Additionally, our method could be deployed on different robot platforms even with complex dynamics and be transferred to real worlds without globally known maps.

These advantages were demonstrated by sufficient simulation comparisons and real implementations. Firstly, the sampling efficiency was validated by comparing with the pure DRL model. Secondly the motion efficiency and the collision avoidance safety were confirmed via comparing with the commonly-used move_base approach and another two DRL-based baselines. Finally, the generalization ability with respect to environments and robot platforms was affirmed through deploying the hierarchical DRL policy on novel and complicated environments using a wheeled mobile robot, a wheeled bipedal robot with complicated dynamics and a quadruped robot. For our current work, we assume that the environment is static, we therefore will try to extend our method in the dynamic environments with crowded pedestrians in the following chapters.

Chapter 3

Navigation in Dynamic Environments with Multi-Frame LiDAR Scans

This chapter incorporates material from the following submission:

W. Zhu and M. Hayashibe, "Sampling efficient deep reinforcement learning for dynamic navigation with raw laser scans," revision and preparation for re-submission.

3.1 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 [31] and optimal reciprocal collision avoidance (ORCA) [30], and deep reinforcement learning (DRL)-based navigation studies [48,49,52–54,74] 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 [55, 56, 75–77]. 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. There-

fore, our end-to-end and straightforward strategy can release the assumption of fully known environments and be generalized into diverse environments.

- 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`.

3.2 Related Work

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[1] 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 [4, 41, 43, 44, 46, 78–80] 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 [81]. 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) [29] and optimal reciprocal collision avoidance (ORCA) [30], can safely generate an action for each moving object based on a reciprocal assumption. Trajectory-planning-based methods [31–33] 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.

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

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 [83, 84]. 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) [48], socially aware (SA)-CADRL [74], and GA3C-CADRL [49] pioneer DRL-based navigation in dynamic environments. Meanwhile, SARL [53], relational graph learning (RGL) [54], and decentralized structural (DS)-RNN [85] 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) [86] DRL model is leveraged to clone the dynamic navigation policy generated from expert demonstrations with only raw RGB-D image observations [75]. However, owing to the generalizability restriction of imitation learning, only relatively simple real implementations are realized in this study. Similarly, another study [76] 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 [55, 56]. 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.

3.3 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. Figure 3.1 shows the

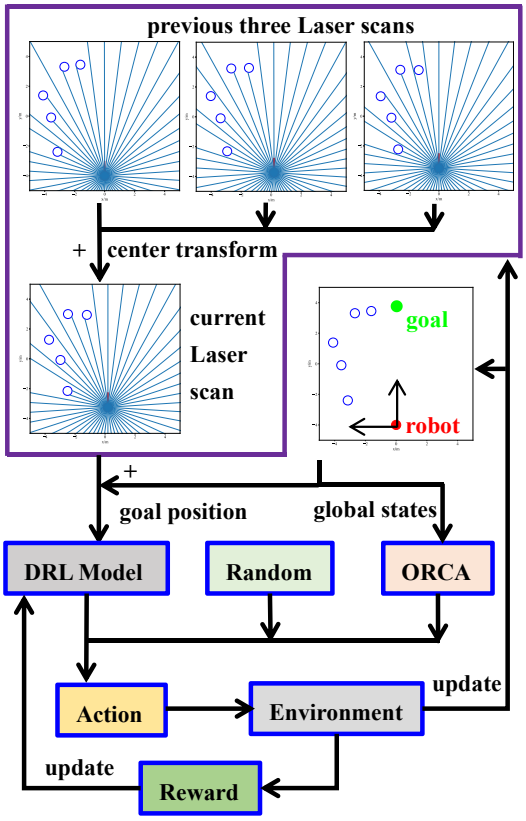


Figure 3.1: Overview of our approach. First, the centers of the previous three laser scans are transformed into the center of the current laser scan. Next, these four laser scans and the goal positionin the robot frame are combined as the observations of the DRL model. The action has three options. One 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.

overview of our approach.

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

$$\begin{aligned} \pi^* &= \underset{\pi}{\operatorname{argmax}} V_\pi(\boldsymbol{s}), \\ V_\pi(\boldsymbol{s}) &= E_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | S_t = \boldsymbol{s} \right] \\ &= \sum_{\boldsymbol{a} \in A} \pi(\boldsymbol{a}|\boldsymbol{s}) Q_\pi(\boldsymbol{s}, \boldsymbol{a}), \end{aligned} \tag{3.3.1}$$

where Q is the action-value function. We utilize the Deep Q-learning networks (DQN) [34] algorithm to iteratively update the action-value function and optimize navigation policy:

$$\begin{aligned} Q(S_t, A_t) \leftarrow & Q(S_t, A_t) + \alpha[R_{t+1} + \\ & \gamma \underset{a}{\max} Q(S_{t+1}, a) - Q(S_t, A_t)], \end{aligned} \tag{3.3.2}$$

where α is a step size parameter.

3.3.2 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 [87] and quadrotor [88] 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 Figure 3.2. Similar to the simplifications in existing studies [53], 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

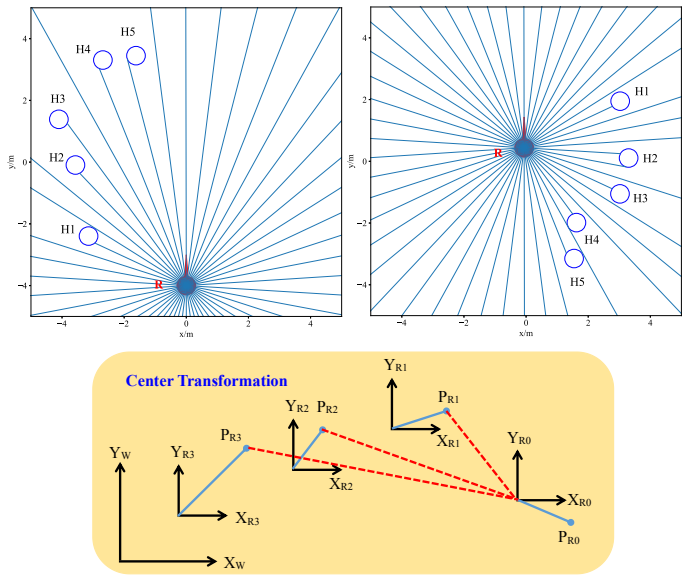


Figure 3.2: Raw laser scans and center transformation in training environments. The left top frame displays the initial states and the right top frame shows the intermediate motion. There is a total of 1800 laser scan beams, and a few are shown in these two sub-figures. The bottom figure illustrates 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.

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 [48, 52–54, 74], 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 di-

mensionality. 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 Figure 3.2, 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 Figure 3.1 and Figure 3.2 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 Figure 3.2. 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.

3.3.3 Elements of the DRL Model

States. The states $\boldsymbol{s}_t$ 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 [56] and another one slightly processing original laser scans to guarantee that all scans start from a fixed direction [55], we transform the centers of previous laser scans into the center of the current laser sensor shown in Figure 3.2. Consequently, we can decouple robot's motion and independently extract surrounding motion features.

Actions. We define the action space as a set of 81 discrete speed pairs. Because the robot motion is assumed to be omnidirectional 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

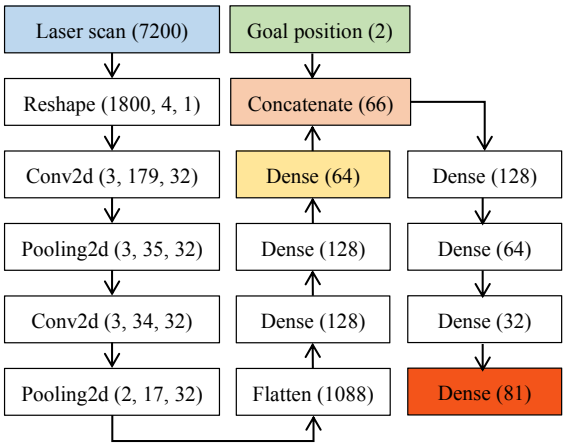


Figure 3.3: Network structure. For each unit except the laser scan and the goal position, the left text represents the network operation, and the right tuple is the output dimension after operation.

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:

$$R(\boldsymbol{s}_t) = R_l(\boldsymbol{s}_t^l) + R_g(\boldsymbol{s}_t^g), \tag{3.3.3}$$

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:

$$R_l(\boldsymbol{s}_t^l) = \begin{cases} -1.0 & d_{\min} \leq r \\ -0.5 \cdot dt \cdot (d_c - d_{\min}) & r < d_{\min} \leq d_c \\ 0 & d_{\min} > d_c \end{cases}. \tag{3.3.4}$$

The second case in (3.3.4) 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:

$$R_g(\boldsymbol{s}_t^g) = \begin{cases} 1.0 & r_t^g \leq r \\ 0.01 \cdot (r_{t-1}^g - r_t^g) & r_t^g > r \end{cases}. \tag{3.3.5}$$

The last case in (3.3.5) encourages the robot to get close to the goal, thus avoiding a possibly invalid exploration.

Network structure. We leverage the DQN algorithm [34] to model the dynamic navigation, with the network structure illustrated in Figure 3.3. 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 Figure 3.1, is selected from the index corresponding to the maximum Q value.

3.3.4 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 Figure 3.1) 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.

3.3.5 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 $\mathrm{T}_1 = 2000$ times to initialize the deep

Algorithm 1: SEDN algorithm

Prefill: Use ORCA to prefill the experience pool with a maximum capacity EX_m;
Pretrain: Pretrain the DRL model T_1 times with a learning rate α_1;
$episode \leftarrow 0$;
while $episode < \mathrm{EP}_m$ **do**
  Randomize the initial human positions;
  $done \leftarrow False$;
  while *not done* **do**
    if $\mathrm{EP}_d < episode$ **then**
      $p_1 = \epsilon_s - (\epsilon_s - \epsilon_e)/\ \mathrm{EP}_d * episode$;
    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 T_2 times with a learning rate α_2;
  $episode \leftarrow episode + 1$;
end

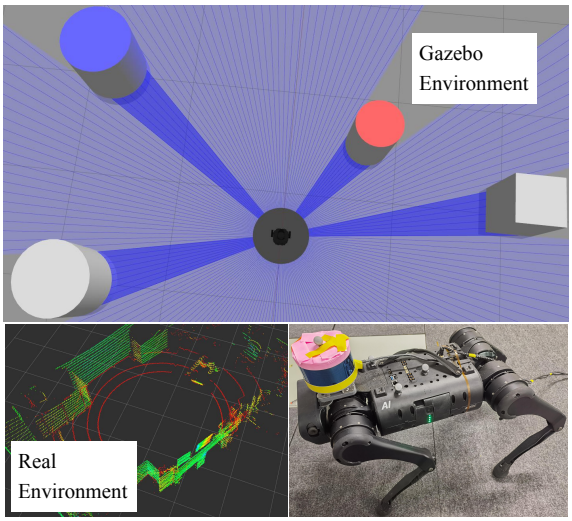
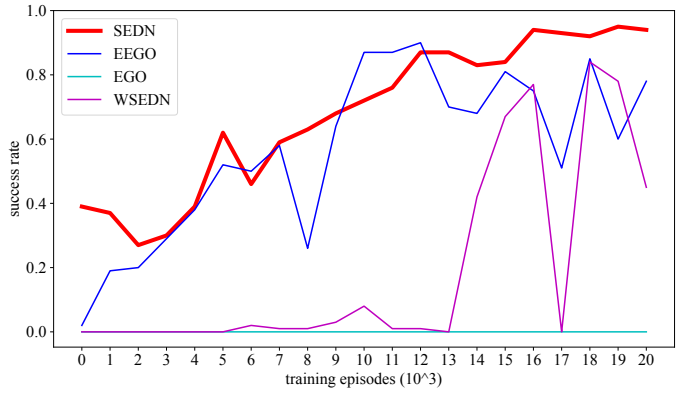


Figure 3.4: Physics-based experiment environments. The top image displays the Gazebo environment and the bottom two figures show the real-world scenario. In Gazebo, surrounding motions are generated by ORCA. In the real world, we transform the 3-D LiDAR point cloud into a 2-D laser scan. We utilize external motion capture system to localize the robot because the quadruped robot's odometer drifts heavily.

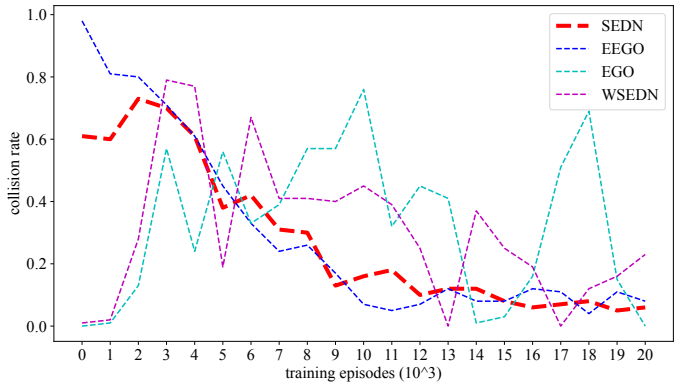
neural networks. Then, we repeat the trial and error for $\mathrm{EP}_m = 2 * 10^4$ episodes. After each episode, we train the model $\mathrm{T}_2 = 20$ times with a smaller learning rate $\alpha_2 = 0.0001$. We decay the exploration factor from $\epsilon_s = 0.8$ to $\epsilon_e = 0.03$ within $\mathrm{EP}_d = 1.6 * 10^4$ episodes to progressively balance exploration and exploitation.

3.4 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 [55]. 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 Figure 3.4.



(a) Success rate

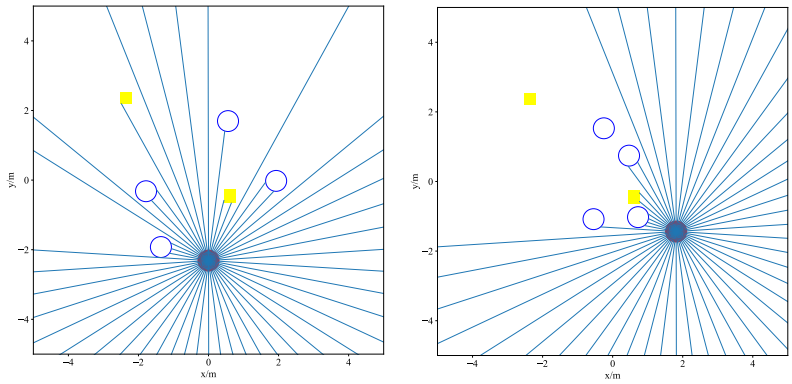


(b) Collision rate

Figure 3.5: 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.

3.4.1 Training

To verify the high sampling efficiency and superior navigation performance of our method, we reproduced a pioneering baseline – EGO [55], for comparison. Because the EGO algorithm is not publicly accessible, we reproduced it according to the aforementioned definitions in Section 3.3. The state space of EGO is four consecutive laser scans without center transformation. EGO's action space is same as our method's. Because EGO does not have ORCA-assisted samples, its success rate shown in Figure 3.5 is zero at all times. Subsequently, we enhanced EGO's sampling efficiency by integrating ORCA, namely EEGO. As shown in Figure 3.5, 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 Figure 3.5, 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,



(a) Scene 1 (b) Scene 2

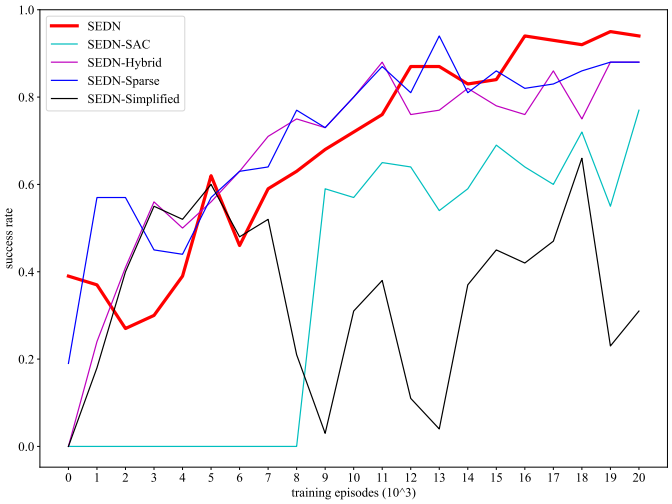
Figure 3.6: Hybrid scenarios with both dynamic and static obstacles. The circles are dynamic objects while the yellow rectangles are static barriers with random positions. The side length of the rectangle varies from 0.3 to 0.4m during training.

WSEDN is able to improve the success rate to 84%, which qualitatively demonstrates that the center transformation for laser scans can improve navigation performance.

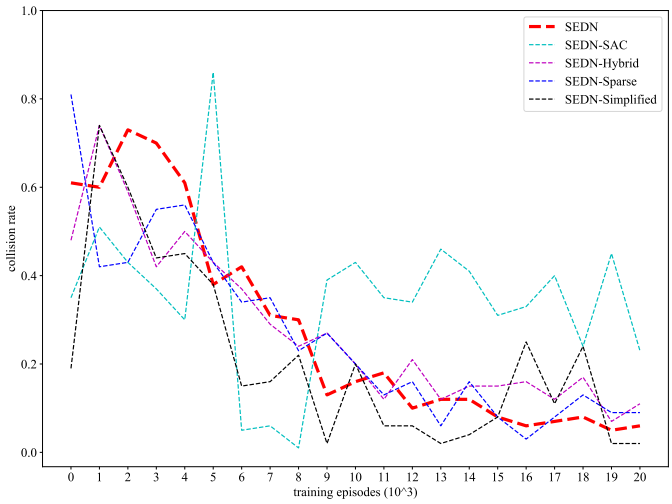
In summary, EEGO and SEDN, with maximum success rates of 90% and 95%, notably outperformed EGO-0% and WSEDN-84%, respectively. This qualitative and quantitative analysis verifies that the ORCA-assisted sampling is able to enhance learning efficiency. Although EEGO can progressively improve the success 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.

3.4.2 Model Revision

Firstly, we removed the CNN shown in Figure 3.3 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 [78]. We removed the CNN because of the low-dimension 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 (3.3.5) to generate more sparse reward – SEDN-Sparse. Finally, we constructed a hybrid environment with both static and dynamic obstacles shown in Figure 3.6 – SEDN-Hybrid. The corresponding training processes of these six baselines are shown in



(a) Success rate



(b) Collision rate

Figure 3.7: Training process of revised SEDN baselines. We omit SEDN-DDPG-Simple and SEDN-SAC-Simple because their success rates are zero at all times.

Figure 3.7.

As shown in Figure 3.7, 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

Table 3.1: 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 [30]	0.43	0.57
EEGO [55]	0.83	0.08
*RGL [54]	0.83	**0.03**
*CADRL [48]	0.94	0.06
*SARL [53]	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		

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.

3.4.3 Evaluation

We reproduced four open-source baselines, namely, ORCA [30], SARL [53], RGL [54], and CADRL [48]. These state-of-the-art baselines assume fully known information including human number, size, position, and speed. Moreover, ORCA is a non-learning multi-agent motion planner. The original ORCA algorithm is able to achieve an 100% success rate because each agent adopts same motion rules and all agents try to avoid each other. For a fair comparison, we assume the robot is invisible to humans, that is the robot actively avoids humans and humans do not avoid the robot. This is why the revised ORCA method frequently fails as shown in Table 3.1. In addition, we implemented another baseline – EEGO, directly using laser scans [55] similar to our observation space. We implemented 500 tests for each

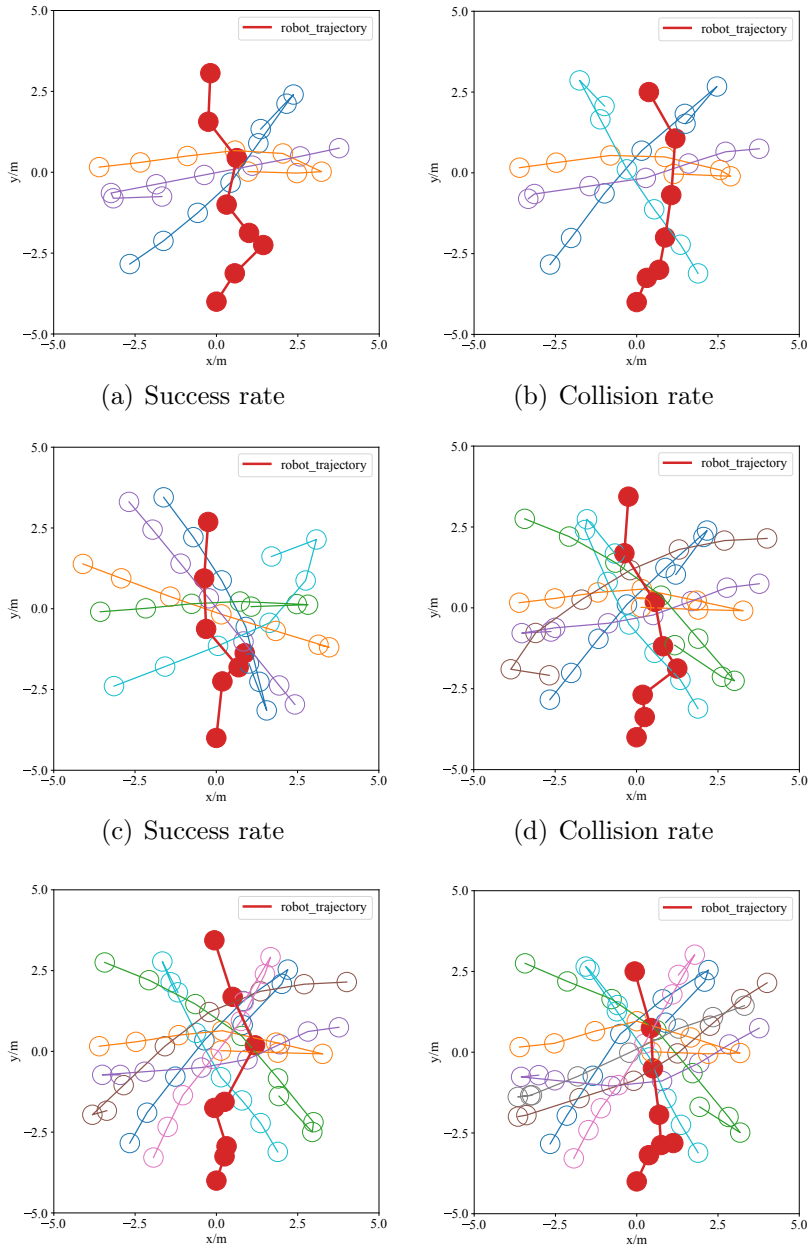
Table 3.2: Comparison with different policies deployed in various human scenarios. The two values in one unit represent the success and collision rates, respectively.

Policy	ORCA [30]	EEGO [55]	**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**

baseline, with the comparison results shown in Table 3.1. 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 3.1, 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 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[2]. The representative trajectories of humans and the robot are illustrated in Figure 3.8. We also compare our method with another 2 baselines that are independent of human numbers, with the results illustrated in Table 3.2. 500 validations were executed for each environment, and our method outperformed the baselines in terms of both the success and collision rate.

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



(a) Success rate (b) Collision rate

(c) Success rate (d) Collision rate

(e) Success rate (f) Collision rate

Figure 3.8: 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.

Moreover, we revised our model and environment settings to create another 6 baselines mentioned in Section IV-B, with the results shown in Table 3.1. For one thing, when we changed our DRL model such as simplifying the network structure (shown in Figure 3.9) and choosing an advanced RL algorithm – Soft Actor-Critic (SAC) (shown in Figure 3.10), the navigation performance was not improved. For another thing, when we dowm-sampled sensor data (shown in Figure 3.11), we could rarely

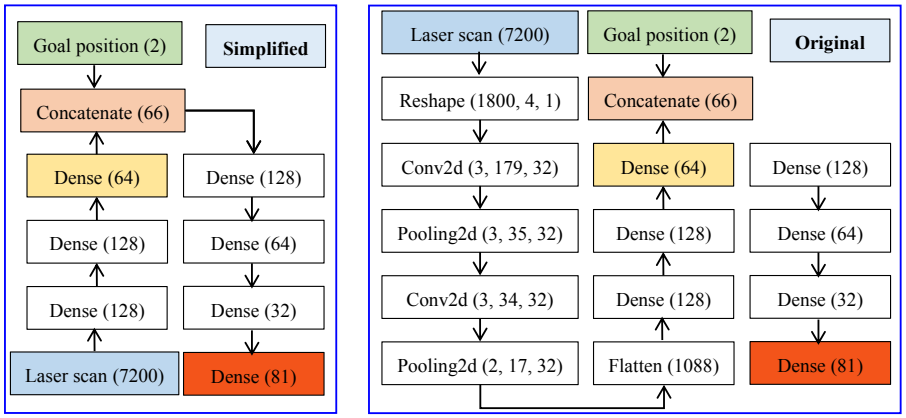


Figure 3.9: Simplified and original network structures. The left figure illustrates the simplified network structure whereas the right figure shows our original network structure with CNNs.

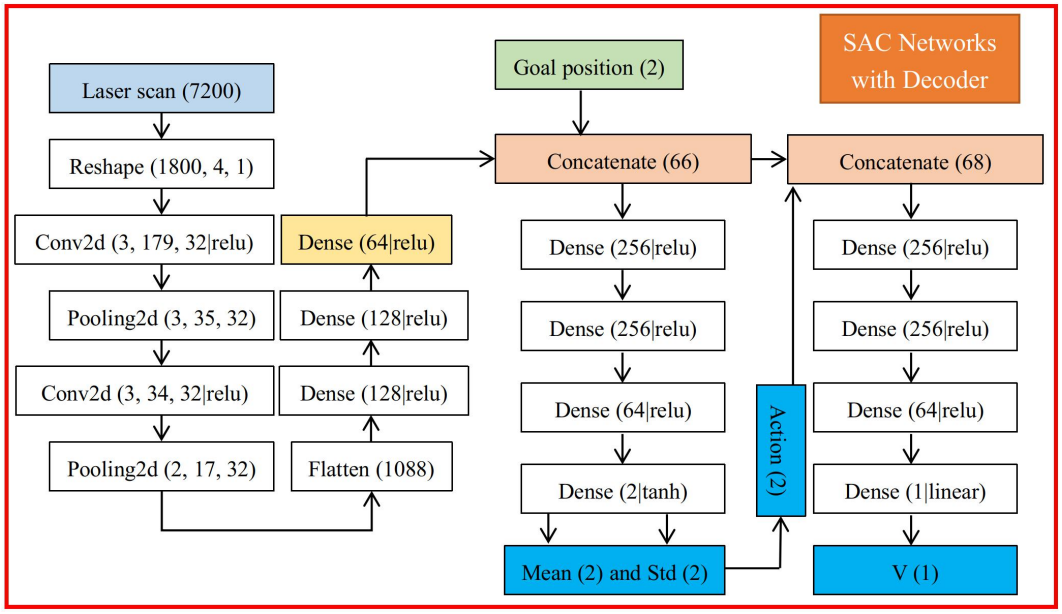


Figure 3.10: Network structure of SAC with a CNN decoder.

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.

3.4.4 Implementation in Physics based Environments

Because the maximum side speed of the omnidirectional quadrupedal robot (shown in Figure 3.4) 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

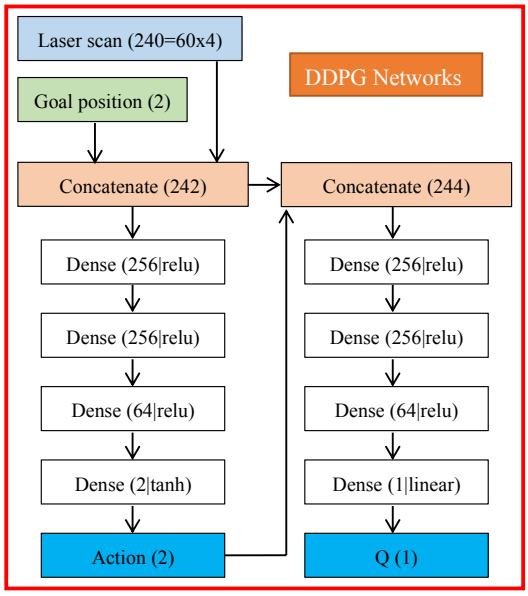


Figure 3.11: Network structure of DDPG with down-sampled sensor data.

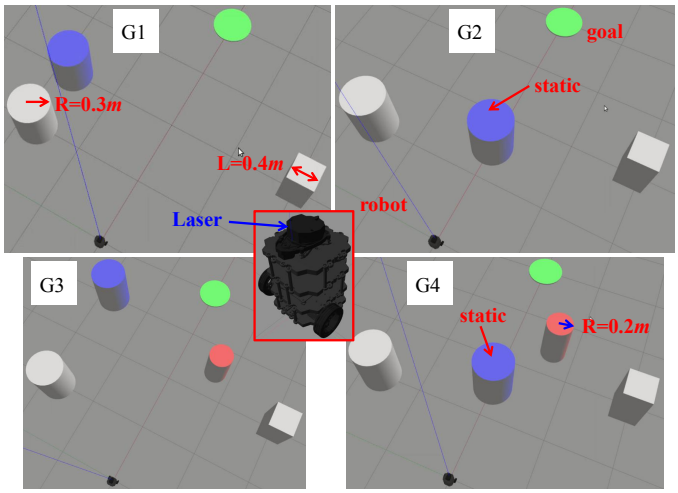


Figure 3.12: Gazebo scenarios with the motion area $4\times4m$. G1 has three dynamic objects, whereas one obstacle is static in G2. Adding one smaller dynamic object into G1 and G2 yields G3 and G4, respectively.

human number from 5 to 3 and initialize these humans around a $2m$-radius 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. 3.12 and 3.13 illus-

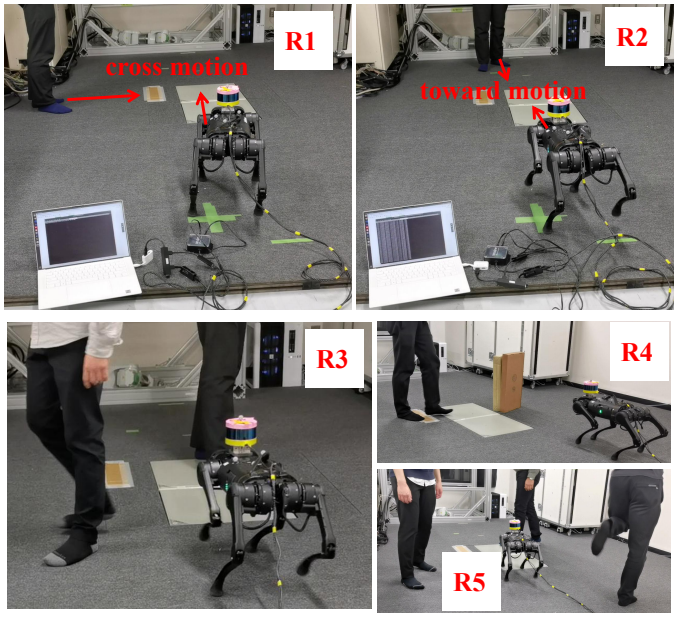


Figure 3.13: Real environments with $2.4 \times 3.4m$ accessible rectangle area. R1 and R2 have one human with a cross and toward motion, respectively. Two humans randomly walk in R3, while one dynamic human and one static obstacle are included in R4. R5 has three moving humans.

trate various Gazebo scenarios and real-world scenarios with diverse shapes, sizes, numbers, and motions of obstacles. 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 toward 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 Figure 3.4. 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[2]. 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

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

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.

3.5 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 non-holonomic 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.

Chapter 4

Navigation in Social Environments with Sequential Occupation Maps

This chapter incorporates material from the following accepted paper:

W. Zhu and M. Hayashibe, "Autonomous navigation system in pedestrian scenarios using a dreamer-based motion planner," *IEEE Robotics and Automation Letters*, accepted, 2023.

4.1 Introduction

Autonomous driving systems are becoming prevalent in human society because of their promising prospects of high efficiency, safety, and intelligence. Additionally, an aging society, labor shortages, and noncontact services during the pandemic promoted the research and development of autonomous mobile robots in hospitals, restaurants, hotels, etc. [2]. However, socially aware robot navigation is a highly complex task because it involves mapping and localization, human detection and behavior analysis, social rules, and decision and planning [6].

On one hand, the modules of perception and motion planning are separately studied by autonomous driving companies and research institutes [17]; thus, human–vehicle interaction is not considered. Moreover, the surrounding humans are individually detected and tracked, and their motions are independently analyzed; therefore, the reciprocal relationships among pedestrians are excluded. On the other hand, state-of-the-art algorithms proposed by academia assume that states such as human number, position, and speed are fully known and simply focus on motion planning, which

limits their generalizability and hinders sim-to-real transfer [10, 51, 52, 89].

In the earlier stages, rule-based motion planners played a dominant role in crowd navigation. Two pioneering approaches include optimal reciprocal collision avoidance (ORCA) [30] and the social force model (SFM) [90]. However, their one-step planning framework resulted in short-sighted, unsafe, and unnatural behaviors [53]. To plan motion over a long horizon, an intuitive strategy is to first predict human trajectories based on a model and subsequently select an optimal path for robots [91, 92]. Nevertheless, human motion models generally focus on individuals, ignoring the social relationships among pedestrians.

Recently, crowd navigation has switched to learning-based methods because of their prominent capability of representing the latent features of human–human and human–robot interactions and planning optimal navigation paths at a long time-scale. The collision avoidance with deep reinforcement learning (CADRL) algorithm [48] and its extension, socially aware CADRL (SA-CADRL), are the main algorithms used in the field of learning-based crowd navigation. These approaches feed all human states, such as positions and speeds, into deep neural networks (DNNs) to extract the implicit reciprocal motion features of humans, which are further fed into deep reinforcement learning (DRL)-based policy neural networks to learn an optimal motion planner. However, these two methods are not generalized to human numbers; thus, DNNs need to be redefined and retrained when the human number changes. Therefore, a subsequent algorithm, long short-term memory reinforcement learning (LSTM_RL), leverages LSTM to represent the relational motion features of humans to allow arbitrary human numbers [49, 50]. Because the inputs of LSTM neural networks are sorted by the distance between the robot and each human in descending order, the relationship among humans is not reciprocal.

To comprehensively describe the reciprocal relationship among pedestrians, the attention mechanism and graph convolutional network (GCN) are broadly embedded in DRL-based crowd navigation [51–54]. However, the aforementioned navigation algorithms assume fully known human states, including human number, position, and speed, and rely heavily on imitation learning and colossal positive datasets collected by rule-based methods such as ORCA, which may result in an early sub-optimum. Moreover, precisely tracking humans and estimating their speeds is difficult in the real world owing to various uncertainties; thus, sim-to-real transfer is a crucial challenge for these methods.

To let go of the assumption of fully known human information in simulations, directly using raw sensor data is a promising alternative [55–57, 61, 93]. The reciprocal

relationship among pedestrians is extracted from consecutive raw sensor data such as high-dimensional LiDAR scans with DNNs. However, direct handling of raw sensor data is inefficient. Consequently, imitation learning and colossal positive datasets are required to initialize DNNs. In addition, the lack of open-source solutions limits further development and comprehensive ablation.

Our study addresses these shortcomings in a comprehensive manner. First, we only detect humans and obtain their positions while excluding human tracking and speed estimation. After localizing the robot and humans, we create an RGB map that can be maturely processed using autoencoder algorithms to represent the instant relationship between humans and the robot. Accordingly, our algorithm can free the assumption of fully known environments and improve generalizability with respect to the human number and speed. In addition, inspired by the Dreamer approach, a model-based RL to address long-horizon tasks from images purely by latent imagination [37], we create a dynamic model with recurrent neural networks (RNNs) to accumulate history information and predict future states over a long horizon, thus reducing the probability of local optima. Moreover, the dynamic model can facilitate policy learning via the model-based DRL framework; thus, we can completely learn an optimal navigation policy without any imitation learning or a massive dataset. The contributions of this study are summarized as follows.

- A complete and publicly accessible autonomous navigation system among pedestrians is developed. We precisely obtain the robot pose using a LiDAR SLAM algorithm, and extract humans via a clustering approach. Moreover, we plan robot motion using a model-based DRL framework to avoid pedestrians and reach a target with a high success rate and navigation efficiency.

- We propose a Dreamer-based motion planning algorithm that can efficiently obtain an optimal motion planner and be generalized to arbitrary human number, variable human speed, and complex human relationships.

- We reproduce several state-of-the-art algorithms for more comprehensive ablation and ensure they are open-sourced. Additionally, sufficient sim-to-real experiments are implemented using domain randomization and system identification techniques.

The code of the whole project is publicly available at `https://github.com/zw199502/navigation_among_pedestrians` and the video is shown at `https://youtu.be/KM2WPpQBfrI`.

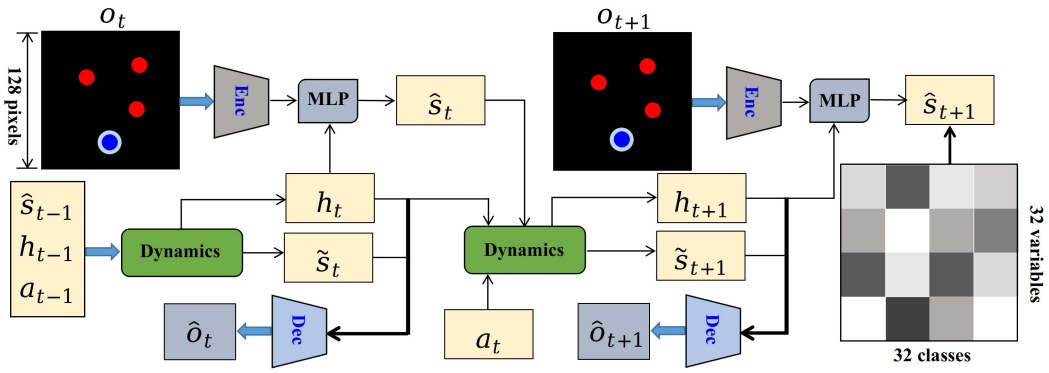


Figure 4.1: Framework of Dreamer-based motion planner with image observation. The multi-layer perceptron (MLP) is used for learning and inference. The encoder network (Enc) comprises convolutional neural networks (CNNs), while the decoder network (Dec) is constructed using transposed CNNs. To propagate historical information, recurrent neural networks (RNNs) are employed in the dynamics module.

4.2 Related Work

4.2.1 Navigation System

A complete navigation system integrates perception, decision, planning, and control modules, which are broadly researched and developed for autonomous driving vehicles [17, 18]. However, these modules are studied separately in both industry and academia. Although there are several mature and open-source solutions, such as Apollo[1] and Autoware[2], publicly accessible navigation systems among pedestrians are rare in the community of service robots. In this study, we construct a complete navigation system, in which the perception and control modules are derived from mature methods and decision and planning are achieved by a novel Dreamer-based motion planner.

4.2.2 Motion Planning among Pedestrians

In the early stage, rule-based motion planners constituted mainstream crowd navigation, such as ORCA [30], SFM [90], and trajectory-prediction-based path optimization [91, 92]. With the rapid development of deep learning, researchers and engineers are focusing on learning-based methods, wherein DRL-based navigation algorithms are attractive because of their promising representation and optimization capabilities [48–60]. CADRL [48] is a pioneering study in the use of DRL for so-

[1]https://github.com/chrislgarry/Apollo-11
[2]https://github.com/autowarefoundation/autoware

cial navigation. However, its value function neglects the social relationships among pedestrians, as it only considers the robot's full state and one pedestrian's observable state. LSTM_RL [49] improves upon CADRL by leveraging LSTM to represent pairs of the robot's state and all pedestrians' states, but its ability to capture reciprocal relationships is limited since pairs are ordered by distance and then fed into LSTM networks. Socially aware RL (SARL) [53] and relational graph learning (RGL) [54] represent state-of-the-art extensions to CADRL and LSTM_RL by using self-attention mechanisms and graph convolutional networks, respectively, to capture interactions and reason about relations between agents. However, these methods assume fully known pedestrian information and require massive positive datasets for learning, leading to degraded performance in real-time settings. EGO [55] and LSTM_EGO [61] offer a more direct mapping of raw sensor data to navigation actions, but open-source solutions for replicating their results are scarce. We propose an open-source Dreamer-based motion planner that can learn completely from zero experience and release the ideal assumptions of fully known environments.

4.2.3 Dreamer

The Dreamer algorithm is a reinforcement learning agent that addresses long-horizon tasks from images purely by latent imagination [37], which yields a large number of achievements in simulated environments, such as Atari games and MuJoCo robots [37–40]. Conversely, we focus more on real implementations of collision-free and socially aware robot navigation by leveraging the key idea of the Dreamer algorithm. A map is created to represent complex scenarios with variable human numbers and random initial states. A dynamic model with a map as a unique observation is learned to represent social relationships among humans. The learned model can facilitate learning complex behaviors, thereby enabling the robot to learn an optimal navigation policy without any prior experience.

4.3 Approach

We first illustrate the formulation of the problem of navigation among pedestrians with model-based RL and subsequently describe the details of the model creation and navigation policy learning using the Dreamer algorithm. In addition, we introduce a completely autonomous navigation system with perception, planning, and control modules. Figure 4.1 shows the overview of the Dreamer-based motion planner.

4.3.1 Problem Formulation

We formulate the problem of crowd navigation as a partially observable Markov decision process (POMDP) defined by a tuple $(\mathcal{S}, \mathcal{H}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \gamma)$, where $\mathcal{S}$ represents the stochastic state space, $\mathcal{H}$ is the deterministic history feature space, $\mathcal{A}$ denotes the action space, $\mathcal{P}$ is the state transition model, $\mathcal{R}$ represents the instant reward space, $\mathcal{O}$ is the observation space, and γ is a discount factor.

The unique observations are the sequenced RGB images $(o_t, o_{t+1}, \cdots, o_{t+K})$, each of which illustrates the instantaneous positions of humans and the robot. We decode the high-dimensional image $o_{t+\tau}$ into the stochastic latent state $s_{t+\tau}$ using considerably fewer variables. In addition, historical state information is accumulated as deterministic feature $h_{t+\tau}$. Given action $a_{t+\tau}$, stochastic state $s_{t+\tau}$, and hidden information $h_{t+\tau}$, the next state $s_{t+\tau+1}$ and accumulated feature $h_{t+\tau+1}$ can be derived from the state transition model $\mathcal{P}$. Given policy π, we can represent the expectation of the value function starting from state $s_{t+\tau}$ as follows:

$$v_\pi(s_{t+\tau}) = \mathbb{E}_\pi \left[\sum_{i=1}^{\infty} \gamma^{i-1} r_{t+\tau+i} \right], \tag{4.3.1}$$

where $r_{t+\tau+i}$ is the instant reward in state $s_{t+\tau+i}$. The goal is to determine policy π^* to maximize the value function.

4.3.2 World Model

Observation. Instead of directly using fully known pedestrian states, including the human number, position, and speed [48, 51–54], or clumsily dealing with raw sensor data [55, 56], we create a map using only the position information of humans and the robot. Therefore, we can reduce the uncertainties of human tracking, speed estimation, and trajectory prediction; generalize our method; and maturely process high-dimensional image observations via representation learning. Figure 4.2 depicts the image observation with the shape $(128, 128, 3)$. Let the size of the motion area be $L \times L$. Thus, the image resolution is $I_r = L/128$.

The image has three RGB channels, with the R channel representing humans and the B channel depicting the robot. We assume that the human is a circle with radius r_h, and the human circle is projected as deep red pixels with the value $p_r = 255$, as shown in Figure 4.2. The pixel value p_r is zero when the pixel is not occupied by a human. Similarly, the robot is assumed to be a circle with radius r_r, which is

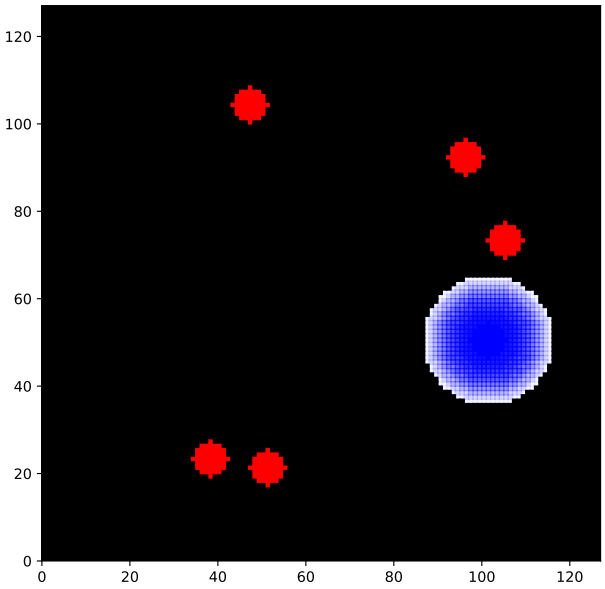


Figure 4.2: Image observation with the position information of humans and the robot.

represented by deep blue pixels with a value $p_b = 255$. By contrast, we inflate the robot circle to define an uncomfortable zone between the robot and humans. The blue pixel value p_b decreases when the pixel is far from the robot rim, and becomes zero when the pixel is outside the uncomfortable area. Note that the G channel is not used to represent the goal position in our current work because we fix the robot goal and randomize the robot's initial position, which is reasonable because the positional relation between the robot and the goal is relative.

Reward and discount factor. The reward function is defined according to the image observation. We first add the R and B channels together to yield a new 2D matrix m_a. If any value of m_a is equal to 510, a collision occurs, the reward is the minimum $r_c = -0.6$, and the discount factor γ is zero. In addition, the reward becomes $r_o = -0.1$ when the robot rim is outside the motion area, and γ remains zero. Another case of zero γ is when the goal distance d_g from the robot is less than the threshold d_r, that is, when the robot reaches the target. The reward is the maximum $r_g = 1.0$. If the robot does not collide with humans, reach the target, or stay outside the motion area, γ is $\bar{\gamma} = 0.99$, and the reward is a combination of the goal distance and the uncomfortable index. The maximum in m_a is m_a^{max} and the uncomfortable index is defined as $d_u = (m_a^{max} - 255)/255$. In summary, the reward

function and discount factor are defined as follows:

$$(r, \gamma) = \begin{cases} (-0.6, 0) & \text{if collision,} \\ (-0.1, 0) & \text{else if outside,} \\ (1.0, 0) & \text{else if reaching,} \\ (0.8 \cdot d_g/L - 0.6 \cdot d_u, 0.99) & \text{else.} \end{cases} \tag{4.3.2}$$

Action. We utilize a quadruped robot, which is an omnidirectional mobile robot. Accordingly, we define the action as two orthogonal velocities v_x and v_y. Additionally, v_x and v_y are continuous instead of discrete, as defined in state-of-the-art approaches [48, 51–54]; therefore, we can generate smoother motions in the physical world.

Autoencoder. We leverage autoencoder technology [94] to reduce the dimension of the image observation and extract the motion features of humans and the robot from sequenced observations. As illustrated in Figure 4.1, we first encode the image observation o_t with convolutional neural networks (CNNs) to extract the feature q_t which is an intermediate vector. Next, we project the combination of q_t and deterministic history information h_t onto the posterior latent state $\hat{s}_t$ with a multi-layer preceptor (MLP). Subsequently, we concatenate h_t and $\hat{s}_t$ and decode the concatenation to restore the observation image $\hat{o}_t$ with transposed CNNs. The autoencoder is summarized as follows:

$$q_t = E_\theta(o_t), \tag{4.3.3a}$$

$$\hat{s}_t \sim p_\theta^{\hat{s}}(q_t, h_t), \tag{4.3.3b}$$

$$\hat{o}_t \sim p_\theta^{\hat{o}}(\hat{s}_t, h_t), \tag{4.3.3c}$$

where θ denotes the weights of the world model network. The loss function of autoencoder can be derived from the likelihood probability represented as below:

$$\mathcal{L}_{AE}^{(t+\tau)} \doteq \log(o_{t+\tau}|\hat{o}_{t+\tau}). \tag{4.3.4}$$

Because motion planning is executed in the latent state space, while the reward r_t and discount factor γ_t are defined in the original observation space, we need another two networks composed by MLP to predict $\hat{r}_t$ and $\hat{\gamma}_t$ from the concatenated h_t and $\hat{s}_t$:

$$\hat{r}_t \sim p_\theta^{\hat{r}}(\hat{s}_t, h_t), \tag{4.3.5a}$$

$$\hat{\gamma}_t \sim p_\theta^{\hat{\gamma}}(\hat{s}_t, h_t). \tag{4.3.5b}$$

Similar to the loss function (4.3.4), another two additional loss functions can be obtained as follows:

$$\mathcal{L}_R^{(t+\tau)} \doteq \log(r_{t+\tau}|\hat{r}_{t+\tau}), \tag{4.3.6a}$$

$$\mathcal{L}_D^{(t+\tau)} \doteq \log(\gamma_{t+\tau}|\hat{\gamma}_{t+\tau}). \tag{4.3.6b}$$

State transition model. It is prohibitively challenging to construct a state transition model in the original observation space with high dimensions; therefore, we model the motions of humans and the robot in the latent state space. We utilize the categorical latent variables to represent the latent state with 32 classes multiplied by 32 variables (shown in Figure 4.1), based on the fact that categorical distributions can naturally capture multi-modal uncertainty of stochastic state transition [95]. Given the posterior latent state $\hat{s}_t$, action a_t, and history motion information h_t, the next latent state $\tilde{s}_{t+1}$ and the next hidden history information h_{t+1} can be predicted using a gated recurrent unit (GRU) neural network as shown in Figure 4.1:

$$h_{t+1} = f_\theta(\hat{s}_t, a_t, h_t), \tag{4.3.7a}$$

$$\tilde{s}_{t+1} \sim p_\theta^{\tilde{s}}(h_{t+1}). \tag{4.3.7b}$$

where f_θ which is composed by a GRU and $p_\theta^{\tilde{s}}$ constructed by a MLP correspond to the dynamics shown in Figure 4.1. The prior distribution $\tilde{s}_{t+\tau}$ is required to be similar to the posterior distribution $\hat{s}_{t+\tau}$ derived from the autoencoder model; thus, the fourth loss function can be defined as the Kullback–Leibler (KL) divergence:

$$\mathcal{L}_{KL}^{(t+\tau)} \doteq -\beta \mathrm{KL}(\hat{s}_{t+\tau}||\tilde{s}_{t+\tau}), \tag{4.3.8}$$

where β is a constant factor weighing the KL divergence loss.

Overall loss function. Given an episode obtained from the interaction between humans and the robot, we select a sequence starting from time step t and ending at $t+K$, where K is a constant. We fill zero to elongate the episode when its length is less than $K+1$ because of early collision, outside motion, or target reaching. The overall loss function along the sequence is represented as follows:

$$\mathcal{L}_{ALL} \doteq \mathbb{E}_{p_\theta}\left[\sum_{\tau=1}^{K}\left[\mathcal{L}_{AE}^{(t+\tau)} + \mathcal{L}_R^{(t+\tau)} + \mathcal{L}_D^{(t+\tau)} + \mathcal{L}_{KL}^{(t+\tau)}\right]\right]. \tag{4.3.9}$$

All world model networks, including (4.3.3a), (4.3.3b), (4.3.3c), (4.3.5a), (4.3.5b), (4.3.7a) and (4.3.7b), are jointly updated using this single overall loss function.

4.3.3 Motion Planner

The state transition model in compact latent space enables trajectory prediction in the long horizon without high-dimensional image observation, which results in a low memory footprint and speedy predictions of thousands of imagined trajectories in parallel [37]. As shown in Figure 4.1, starting from the latent state $\hat{s}_{t+\tau}$ and history information $h_{t+\tau}$, a considerable number of episodes with H horizon can be swiftly generated. Consequently, we can efficiently leverage imagined episodes to optimize the navigation policy.

For the imagination process, we create an actor network and a critic network using MLP to map the current latent state and historic motion information into the action and value function, respectively:

$$\bar{a}_{t+\tau} \sim p^{\bar{a}}_{\phi}(\bar{s}_{t+\tau}, \bar{h}_{t+\tau}), \tag{4.3.10a}$$

$$\bar{v}_{t+\tau} \sim p^{\bar{v}}_{\psi}(\bar{s}_{t+\tau}, \bar{h}_{t+\tau}), \tag{4.3.10b}$$

where ϕ and ψ denote the weights of the actor and critic networks, respectively. Additionally, the reward $\bar{r}_{t+\tau}$ and discount factor $\bar{\gamma}_{t+\tau}$ are predicted using (4.3.5a) and (4.3.5b), respectively. With the imagined episode having a long horizon, we can evaluate the value function with a multi-step RL framework because it yields a better unbiased estimation than the one-step RL algorithm [96]:

$$\begin{aligned} v_i(s_\kappa) &\doteq \mathbb{E}_{(p^{\hat{r}}_{\theta}, p^{\hat{\gamma}}_{\theta}, p^{\hat{s}}_{\theta})} \left[\sum_{n=\kappa}^{h-1} (\bar{\gamma}_n^{n-\kappa} \bar{r}_n) + \bar{\gamma}_h^{h-\kappa} \bar{v}_\psi(s_h) \right], \\ & i = 1, 2, \cdots, H, \\ & h = \min(\kappa + i, t + \tau + H), \\ v_\lambda(s_\kappa) &\doteq (1-\lambda) \sum_{n=1}^{H-1} (\lambda^{n-1} v_n(s_\kappa)) + \lambda^{H-1} v_H(s_\kappa), \end{aligned} \tag{4.3.11}$$

where λ is another discount factor that weighs the value function, and κ ranges from $t+\tau$ to $t+\tau+H$. We have two objectives: to determine a policy to maximize the value function, and to minimize the error between the estimated value function and the predicted value from the critic network. Therefore, the weights of the critic and actor networks can be updated as follows:

$$\min_{\psi} \mathbb{E}_{(p_\theta, p_\phi)} \left(\sum_{\kappa=t+\tau}^{t+\tau+H} \frac{1}{2} \| \bar{v}_\kappa^{mean} - v_\lambda(s_\kappa) \| \right), \tag{4.3.12a}$$

$$\max_{\phi} \mathbb{E}_{(p_\theta, p_\phi)} \left(\sum_{\kappa=t+\tau}^{t+\tau+H} v_\lambda(s_\kappa) \right), \tag{4.3.12b}$$

where $\bar{v}_n^{mean}$ is the mean of the distribution $\bar{v}_\kappa$.

4.3.4 Algorithm Summary

Algorithm 2: Dreamer-based Motion Planner

$T \leftarrow 0$;
Initialize experience buffer $\mathcal{D}$ with random episodes;
Update T according to the buffer size;
Initialize weights, θ, ψ and ϕ via pre-training;
while $T < T_{max}$ **do**
  $t \leftarrow 0$, $\gamma_t \leftarrow \bar{\gamma}$, $r_t \leftarrow 0$, $h_t \leftarrow \mathbf{0}$, $a_t \leftarrow \mathbf{0}$;
  $o_t \leftarrow env.reset()$;
  $episode \leftarrow [(o_t, r_t, a_t, \gamma_t)]$;
  while $t < t_{max}$ ***and*** $\gamma_t > 0$ **do**
    $q_t = E_\theta(o_t)$;
    $s_t \sim p^s_\theta(q_t, h_t)$;
    $a_{t+1} \sim p^a_\phi(s_t, h_t)$;
    $o_{t+1}, r_{t+1}, \gamma_{t+1} \leftarrow env.step(a_{t+1})$;
    $h_{t+1} \leftarrow f_\theta(s_t, a_t, h_t)$;
    $episode.append([(o_{t+1}, r_{t+1}, a_{t+1}, \gamma_{t+1})])$;
    $t \leftarrow t + 1$;
    Select B batches of sequenced data from $\mathcal{D}$;
    Fill $\mathbf{0}$ until $\mathcal{B} = \{(o_\tau, r_\tau, a_\tau, \gamma_\tau)\}_{\tau_0}^{\tau_0+K}$;
    Update the world model with the batches;
    Imagine episodes at H horizon from (s_τ, h_τ);
    Update the motion planner with the episodes;
  end
  Add *episode* to $\mathcal{D}$;
  $T \leftarrow T + t$;
end

In the simulation, we assume that the human motion is generated by ORCA [30]. Additionally, the robot is assumed to be invisible to humans; otherwise, it will be difficult to differentiate whether our motion planner is effective or whether humans avoid the robot. The simulation is *reset* when collision, outside motion, target reaching, or timeout occurs. When the episode length is greater than t_{max}, it is terminated as a timeout. We use the *step* function to update the positions of humans and the robot and obtain the instant reward and discount factor. Simulation interaction samples are collected to update the world model. We subsequently utilize the world model to imagine episodes in the latent space, which are used to update the motion planner. Next, a motion planner is used to generate new simulation episodes. We alternately update the world model and motion planner until a stable navigation policy is acquired. This algorithm is summarized in Algorithm 2.

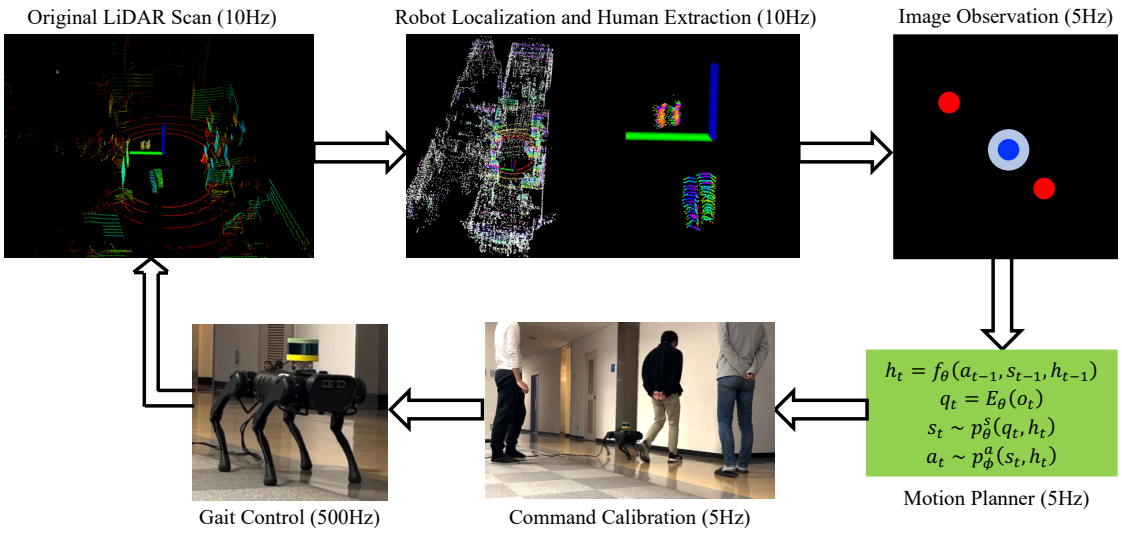


Figure 4.3: Complete autonomous navigation system. The robot localization and human extraction are executed at 10Hz due to the limitation of sensoring frequency. We plan the motion at the speed level every 0.2s while the level of robot gait generation corresponding to the desired speed is run at 500Hz.

4.3.5 Complete Navigation System

As stated in existing studies, wheeled mobile robots can localize themselves with the wheel odometer [48, 51–54], which however drifts heavily with the increase in motion time. External motion capture systems are commonly used to precisely obtain the robot position and orientation [50]. Nevertheless, such systems are expensive and impractical in a real human society. A navigation system without any accurate and internal localization module is incomplete and cannot be applied to society. Additionally, these studies have not focused on publicly accessible real-world implementations, such as human detection, speed estimation, and trajectory prediction. Conversely, we leverage the LiDAR odometry and mapping (LOAM) SLAM algorithm [97] to accurately localize the robot. In addition, we extract humans using a clustering approach [98]. Because we can obtain the sequenced position information of humans and the robot, we can extract the latent motion feature using our algorithm without individually estimating the human speed and future trajectory. Because our algorithm directly outputs two orthogonal velocities v_x and v_y, we need to further match v_x and v_y to the speed command of the quadruped robot[3] used for practical implementations. 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 [70]. We made our system (shown in Figure 4.3) open-source for easier deployment in the autonomous navigation community.

[3]https://m.unitree.com/products/a1/

4.4 Experiments

We used three simulation scenarios: Simulation one is for a comprehensive comparison, Simulation two is designed to verify the generalizability of our method, and Simulation three is constructed for sim-to-real transfer. Subsequently, we directly deployed the policy learned in Simulation three into various real scenarios without any retraining or fine-tuning.

4.4.1 Simulation

Training for comparison. The side length of the motion area is $L = 10$m. Similar to the original settings of RGL [54], we assume a human number fixed at 5, human radius $r_h = 0.3$m, and each human is randomly initialized around a circle with a radius of 4m. The initial position is $(x_0^{(i)}, y_0^{(i)})$, and the goal is $(-x_0^{(i)}, -y_0^{(i)})$. The i-th human moved back and forth from these two positions with a preference speed of 1m/s. Additionally, reciprocal motion among humans is generated by ORCA [30] with the robot being invisible. The robot radius was $r_r = 0.3$m and the maximum values of v_x and v_y were both 1m/s. For a fair comparison, we assumed that the robot moves from (-4.0, 0.0)m to (4.0, 0.0)m. We reproduced 9 popular baselines: the first is A-Star algorithm for relatively static environments [21], one is Dynamic Window Approach (DWA) [23] which is maturely used for service robots, another non-learning method is called ORCA [30], whereas the other 6 are learning-based approaches, called CADRL [48], LSTM_RL [49], SARL [53], RGL [54], EGO [55], and LSTM_EGO [61] respectively.

A-Star is broadly used for path planning in relatively static environments. We select the waypoints on the planned path to guide the robot motion. Because of the highly dynamic environments, A-Star's success rate is 42% (shown in Table 4.1) although we fine-tune parameters. DWA generally assumes that obstacles are static at each planning step. In this study, we extend it to adapt to dynamic environments. More specifically, we calculate the planning cost along multiple time steps in the future and obstacles' positions are predicted based on their velocities at present time step. The modified DWA method yields a high success rate (95%) shown in Table 4.1. However, it still causes failures because it assumes constant velocity for prediction. As another none-learning approach, ORCA assumes that the agent's states, including shape, size, position, and speed, are fully known. Based on this information, it generates an optimal collision-free action in one step. However, this one-step planning approach may result in short-sighted, unsafe, and unnatural

behaviors. For a fair comparison, the robot is invisible to humans; otherwise, it is difficult to differentiate whether the robot tries to avoid humans or humans actively avoid the robot. This revision results in frequent failures (shown in Table 4.1) because the reciprocal assumption is violated.

In contrast, DRL-based crowd navigation algorithms, which utilize a value function that can represent accumulated return over a long horizon, have become increasingly popular due to their ability to address these issues. For example, CADRL was a pioneering study in this area, but its value function only considered the pair of the robot and one human, making it unable to represent relational interactions among humans. As a result, LSTM_RL was developed to pair the robot with all humans, but it still only captures partial interactions because it sorts the pairs by distance before feeding them into the LSTM networks. To more comprehensively represent social interactions among humans, SARL utilizes a self-attention mechanism to capture interactions within pedestrians, while RGL embeds a graph convolutional network to reason about relations between agents and compute interactions between them. However, these methods all require fully known human states. In contrast, EGO and LSTM_EGO can handle both static and dynamic obstacles of different shapes, sizes, and numbers, as they directly map raw sensor data to navigation actions. EGO uses continuous LiDAR scans, while LSTM_EGO uses one frame of LiDAR scan and embeds LSTM to deal with sequential scans. However, a downside of these methods is the difficulty of efficiently learning a feasible navigation policy. While the other four have open-source solutions, publicly accessible resources for EGO and LSTM_EGO are rare. Therefore, we specially created a simulator that could generate 2D LiDAR scans and constructed neural networks for policy learning. We named our method navigation among pedestrians with a Dreamer-based motion planner (NPD). The learning processes of the six methods are illustrated in Figure 4.4, and the final evaluation is shown in Table 4.1. Please note that Figure 4.4 displays the success rate of 100 evaluation episodes. Each episode may result in success, collision, overtime, or outside motion. Our analysis revealed that when the success rate was at or above 90%, the number of collision cases decreased to less than 5, sometimes even to 1, whereas most of the cases were overtime. Notably, in overtime cases, the robot was very close to the target. We hypothesized that the image resolution may have contributed to this phenomenon. As the robot was approaching the target, we reclassified the overtime cases as success in the final 500 tests.

Because baselines CADRL, LSTM_RL, SARL, and RGL initialized the neural networks with imitation learning and fill the experience pool with a large number of positive samples ahead of the training, they could swiftly learn a feasible navi-

Table 4.1: Final Evaluation. 500 random tests are executed with the best neural networks saved during the training.

Method	SR	NT (s)	AT (s)
A-Star	0.42	15.85	5.0e-4
*DWA	0.95	11.94	2.6e-3
*ORCA-Ideal	0.92	12.32	**5e-5**
*ORCA	0.47	10.83	**5e-5**
*CADRL	0.80	12.40	0.035
*LSTM_RL	0.97	10.95	0.067
*SARL	0.98	**10.75**	0.060
*RGL	0.97	11.22	0.067
EGO	0.54	12.67	1.5e-3
LSTM_EGO	0.52	15.08	7.5e-4
NPD(ours)	**0.99**	11.15	3.3e-3

*These methods require fully known human information.
SR: success rate; NT: navigation time; AT: action time

Table 4.2: Test results with fewer positive samples.

Method	Success Rate	average time (s)
CADRL	0.80	12.40
CADRL*	0.59	12.00
LSTM_RL	0.97	10.95
LSTM_RL*	0.98	10.67
SARL	0.98	10.75
SARL*	0.99	10.66
RGL	0.97	11.22
RGL*	0.52	15.28

*Fewer positive samples

gation policy. Although EGO's initialization was same, its learning was notably unstable and it could only reach a success rate of 0.54. The initial data settings of LSTM_EGO were same as ours, however, its success rate was 0.52, significantly lower than our method's. In addition, LSTM_EGO required the longest navigation time, averaging 15.08s. We found that the success rates of all the baselines which require prior initialization became zero at all times when we removed imitation learning and the massive positive dataset. Therefore, we excluded the training results of these baselines without positive samples. Nevertheless, we present the results of these baselines with fewer positive samples (50 episodes) in Figure 4.5 and Table 4.2. The results show that reducing positive samples does not significantly affect

Table 4.3: Quantitative analysis when adding uniform noises to the human action originally generated by ORCA.

Method/Noise	0.05	0.10	0.15	0.20
RGL	0.98	0.98	0.97	0.97
NPD(ours)	0.96	0.95	0.97	0.94

the learning performance of LSTM_RL and SARL. However, the learning efficiency of CADRL and RGL deteriorates with the reduction of positive samples.

Comparatively, our method could completely learn from zero experience and yielded a stable convergence and high success rate, which indicates that our method does improve learning efficiency. As shown in Table 4.1, our approach quantitatively either outperformed or behaved similar as the other six learning baselines with respect to both the success rate and average navigation time. We found that the failure of our method initially occurred when two of the humans closely surrounded the robot. Additionally, the collision occurred within five time steps of 1 second. We believe that this short sequence results in the most of the remaining 1% failures. Although ORCA requires a short navigation time, it produced the lowest success rate, whereas our method was able to adequately balance the navigation time and success rate. Because our final goal is to deploy the navigation policy on real robot platforms, real-time performance should be another evaluation factor. The time used to generate an action when given observations is referred to action time. ORCA's action time was only tens of microseconds because ORCA is a non-learning approach. However, CADRL, LSTM_RL, SARL, and RGL required tens of milliseconds to derive an action because they only have a value network and need to inquire each action choice to obtain the best one. On the contrary, EGO, LSTM_EGO, and NPD(ours) have both an actor network and a critic network, therefore, they are able to quickly access an optimal action from the actor network. Because the planning frequency in the real world is 5Hz, our method's action time (3.3e-3) is acceptable for real implementations. The aforementioned test assumes that human motions follow ORCA rules. To verify the generalizability of the trained policy with respect to human motion modals, we deviated the human action originally generated using ORCA by adding uniform noises. Additionally, we used RGL as a benchmark for comparison, and the results are presented in Table 4.3. Since the maximum speed of humans is 1.0m/s, we added uniform noises with the bounds of 0.05, 0.1, 0.15, and 0.2, respectively. We tested each case 500 times and observed that RGL outperformed our method slightly. However, our method demonstrated the ability to handle deviated human motions. Additionally, we found an interesting result that the performance with higher noise (0.15) was slightly better than that with lower noise (0.05), which

could be a topic of future exploration.

Training for generalizability verification. we trained our model in a more challenging navigation scenario (shown in Figure 4.6) to further verify the generalizability of our approach. Similar to the scenario shown in Figure 4.2, the motion area is 10×10m. Differently, the obstacle number in the revised environment is up to 7, which makes the training scenario denser and more complicated. Moreover, the number of moving humans changes from 1 to 4. Additionally, we add rectangular static obstacles whose number is variable from 1 to 3 and side length ranges from 0.3m to 0.4m. Baselines CADRL, LSTM_RL, SARL, and RGL assume that the obstacles are circles and their number is fixed during the whole training, whereas baselines EGO and LSTM_EGO are independent of obstacle number and shape. We choose LSTM_EGO as the ablation simply because it is a more recent study. The training process is depicted in Figure 4.7. Although the environment becomes complex, our method can still reach a 93% success rate, significantly outperforming LSTM_EGO whose success rate is below 40% at all times.

Training for policy transfer.

To enable sim-to-real transfer, we leveraged the domain randomization technique to improve the generalizability of the simulated navigation policy. Because the maximum sideward speed of the real quadruped robot was 0.27m/s, we first constrain the maximum v_y as 0.27m/s and the maximum v_x 0.3m/s. Note that we did not enlarge the forward speed v_x for slow and stable motion in our small real scenarios with a size 3×3m. Different from the simulation configurations considered for comparison, we narrowed the motion area to $L = 6$m, and the human number changed from 1 to 3. Additionally, we randomized the initial robot position while keeping the goal fixed at $(1.0, 0.0)$. Moreover, the initial human position was randomized over the entire motion area, while the human goal was distributed around the margin of the motion area. The preference speed of humans ranges from 0.15m/s to 0.3m/s. For RL-based baselines, obtaining a feasible navigation policy is challenging if the environment significantly changes. Moreover, certain baselines, such as CADRL and SARL, do not allow variable human numbers during training. Although we introduced a large number of stochastics, our algorithm could produce stable policy optimization while LSTM_EGO failed in obtaining a feasible navigation policy, as shown in Figure 4.9. The trained policy achieved a 95% success rate in the final evaluation with 500 random settings. Figure 4.8 shows six representative episodes with different human numbers and complex human motions.

In addition to comparing with the learning-based baselines LSTM_EGO and RGL,

we conducted ORCA as another ablation. We found that ORCA could significantly improve navigation success rate from 0.47 to 0.98 when human number was decreased from 5 to 3. Although ORCA (0.98) outperformed our method (0.95) in simulation, we found that the robot was inclined to move outside of the specific area in real scenarios. The possible reason may be inaccurate human tracking and speed estimation. Conversely, our method only needs human positions, without considering the errors of human tracking and speed estimation, therefore, our approach is able to proficiently deal with various real-world scenarios.

To align the robot collision margin with the actual robot platform, we replaced the inflated circular margin with a rectangular one (shown in Figure 4.10). This modification enabled us to match the collision margin of the simulated robot with that of the real robot platform. The collision margin has a length and width of 0.5m and 0.3m, respectively, identical to the collision margin of our real robot platform. The corresponding learning is illustrated in Figure 4.9, which indicates that our method can deal with different robot collision margins.

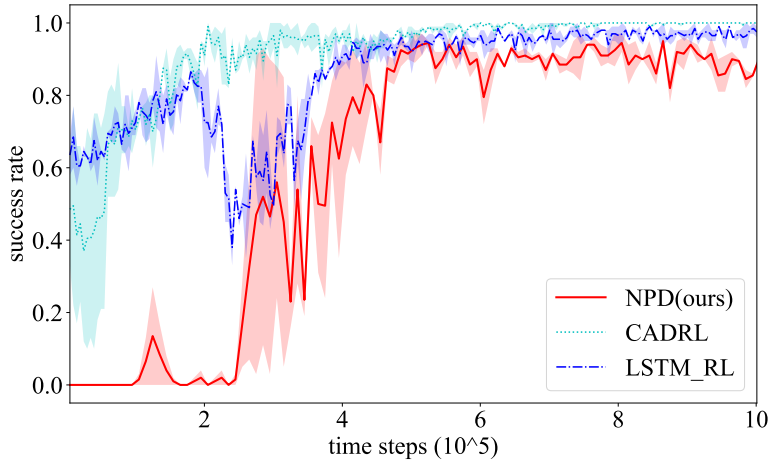
4.4.2 Real Implementations

We deployed the policy learned from the simulation on a quadruped robotic platform equipped with a Velodyne VLP-16 LiDAR. The tests are shown in the attached videos. Although the human number changed from 1 to 3 and human motions are diversified, our simulated navigation policy could be directly transferred into real scenarios without any retraining or fine-tuning, which shows the potential of our method to model complex reciprocal human relations and navigate robots among pedestrians in the real world.

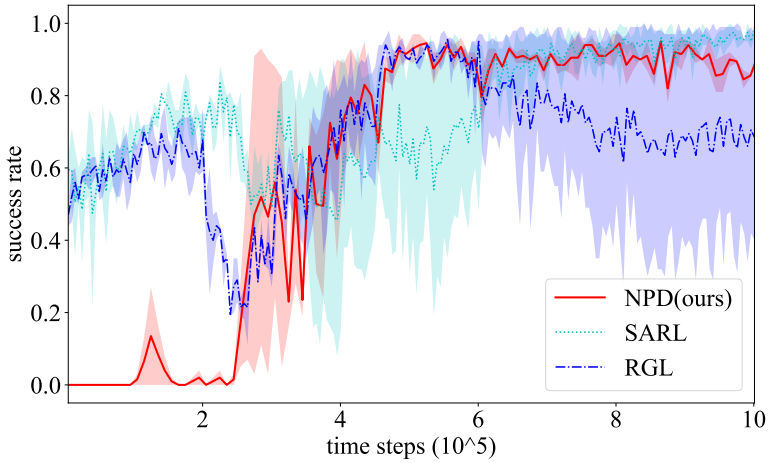
4.5 Conclusions

This paper presented an autonomous navigation system with a Dreamer-based motion planner. We let go of the assumption of fully known human states and only utilized the human position information. The human positions and robot location were projected onto a image. From the sequenced image observations, we extracted reciprocal relationships among pedestrians through representation learning. In addition, we created a state transition model using the extracted latent information to imagine episodes for reinforcement learning. Sufficient simulation ablations demonstrated that our method could learn from zero experience with high efficiency and

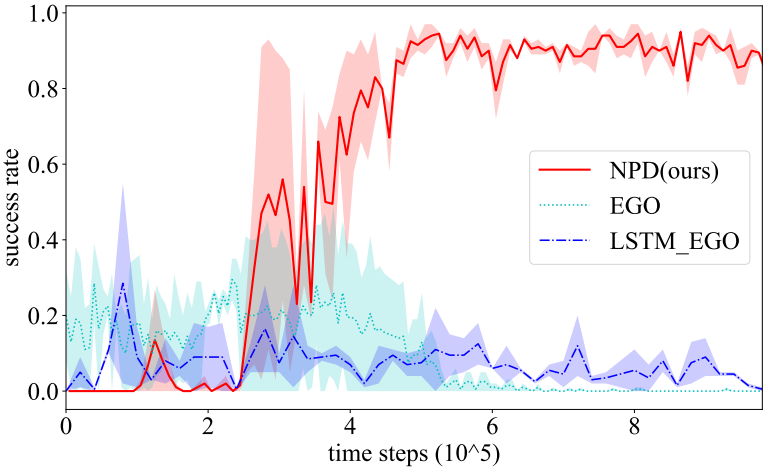
outperformed state-of-the-art algorithms. In addition, we leveraged the techniques of system identification, domain randomization, clustering, and LiDAR SLAM to enable sim-to-real transfer. Adequate real implementations illustrated the potential of our method to model complex reciprocal human relations and navigate the robot among pedestrians in the physical world. Our future study will focus on accurate human detection, precise robot localization, and universal navigation policy.



(a) Group one: CADRL and LSTM_RL

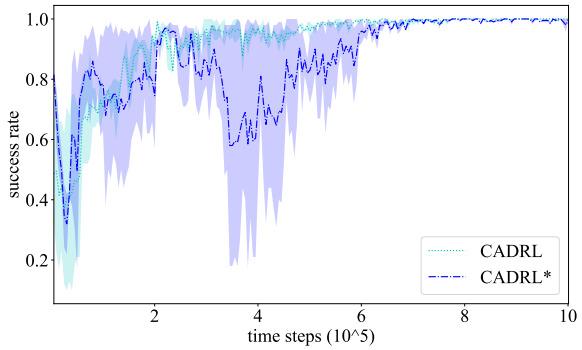


(b) Group two: SARL and RGL

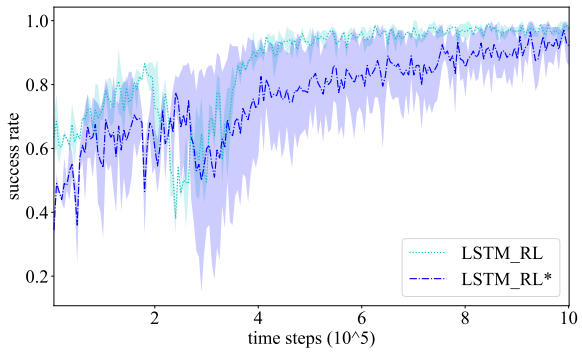


(c) Group three: EGO and LSTM_EGO

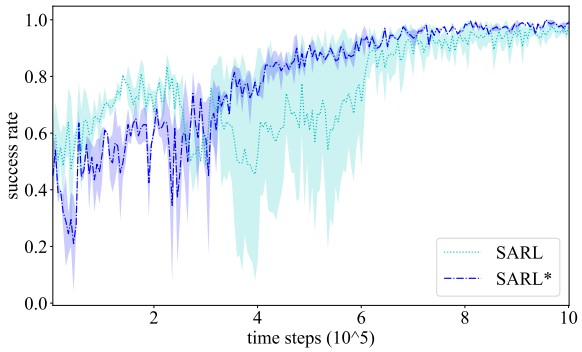
Figure 4.4: Learning ablations. At certain time step, we evaluate the policy 100 times and calculate the corresponding success rate of collision-free and target-reaching navigation. Group one include two baselines, CADRL and LSTM_RL, which have a stable learning process while the training of another two baselines in Group two, SARL and RGL, vibrates intensely. EGO and LSTM_EGO in Group three can not reach a high success rate of crowd navigation and their learning is extremely unstable.



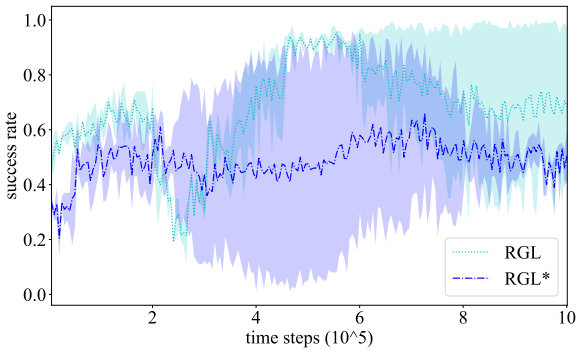
(a) CADRL



(b) LSTM_RL

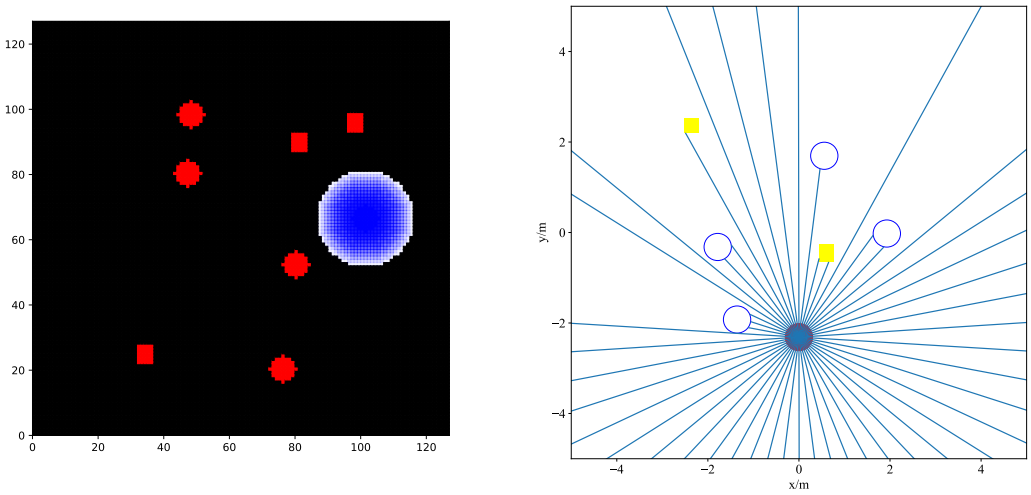


(c) SARL



(d) RGL

Figure 4.5: Baselines with fewer positive samples. METHOD has 2000 positive episodes whereas the experience pool of METHOD* is initialized by 50 positive episodes.



(a) NPD environment (b) LSTM_EGO environment

Figure 4.6: Complex environments. (a) The red circles represent moving humans whose number changes from 1 to 4 whereas the rectangles stand for static obstacles whose number is variable from 1 to 3 and side length ranges from 0.3m to 0.4m. The robot with an inflation area is illustrated by the blue circle. (b) The number and shape of moving humans and static obstacles are same as those illustrated in (a). We specially designed a LiDAR scan environment to train LSTM_EGO, where the LiDAR scan is composed by 1800 beams to comprehensively detect surrounding objects.

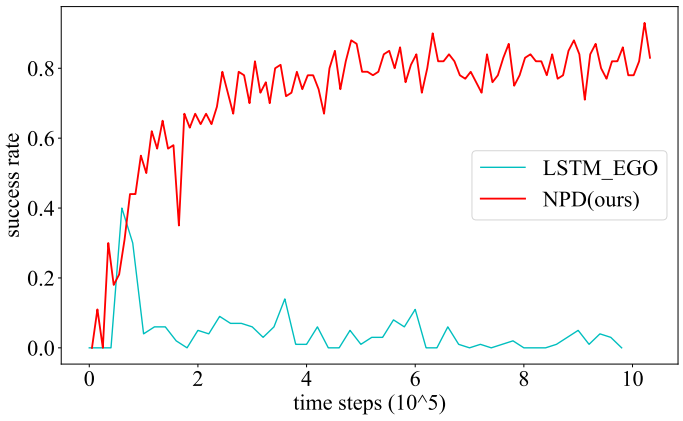
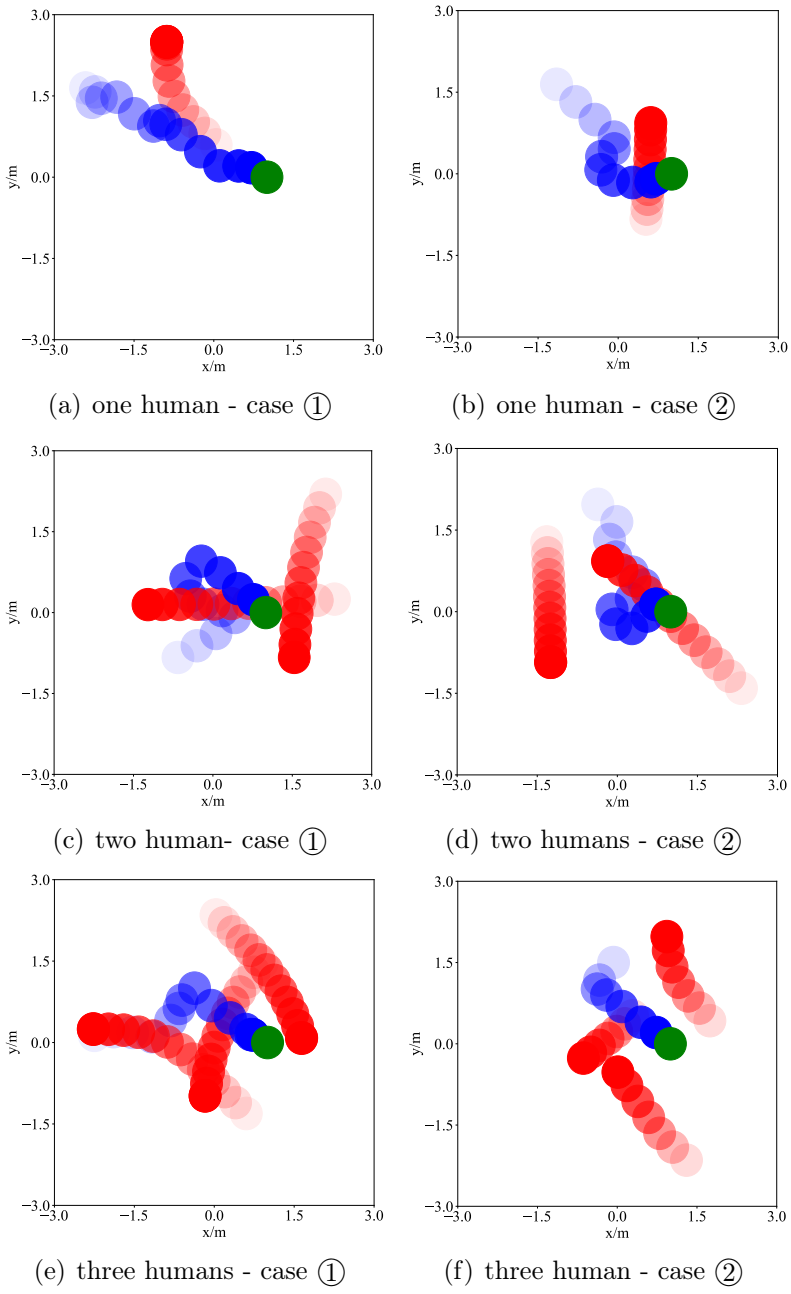


Figure 4.7: Training in complex environments.



(a) one human - case ①

(b) one human - case ②

(c) two human- case ①

(d) two humans - case ②

(e) three humans - case ①

(f) three human - case ②

Figure 4.8: Trajectory visualization. The red circle represents the human, the blue is the robot, and the green stands for the goal. The object moves at the direction that the circle opacity increases. The top two figures shows the scenarios with one human, the middle two humans, and the bottom three humans.

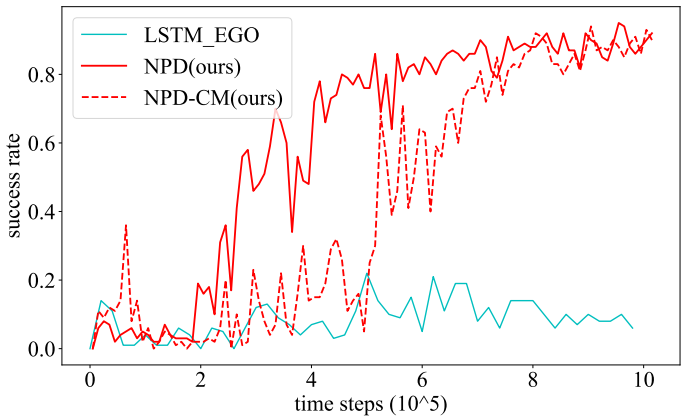


Figure 4.9: Learning process with stochastic configurations in simulation. Different from the training for comparison, this training is executed in stochastic environments with variable human numbers and distributions to learn a more generalized navigation policy. NPD considers the robot's collision margin as a circular shape, whereas NPD-CM uses a more accurate collision margin that is a circumscribed rectangle around the physical robot. RGL's learning is omitted because its success rate is zero at all times.

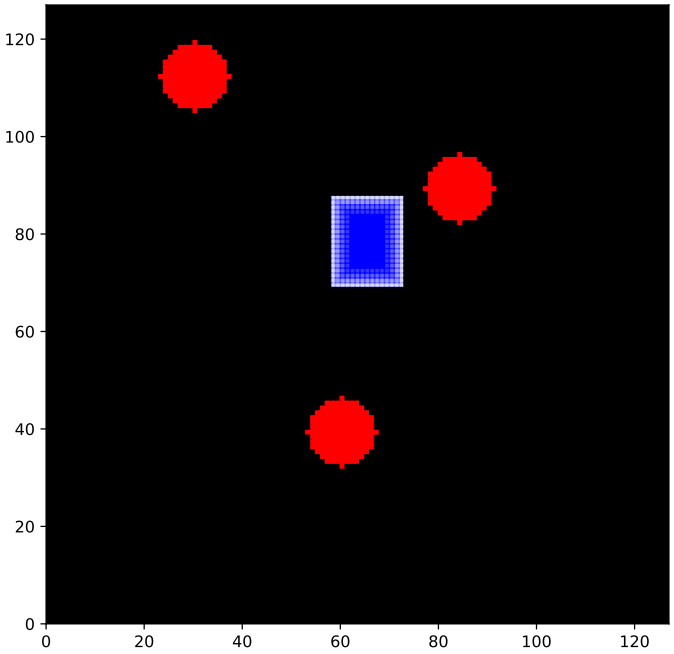


Figure 4.10: Robot collision margin is a rectangle whereas humans are represented by circles.

Chapter 5

Navigation in Social Environments with Sequential LiDAR Scans

This chapter incorporates material from the following paper which is under review:

W. Zhu and M. Hayashibe, "Learn to navigate in dynamic environments with normalized LiDAR scans," under review.

5.1 Introduction

Self-driving cars have garnered significant interest and research over the years, and recent advances in machine learning, artificial intelligence, and computer vision have resulted in highly sophisticated autonomous driving vehicle technology. Tech giants, including IT and car manufacturers, have invested substantial resources in developing self-driving car prototypes and testing them on public roads [99]. These vehicles hold the potential to offer significant benefits such as improved safety, reduced traffic congestion, increased energy efficiency, and greater mobility for individuals who cannot drive. However, concerns surrounding the safety and security of self-driving cars, as well as their impact on employment in the transportation sector, remain a point of contention. Although outdoor self-driving cars have yet to be produced on a massive scale, service mobile robots are already making their presence felt in closed environments within 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 [100]. However, safely

and efficiently navigating service mobile robots in human-rich and highly dynamic environments remains an exceedingly difficult task.

Map-based navigation strategies have been widely utilized for mobile robots in stable environments [101, 102]. Various studies have employed high-resolution maps to search for global paths via efficient planning algorithms, such as A-Star (A*) [103] and rapidly-exploring random tree (RRT) [104]. Local motion planners, such as dynamic window approach (DWA) [23] and directional approach [105], 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 can be 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 DRL navigation methods have become increasingly popular in the research literature [106]. However, early studies [4, 41, 43, 44, 46, 78–80] 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 [48, 49, 52–54, 74]. 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 [55, 56, 61, 75–77]. Nevertheless, commonly used simulators which can generate raw sensor observations can not significantly accelerate running, such as Gazebo [107] and V-REP [108]. 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 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 fixed circle shape, and static obstacles are rectangles with variable sizes. However, 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 with a fixed radius 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 [55] or high-dimensional depth images [61] which require colossal networks to extract latent features, 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. We name our approach Learn to Navigate in Dynamic environments with Normalized LiDAR 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 point-clouds. Additionally, we re-generate 2D LiDAR scans from the normalized obstacles to be consistent with simulated settings.

- 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 extensively compared it with state-of-the-art baselines. Additionally, we conducted extensive real experiments to highlight the potential of sim-to-real transfer.

Our project is open-sourced at `https://github.com/zw199502/LSTM_EGO` and attached videos are shown at `https://youtu.be/Eiyp8V8EjWo`.

5.2 Related Work

5.2.1 Robot Simulators

In order to obtain a significant number of samples for Deep Reinforcement Learning (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 [109], its extensions such as DMC [110] and Robomimic [111], 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 [107] and V-REP [108] 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 [112] and iGibson [113] 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.

5.2.2 Robot Navigation in Stable Environments

Map-based navigation techniques have found widespread application in various industrial scenarios, such as robotics logistics in hotels [114] and restaurants [13]. 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 [4, 41, 43, 44, 46, 78–80]. 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.

5.2.3 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 [14]. However, predicting trajectories is a complex task as it involves object tracking and intricate interactions between moving objects. 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 [115], can be time-consuming.

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 [96]. 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 [48] 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 [49] 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 [53] and graph convolutional networks (GCNs) [54] 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 [55,56,117], sequential images [61,118], or multi-sensor observations [76]. In hybrid environments where dynamic humans and

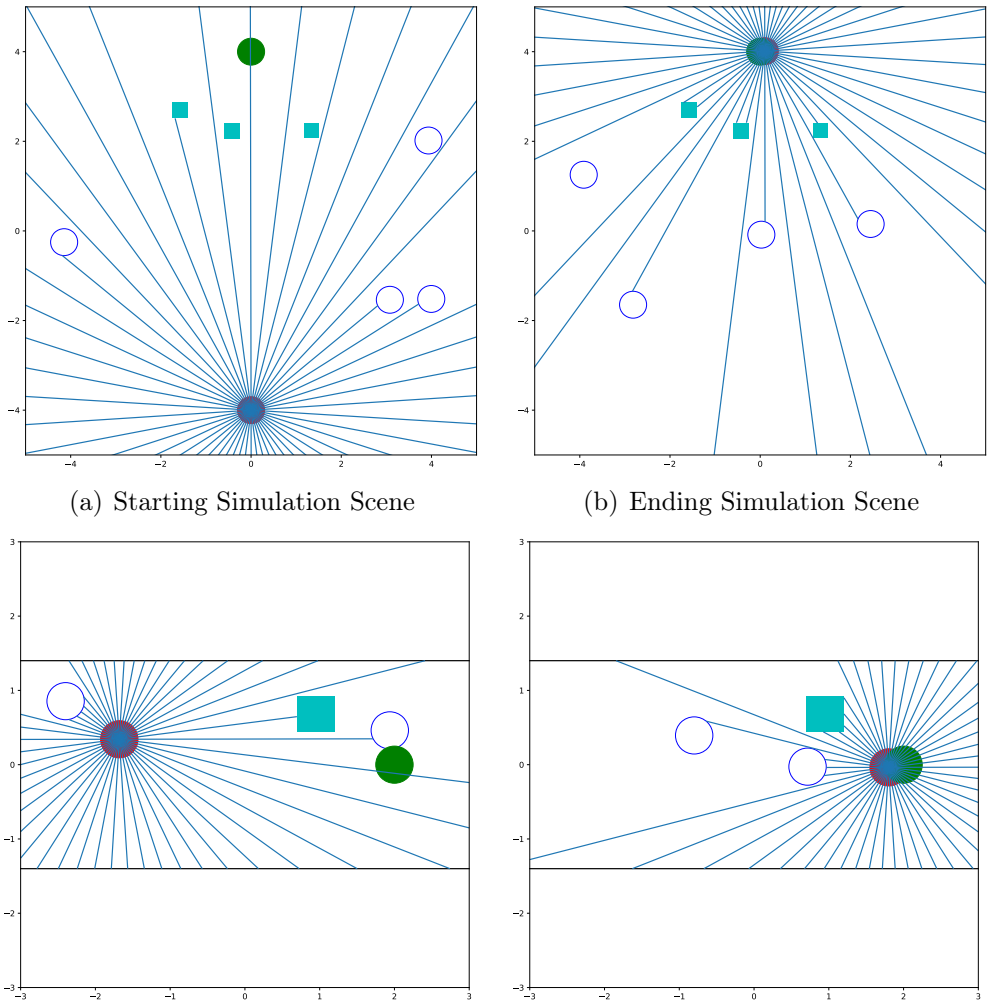
static obstacles such as walls and boxes coexist, a feasible strategy is to combine fully known human states and partially observable raw sensor data [116] as observations. Unfortunately, there are few open-source solutions available. In addition, because of the requirement of large CNNs to be embedded into policy networks to process high-dimensional observations, it is necessary to equip physical robots with a powerful computer, which limits 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. Furthermore, we normalize real-world obstacles to match the simulated settings, allowing for successful sim-to-real transfer.

5.3 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 collision-free and target-approaching robot actions.

5.3.1 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 Figure 5.1) 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 [30], 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 for more authentic perception..



(a) Starting Simulation Scene (b) Ending Simulation Scene

(c) Starting Real Scene (d) Ending Real Scene

Figure 5.1: Specially designed simulator. To illustrate our approach, we represent dynamic humans as circles and static obstacles as rectangles. The goal is a green solid circle without a collision margin. Our robot is equipped with an ego-centric LiDAR sensor, and we use representative scenes (a) and (b) for simulation ablations, and selective frames (c) and (d) for real-world implementations. To ensure fair comparison with other baselines, the robot's initial location in (a) and goal position in (b) remain fixed in all training and testing episodes. However, in (c), we randomly set the robot's initial location to generalize real-world situations. The goal position in (d) remains fixed as it is relative to the robot's frame. We handle both moving and steady obstacles of varying shapes, sizes, and numbers in both simulation and real scenes. Due to the slower motion of our real robot platform, the real scenes are smaller (2.8×6m) than the simulation scenes (10×10m) with a larger motion area.

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

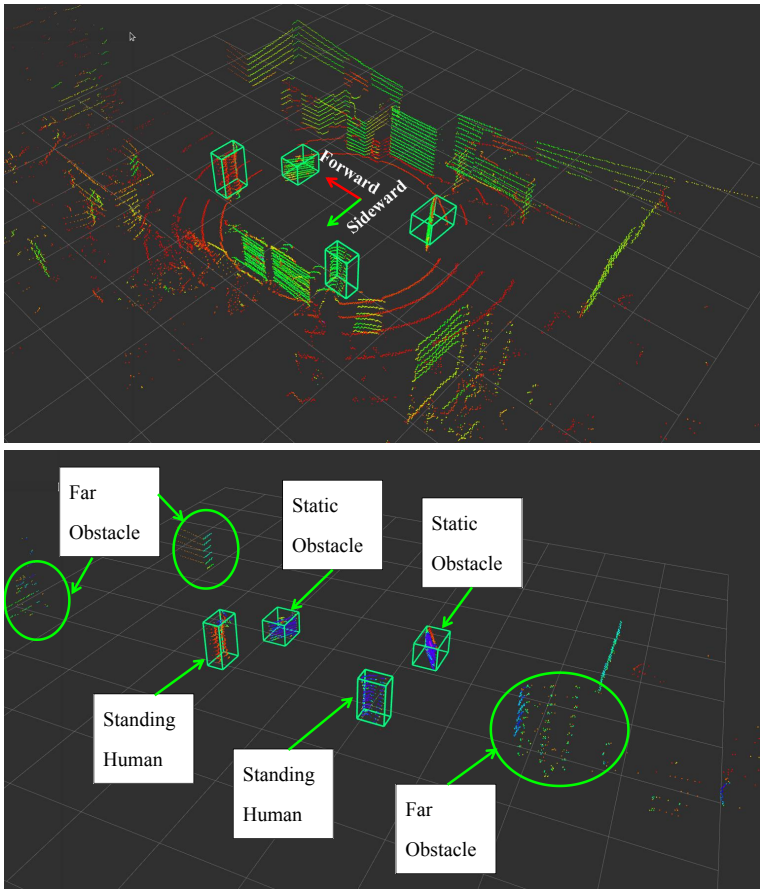


Figure 5.2: Obstacle clustering. The upper figure shows a raw LiDAR scan whereas the bottom one illustrates the clustering result. The target obstacles are bounded by boxes.

lines of the rectangle. Finally, we select the intersection point that is closest to the robot and represent the beam up to that point. This method is straightforward, but its complexity increases as the number of obstacles increases, which makes it suboptimal. 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. Conversely, when there is only one circle and one rectangle, the calculation time is reduced to 4.8ms. Since our motion planning frequency is 5Hz, we can disregard the generation time of LiDAR beams.

5.3.2 Obstacle Normalization

The studies which directly utilize raw real-world sensor observations [55,56] have the challenge of sim-to-real transfer because simulated configurations can not represent 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 [119] to individually extract obstacles from the 3D LiDAR pointclouds captured in the real world, as depicted in Figure 5.2. 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 [97] to locate the robot. Subsequently, we further filter out unrelated points in the world frame. We define a maximum forward and sideward distance, thereby refining the region of interest (ROI) in the world frame, similar to the motion area illustrated in Figure 5.1-c. Finally, we remove the points outside the ROI, and the remaining points are shown in the bottom image of Figure 5.2, which are then used for clustering.

Intuitively, two points, p_i and p_j $(i \neq j)$, belong to a same cluster C^k if the distance is within a threshold d:

$$p_i \in C^k,\ p_j \in C^k,\ if\ \|p_i - p_j\|_2 < d. \tag{5.3.1}$$

In addition, if $p_m \in C^k$ and $p_i \in C^k$ $(m \neq i)$, and $p_m \in C^k$ and $p_j \in C^k$ $(m \neq j)$; then $p_m \in C^k$, $p_i \in C^k$, and $p_j \in C^k$. If two adjacent points come from a same obstacle, the distance between these two points increases if the obstacle is far from the sensor. Therefore, we can define an adaptive threshold d as follows:

$$d = 2 \cdot r \cdot \tan \frac{\Theta}{2}, \tag{5.3.2}$$

where r is the distance from the point to the sensor and Θ is the vertical angular resolution of the 3D LiDAR sensor. We frame each cluster and normalize it as a circle or rectangle in the world frame as illustrated in Figure 5.1-c. Subsequently, we re-generate LiDAR scans to reduce sim-to-real discrepancies.

5.3.3 Algorithm Framework

To capture dynamic motion features, DRL-based policy networks generally utilize continuous raw sensor observations as input [55, 56]. However, this requires colossal CNNs to decode high-dimensional observations, especially when collected over a long time horizon [55]. Additionally, if the time horizon is short [56], extracting long-term motion features becomes challenging. To address these challenges, we

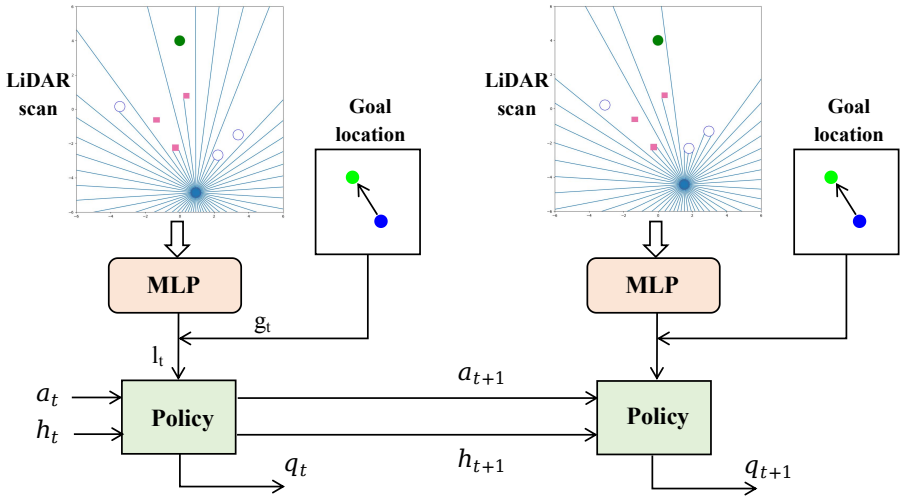


Figure 5.3: Structure of LNDNL. We only require a multi-layer perceptron (MLP) to pre-process a single-frame LiDAR scan. The inputs of our policy networks include history information h_t, a processed LiDAR scan l_t, and the goal position g_t in the robot frame. The action-value function q_t is represented by a MLP with the inputs of h_t, l_t, g_t, and a_t. The policy network projects h_t, l_t, and g_t into next action a_{t+1}.

introduce LSTM as a lightweight method to accumulate and propagate historical motion features (as illustrated in Figure 5.3).

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

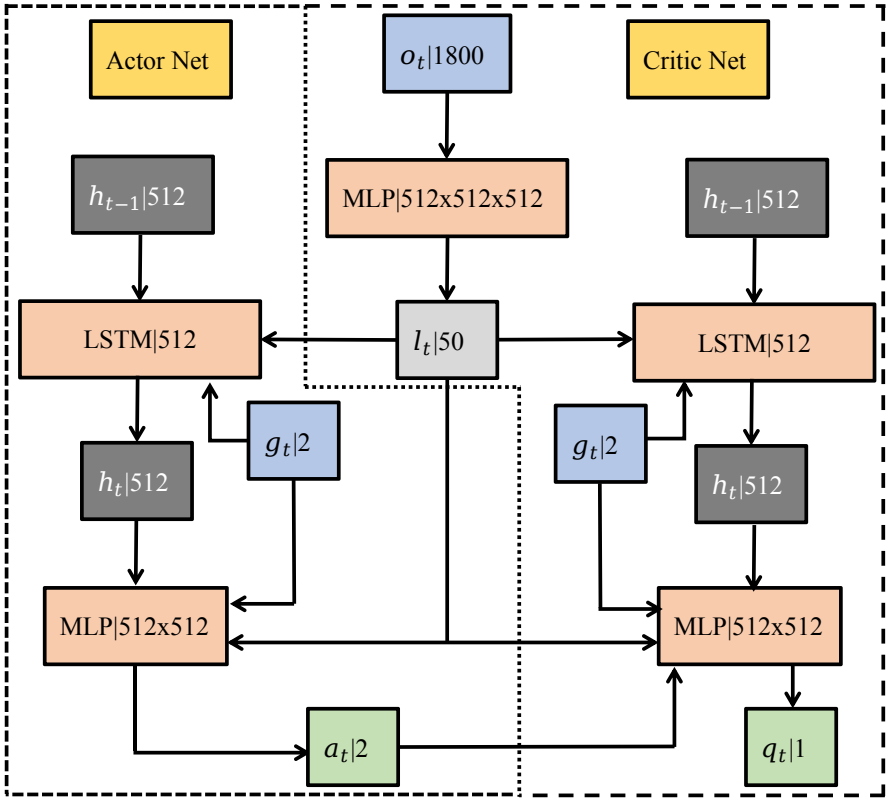


Figure 5.4: Network structure. A|B: A is a variable or network unit and B represents its dimension. MLP|AxBx...: MLP is a multi-layer perceptron and AxBx... denotes layer units.

networks (shown in Figure 5.4) are represented as follows:

$$\begin{aligned} l_t &= f_\theta(o_t), \\ s_t &= \{l_t, h_t, g_t\}, \\ q_t &= f_\psi(s_t, a_t), \\ a_{t+1} &= f_\phi(s_t), \\ h_t &= f_\varphi(h_{t-1}, l_t, g_t). \end{aligned} \tag{5.3.3}$$

We leverage the twin delayed deep deterministic policy gradient (TD3) algorithm [120] to optimize the weights $\Phi = \langle \theta, \psi, \phi, \varphi \rangle$ in Eq. (5.3.3). Different from the original TD3 algorithm using fully observable states, we have another network f_θ to pre-process high-dimensional LiDAR observations. As a consequence, both the actor network and critic network receive the pre-processed data from f_θ. To keep consistency, we only update θ using the loss from the critic network while the θ^* used for actor network is a duplication of θ. Additionally, we have anther LSTM network f_φ to accumulate and propagate history motion features. Different from θ singly updated according to the loss from critic network, two independent f_{φ^*} and

f_φ are separately embedded into the actor network and critic network, and φ^* and φ are correspondingly updated.

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 and 0.3m/s in real-world situations. In addition, the velocities are continuous instead of discrete values in state-of-the-art baselines [48, 50, 53, 54]. The observation at each time step is a 1800D ego-centric LiDAR scan and each beam length ranges from $b_{\min}$ to $b_{\max}$. $b_{\min}$ to $b_{\max}$ are slightly different in simulation and real-world situations. The navigation goal is to avoid obstacles and reach a target. Therefore, we define a reward function as follows:

$$r_t = \begin{cases} -r_c & \text{if } d_s < d_c, \\ 1.0 & \text{else if } g_t < g_r, \\ \omega_c \cdot (d_s - r_r - d_u) & \text{else if } d_s < d_u, \\ \omega_g \cdot (g_{t-1} - g_t) & \text{else.} \end{cases} \tag{5.3.4}$$

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.

Parameters. The learning rate of all networks is 3×10^{-4}. The discount factor γ is 0.99. To guarantee a stable learning, target networks are simultaneously created with source networks. Let Φ^* be the weights of target networks and Φ represent the weights of source networks. We update Φ^* using a momentum coefficient $\eta = 0.005$: $\Phi^* = (1 - \eta)\Phi^* + \eta\Phi$. We leverage the TD3 algorithm, which incorporates several tricks into deep deterministic policy gradient (DDPG) [35] to avoid dramatically overestimating action value function. One trick is the delayed update for actor network. We update the action network once after every two updates of the critic network. Another trick is to add noise to the action output from the target actor network. We define the noise as a normal distribution with 0 mean and 0.25 standard deviation. At the meanwhile, we add the same noise to the action yielded from the source actor network for exploratory interactions with environments. Because we train our model in two different situations – simulation ablation and real implementation, the parameters in Eq. (5.3.4) are separately tuned to improve navigation policy. In addition, the LiDAR parameters vary depending on the size of the motion

Table 5.1: Reward parameters used in simulation ablation and real implementation

	r_c	d_c, g_r, r_r (m)	d_u (m)	w_c	w_g	$b_{\min}$ (m)	$b_{\max}$ (m)
Sim	0.3	0.3	0.2	0.5	0.1	0.3	6.0
Real	0.4	0.25	0.2	0.5	0.3	0.26	4.0

area and the maximum speed of the robot. These parameters are list in Table 5.1.

5.4 Experiments

We conducted two sets of simulation ablations to demonstrate the superior performance of our approach. The first group, named Sim1, consisted of seven baselines executed in a relatively simple scenario that involved only dynamic circle humans with a fixed number. The second group, named Sim2, consisted of two baselines that were tested in a more challenging situation that mixed dynamic circle humans and static rectangle obstacles whose number was changeable. We also implemented two sets of training, named Real1 and Real2 respectively, for real-world applications, both of which included the baseline mentioned in the fourth chapter. Differently, Real1 only involved dynamic circle humans, while Real2 included hybrid obstacles similar to those in Sim2. Since our study does not focus on classifying obstacles as humans or not, we supplemented another training named Real3, which assumes all obstacles have a fixed circular shape and are either dynamic or static. To enhance sim-to-real transferability in a quadruped robot, we have made improvements to the observation and action definition in Real3.

5.4.1 Simulation Ablation

To conduct a comprehensive comparison with the state-of-the-art baselines described in the related work section, which includes CADRL [48], LSTM_RL [49], SARL [53], and RGL [54], we designed a simple scenario that includes five moving humans in a 10×10m area, named Sim1. These baselines 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 baselines are available, there

Table 5.2: Final Evaluation. 500 random tests are executed with the best neural networks saved during the training.

Method	SR	NT	AT	NV
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
SEDN[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]These methods require fully known human information.

[2]These methods require imitation learning and positive samples.

SR: success rate; NT: navigation time (s)

AT: action time (s); NV: network variables (10^5)

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 [55,56]. Additionally, we compared the SEDN and NPD algorithms discussed in Chapters 3 and 4, respectively. Specifically, SEND improves EGO by transforming the centers of all LiDAR scans into a same location to individually extract surrounding motion features. 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 Figure 5.5 and provide a quantitative analysis in Table 5.2.

The results presented in Figure 5.5 show that our method exhibits significantly better learning efficiency than EGO, SEDN, 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 baseline approaches rely heavily on these prior experiences. In Table 5.2, 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

Table 5.3: The influence of obstacle density and type. 500 random tests are executed for each scenario.

1S0D	0S1D	2S0D	1S1D	0S2D	3S0D	2S1D	1S3D	0S3D
0.98	0.96	0.94	0.89	0.86	0.86	0.80	0.74	0.73
XSYD, X static obstacle(s) and Y dynamic obstacle(s).								

fewer due to their simpler observations. Despite our network being comparatively larger, it is still more lightweight than the study in Chapter 4, NPD. Additionally, the action time of our network outperforms EGO and SEDN, even though they have smaller networks. The advantages of a short action time and an acceptable network scale enable sim-to-real transfer on compact robots with limited hardware resources.

5.4.2 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 Figure 5.1-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 SEDN and NPD as our baselines for Sim2. As shown in Figure 5.6, 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.

5.4.3 Real Implementation

We defined two real-world scenarios for practical implementation. In the first scenario, motion area is narrowed to 6.0×2.8m, and a maximum of three obstacles are present, resulting in a notably higher obstacle density ($3/(6.0 \times 2.8) = 0.18$) than in Sim2 ($7/(10 \times 10) = 0.07$). In the second scenario, called Real2, the environment is a mix of dynamic circle humans and static rectangle obstacles with side lengths ranging from 0.3 to 0.6m (shown in Figure 5.1-c). Conversely, Real1 includes only dynamic circle humans, with their numbers ranging from one to three. The maxi-

mum velocity of all agents is decreased to 0.3m/s because of the motion constraint of our robot platform. Because the baseline SEDN requires imitation learning and a great number of supervised samples, we disregarded it as a baseline. Instead, we chose NPD for comparison because it is able to learn from zero experience, which is similar to the prior-experience-free characteristic of our approach. Figure 5.7 shows the training process, and our method outperforms NPD in terms of learning efficiency. We thought NPD's colossal networks result in the time-consuming learning. Additionally, our method shows stable training in both Real1 and Real2 scenarios, whereas NPD's training fluctuates violently in Real2. However, the success rate of our method in the real-world scenarios converges to 0.8 and peaks at 0.89, which is lower than that of the simulated scenarios. This could be due to the dense environments, which make it harder for our method to navigate successfully. Specifically in Real2, Table 5.3 illustrates how the obstacle density influences the success rate. When obstacles are densely distributed in a narrow motion area, the success rate is significantly degraded, especially when the percentage of dynamic obstacles is high.

We found that the real quadruped robot frequently and fiercely vibrated along the sideward locomotion direction when we directly transferred the navigation policy learned in Real1. To enable a more stable sideward locomotion, we designed another real scenario, name Real3, and re-defined the observation and action space. Specifically, in addition to the LiDAR scan and goal position g_t, we added the last action command a_{t-1} as another state. Correspondingly, we concatenated g_t and a_{t-1} to replace g_t shown in Figure 5.3 and 5.4. Moreover, the action space is a variation with respect to a_{t-1} and its maximum forward value is 0.1m/s and sideward value is 0.05m/s to enable a stable velocity change between two adjacent time times. In addition, the gait generator of the quadruped robot is unstable when the sideward locomotion is fast. Therefore, we constrain the maximum sideward speed as 0.15m/s. In contrast, the maximum forward speed is 0.3m/s. Please note that the human motion has no constraints except that both the maximum sideward and forward speeds are 0.3m/s. Despite of the robot locomotion constraints, our learning framework is able to obtain a feasible navigation strategy. The training process is illustrated in Figure 5.7, where the peak success rate is 86%. The sim-to-real transfer is shown in the attached videos, with the shots shown in Figure 5.8.

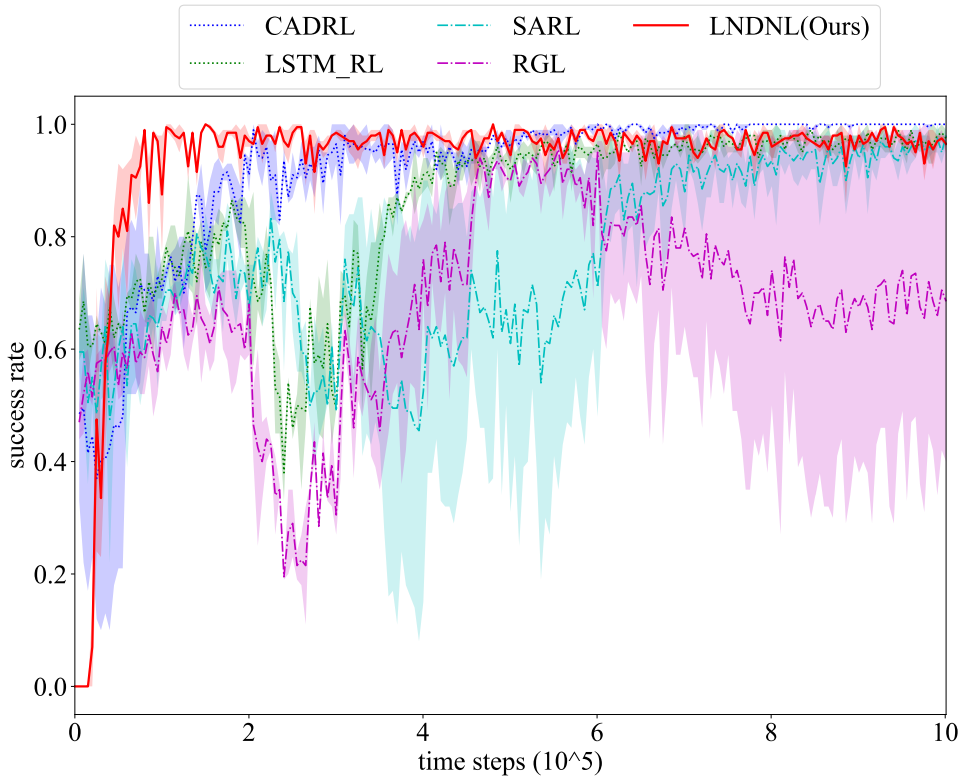
5.4.4 Limitations

Although our approach outperformed state-of-the-art baselines in both simulation and real implementation, its performance was degraded when obstacles are dense in

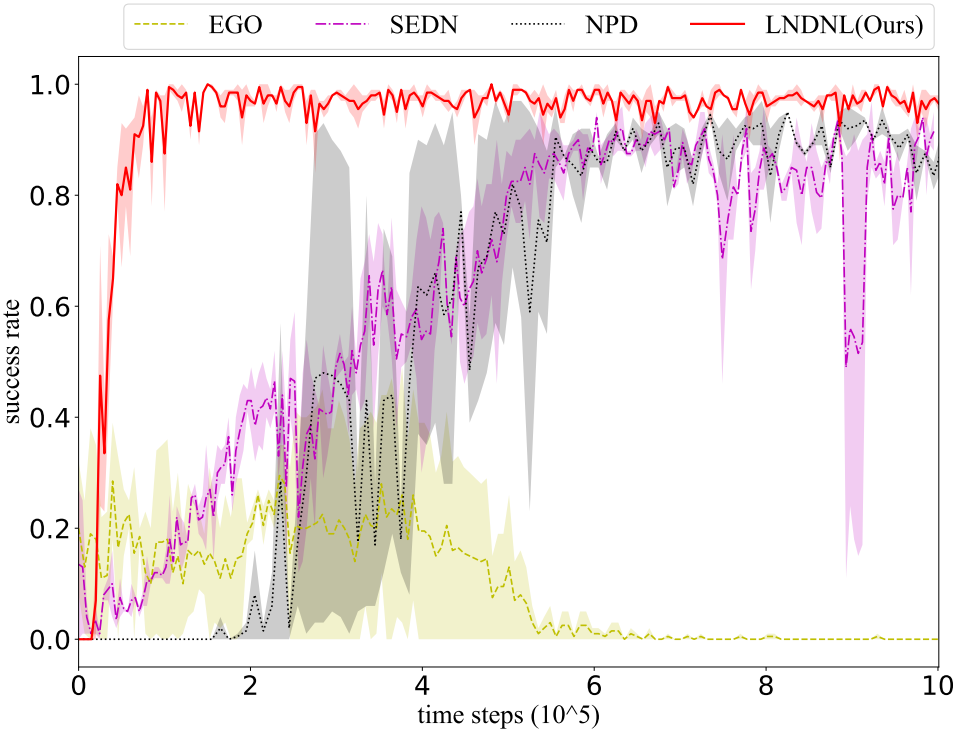
environments. Moreover, we found that the movements of the real quadruped robot was not smooth although we re-defined and constrained the action space, which further resulted in inaccurate obstacle normalization. Our future studies will focus on improving the stability of sim-to-real transfer and the learning performance of real implementation. Additionally, the robot's motion area is constricted because of wired cables. We will mount an on-board computer to realize wireless and remote control. Because of the relatively lightweight networks, we are able to deploy our policy networks on compact devices.

5.5 Conclusions

This study presents a navigation strategy that directly projects sequential LiDAR observations into robot actions through LSTM-DRL. To expedite training, we developed a simulator that can 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. However, we observed a degradation in navigation performance when we narrowed the motion area and the robot movements in the real-world were not smooth. Therefore, our future studies will concentrate on enhancing real-world implementations.



(a) Fully Observable States



(b) Partially Observable States

Figure 5.5: 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, SEDN, and NPD are partially observable. In addition, EGO and SEDN directly use continuous LiDAR scans whereas NPD leverages sequential occupation maps.

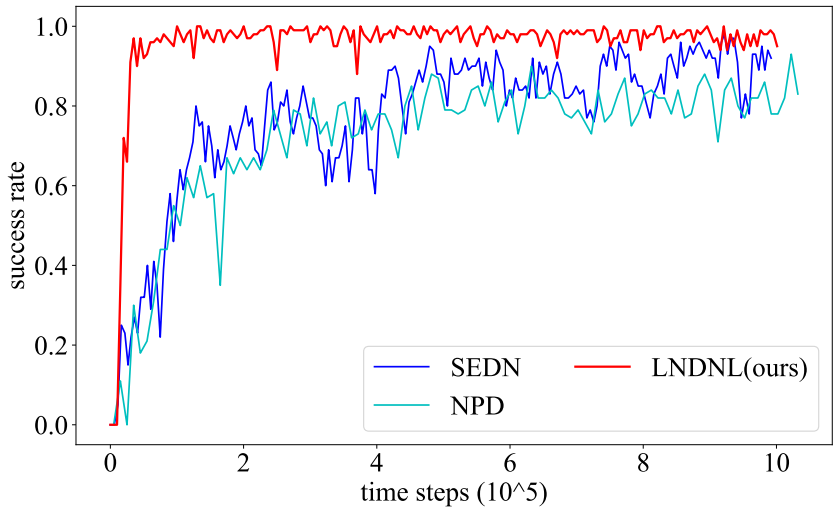


Figure 5.6: Training process in more challenging environments.

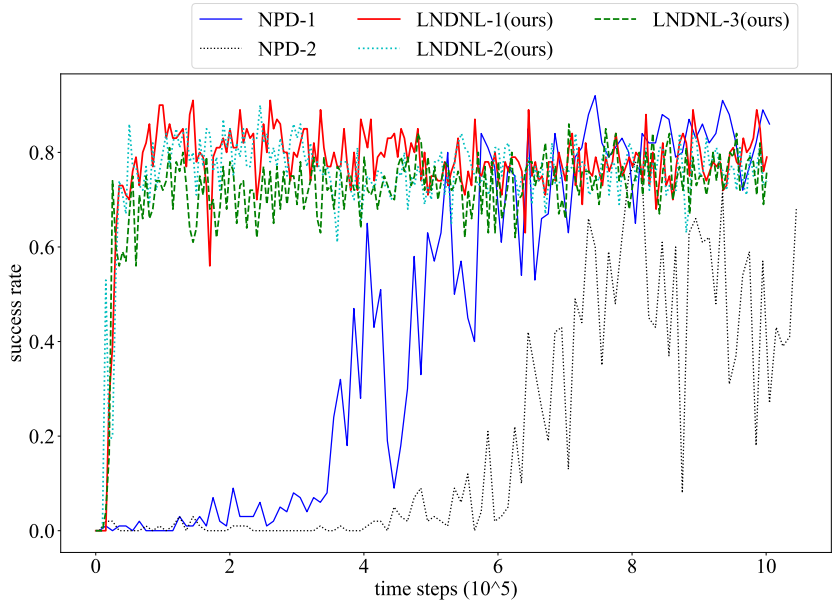


Figure 5.7: Training process in real-world scenarios. NPD-1(2) and LNDNL-1(2) represent the training results in Real1(Real2). LNDNL-3 illustrates the learning process in Real3.

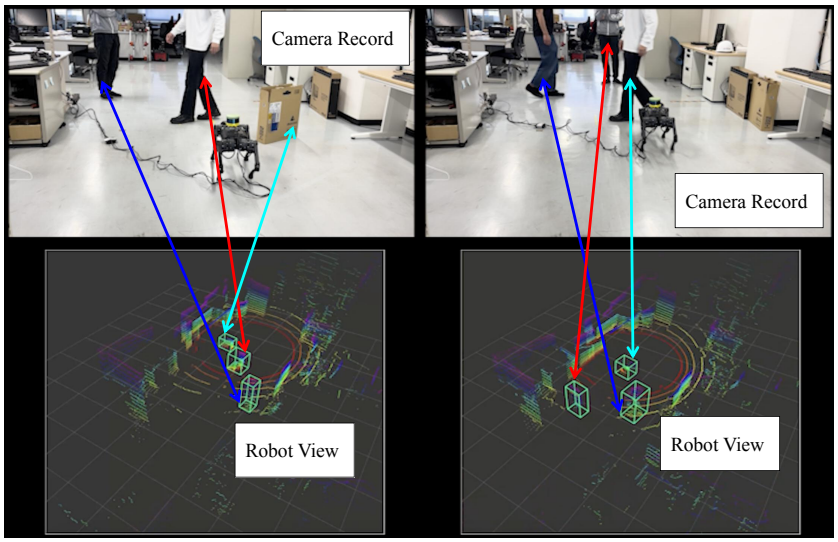


Figure 5.8: Shots of real-world experiments.

Chapter 6

Conclusions and Future Work

6.1 Summary

In this dissertation, we have studied the robot navigation in diverse environments through deep reinforcement learning (DRL). The aim of this dissertation is to enhance the collision avoidance capabilities of physical mobile robots operating in dynamic environments through DRL. In order to achieve this objective, we developed four DRL models which were trained to learn collision-free navigation policies for different scenarios. Furthermore, to make the system applicable in practical situations, we employed a combination of conventional controllers, domain randomization, system identification, and localization algorithms. To evaluate the effectiveness of our proposed approach, we conducted a series of experiments on various simulated environments and real-world scenarios with dynamic obstacles. Our results indicate that our approach surpasses traditional methods in terms of collision avoidance and navigation efficiency. In addition, our proposed navigation systems were shown to adapt well to changes in the environment, making them suitable for practical applications.

In Chapter 2, our proposed navigation structure, called the Hierarchical DRL-based (HDRL) framework, can adapt to unknown environment configurations and is highly efficient in terms of sampling. This framework is also capable of transferring simulated navigation to real-world situations. Our low-level DRL policy enables the robot to navigate toward the target while keeping a safe distance from obstacles. In addition, a high-level DRL policy is used to further improve navigation safety. To reduce the state space and prevent sparse rewards, we select a waypoint along the path from the robot to the final destination as a sub-goal. The path is generated based on either a local or global map, which significantly enhances the sampling

efficiency, safety, and generalization capability of the proposed DRL framework. Furthermore, by deriving a target-directed representation for the action space based on the sub-goal, we can increase motion efficiency while reducing the action space. Finally, we demonstrate the effectiveness of our navigation strategy by deploying it on a wheel-legged biped robot and a quadruped robot in initially unknown environments.

The algorithm in Chapter2 assumed that obstacles would remain static after their initial configuration of position and orientation. Progressively, in Chapter 3, we faced a more difficult challenge of dealing with a combination of static and dynamic obstacles. To address this challenge, we developed a Sampling Efficient DRL framework for Dynamic Navigation (SEDN), that utilizes continuous raw LiDAR scans. To train the DRL policy efficiently and simulate LiDAR scans, we designed a kinematics-based simulator. The learned navigation policy can be directly applied into a physics-based Gazebo simulator and real-world scenarios, where we utilized a quadruped robot equipped with a LiDAR sensor. We also demonstrated the policy's generalization ability across diverse environments that have never been explored before. Given the continuous raw LiDAR scans represent the coupled motions of the robot and surrounding humans, we transformed the center of previous LiDAR scans to the center of the current scan to individually extract surrounding motion features. Additionally, we integrated optimal reciprocal collision avoidance (ORCA) to generate auxiliary action alternatives to enhance the sampling efficiency. Through various ablations and real-world implementations, our approach demonstrated remarkable learning efficiency, superior generalization ability, and strong adaptability in real-world scenarios.

Although the navigation strategy in Chapter 3 has yielded significant achievements in dynamic environments, there are several drawbacks to be solved. Firstly, the practical application of the study in Chapter 3 is limited due to significant differences in obstacle margins between simulations and real-world scenarios. Additionally, the robot's reliance on external localization systems further hinders its practicality. Furthermore, the need for imitation learning and supervised samples in the study of Chapter 3 may result in sub-optimal navigation policies. To address these limitations, our further improved research in Chapter 4 integrates human detection and robot localization modules to create an autonomous navigation system. We use a model-based and sampling-efficient DRL framework to eliminate the need for supervised demonstrations and enable realistic and practical applications in human society. Our approach involves creating a collision-free Navigation system in diverse Pedestrian scenarios using a Dreamer-based (NPD) motion planner. We leverage system identification, domain randomization, clustering, and LiDAR SLAM tech-

niques for practical deployment. Our approach outperforms state-of-the-art baselines in rigorous comparisons, with superior results consistently demonstrated. Our extensive real-world experiments provide compelling evidence for the efficacy of our method in modeling complex, reciprocal human relations, and successfully navigating robots among pedestrians.

We further improved our navigation strategy by combining the findings from our second and third studies in Chapter 3 and 4. Our new approach has multiple advantages, including faster and more stable learning, lower computational requirements, and improved adaptability. Instead of relying on consecutive LiDAR scans, we use long short-term memory (LSTM) to extract motion features from ego-centric and sequential LiDAR scans. This allows us to reduce the size and complexity of our networks, which results in shorter training times and lower hardware requirements. We also employ a model-free DRL framework to handle deviated human motion models, rather than the Dreamer algorithm used in Chapter 4. To make our approach more practical for real-world scenarios, we use an improved clustering algorithm to extract obstacle positions from raw LiDAR scans. We then normalize the obstacles as circles and rectangles, which allows us to generate new LiDAR scans that are more generalizable to real-world situations. We tested our new navigation strategy extensively through simulations and practical implementations, and our results show that it outperforms the methods used in Chapter 3 and 4.

Overall, this dissertation aims to present a robust and adaptable navigation system for autonomous agents that can be applied to various domains, including robotics, autonomous driving, and unmanned aerial vehicles, contributing to the field of end-to-end robot navigation in dynamic environments. Furthermore, the proposed approach has the potential to address related issues such as multi-agent collision avoidance, which is a critical topic in robotics and autonomous systems.

6.2 Future Work

There are several improvements and further investigations that can be done to the studies carried out in this thesis.

The study presented in Chapter 5 has room for improvement. While the simulation ablations showcase the advantages of our approach in terms of learning efficiency, navigation performance, and hardware usage, the real-world implementation needs to show better performance, particularly in terms of the success rate of collision-

free and target-reaching navigation. Additionally, the current implementation has a small motion area with fewer than three obstacles, so our future research will concentrate on navigating crowds in environments with a higher density of humans, such as restaurants, hotels, and hospitals. To achieve our objective of enabling robots to navigate seamlessly in human societies, we will explore new methods to optimize our algorithms for human-robot interactions. This includes developing advanced perception and learning techniques to ensure robots can detect and navigate around humans and other obstacles accurately. Additionally, we plan to enhance our robot's ability to learn and adapt to new environments quickly.

A critical challenge that we face in all our studies is the transfer of simulation to the real world. The motion area of our robot platform is constrained by the use of wired cables for data transmission. To overcome this issue, we plan to remove these cables and incorporate an on-board computer, enabling the robot to operate in a larger workspace. Moreover, we plan to integrate state-of-the-art localization and mapping techniques to enhance the robot's ability to perceive and navigate through dynamic environments. We aim to develop a robust and accurate system that can detect and localize objects and obstacles in real-time. We also plan to explore new methods to generate smoother and more stable locomotion in the real world. This includes developing advanced control algorithms that can stably and smoothly track the commands generated from DRL-based navigation policies. Finally, we will evaluate our approach using real-world experiments and datasets to demonstrate the effectiveness of our methods.

Bibliography

[1] X. Xiao, B. Liu, G. Warnell, and P. Stone, "Motion planning and control for mobile robot navigation using machine learning: a survey," *Autonomous Robots*, vol. 46, pp. 569-597, 2022.

[2] S. Kim, J. Kim, F. B. Baiden, M. Giroux, and Y. Choi, "Preference for robot service or human service in hotels? Impacts of the COVID-19 pandemic," *International Journal of Hospitality Management*, vol. 93, no. 102795, 2021.

[3] K. Wu, H. Wang, M. A. Esfahani, and S. Yuan, "Learn to navigate autonomously through deep reinforcement learning," *IEEE Transactions on Industrial Electronics*, vol. 69, no. 5, pp. 5342-5352, 2022.

[4] 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.

[5] Y. Cai, T. Luan, H. Gao, H. Wang, L. Chen, Y, Li, M. Sotelo, and Z. Li, "YOLOv4-5D: An effective and efficient object detector for autonomous driving," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, no. 4503613, pp. 1-13, 2021.

[6] R. Moller, A. Furnari, S. Battiato, A. Harma, and G. M. Farinella, "A survey on human-aware robot navigation," *Robotics and Autonomous Systems*, vol. 145, no. 103837, 2021.

[7] H. Michael, A. S. Matveev, and A. V. Savkin, "Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey," *Robotica*, vol. 33, no. 3, pp. 463-497, 2015.

[8] R. Liu, J. Wang, and B. Zhang, "High definition map for automated driving: Overview and analysis," *The Journal of Navigation*, vol. 73, no. 2, pp. 324-341, 2020.

[9] T. Randhavane, A. Bera, E. Kubin, A. Wang, K. Gray, and D. Manocha, "Pedestrian dominance modeling for socially-aware robot navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[10] K. D. Katyal, G. D. Hager, and C. M. Huang, "Intent-aware pedestrian prediction for adaptive crowd navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[11] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 6, pp. 2064-2076, 2020.

[12] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674-691, 2021.

[13] 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.

[14] 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.

[15] A. J. Sathyamoorthy, J. Liang, U. Patel, T. Guan, R. Chandra, and D. Manocha, "DenseCAvoid: Real-time navigation in dense crowds using anticipatory behaviors," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[16] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, no. 37, vol. 3, pp. 362-386, 2020.

[17] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443-58469, 2020.

[18] D. Omeiza, H. Webb, M. Jirotka, and L. Kunze, "Explanations in autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 10142-10162, 2022.

[19] D. Filliat and J. Meyer, "Map-based navigation in mobile robots: I. a review of localization strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 243-282, 2003.

[20] Y. Wang and W. Chen, "Hybrid map-based navigation for intelligent wheelchair," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[21] C. W. Warren, "Fast path planning using modified A* method," in *IEEE International Conference on Robotics and Automation (ICRA)*, 1993.

[22] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[23] 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.

[24] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *International Conference on Robotics and Automation (ICRA)*, 2019.

[25] H. Wang, C. Hu, W. Cui, and H. Du, "Multi-objective comprehensive control of trajectory tracking for four-in-wheel-motor drive electric vehicle with differential steering," *IEEE Access*, vol. 9, pp. 62137-62154, 2021.

[26] M. B. Radac and T. Lala, "Hierarchical cognitive control for unknown dynamic systems tracking," *Mathematics*, vol. 9, no. 21, article no. 2752, 2021.

[27] G. Oriolo, G. Ulivi, and M. Vendittelli, "Real-time map building and navigation for autonomous robots in unknown environments," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 28, no. 3, pp. 316-333, 1998.

[28] D. H. Lee, S. S. Lee, C. K. Ahn, P. Shi, and C. C. Lim, "Finite distribution estimation-based dynamic window approach to reliable obstacle avoidance of mobile robot," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 9998-10006, 2021.

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

[30] J. Berg, S. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," *Robotics Research*, Springer, Berlin, Heidelberg, 2011, pp. 3-19.

[31] F. Belkhouche, "Reactive path planning in a dynamic environment," *IEEE Transactions on Robotics*, vol. 25, no. 4, pp. 902-911, 2009.

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

[33] 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.

[34] V. Mnih, K. Kavukcuoglu, D. Silver, et al., "Human-level control through deep reinforcement learning," *Nature*, no. 518, pp. 529–533, 2015.

[35] T. Lillicrap, J. Hunt, A. Pritzel, et al., "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[36] L. Kaiser, M. Babaeizadeh, P. Milos, et al., "Model-based reinforcement learning for atari," *arXiv preprint arXiv:1903.00374*, 2019.

[37] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *International Conference on Learning Representations (ICLR)*, 2020.

[38] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba, "Mastering atari with discrete world models," in *International Conference on Learning Representations (ICLR)*, 2021.

[39] M. Okada and T. Taniguchi, "Dreaming: Model-based reinforcement learning by latent imagination without reconstruction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[40] F. Deng, I. Jang, and S. Ahn, "Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations," in *International Conference on Machine Learning (PMLR)*, 2022.

[41] 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.

[42] J. Zhang, J. Springenberg, J. Boedecker, et al., "Deep reinforcement learning with successor features for navigation across similar environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[43] 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.

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

[45] C. Sampedro, H. Bavle, A. Ramos, et al., "Laser-based reactive navigation for multirotor aerial robots using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[46] 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.

[47] G. Kahn, A. Villaflor, B. Ding, et al., "Self-supervised deep reinforcement learning with generalized computation graphs for robot navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[48] 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.

[49] 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.

[50] 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.

[51] Y. Chen, C. Liu, B. E. Shi, and M. Liu, "Robot navigation in crowds by graph convolutional networks with attention learned from human gaze," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2754-2761, 2020.

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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.

[57] X. Huang, H. Deng, W. Zhang, R. Song, and Y. Li, "Towards multi-modal perception-based navigation: A deep reinforcement learning method," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4986-4993, 2021.

[58] C. Arpino, C. Liu, P. Goebel, R. Martin, and S. Savarese, "Robot navigation in constrained pedestrian environments using reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[59] U. Patel, N. K. S. Kumar, A. J. Sathyamoorthy, and D. Manocha, "DWA-RL: Dynamically feasible deep reinforcement learning policy for robot navigation among mobile obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

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

[61] 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.

[62] T. Bailey and H. Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108-117, 2006.

[63] R. Fahad, and M. Hayashibe, "Towards robust wheel-legged biped robot system: Combining feedforward and feedback control," in *IEEE/SICE International Symposium on System Integration (SII)*, 2021.

[64] A. Pfrunder, P. Borges, A. Romero, et al., "Real-time autonomous ground vehicle navigation in heterogeneous environments using a 3D LiDAR," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[65] G. Desouza, and A. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 237-267, 2002.

[66] J. Berg, J. Snape, S. J. Guy, and D. Manocha, "Reciprocal collision avoidance with acceleration-velocity obstacles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[67] K. Guo, D. Wang, T. Fan, and J. Pan, "VR-ORCA: Variable responsibility optimal reciprocal collision avoidance," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4520-4527, 2021.

[68] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han and Y. Zhao, "Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5435-5444, 2021.

[69] K. Tsunekawa, F. Leiva, and J. Solar, "Visual navigation for biped humanoid robots using deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3247-3254, 2018.

[70] W. Zhu, X. Guo, D. Owaki, K. Kutsuzawa and M. Hayashibe, "A survey of sim-to-real transfer techniques applied to reinforcement learning for bio-inspired robots," *IEEE Transactions on Neural Networks and Learning Systems*, doi: 10.1109/TNNLS.2021.3112718.

[71] Y. Duan, X. Chen, R. Houthooft, et al., "Benchmarking deep reinforcement learning for continuous control," in *International Conference on Machine Learning (ICML)*, pp. 1329-1338, 2016.

[72] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *International Conference on Machine Learning (PMLR)*, 2017.

[73] F. Raza, W. Zhu, and M. Hayashibe, "Balance stability augmentation for wheel-legged biped robot through arm acceleration control," *IEEE Access*, vol. 9, pp. 54022-54031, 2021.

[74] 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.

[75] 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.

[76] 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.

[77] 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.

[78] 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.

[79] 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.

[80] 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.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

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

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

[88] 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.

[89] Q. Li, W. Lin, Z. Liu, and A. Prorok, "Message-aware graph attention networks for large-scale multi-robot path planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5533-5540, 2021.

[90] G. Ferrer, A. Garrell, and A. Sanfeliu, "Robot companion: A social-force based approach with human awareness-navigation in crowded environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[91] S. Paris, J. Pettre, and S. Donikian, "Pedestrian reactive navigation for crowd simulation: A predictive approach," *Computer Graphics Forum*, vol. 26, no. 3, pp. 665-674, 2007.

[92] M. Kuderer, H. Kretzschmar, C. Sprunk, and W. Burgard, "Feature-based prediction of trajectories for socially compliant navigation," in *Robotics: Science and Systems (RSS)*, 2012.

[93] 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, 2023.

[94] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232-242, 2016.

[95] M. Okada and T. Taniguchi, "DreamingV2: Reinforcement learning with discrete world models without reconstruction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[96] R. S. Sutton and G. B. Andrew, "Reinforcement learning: An introduction," *MIT press*, 2018.

[97] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," in *Robotics: Science and Systems (RSS)*, 2014.

[98] D. Matti, H. K. Ekenel, and J. P. Thiran, "Combining LiDAR space clustering and convolutional neural networks for pedestrian detection," in *IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017.

[99] C. Badue, R. Guidolini, R. V. Carneiro, et al., "Self-driving cars: A survey," *Expert Systems with Applications*, vol. 165, no. 113816, 2021.

[100] G. A. Zachiotis, G. Andrikopoulos, R. Gornez, K. Nakamura, and G. Nikolakopoulos, "A survey on the application trends of home service robotics," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018.

[101] J. Cheng, H. Cheng, M. Meng, and H. Zhang, "Autonomous navigation by mobile robots in human environments: A survey," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2018.

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

[103] 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.

[104] 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.

[105] 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.

[106] 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.

[107] 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.

[108] 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.

[109] 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.

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

[111] 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.

[112] 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.

[113] 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.

[114] 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.

[115] 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.

[116] 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.

[117] 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.

[118] 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.

[119] 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.

[120] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International Conference on Machine Learning (PMLR)*, 2018.

Acknowledgements

I would like to express my immense gratitude to Professor Mitsuhiro Hayashibe, my advisor, for the incredible opportunity to work at the Neuro-Robotics Lab and his unwavering support and guidance during my three-year doctoral program. I would also like to extend my heartfelt appreciation to Associate Professor Dai Owaki and Assistant Professor Kyo Kutsuzawa for their invaluable feedback on my papers and presentations. Additionally, I wish to express my sincere appreciation to Professor Yongchun Fang, Associate Professor Xian Guo, Dr. Xuetao Zhang, and Dr. Haiming Gao from my previous lab for their invaluable career advice.

I would like to give special thanks to Dr. Fahad Raza for his fundamental work in teaching me how to operate the robot platform. Without his guidance, I would not have been able to start my research, and I am grateful for the fantastic experience of co-working with him.

My deep gratitude also goes to Taku Sugiyama, Shunsuke Koseki, and Yoshida Takashi, who helped me immerse myself in Japanese culture and live a comfortable life in Japan. I am equally thankful for the support and camaraderie of my colleagues at the Neuro-Robotics Lab, including Adam Zaki, Amged Elshiekh, Chu Zheng, Christopher Herneth, Lucas Sulpice, Keli Shen, Jiazheng Chai, Orvin Demsy, Atsushi Hamada, Jihui Han, Guanda Li, Yan Guo, Yuchen Wang, Tianjian Yuan, Youchun Ma, Hannan Ahmed, and Chatrin Phunruangsakao.

I would also like to express my gratitude to my thesis committee members, Professor Mitsuhiro Hayashibe, Professor Yasuhisa Hirata, Professor Kazuya Yoshida, and Professor Xuebo Zhang, for their thoughtful questions and valuable comments on my research project.

My parents deserve my most profound gratitude for their unwavering support in all my endeavors. Finally, I would like to extend my warmest thanks to Hongyuan Yu, Xuetao Zhang, Guanda Li, Yan Guo, Yoshida Takashi, and Amged Elshiekh for their lasting friendship.