LEARNING STRUCTURED INTERACTION MODELS FOR ROBOT NAVIGATION IN HUMAN ENVIRONMENTS

BY

SHUIJING LIU

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois Urbana-Champaign, 2024

Urbana, Illinois

Abstract

Robots are becoming increasingly prevalent in our daily lives. However, these autonomous agents work best in isolation. Integrating robots into human environments remains challenging due to the complex and dynamic nature of human-robot interactions. This thesis addresses the challenge of robot navigation in human environments by proposing structured learning methods that enhance robots' ability to interact with humans in a safe, efficient, and socially aware manner.

We explore two types of interactions: implicit interactions through motion and explicit interactions through language. We build navigation systems that predict human intentions, reason about subtle interactions among agents, and plan paths for robots to fulfill tasks. In previous work, model-based approaches and end-to-end learning methods have demonstrated their own advantages and limitations. To combine the best of both worlds, this thesis develops structured ML systems by incorporating the predictions of human behaviors into robot planning. In addition, we formulate interactive scenarios using graph structures, which enables robots to achieve a more nuanced understanding of human behavior and interaction dynamics. We propose intention and interaction-aware robot decision-making systems for navigation in human spaces. Through simulation benchmarks, real world experiments, and user studies, we demonstrate robot's capability to navigate in various complex human environments.

Our contributions span three applications: (1) driver trait inference for autonomous vehicle navigation at T-intersections, (2) crowd navigation using attention-based spatio-temporal graphs, and (3) assistive navigation for persons with visual impairments using dialogue and visual-language grounding. Through these applications, we demonstrate the effectiveness of structured learning in robotic tasks that involve human-robot interaction. This thesis proposes algorithms and tools for deploying robots in real-world human settings, advancing the field of learning-based human-robot interaction.

To all of those who have trusted, encouraged, and supported me. They have shaped me into a resilient yet compassionate person who is committed to an ambition in engineering.

Acknowledgments

When I started in Illinois in 2014, I did not plan to stay there for a decade and definitely did not plan to do a PhD. There are a lot of people to thank for making my time in Illinois a great experience.

I am profoundly thankful to my advisor, Professor Katie Driggs-Campbell, for her guidance and support throughout my PhD journey. In research, she has been encouraging me to navigate independently from the very beginning of my PhD. Meanwhile, she always provides acute insights in our discussions on a large span of topics. Outside of research, Katie is not only a good advisor who cares about the well-being of her students, but also a role model that I would like to consistently learn from throughout my professional career. Thanks to her mentorship, I achieved way beyond my expectations from my PhD.

I am also grateful to the members of my thesis committee, Professor Nancy M. Amato, Professor Kris Hauser, and Professor Saurabh Gupta for their valuable feedback on my thesis. I would also like to thank many other professors who collaborated with me: Professor Junyi (Jenny) Geng, Professor D. Livingston McPherson, Professor Wendy A. Rogers, Professor Yunzhu Li, and Professor Girish Chowdhary.

Human Centered Autonomy (HCA) Lab has been a fantastic place for research. The wide range of people and projects in HCA lab have helped me to develop a wide research spectrum. I would like to thank all of my collaborators: Peixin Chang for helping me jump-start my PhD and teaching me how to do good research hand-by-hand, Neeloy Chakraborty for trusting me as a first-time mentor and continually growing beyond me, Zhe Huang and Haonan Chen for introducing me to areas out of my comfort zone, Kaiwen Hong for his up-to-date knowledge of state-of-the-art, Aamir Hasan for acting as the pseudo project manager who keeps everything organized, Weihang (Eric) Liang for his extensive knowledge on computer vision and robot hardware, Ye-Ji Mun, Pulkit Katdre, Tianchen Ji, John Pohovey, and Fatemeh Cheragi. They are not only the best co-workers but also great friends who provide my journey with rewards and fun. I would also like to thank Peter Du for insightful discussions with his sharp mind and sense of humor.

Before joining HCA lab, when I was an undergraduate student, I had already met a set of amazing collaborators who are still influencing me today. I owe my sincere thanks to Anay Pattanaik and Subhalakshmi Kumar for their mentorship. I also thank my undergraduate collaborators: Ke (Richard) Xu, Zhenyi Tang, and Andrew Kuznetsov for exploring the realms of research with me.

I have made many lifelong friends in Illinois. I would like to thank Sijia (Scarlett) Huo for being my dormmate in freshman year and my travel companion to places around the world, Yangxue (Sherry) Yu for co-founding a student organization on handcraft making with me, Xiangyuan Zhang for having hotpot dinner with me on Fridays, and Qian Jiang and Binyuan Liu for teaching me how to play board games. In addition, I would also like to thank my coursemates who helped me survive my degree, particularly Yu Meng, Huiyuan (Fred) Liu, Zhikai Guo, and Yijun Feng.

Since the job market has been extremely tough when I am about to graduate, I would like to thank everyone who helped me find a job, in particular but not limited to: Feng Tao for encouraging me to pursue

a faculty career, Shaoyu Meng for giving me countless referrals, as well as Jialu Li, Pranay Thangeda, and Haresh Karnan for being my job hunting buddies (Fortunately, we are all hired as of today).

Again, I thank my family members for their love and support, including my partner Peixin Chang, my parents Qing Luo and Dongxu Liu, my aunt Wen Luo, and my cousin Justin Luo.

Last but not least, this material is based upon work supported by the National Science Foundation (NSF) No. 2143435, University of Illinois Urbana-Champaign campus research board, as well as National Institute on Disability, Independent Living, and Rehabilitation Research (NIDILRR) No. 90REGE0021.

Table of contents

List of Figures

List of Tables

List of Abbreviations

HCA	Human Centered Autonomy. iv
ML	Machine Learning. 1
VAE	Variational Autoencoder. 2
HRI	Human-Robot Interaction. 3
St-graph	Spatio-temporal Graph. 3
RL	Reinforcement Learning. 3
PwVI	Persons with visual impairments. 3
RNN	Recurrent Neural Networks. 7
CVAE	Conditional Variational Autoencoder. 8
TTC	Time-to-Collision. 8
POMDP	Partially Observable Markov Decision Process. 8
IDM	Intelligent Driver Model. 9
GRU	Gated Recurrent Unit. 10
LSTM	Long Short-term Memory Networks. 10
SOS	Start-of-sequence. 11
KL divergence	Kullback–Leibler divergence. 11
MLP	Multi-layer Perceptron. 12
PPO	Proximal Policy Optimization. 12
ORCA	Optimal Reciprocal Collision Avoidance. 18
SF	Social Force. 18
MDP	Markov Decision Process. 18
DS-RNN	Decentralized Structural-RNN. 19
RVO	Reciprocal Velocity Obstacle. 19
S-RNN	Structural-RNN. 20
FC	Fully Connected. 21

FoV	Field of View. 24
RGL	Relational Graph Learning. 24
CADRL	Collision Avoidance with Deep Reinforcement Learning. 26
OM-SARL	Occupancy Map SARL. 26
Attn	Attention. 26
ROS	Robot Operating System. 29
HH	Human-Human. 31
RH	Robot-Human. 31
Const vel	Constant Velocity. 38
GST	Gumbel Social Transformer. 38
SR	Success Rate. 39
NT	Navigation Time. 39
PL	Path Length. 39
ITR	Intrusion Time Ratio. 39
SD	Social Distance during Intrusions. 39
E2E	End-to-End. 44
OH	Obstacle-Human. 48
RRT	Rapidly Exploring Random Tree. 53
DRAGON	Dialogue-based Robot for Assistive Navigation with Visual-Language Grounding. 54
SR	Speech Recognition. 54
NLU	Natural Language Understanding. 55
CLIP	Contrastive Language-Image Pre-training. 55
RFID	Radio-Frequency Identification. 57
SLAM	Simultaneous Localization and Mapping. 57
DWA	Dynamic Window Approach. 59
TTS	Text-to-Speech. 59
VQA	Visual-Question-Answering. 61
LR	Landmark Recognition. 64
EnvDes	Environment Description. 64
NavAdj	Navigation Preference Adjustment. 64
LLM	Large Language Model. 70

Chapter 1

Introduction

1.1 Overview & Challenges

The past decade has witnessed numerous breakthroughs in robotics, ranging from da Vinci Surgical System to Tesla Optimus. However, these robots work best in isolation, such as highly controlled laboratories or restricted areas in warehouses. To deploy robots in human environments, researchers and engineers have made numerous efforts in fields such as autonomous driving, last-mile delivery robots, and household robots. Unfortunately, these products are far from perfect today. In 2019 and 2023, two fatal crashes of autonomous vehicles, with a cyclist and a pedestrian respectively, have raised public concern [1], [2]. For delivery robots, at least for some portion of the time, they are teleoperated by humans and thus are not fully autonomous [3]. For household robots, a state-of-the-art example is Amazon Astro, which has limited functionalities besides what a virtual digital assistant can offer [4]. The root cause of these limitations can be traced to a core challenge in deploying robots in human environments: Subtle and highly dynamic interactions are always happening among different agents. These interactions are prevalent in human environments yet difficult to infer, which poses significant challenges for robot planning.

To address the aforementioned problem, this thesis focuses on robot navigation in interactive human environments. Two types of interactions exist while robots navigate among humans: Implicit interactions through motions (e.g., two agents yielding to each other when crossing a corridor) and explicit interactions through communication (e.g., speech, text, interfaces). To take complicated and long-term responsibilities such as museum guides and babysitters, robots must adeptly manage both types of interactions. Achieving this goal involves predicting human intentions from motions or language, reasoning about interactions, and planning accordingly to achieve task goals.

A large body of previous research uses model-based approaches. In these works, robots make decisions based on mathematical models of human navigation behaviors or conversation patterns [5], [6]. However, these model-based approaches make assumptions, such as humans are fully cooperative. Thus, the performance of these methods degrades in real-world scenarios with emergent phenomena, such as uncooperative humans [7], [8]. To alleviate this problem, more recent works have developed end-to-end machine learning (ML) pipelines for robots to learn to navigate or hold conversations with humans [9], [10]. While promising, training these models consumes a large amount of data and computation resources, which cannot be afforded by all real-world robotic applications, especially those in low data regimes.

Thus, the goal of this thesis is to propose structured learning methods that introduce structures into ML models. By combining the strengths of both worlds, robots could achieve a more nuanced understanding of human behavior and interaction dynamics. Structured learning-based frameworks enhance robots' ability to navigate in a safe and socially aware manner in complex human environments where large amounts of real-world datasets do not exist.

1.2 Problem Statement

This thesis develops structured ML systems for robot decision making in human environments. The following research problems are investigated:

1. Consider a navigation scenario with a pedestrian or vehicle crowd, such as a traffic intersection, a sidewalk, or a busy lobby. We assume an explicit communication channel among agents does not exist. Other agents run their own policies and may have different internal states (e.x. intended goals, walking or driving styles, etc), both of which are not observable to the robot. Instead, the robot can only observe the positions of a variable number of agents across time with a limited sensor range. Given a predefined goal, the research question is how to control a robot to the goal while avoiding collisions with other agents in a safe, efficient, and socially aware manner?

2. Suppose a human user gives verbal commands such as "Take me to the kitchen," how can the robot match the user's intended destinations to the environment and fulfill the corresponding task in an intuitive manner? In addition, to enhance the user's understanding of the surrounding environment, how can language be used to present semantic information from robot sensors?

1.3 Contributions

As shown in Fig. 1.1, this thesis develops structured robot navigation systems in various interaction scenarios with various learning methods. We present efforts to uncover structures of interactive behaviors, a key factor that determines the navigation performance of the robot. We develop ML models to represent the interactions among different agents, and incorporate the learned representations of interactions into the path planning stack of robots. Our work demonstrates the synergy between intention prediction and planning, as well as the power of injecting structures into multi-agent problems. Our case studies include three interactive navigation tasks with high impact on real-world applications: Our applications of implicit interactions include navigation in an uncontrolled intersection for an autonomous vehicle and in pedestrian crowds for a mobile robot. Our application of explicit interactions is a robot guide for persons with visual impairments. More specifically, we present the following contributions.

1.3.1 Driver Trait Inference for T-Intersection Navigation

In autonomous driving, the ego vehicle needs to infer and adapt to the internal traits of its surrounding road participants such as aggressiveness and distraction levels. We formulate the trait inference as a representation learning problem. Previous works have used supervised learning to classify the traits, yet trait labels are expensive to obtain for real driving datasets [11], [12]. The key critical innovation is an unsupervised approach to learn a representation of driver traits with a recurrent network based on variational autoencoder (VAE). The learned representation can form clusters of different traits from observed trajectories. Incorporated with

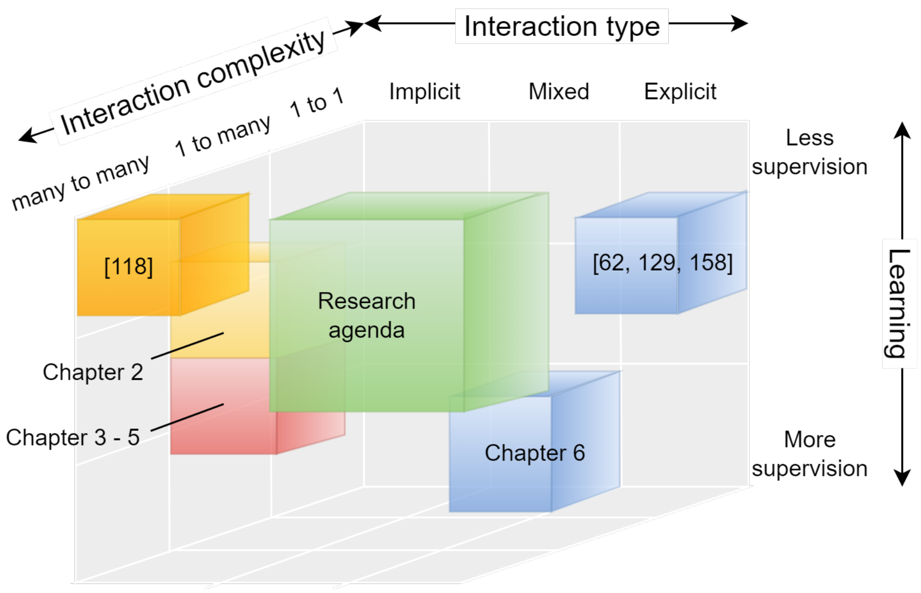


Figure 1.1: **An illustration of thesis overview.** This thesis explores the space of problems related to the complexity of human-robot interactions (HRI), the type of interaction, and level of supervision in the machine learning.

inferred traits, the ego vehicle is able to learn an adaptive policy and shows better navigation performance compared with state-of-the-art baselines [13], [14].

1.3.2 Crowd Navigation with attention-based spatio-temporal graphs

Compared with driving, robot navigation in a moving crowd has fewer constraints and more free-form interactions. Yet few previous works account for the heterogeneous interactions among all observed agents and obstacles through both space and time [5], [15], [16]. For the first time, we model the crowd navigation problem as a heterogeneous spatio-temporal graph (st-graph), which captures various interactions among agents and obstacles through space and time. From the st-graph, we derive a novel reinforcement learning (RL) policy network architecture for the robot with an attention mechanism, enabling the robot to attend to important humans, such as those nearly colliding with the robot. We first start with a sparse st-graph that captures some interactions in sparse crowds and constrained environments. Later on, we propose a dense st-graph that captures all interactions among dynamic agents in dense crowds, as well as interactions among static and dynamic agents in constrained environments.

To avoid short-sighted robot behaviors, I propose an intention-aware RL formulation that enables proactive planning in a general and scalable manner. By incorporating predicted human behavior into the observation space and reward function, the intention-aware RL framework encourages the robot to avoid intrusions into the intended path of human agents. The main benefits are improved safety and social awareness in simulated and real-world navigation in pedestrian-rich environments.

1.3.3 Assistive navigation with dialogue and visual-language grounding

Persons with visual impairments (PwVI) have difficulties understanding and navigating spaces around them [17], [18]. Current wayfinding technologies either focus solely on navigation or provide limited communication about the environment [19], [20]. Motivated by recent advances in visual-language grounding and semantic navigation [21], [22], we propose a guiding robot powered by a dialogue system and the ability to associate the environment with natural language. By understanding the commands from the user, the robot is able to guide the user to the desired landmarks on the map, describe the environment, and answer questions from visual observations. Through effective utilization of dialogue, the robot can ground the user's freeform language to the environment, and give the user semantic information through spoken language. With dialogue and grounding, the users can communicate with the robot and navigate to places of interest intuitively and efficiently.

1.4 Thesis Structure

The rest of this thesis is structured as follows:

- Chapter 2 presents work for driver trait inference for T-intersection navigation.
  The contents of this chapter are based on the following paper:
  - Shuijing Liu, Peixin Chang, Haonan Chen, Neeloy Chakraborty, and Katherine Driggs-Campbell. "Learning to Navigate Intersections with Unsupervised Driver Trait Inference." In IEEE International Conference on Robotics and Automation (ICRA), 2022, pp. 3576-3582. URL: https://arxiv.org/abs/2109.06783.
- Chapter 3 presents the first version of our work for crowd navigation in human crowds with sparse interactions. Our network incorporates robot-human interactions in the st-graph and robot policy network.
  The contents of this chapter are based on the following paper:
  - Shuijing Liu*, Peixin Chang*, Weihang Liang†, Neeloy Chakraborty†, and Katherine Driggs-Campbell. "Decentralized Structural-RNN for Robot Crowd Navigation with Deep Reinforcement Learning." In IEEE International Conference on Robotics and Automation (ICRA), 2021, pp. 3517–3524. URL: https://arxiv.org/abs/2011.04820[1].
- Chapter 4 presents the second version of our work for intention-aware crowd navigation, which incorporates all spatio-temporal interactions among dense and highly interactive agents. We also incorporate the future intentions of pedestrians in the planning framework.
  The contents of this chapter are based on the following paper:
  - Shuijing Liu, Peixin Chang, Zhe Huang, Neeloy Chakraborty, Kaiwen Hong, Weihang Liang, D. Livingston McPherson, Junyi Geng, and Katherine Driggs-Campbell. "Intention Aware Robot Crowd Navigation with Attention-Based Interaction Graph." In IEEE International Conference on Robotics and Automation (ICRA), 2023, pp. 12015-12021. URL: https://arxiv.org/abs/2203.01821.
- Chapter 5 presents the third version of our work for crowd navigation in constrained environments. We extend the st-graph to include static obstacles and environmental constraints in a scalable fashion, so that the robot can avoid collisions with both humans and obstacles.
  The contents of this chapter are based on the following paper:

[1]We have obtained permission from the publishers to reprint these articles.

- Shuijing Liu, Neeloy Chakraborty, Kaiwen Hong, and Katherine Driggs-Campbell. “Structured Attention Graph for Constrained Robot Crowd Navigation with Low Fidelity Simulation.” In preparation, 2024.

- Chapter 6 presents a mobile robot for Assistive navigation with dialogue and visual-language grounding. The contents of this chapter are based on the following paper:
  - Shuijing Liu, Aamir Hasan, Kaiwen Hong, Runxuan Wang, Peixin Chang, Zachery Mizrachi, Justin Lin, D. Livingston McPherson, Wendy A. Rogers, and Katherine Driggs-Campbell. “DRAGON: A Dialogue-Based Robot for Assistive Navigation with Visual Language Grounding.” IEEE Robotics and Automation Letters (RA-L), vol. 9, no. 4, pp. 3712-3719, 2024. URL: https://arxiv.org/abs/2307.06924.

We conclude the thesis and point out directions for future work in Chapter 7. A literature review is provided for each contribution in Chapter 2 to 6.

Chapter 2

Driver Trait Inference for T-Intersection Navigation

2.1 Introduction

To successfully navigate through uncontrolled intersections, autonomous vehicles must carefully reason about how to interact with different types of human drivers [11]. It is important for the vehicle to infer the traits of human drivers, such as the propensity for aggression or cooperation, and adjust its strategies accordingly [14], [23]. Inspired by recent advancements of unsupervised learning and deep reinforcement learning, we propose a novel pipeline to learn a representation of traits of other drivers, which is used for autonomous navigation in uncontrolled intersections.

Trait inference is challenging yet essential for the navigation of autonomous vehicles for two reasons. First, the environment is not fully observable to the ego vehicle, since each traffic participant runs its own policy individually and has its own internal states. The ego vehicle needs to interpret the hidden states such as driving styles and intents of other agents to understand future behaviors that may influence planning [14], [23]. Second, uncontrolled intersections are less structured since traffic lights and stop signs are not present to coordinate agent behaviors. The traffic participants implicitly interact and negotiate with each other, making the environment complex and potentially dangerous [24]–[27]. By inferring the traits of other drivers, the ego vehicle can be cautious when other drivers are aggressive or irrational and bold when they are passive or cooperative, improving both the safety and efficiency of interactive navigation.

To address the above problems, Morton *et al* learns a latent representation of driver traits, which is fed into a feedforward policy to produce multimodal behaviors [13]. However, the feedforward policy only considers current states and actions which are not sufficient to fully express long-term properties of drivers such as traits. As a result, this representation fails to distinguish between different traits. Ma *et al* classifies driver traits with supervised learning to aid navigation in intersections [14] but has the following two problems. First, the trait labels are expensive to obtain and usually do not exist in most real driving datasets [11], [12]. Second, the navigation policy is trained with ground truth trait labels instead of predicted traits. When the trait classifier and the policy are combined in testing, intermediate and cascading errors cause severe performance degradation.

In this chapter, we study the same uncontrolled T-intersection navigation problem as in [14], which is shown in Fig. 2.1. Before entering the T-intersection, the ego vehicle needs to infer the latent driving styles of

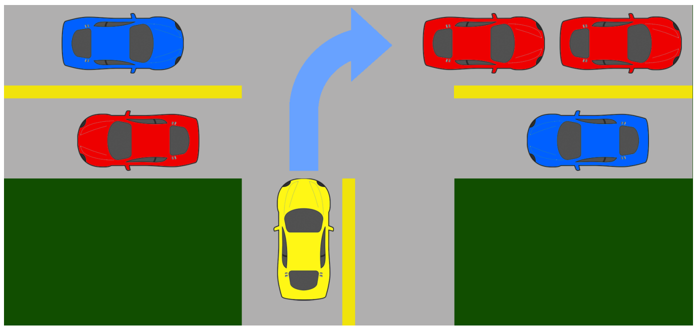


Figure 2.1: **The T-intersection scenario in left-handed traffic.** The goal of the ego car (yellow) is to take a right turn and merge into the upper lane without colliding with other cars. The conservative car (blue) yields to the ego car while the aggressive cars (red) do not yield.

other drivers to determine whether they are willing to yield to the ego vehicle. Based on the inferred driving style, the ego vehicle must intercept the drivers who will yield to achieve the goal.

We seek to create a pipeline that first learns a representation of driver traits in an unsupervised way, and then uses the trait representation to improve navigation in the T-intersection. In the first stage, we encode the trajectories of other drivers to a latent representation using a variational autoencoder (VAE) with recurrent neural networks (RNN). Since trajectory sequences reveal richer information about driver traits than single states, the VAE+RNN network effectively learns to distinguish between different traits without any explicit trait labels. In the second stage, we use the latent representations of driver traits as inputs to the ego vehicle's navigation policy, which is trained with model-free reinforcement learning (RL). With the inferred traits, the ego vehicle is able to adjust its decisions when dealing with different drivers, which leads to improved performance. Since the RL policy is trained with the learned representations instead of ground truth labels, our pipeline is much less sensitive to cascading errors from the trait inference network.

We present the following contributions:

1. We propose a novel unsupervised approach to learn a representation of driver traits with a VAE+RNN network;

2. We use the learned representation to improve the navigation of an autonomous vehicle through an uncontrolled T-intersection;

3. Our trait representation and navigation policy exhibit promising results and outperform previous works.

This chapter is organized as follows: We review related works in Section 2.2. We formalize the problem and propose our method in Section 2.3. Experiments and results are discussed in Section 2.4. We conclude the paper in Section 2.5.

2.2 Related Works

2.2.1 Driver internal state estimation

Driver internal state estimation can be divided into *intent estimation* and *trait estimation* [28]. Intent estimation often uses methods such as probabilistic graph model and unparameterized belief tracker to predict the future maneuvers of other drivers [26], [29], [30], which can then inform downstream planning for the ego driver [26], [30], [31]. Trait estimation infers the properties of drivers such as driving styles, preferences, fatigue, and level of distraction [28]. Some works distinguish between distracted and attentive drivers for behavior prediction and cooperative planning [32], [33]. Driving style recognition has been addressed with both unsupervised and supervised learning methods, which we will discuss in detail below [13], [14], [34], [35].

Morton *et al* propose a method that first encodes driving trajectories with different driving styles to a latent space. Then, the latent encodings and the current driver states are fed into a feedforward policy that produces multimodal actions [13]. The encoder and the policy are optimized jointly. However, since the feedforward policy only considers the relations between current states and actions, the joint optimization encourages the encoder to encode short-term information of the trajectories such as accelerations while ignoring the persistent properties such as traits. In contrast, since our method uses an RNN decoder to reconstruct the trajectories, the encoder must encode the trait information. Ma *et al* propose a graph neural network that classifies the driving styles with supervised learning, which requires large amounts of labeled data and is difficult to scale in reality [14]. In addition, since the definitions of trait properties vary by individuals and cultures, manually labeled trait information will likely be noisy and inconsistent. In contrast, our method is unsupervised and does not need any trait labels.

2.2.2 Representation learning for sequential data

Contrastive learning is widely used to learn representations from sequential data such as videos and pedestrian trajectories [36]–[38]. However, the performance of contrastive learning is sensitive to the quality of negative samples, and finding efficient negative sampling strategies remains an open challenge [39].

Another category of representation learning methods is VAE and its variants [40], [41]. Bowman *et al* introduce an RNN-based VAE to model the latent properties of sentences [42], which inspires us to learn the traits of drivers from their trajectories. Conditional VAEs (CVAE) are widely used in pedestrian and vehicle trajectory predictions since discrete latent states can represent different behavior modes such as braking and turning [43]–[46]. While these behavior modes change frequently, the driver traits that we aim to learn are persistent with each driver [28]. Also, the goal of these CVAEs is to generate multimodal trajectory predictions while, to the best of our knowledge, our work is the first to infer driver traits with VAE for autonomous driving.

2.2.3 Autonomous driving in uncontrolled intersections

Autonomous navigation through uncontrolled intersections is well studied and has had many successful demonstrations [14], [26], [47], [48]. Some heuristic methods use a time-to-collision (TTC) threshold to decide when to cross [48], [49]. However, the TTC models assume constant velocity for the surrounding vehicles, which ignores the interactions and internal states of drivers. Also, the TTC models can be overly cautious and cause unnecessary delays [47]. Another line of works formulates the problem as a partially observable Markov decision process (POMDP), which accounts for the uncertainties and partial observability in the intersection scenario but is computationally expensive to solve [26], [50], [51]. RL-based methods use neural networks as function approximators to learn driving policies. Isele *et al* learns to navigate in occluded intersections using deep Q-learning [47]. Ma *et al* focuses on navigation in a T-Intersection where the drivers exhibit different traits [14]. However, the navigation policy is trained with ground truth traits and only uses the inferred traits in testing. Thus, the performance of the RL policy is sensitive to the intermediate errors from the trait inference module. In contrast, our method trains the RL policy directly with inferred trait representations, which eliminates the problems caused by the intermediate errors.

2.3 Methodology

In this section, we first present our unsupervised approach to represent driver traits from unlabeled driving trajectories with a VAE+RNN network. Then, we discuss how we use the inferred traits to improve the navigation policy.

2.3.1 Preliminaries

Consider a T-intersection environment with an ego vehicle and n surrounding vehicles. The number of surrounding vehicles n may change at any timestep t. Suppose that all vehicles move in a 2D Euclidean space. Let o_0^t be the state of the ego vehicle and o_i^t be the observable state of the i-th surrounding vehicle at time t, where $i \in \{1, ..., n\}$. The state of the ego vehicle o_0^t consists of the position (p_x, p_y) and the velocity (v_x, v_y) of the vehicle. The observable state of each surrounding vehicle o_i^t consists of its position (p_x^i, p_y^i). In contrast to the previous work [14], o_i^t does not include other vehicles' velocities because they are hard to accurately measure without special facilities and algorithms in the real world [52], [53]. In addition, each surrounding vehicle has a latent state z_i that indicates the aggressiveness (i.e., the trait) of the i-th driver. We assume that the driving style of each driver z_i does not change throughout an episode. The positions of surrounding vehicles $o_1^t, ..., o_n^t$ are observable to the ego vehicle, while the latent states $z_1, ..., z_n$ are not.

2.3.2 Trait representation learning

Data collection

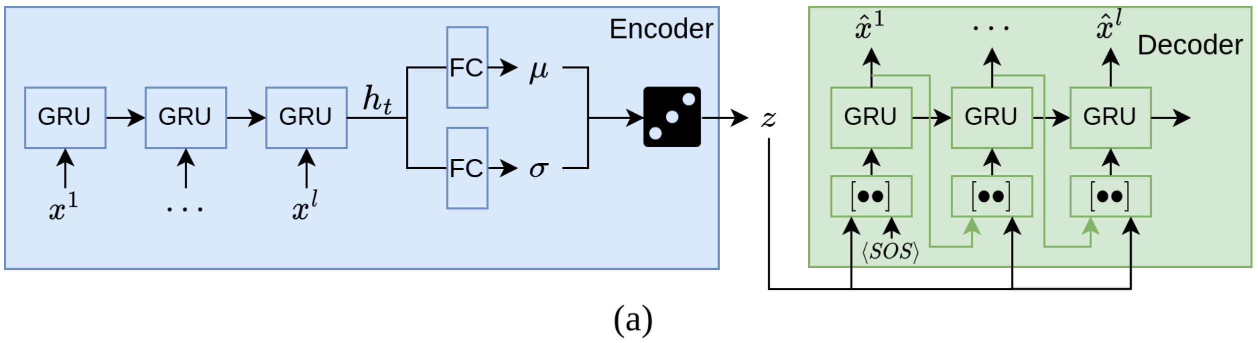
For each vehicle in the T-Intersection, the vehicle in front of it has the most direct influence on its behaviors. For this reason, in the trait representation learning stage, we define the observable state of each vehicle to be $x = (\Delta p_x, \Delta p_{x,f})$, where Δp_x is the horizontal offset from the vehicle's starting position in the trajectory, and $\Delta p_{x,f}$ is the horizontal displacement of the vehicle from its front vehicle. Then, the trajectory of each driver is a sequence of states $\mathbf{x} = [x^1, ..., x^l]$, where l is the length of the trajectory.

We only keep the longitudinal state information because all vehicles move in horizontal lanes and the lateral states are not insightful in this setting, except indicating which lane the vehicle is in. We rotate the trajectories so that all trajectories in the dataset are aligned in the same direction. Thus, the lane and the directional information is indistinguishable within the trajectory data, allowing the network to focus on learning the trait difference instead of other differences between vehicles.

To collect the trajectory data, we run a simulation of the T-intersection scenario without the presence of the ego car and record the trajectories of all surrounding vehicles, which are controlled by the Intelligent Driver Model (IDM) [54]. Learning trait representations from this dataset allows the ego car to infer the traits from the trajectories of other drivers before deciding to intercept or wait. The dataset is denoted as $\{\mathbf{x}_j\}_{j=1}^N$, where N is the total number of trajectories.

Network architecture

We use the collected dataset to train the VAE+RNN network to learn a representation of traits, as shown in Fig. 2.2a. The VAE network consists of an encoder, which compresses a trajectory $\mathbf{x}$ to a distribution of a



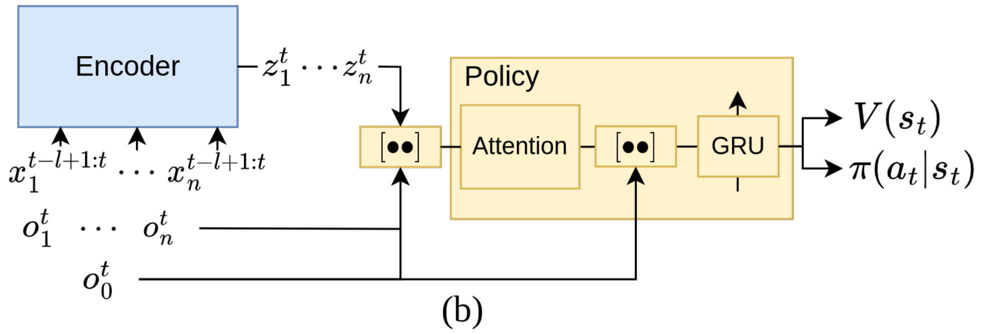


Figure 2.2: **The network architectures.** (a) The VAE+RNN network for trait representation learning. The fully connected layers before and after GRUs are eliminated for clarity. The dice represents the reparameterization trick. The start-of-sequence state is denoted by $\langle SOS \rangle$. (b) The navigation policy network. The weights of the encoder (blue) is fixed and only the yellow part is trained with RL. We use [●●] to denote concatenation. For illustration purposes, we assume that the inferred trait $z_1, ..., z_n$ is updated at time t.

latent trait vector z, and a decoder, which reconstructs the trajectory from the latent vector z. Both the encoder and the decoder are gated recurrent unit (GRU) networks since GRUs are more computationally efficient than long short-term memory networks (LSTM).

Given a trajectory $\mathbf{x} = [x^1, ..., x^l]$, the encoder GRU first applies a non-linear embedding layer f_{encoder} to each state x^t and then feeds the embedded features to the GRU cell:

$$h_e^t = \text{GRU}\left(h_e^{t-1}, f_{\text{encoder}}(x^t)\right) \tag{2.1}$$

where h_e^t is the hidden state of the encoder GRU at time $t \in \{1, ..., l\}$. After the entire trajectory is fed through the encoder GRU, we take the last hidden state h_e^l as the encoded latent feature of the trajectory $\mathbf{x}$ and apply fully connected layers f_μ and f_σ to get the Gaussian parameters of the latent driving style $z \in \mathbb{R}^2$ in a two-dimensional latent space:

$$\mu = f_\mu(h_e^t), \quad \sigma_i = f_\sigma(h_e^t). \tag{2.2}$$

Finally, we use the reparameterization trick to sample z from $\mathcal{N}(\mu, \sigma)$ for efficient learning: $z = \mu + \epsilon\sigma, \epsilon \sim \mathcal{N}(0, I)$.

In the decoding stage, since the driving style of each driver does not change over time, we treat the latent state z as part of the vehicle state instead of the initial hidden state of decoder GRU. To this end, at each timestep t, we concatenate the reconstructed state $\hat{x}^{t-1}$ from the last timestep and the latent state z from the encoder. Then, we embed the joint states using f_{decoder}, feed the embeddings into the next decoder GRU cell, and apply another embedding g_{decoder} to the next hidden state h_d^t, which outputs the next reconstructed state $\hat{x}^t$:

$$\begin{aligned} h_d^t &= \text{GRU}\left(h_d^{t-1}, f_{\text{decoder}}([\hat{x}^{t-1}, z])\right) \\ \hat{x}^t &= g_{\text{decoder}}(h_d^t). \end{aligned} \tag{2.3}$$

In the first timestep, we use a special start-of-sequence (SOS) state, which is similar to the start-of-sequence symbol in natural language processing to reconstruct $\hat{x}^1$ [55]. The process in Eq. 2.3 repeats until we reconstruct the whole trajectory $\hat{\mathbf{x}} = [\hat{x}^1, ..., \hat{x}^l]$.

The objective for training our VAE+RNN network is

$$\mathcal{L} = \beta D_{KL}\left(\mathcal{N}(\mu, \sigma)||\mathcal{N}(0, I)\right) + ||\mathbf{x} - \hat{\mathbf{x}}||_2 \tag{2.4}$$

where D_{KL} is the Kullback–Leibler (KL) divergence. The first term regularizes the distribution of the latent trait z to be close to a prior with a standard normal distribution. The second term is the reconstruction loss and measures the $L2$ error of the reconstructed trajectories from the original trajectories. The two terms are summed with a weight β.

By optimizing Eq. 2.4, our network learns latent encodings that represent the trait of each trajectory without any ground truth trait labels. Note that we also make no assumptions on the number of trait classes or the semantic meanings of the trait classes. Thus, our network has the potential to generalize to real trajectory datasets with more complex traits.

2.3.3 Navigation policy learning

We model the T-intersection scenario as a POMDP, defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathcal{P}, \gamma \rangle$. Suppose that there are n surrounding vehicles at timestep t. We use $o_t = [o_0^t, o_1^t, ..., o_n^t] \in \mathcal{O}$ to denote the observations of the ego vehicle, where $\mathcal{O}$ is the observation space. Let $u_i^t = [o_i^t, z_i] \in \mathbb{R}^4$ be the state of the i-th surrounding vehicle. Since the ego vehicle is influenced by all surrounding vehicles, we use $s_t = [o_0^t, u_1^t, ..., u_n^t] \in \mathcal{S}$ to denote the state of the POMDP, where $\mathcal{S}$ is the state space. And $\mathcal{P} : \mathcal{S} \rightarrow \mathcal{O}$ is the set of conditional observation probabilities.

At each timestep t, the ego vehicle chooses the desired speed for the low-level controller $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|s_t)$, where $\mathcal{A} = \{0, 0.5, 3\}$m/s is the action space. In return, the agent receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $\mathcal{T}(\cdot|s_t, a_t)$. Meanwhile, all other vehicles also take actions according to their policies and move to the next states with unknown state transition probabilities. The episode continues until t exceeds the maximum episode length T, the ego vehicle reaches its goal, or the ego vehicle collides with any other vehicle. The goal of the agent is to maximize the expected return, $R_t = \mathbb{E}[\sum_{i=t}^{T} \gamma^{i-t} r_i]$, where γ is a discount factor. The value function $V^\pi(s)$ is defined as the expected return starting from s, and successively following policy π.

To handle the unknown transitions and environment complexity, we train our policy network illustrated in Fig. 2.2b using model-free deep RL. During RL training, we fix the trainable weights of the encoder. For every l timesteps, we feed the trajectories of the surrounding vehicles over the past l timesteps to the encoder and infer the driving style z_i^t of each driver i. To improve efficiency, we only update the latent states of the drivers in the lanes that the ego car has not passed yet, as shown in Fig. 2.5. Since the ego car is not allowed to move backward, the latent states of the drivers that the ego car has already passed or the drivers that have already yielded to the ego car have no effect on the ego car's decisions and thus are not updated anymore.

Our policy network is a GRU with an attention module. We concatenate the state of each driver u_i^t

with the state of the ego vehicle o_0 to obtain $q_i^t = [u_i^t, o_0^t]$, where $q_i^t \in \mathbb{R}^8$ and $i \in \{1, ..., n\}$. We feed each concatenated state q_i^t into an attention module which assigns attention weights to each surrounding vehicle. Specifically, we embed q_i^t using a multi-layer perceptron (MLP) denoted as f_{emb} and calculate the mean of the embeddings of each surrounding vehicle:

$$m^t = \frac{1}{n}\sum_{i=1}^{n} e_i^t, \quad e_i^t = f_{emb}(q_i^t) \tag{2.5}$$

where m^t is the resulting mean. The weighted feature of each surrounding vehicle c_i^t is calculated by

$$c_i^t = \alpha_i^t \cdot e_i^t, \quad \alpha_i = f_{attn}([e_i^t, m^t]) \tag{2.6}$$

where $\alpha_i^t \in \mathbb{R}$ is the attention score for the i-th vehicle, and f_{attn} is another MLP. We then concatenate the sum of $c_1^t, ..., c_n^t$ with the state of the ego vehicle o_0^t and feed the result to a GRU:

$$h_\pi^t = \text{GRU}\left(h_\pi^{t-1}, \left[\sum_{i=1}^{n} c_i^t, o_0^t\right]\right) \tag{2.7}$$

Finally, the hidden state of the GRU h_π^t is fed to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$. We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [56].

2.4 Experiments and results

In this section, we first describe the simulation environment for training. We then present our experiments and results for trait representation and navigation policy.

2.4.1 Simulation environment

Fig. 2.5 shows our simulation environment adapted from [14]. At the beginning of an episode, the surrounding vehicles are randomly placed in a two-way street with opposite lanes and we assume that they never take turns or change lanes. The number of surrounding vehicles varies as vehicles enter into or exit from the T-intersection. We assume that all cars can always be detected and tracked. The surrounding vehicles are controlled by IDM [54]. Conservative drivers vary their front gaps from the preceding vehicles between $0.5m$ and $0.7m$ and have the desired speed of $2.4m/s$. Aggressive drivers vary their front gaps between $0.3m$ and $0.5m$ and have the desired speed of $3m/s$. If the ego car moves in front of other cars, conservative drivers will yield to the ego car, while aggressive drivers will ignore and collide with the ego car.

The ego car starts at the bottom of the T-intersection. The goal of the ego car is to take a right turn to merge into the upper lane without colliding with other cars. The ego car is controlled by a longitudinal PD controller whose desired speed is set by the RL policy network. The right-turn path of the ego car is fixed to follow the traffic rule. The ego car also has a safety checker that clips the magnitude of its acceleration if it gets dangerously close to other cars.

Let S_{goal} be the set of goal states, where the ego vehicle successfully makes a full right-turn, and S_{fail} be the set of failure states, where the ego vehicle collides with other vehicles. Let $r_{speed}(s) = 0.05 \times \|v_{ego}\|_2$ be a small reward on the speed of the ego vehicle and $r_{step} = -0.0013$ be a constant penalty that encourages the ego vehicle to reach the goal as soon as possible. The reward function is defined as

$$r(s,a) = \begin{cases} 2.5, & \text{if } s \in S_{goal} \\ -2, & \text{if } s \in S_{fail} \\ r_{speed}(s) + r_{step}, & \text{otherwise.} \end{cases} \tag{2.8}$$

2.4.2 Trait inference

Baseline

We compare the latent representation of our method with the method proposed by Morton *et al* [13]. The

Table 2.1: Testing results of simple supervised classifiers with learned latent representations

Method	Accuracy
Ours	**98.08%**
Morton *et al.*	60.22%

encoder of the baseline is the same as our encoder except that the baseline takes the longitudinal acceleration at each timestep as an extra input. The policy network of the baseline is a 4-layer multilayer perceptron (MLP) network that imitates the IDM policy from the trajectory dataset.

Training and evaluation

Our dataset contains approximately 696,000 trajectories from both classes, and the train/test split ratio is 2:1. We train both methods for 1000 epochs with a decaying learning rate 5×10^{-4}. The weight of the KL divergence loss β is 5×10^{-8} for both methods.

To better understand the learned latent representation and how it provides trait information to the navigation policy, we fix the trainable weights of the encoders, train linear support vector classifiers using the inferred latent states as inputs, and record the testing accuracy of the classifiers. To visualize the learned representations, we feed a set of testing trajectories into the encoder and visualize their latent representations.

Results and interpretability

In Table 2.1, the supervised classifier with our latent representation achieves much higher classification accuracy than that of Morton *et al.* Together with Fig. 2.3a, we show that our representation successfully separates the vehicle trajectories with different traits in most cases. For example, in Fig. 2.3c, #1, #5, and #6 are in the aggressive cluster, while #2, #7, and #8 are in the conservative cluster. The overlapped region encodes the trajectories with ambiguous traits, such as very short trajectories (#3) and the trajectories with ambiguous front gaps. For example, the average front gap of #4 is between the aggressive #6 and the conservative #8. Besides the binary traits, our latent representation also captures other meaningful information. First, the trajectories of the cars whose front cars go out of the border, such as #1 and #2, are mapped into the fan-shaped peripherals of the two clusters respectively. Second, in the central regions of the two clusters, the trajectories with larger average speeds are mapped in the lower left part (#5 and #7), while those with smaller average speeds are in the upper right part (#6 and #8).

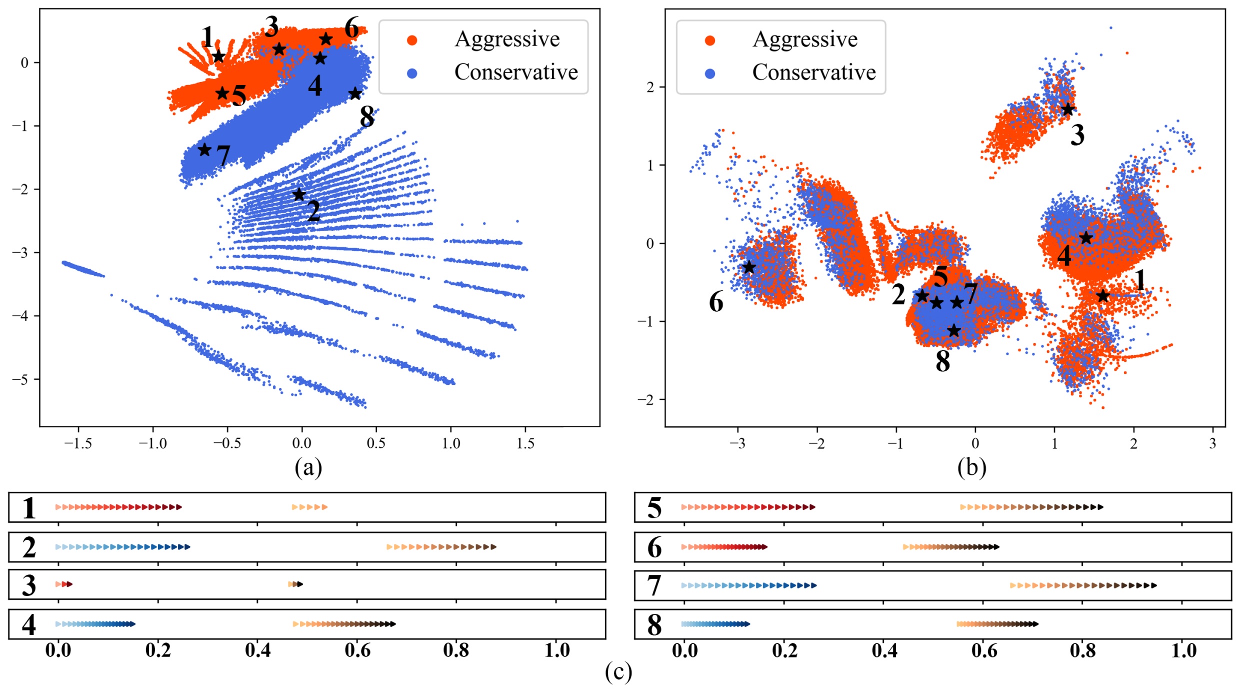


Figure 2.3: **Visualizations of the latent representations of two driver traits.** (a) Our method. (b) The method by Morton *et al.* (c) The original processed trajectories corresponding to the labeled latent vectors in (a) and (b). The x-axis is the horizontal displacement in meters. Each triangle marker indicates the position of a car at each timestep and the markers become darker over time. Denser triangles indicate smaller velocities. The blue and red trajectories indicate conservative and aggressive cars respectively. The brown trajectories indicate the front cars. If the front car goes out of the boundary before the trajectory ends, the brown trajectories will be shorter than the red or blue trajectories, such as #1 and #2.

From Table 2.1 and Fig. 2.3b, the baseline suffers from severe model collapse and the representation fails to separate the two traits. The reason is that the MLP policy only considers current state-action pairs, which encourages the encoder to only encode features within short time windows such as accelerations while ignoring the properties of the trajectories such as traits. Despite the model collapse, the baseline still learns some meaningful representations. For example, the trajectories with uniform speeds together such as #5 and #7 and forms separate clusters for the decelerating trajectories such as #6. Therefore, we conclude that compared with single states, trajectories exhibit richer information about traits and are better suited for trait representation learning.

2.4.3 Navigation with inferred traits

Baselines and ablations

We use the following two baselines: (1) The pipeline proposed by Ma *et al*, which trains a supervised trait predictor and an RL policy with binary ground truth trait labels separately and combines them at test time [14]; (2) We use the latent representation by Morton *et al* to train a policy network using the same method described in Sec. 2.3.3. In addition, we use an RL policy trained with ground truth labels as an oracle, and another RL policy trained without any trait information as a naïve model. To examine the effectiveness of the attention mechanism, we train an ablated model of our method without the attention module. For a fair comparison, the architectures of all policy networks are kept the same.

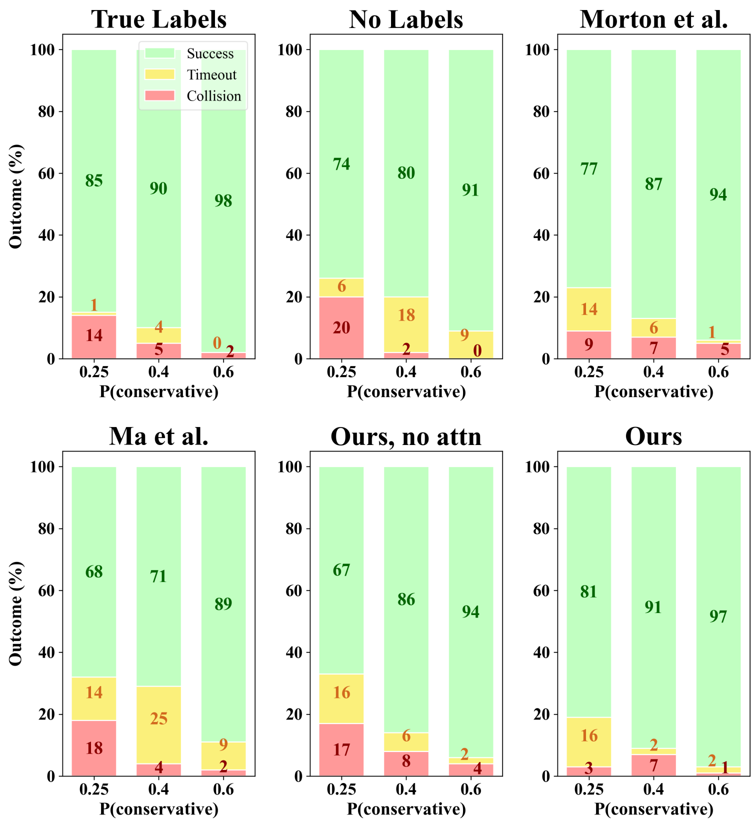


Figure 2.4: **Success, timeout, and collision rates w.r.t. different driver trait distributions.** P(conservative) is the probability for each surrounding driver to be conservative. The task difficulty increases as P(conservative) decreases. The numbers on the bars indicate the percentages of the corresponding bars.

Training

We run three experiments with different proportions of two types of drivers, as shown in Fig. 2.4. We train the policy networks for all methods and ablations for 1×10^7 timesteps with a decaying learning rate 1×10^{-4}. To accelerate and stabilize training, we run twelve instances of the simulation environment in parallel for collecting the ego car's experiences. At each policy update, 30 steps of 6 episodes are used. For Ma *et al*, we pretrain a trait classification network with a 96% testing accuracy and use the classifier to infer traits for the RL policy.

Evaluation

We test all models with 500 random unseen test cases. We measure the percentage of success, collision, and timeout episodes as the evaluation criteria.

Results

As shown in Fig. 2.4, the success rates of our method are closest to the oracle who has access to true trait labels in all experiments, with an average difference of 2%. We believe the main reason is that our trait

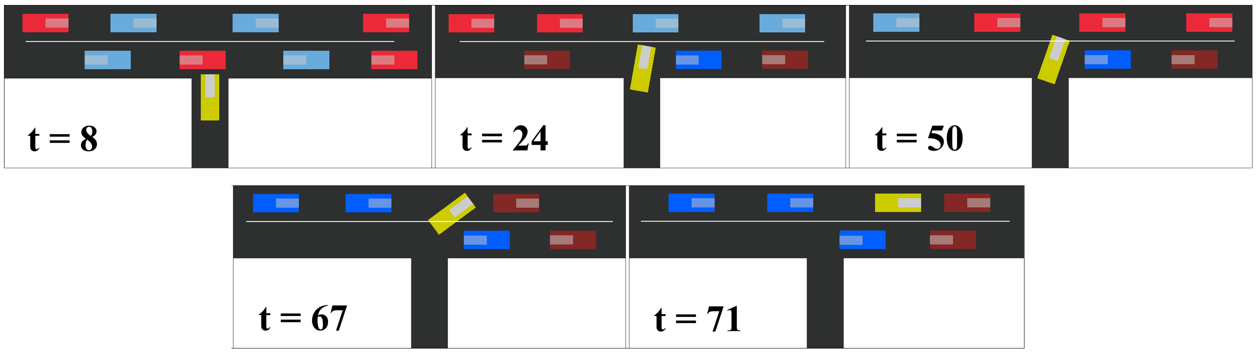


Figure 2.5: **Qualitative result of our method.** The ego car is in yellow, the conservative cars are in blue, and the aggressive cars are in red. As mentioned in Sec. 2.3.3, the latent traits of the light blue and light red cars are updated periodically, while those of the dark blue and dark red cars are not updated and stay the same as before.

representation effectively captures the trait differences of the surrounding drivers and aids the decision-making in RL. The performance gap between the oracle and our method is caused by the drivers with ambiguous traits, as well as the fact that the learned representation is noisier than the true labels. Although the model with no labels can implicitly infer traits to some extent in RL training, our policy is able to utilize the existing trait representation and focuses more on the decision-making of the ego vehicle, which leads to better navigation performance. Fig. 2.5 shows a successful episode of our method, where the ego car waits until a conservative car appears, cuts in the front of the conservative cars when passing both lanes, and completes the right turn.

Compared with our model, the model by Morton *et al* has a lower success rate especially in more challenging experiment settings when $P(\text{conservative})$ is smaller. The reason is that the latent representation does not distinguish between different traits and only provides very limited useful information to RL. This implies that the policy still needs to implicitly infer the traits while being distracted by the latent representation, which negatively affects the performance.

For Ma *et al*, the trait classifier and the RL policy both have good performance when tested separately. However, when the two modules are combined together, intermediate and cascading errors significantly lower the success rates. The performance drop is due to the distribution shift between the true traits and inferred traits. Since the policy is trained with true traits, it fails easily whenever the trait classifier makes a small mistake. Moreover, Ma *et al* requires trait labels, but our trait representation is learned without any labels and still outperforms Ma *et al* in navigation.

2.4.4 Attention vs. no attention

The second from the rightmost graph in Fig. 2.4 shows that the removal of the attention module results in $3\% \sim 14\%$ lower success rates. Since the attention module assigns different weights to the cars with different traits and in different positions, the policy is able to focus on the cars which have a bigger influence on the ego car, which validates the necessity of the attention mechanism.

2.5 Summary

We propose a novel pipeline that encodes the trajectories of drivers to a latent trait representation with a VAE+RNN network. Then, we use the trait representation to improve the navigation of an autonomous vehicle through an uncontrolled T-intersection. The trait representation is learned without any explicit supervision. Our method outperforms baselines in the navigation task and the trait representation shows interpretability. Possible directions to explore in future work include (1) validating our method with real driving trajectory data, (2) generalizing our model to more sophisticated driver internal states and more navigation tasks, and (3) incorporating occlusions and limited sensor range of the ego vehicle to close the gap between the simulation and the real world.

Code implementation and videos are available at https://github.com/Shuijing725/VAE_trait_inference.

Chapter 3

Decentralized Structural-RNN for Robot Crowd Navigation

3.1 Introduction

As mobile robots are becoming prevalent in people's daily lives, autonomous navigation in crowded places with other dynamic agents is an important yet challenging problem [57], [58]. Inspired by the recent applications of deep learning in robot control [59]–[62] and in graph modeling [63], we seek to build a learning-based graphical model for mobile robot navigation in pedestrian-rich environments.

Robot crowd navigation is a challenging task for two key reasons. First, the problem is decentralized, meaning that each agent runs its own policy individually, which makes the environment not fully observable to the robot. For example, other agents' preferred walking style and intended goals are not known in advance and are difficult to infer online [64]. Second, the crowded environment contains both dynamic and static agents, who implicitly interact with each other during navigation. The ways agents influence each other are often difficult to model [65], making the dynamic environment harder to navigate and likely to produce emergent phenomenon [66].

Despite these challenges, robot crowd navigation is well-studied and has had many successful demonstrations [16], [67], [68]. Reaction-based methods such as Optimal Reciprocal Collision Avoidance (ORCA) and Social Force (SF) use one-step interaction rules to determine the robot's optimal action [5], [68], [69]. Another line of works first predict other agents' future trajectories and then plan a path for the robot [70]–[73]. However, these two methods suffer from the *freezing robot problem*: in dense crowds, the planner decides that all paths are unsafe and the robot freezes, which is suboptimal as a feasible path usually exists [7].

More recently, learning-based methods model the robot crowd navigation as a Markov Decision Process (MDP) and use Deep V-Learning to solve the MDP [15], [16], [74]–[76]. In Deep V-Learning, the agent chooses an action based on the state value approximated by neural networks. However, Deep V-Learning is typically initialized by ORCA using supervised learning and, as a result, the final policy inherits ORCA's aforementioned problems. Moreover, to choose actions from the value network, the dynamics of the humans are assumed to be known to the robot and are deterministic, which can be unrealistic in real applications.

In this chapter, we seek to create a learning framework for robot crowd navigation using spatio-temporal reasoning trained with model-free deep reinforcement learning (RL). We model the crowd navigation scenario as a decentralized spatio-temporal graph (st-graph) to capture the interactions between the robot and

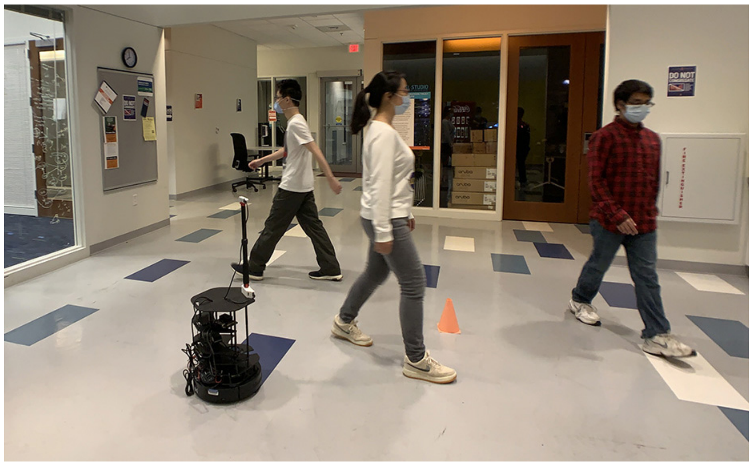


Figure 3.1: **Real-world crowd navigation with a TurtleBot 2i.** The orange cone on the floor denotes the robot goal. The TurtleBot is equipped with cameras for localization and human tracking.

multiple humans through both space and time. Then, we convert the decentralized st-graph to a novel end-to-end decentralized structural-RNN (DS-RNN) network. Using model-free RL, our method directly learns a navigation policy without prior knowledge of any agent's dynamics or expert policies. Since the robot learns entirely from its own experience, the resulting navigation policy easily adapts to dense human crowds and partial observability and outperforms previous methods in these scenarios.

We present the following contributions: (1) We propose a novel deep neural network architecture called DS-RNN, which enables the robot to perform efficient spatio-temporal reasoning in crowd navigation; (2) We train the network using model-free RL without any supervision, which both simplifies the learning pipeline and avoids the network from converging to a suboptimal policy too early; (3) Our method demonstrates better performance in challenging navigation settings compared with previous methods. For more details, our code is available at `https://github.com/Shuijing725/CrowdNav_DSRNN`.

3.2 Related Works

3.2.1 Reaction-based crowd navigation methods

Robot navigation in dynamic environments has been studied for over two decades [67], [77]–[79]. A subset of these works specifically focuses on robot navigation in pedestrian-rich environments or crowd navigation [57], [80].

Reaction-based methods such as Reciprocal Velocity Obstacle (RVO) and ORCA model other agents as velocity obstacles to find optimal collision-free velocities under reciprocal assumption [5], [68], [81]. Another method named Social Force models the interactions in crowds using attractive and repulsive forces [69]. However, these algorithms suffer from the *freezing robot problem* [7]. In addition, since the robot only uses the current states as input, the generated paths are often shortsighted and unnatural.

In contrast, we train our network with model-free RL to mitigate the freezing problem. Also, our network contains RNNs that take a sequence of trajectories as input to encourage longsighted behaviors.

3.2.2 Learning-based crowd navigation methods

With the recent advancement of deep learning, imitation learning has been used to uncover policies from demonstrations of desired behaviors [82], [83]. Another line of works use Deep V-Learning, which combines supervised learning and RL [15], [16], [74]–[76], [84]. Given the state transitions of all agents, the planner first calculates the values of all possible next states from a value network. Then, the planner chooses an action that leads to the state with the highest value. To train the value network, Deep V-Learning first initializes the network by supervised learning using trajectories generated by ORCA, and then fine-tunes the network with RL. Using a single rollout of the policy, Monte-Carlo value estimation calculates the ground-truth values for both supervised learning and RL.

Deep V-Learning has demonstrated success in simulation and/or in the real world but still suffers from the following drawbacks: (1) Deep V-Learning assumes that the state transitions of all surrounding humans are known and well-defined, which are in fact highly stochastic and difficult to model; (2) Since the networks are pre-trained with supervised learning, they share the same disadvantages with the demonstration policy, which are hard to be corrected by RL; (3) Monte-Carlo value estimation is not scalable with increasing time horizon; and (4) To achieve the best performance, Deep V-Learning needs state information of all humans. If applied to real robots, a real-time human detector with a 360° field of view is required, which can be expensive or impractical.

To tackle these problems, we introduce a policy network trained with model-free RL, which does not need state transitions, Monte-Carlo value estimation, or expert supervision. Further, we show that incorporating both spatial and temporal reasoning in our network improves performance in challenging navigation environments over prior methods.

3.2.3 Spatio-temporal graphs and structural-RNN

St-graph is a type of conditional random field [85] with wide applications [63], [86]–[88]. St-graphs use nodes to represent the problem components and edges to capture the spatio-temporal interactions [63]. With each node or edge governed by a factor function, st-graph decomposes a complex problem into many smaller and simpler factors.

Jain *et al* propose a general method called structural-RNN (S-RNN) that transforms any st-graph to a mixture of RNNs that learn the parameters of factor functions end-to-end [63]. S-RNNs have been applied to research areas such as human tracking and human trajectory prediction [89], [90]. However, the scope of these works is restricted to learning from static datasets. Applying st-graph to crowd navigation poses extra challenges in data collection and decision-making under uncertainty. Although some works in crowd navigation have used graph convolutional network [91] to model the robot-crowd interactions [76], [84], to the best of our knowledge, our work is the first to combine S-RNN with model-free RL for robot crowd navigation.

3.3 Methodology

In this section, we first formulate the robot decision making in crowd navigation as an RL problem. Then, we present our approach to model crowd navigation scenario as an st-graph, which leads to the derivation of our DS-RNN network architecture.

3.3.1 Problem formulation

Consider a robot interacting with an episodic environment with other humans. We model this interaction as an MDP, defined by the tuple $\langle \mathcal{S}, \mathcal{A}, P, R, \gamma, \mathcal{S}_0 \rangle$. Suppose that all agents move in a 2D Euclidean space. Let $\mathbf{w}^t$ be the robot states and $\mathbf{u}_i^t$ be the i-th human's states observable by the robot. Then, the state $s_t \in \mathcal{S}$ for the MDP is $s_t = [\mathbf{w}^t, \mathbf{u}_1^t, ..., \mathbf{u}_n^t]$ assuming a total number of n humans was involved at the timestep t. The robot state $\mathbf{w}^t$ consists of the robot's position (p_x, p_y), velocity (v_x, v_y), goal position (g_x, g_y), maximum speed v_{max}, heading angle θ, and radius ρ. Each human state $\mathbf{u}_i^t$ consists of the human's position (p_x^i, p_y^i). In contrast to previous works [5], [15], [16], [84], the human state $\mathbf{u}_i^t$ does not include the human's velocity and radius because they are hard to be measured accurately in the real world.

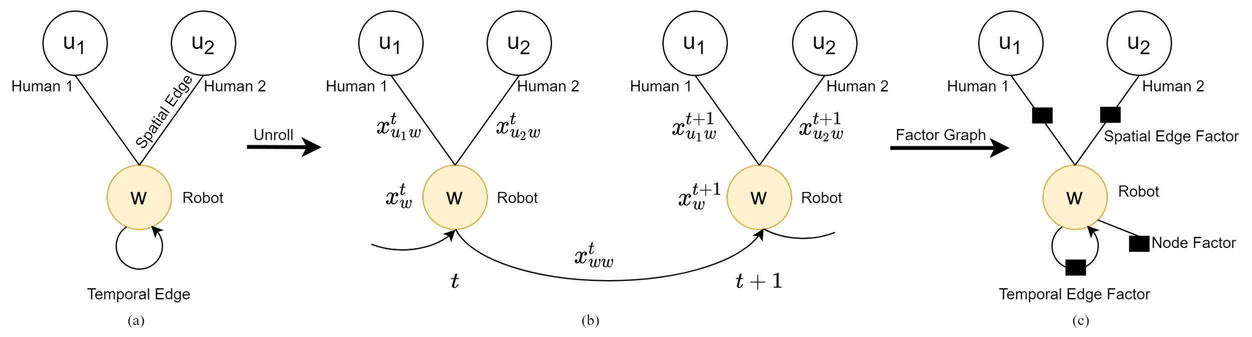


Figure 3.2: **Conversion from the st-graph to the factor graph.** (a) St-graph representation of the crowd navigation scenario. We use w to denote the robot node and u_i to denote the i-th human node. (b) Unrolled st-graph for two timsteps. At timestep t, the node feature for the robot is x_w^t. The spatial edge feature between the i-th human and the robot is $x_{u_i w}^t$. The temporal edge feature for the robot is x_{ww}^t. (c) The corresponding factor graph. Factors are denoted by black boxes.

In each episode, the robot begins at an initial state $s_0 \in \mathcal{S}_0$. At each timestep t, the robot takes an action $a_t \in \mathcal{A}$ according to its policy $\pi(a_t|s_t)$. In return, the robot receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $P(\cdot|s_t, a_t)$. Meanwhile, all other humans also take actions according to their policies and move to the next states with unknown state transition probabilities. The process continues until t exceeds the maximum episode length T, the robot reaches its goal, or the robot collides with any humans.

Let $\gamma \in (0, 1]$ be the discount factor. Then, $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ is the total accumulated return from timestep t. The goal of the robot is to maximize the expected return from each state. The value of state s

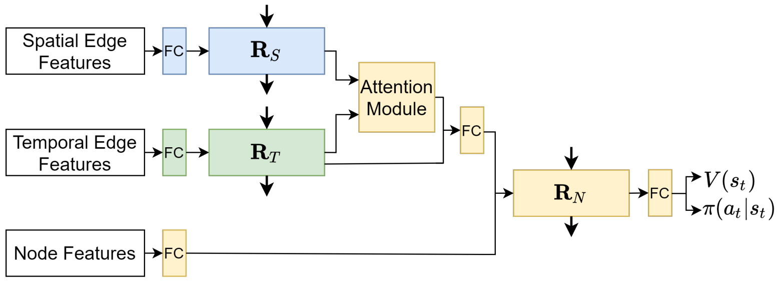


Figure 3.3: **DS-RNN network architecture.** The components for processing spatial edge features, temporal edge features, and node features are in blue, green, and yellow, respectively. Fully connected layers are denoted as FC.

under policy π, defined as $V(s) = \mathbb{E}[R_t | s_t = s]$, is the expected return for following policy π from state s.

3.3.2 Spatio-Temporal Graph Representation

We formulate the crowd navigation scenario as a decentralized st-graph. Our graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}_S, \mathcal{E}_T)$ consists of a set of nodes $\mathcal{V}$, a set of spatial edges $\mathcal{E}_S$, and a set of temporal edges $\mathcal{E}_T$. As shown in Fig. 3.2a, the nodes in the st-graph represent the agents, the spatial edges connect two different agents at the same timestep, and the temporal edges connect the same nodes at adjacent timesteps. We prune the edges and nodes not shown in Fig. 3.2a as they have little effect on the robot's decisions[1]. The corresponding unrolled st-graph is shown in Fig. 3.2b.

The factor graph representation of the st-graph factorizes the robot policy function into the robot node factor, spatial edge factors, and robot temporal edge factor. At each timestep, the factors take the node or edge features as inputs and collectively determine the robot's action. In Fig. 3.2c, factors are denoted by black boxes and have parameters that need to be learned.

We choose $x^t_{u_i w}$ to be the vector pointing from humans' position to the robot position, $(p^i_x - p_x, p^i_y - p_y)$, x^t_{ww} to be robot velocity (v_x, v_y), and x^t_w to be $\mathbf{w}^t$. To reduce the number of parameters, all spatial edges share the same factor. This parameter sharing is important for the scalability of our st-graph because the number of parameters is kept constant with an increasing number of humans [63].

3.3.3 Network Architecture

As shown in Fig. 3.3, we derive our network architecture from the factor graph representation of the st-graph motivated by [63]. In our network, we represent each factor with an RNN, referred to as spatial edgeRNN $\mathbf{R}_S$, temporal edgeRNN $\mathbf{R}_T$, and nodeRNN $\mathbf{R}_N$ respectively. We use W and f to denote trainable weights and fully connected layers throughout this section.

The spatial edgeRNN $\mathbf{R}_S$ captures the spatial interactions between humans and the robot. $\mathbf{R}_S$ first applies a non-linear transformation to each spatial feature $x^t_{u_i w}$ and then feeds the transformed results to the RNN cell:

$$h^t_{u_i w} = \text{RNN}\left(h^{t-1}_{u_i w}, f_{\text{spatial}}(x^t_{u_i w})\right) \tag{3.1}$$

where $h^t_{u_i w}$ is the hidden state of the RNN at time t for i-th human and the robot. Due to the parameter sharing mentioned in Section 3.3.2, the spatial edge features between all human-robot pairs are fed into the same spatial edgeRNN.

The temporal edgeRNN $\mathbf{R}_T$ captures the dynamics of the robot's own trajectory. Similar to $\mathbf{R}_S$, $\mathbf{R}_T$ applies a linear transformation to the temporal edge feature and processes the results with its RNN cell:

$$h^t_{ww} = \text{RNN}\left(h^{t-1}_{ww},\ f_{\text{temporal}}(x^t_{ww})\right) \tag{3.2}$$

where h^t_{ww} is the hidden state of the RNN at time t.

The outputs of two edgeRNNs are fed into an attention module which assigns attention weights to each spatial edge. The attention mechanism is similar to the *scaled dot product attention* in [92]. Let V^t be the output of $\mathbf{R}_S$ at time t, $V^t = [h^t_{u_1 w}, ..., h^t_{u_n w}]^\top$, where n is the number of spatial edges or humans. Both V^t

[1]From experiments, we find that a network derived from a full st-graph as in [63] performs very similarly to DS-RNN.

and h_{ww}^t are first put through linear transformations:

$$Q^t = V^t W_Q, \quad K^t = h_{ww}^t W_K \tag{3.3}$$

where $Q^t \in \mathbb{R}^{n \times d_k}$, $K^t \in \mathbb{R}^{1 \times d_k}$, and d_k is a hyperparameter for the attention size. The attention weight at time t, α^t, is calculated as

$$\alpha^t = \text{softmax}\left(\frac{n}{\sqrt{d_k}} Q^t (K^t)^\top\right) \tag{3.4}$$

The output of the attention module at time t, v_{att}^t, is the weighted sum of spatial edges:

$$v_{\text{att}}^t = (V^t)^\top \alpha^t \tag{3.5}$$

The nodeRNN $\mathbf{R}_N$ uses the robot state, x_w^t, the weighted hidden states of $\mathbf{R}_S$, v_{att}^t, and the hidden states of temporal edgeRNN, h_{ww}^t, to determine the robot action and state value at each time t. The nodeRNN concatenates v_{att}^t and h_{ww}^t and embeds the concatenated results and the robot state with linear transformations:

$$e^t = f_{\text{edge}}([v_{\text{att}}^t, h_{ww}^t]), \quad n^t = f_{\text{node}}(x_w^t) \tag{3.6}$$

Both e^t and n^t are concatenated and fed into $\mathbf{R}_N$ to get the nodeRNN hidden state.

$$h_w^t = \text{RNN}\left(h_w^{t-1}, [e^t, n^t]\right) \tag{3.7}$$

Finally, the h_w^t is input to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$. We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [56] and we adopt the PPO implementation from [93]. To accelerate and stabilize training, we run twelve instances of the environment in parallel for collecting the robot's experiences. At each policy update, 30 steps of six episodes are used.

By identifying the independent components of robot crowd navigation, we split the complex problem into smaller factors, and use three RNNs to efficiently learn the parameters of the corresponding factors. By combining all components above, the end-to-end trainable DS-RNN network performs spatial and temporal reasoning to determine the robot action.

3.4 Simulation Experiments

In this section, we describe the simulation environment for training and present our experimental results in simulation.

3.4.1 Simulation environment

Fig. 3.4 shows our 2D simulation environments adapted from [16]. We use holonomic kinematics for each agent, whose action at time t consists of the desired velocity along the x and y axis, $a_t = [v_x, v_y]$. All humans are controlled by ORCA with randomized maximum speed and radius. We assume that humans react only to other humans but not to the robot. This invisible setting prevents our model from learning an extremely aggressive policy in which the robot forces all humans to yield while achieving a high reward. We also assume that all agents can achieve the desired velocities immediately, and they will keep moving with these velocities

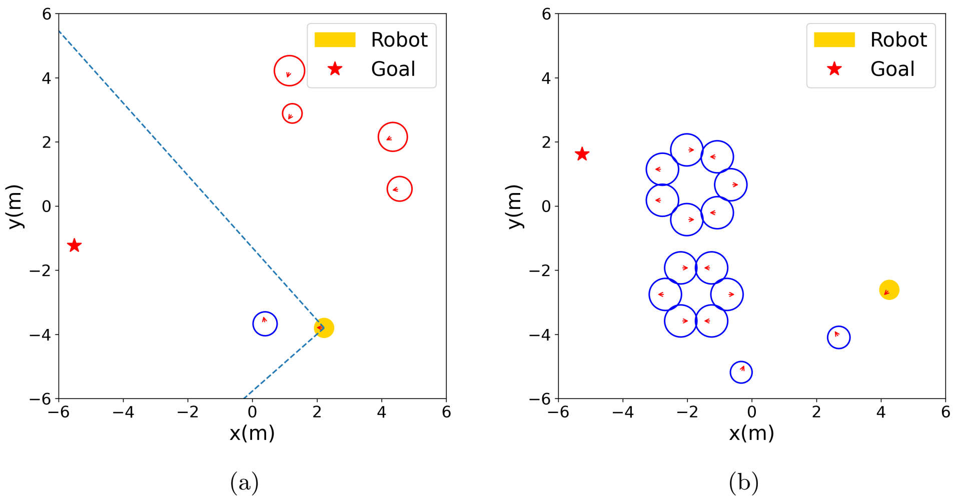


Figure 3.4: **Illustration of our simulation environment.** In a $12m \times 12m$ $2D$ plane, the humans are represented as circles, the orientation of an agent is indicated by a red arrow, the robot is the yellow disk, and the robot's goal is the red star. We outline the borders of the robot FoV with dashed lines. The humans in the robot's FoV are blue and the humans outside are red.

for Δt seconds. We define the update rule for an agent's position p_x, p_y as follows:

$$\begin{aligned} p_x[t+1] &= p_x[t] + v_x[t]\Delta t \\ p_y[t+1] &= p_y[t] + v_y[t]\Delta t \end{aligned} \tag{3.8}$$

Environment configurations

Fig. 3.4a shows the FoV Environment, where the robot's field view (FoV) is within 0° to 360° and remains unchanged in each episode. The robot assumes that the humans out of its view proceed in a straight line with their last observed velocities. There are always five humans, whose starting and goal positions are randomly placed on a circle with radius $6m$. The FoV Environment simulates the limited sensor range of a robot, since deploying several sensors to obtain a 360° FoV is usually expensive and unrealistic in the real world.

Fig. 3.4b shows the Group Environment, where the robot's FoV is 360° but the number of humans is large and remains the same in each episode. Among these humans, some form circle groups in random positions and do not move while the rest of them move freely. The Group Environment simulates the scene of a dense crowd with both static and dynamic obstacles. We use this environment to evaluate whether the robot policies have the *freezing robot problem*.

To simulate the variety of complexities in real-world crowd navigation, we add the following randomness and features not included in the original simulator from [16]. When an episode begins, the robot's initial position and goal are chosen randomly. In addition, all humans occasionally change their goal positions within an episode. Finally, to simulate a continuous human flow, immediately after humans arrive at their goal positions, they will move to new random goals instead of remaining stationary at their initial destinations.

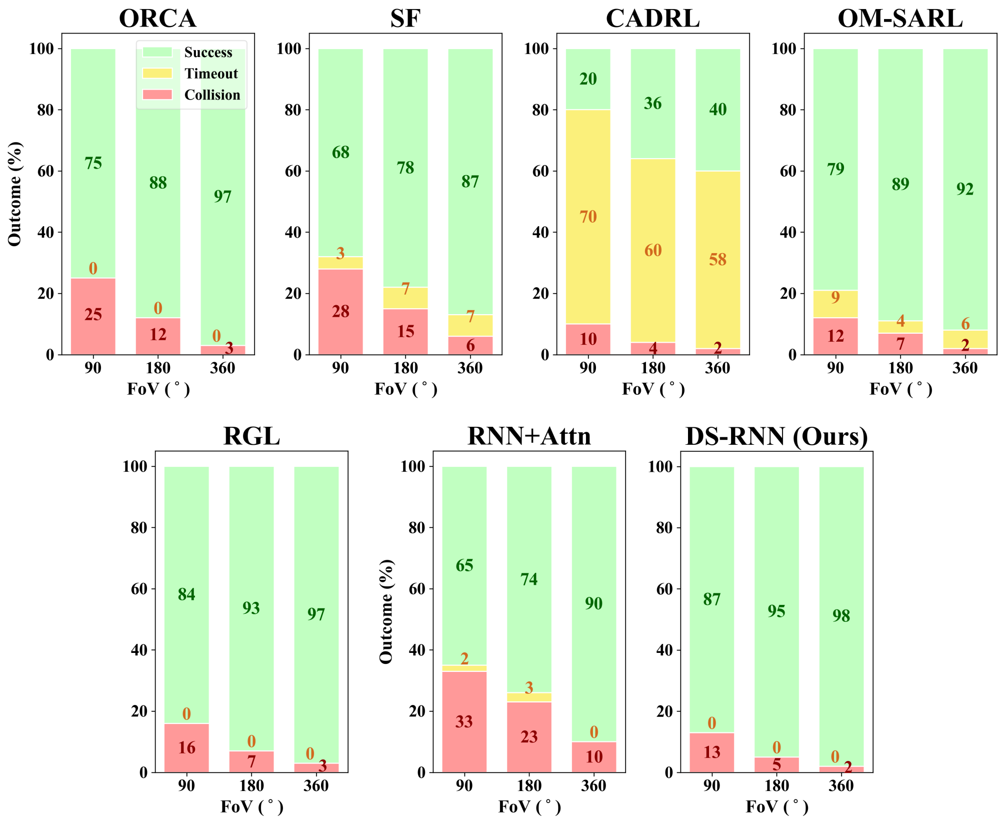


Figure 3.5: **Success, timeout, and collision rates w.r.t. different FoV.** The numbers on the bars indicate the percentages of the corresponding bars.

Reward function

The reward function awards the robot for reaching its goal and penalizes the robot for colliding with humans or getting too close to humans. In addition, we add a potential-based reward shaping to guide the robot to approach the goal:

$$r(s_t, a_t) = \begin{cases} -20, & \text{if } d^t_{min} < 0 \\ 2.5(d^t_{min} - 0.25), & \text{if } 0 < d^t_{min} < 0.25 \\ 10, & \text{if } d^t_{goal} \leq \rho_{robot} \\ 2(-d^t_{goal} + d^{t-1}_{goal}), & \text{otherwise.} \end{cases} \tag{3.9}$$

where d^t_{min} is the minimum separation distance between the robot and any human at time t, and d^t_{goal} is the $L2$ distance between the robot position and goal position at time t. Intuitively, the robot gets a high reward when it approaches the goal while maintaining a safe distance from all humans.

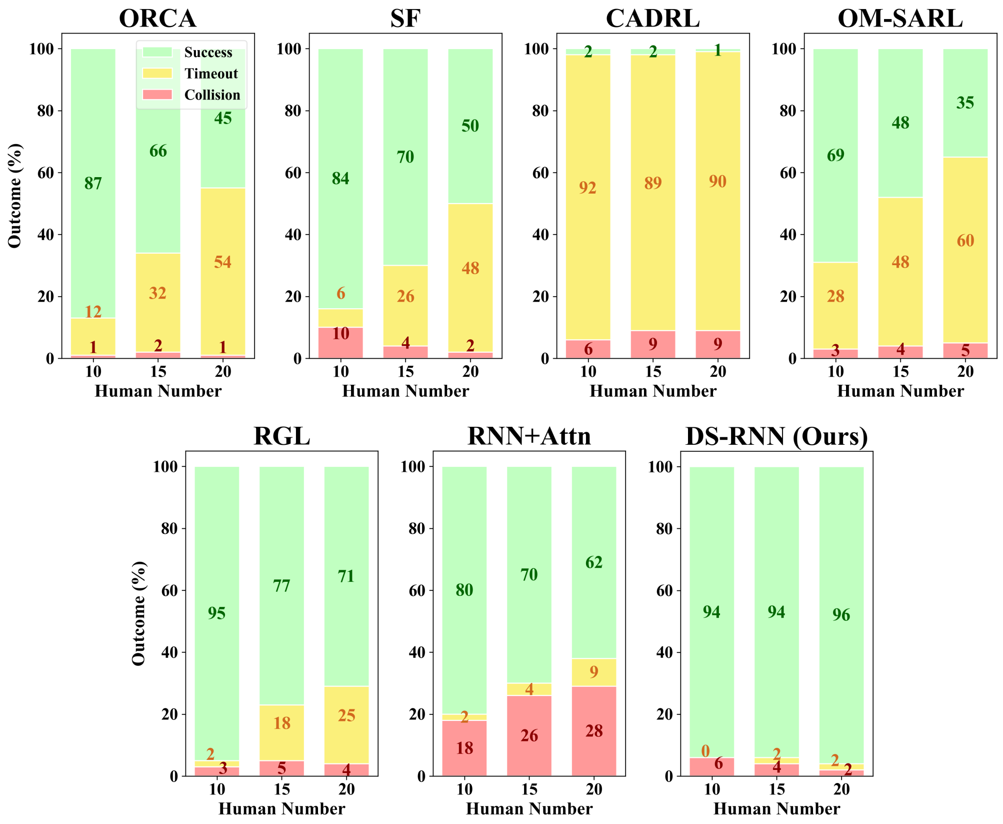


Figure 3.6: **Success, timeout, and collision rates w.r.t. different number of humans.**

3.4.2 Experiment setup

Baselines and Ablation Models

We compare the performance of our model with the representatives of the three types of methods in Section 3.2. We use ORCA and SF as the baselines for reaction-based methods; Relational Graph Learning (RGL) [84] as a baseline for both trajectory prediction based methods and Deep V-Learning; and CADRL [15] and OM-SARL [16] as the baselines for Deep V-Learning.

To remove the performance gain caused by other factors such as model-free RL and RNN, we also implement an ablation model, called RNN+Attn, by adding an RNN to the end of OM-SARL network. For RNN+Attn, the attention module assigns attention weights on the state features of humans. The weighted human features are then concatenated with robot state features to form the joint state features which are passed to an RNN network with the same size and sequence length as the robot nodeRNN in our model. Both networks are trained using PPO with the same hyper-parameters and thus the results serve as a clean comparison to highlight the benefits of our DS-RNN.

Training

We use the same reward as defined in Equation 3.9 for CADRL, OM-SARL, RGL, RNN+Attn, and DS-RNN. The network architectures of all methods are kept the same in all experiments. We train DS-RNN and RNN+Attn for 1×10^7 timesteps with a learning rate 4×10^{-5}. We train all baselines as stated in the original papers.

Table 3.1: Navigation time (second) in two environments.

Method	FoV			Number of Humans		
	90°	180°	360°	10	15	20
ORCA	**9.12**	9.96	10.40	15.94	19.09	19.66
SF	20.86	24.28	23.96	24.60	30.26	31.12
CADRL	30.84	27.54	33.46	32.62	36.81	41.75
OM-SARL	18.32	13.70	21.04	27.25	23.79	29.24
RGL	9.54	**9.48**	**9.59**	**13.22**	**14.75**	16.44
RNN+Attn	16.57	14.00	10.96	16.01	21.31	25.55
DS-RNN	11.83	10.99	11.79	13.51	15.64	**15.52**

Evaluation

We evaluate the performance of all the models with six experiments: for the FoV Environment, the FoV of the robot is 90°, 180°, or 360°; for the Group Environment, the number of humans is 10, 15, or 20. For each of the six experiments, we test all the models with 500 random unseen test cases. We measure the percentage of success, collision, and timeout episodes, as well as the average navigation time of the successful episodes.

3.4.3 Results

Spatio-temporal reasoning

We show the effectiveness of our DS-RNN architecture by comparing it with RNN+Attn. In Fig. 3.5 and Fig. 3.6, compared with RNN+Attn, our model exhibits higher success rates and lower collision and timeout rates in all settings. In Table 3.1, our model has shorter navigation time because DS-RNN often finds a better path (Fig. 3.7e and 3.7f). We believe the main reason is that by formulating the crowd navigation into an st-graph, we decompose the robot decision making into smaller factors and feed each RNN with only relevant edge or node features. In this way, the three RNNs are able to learn their corresponding factors more effectively. By combining all factors (RNNs), the robot is able to explicitly reason about the spatial relationships with humans and its own dynamics to take actions. In contrast, RNN+Attn does not have such spatio-temporal reasoning and learns all factors with one single RNN, which explains its lower performance.

Comparison with traditional methods

We compare the performance of our model with those of ORCA and SF. As shown in Fig. 3.6, in the Group Environment, ORCA and SF exhibit high timeout rates, which increases significantly as the number of humans increases. This observation indicates that the *freezing robot problem* is prevalent in these mixed static and dynamic settings (Fig. 3.7a). Also, as Table 3.1 suggests, the large navigation times show that both methods are overly conservative in dense crowds. In the FoV Environment, our method also outperforms ORCA and SF in most metrics, as shown in Fig. 3.5 and Table 3.1, because our method explores the environment and learns from the past experience during RL training. Combined with spatio-temporal reasoning, our method is able to better adapt to dense and partially observable environments. In addition, with our method, the robot is long-sighted because RL optimizes the policy over cumulative reward and the RNNs takes a sequence of trajectories to make decisions while ORCA and SF only consider the current state (Fig. 3.7c).

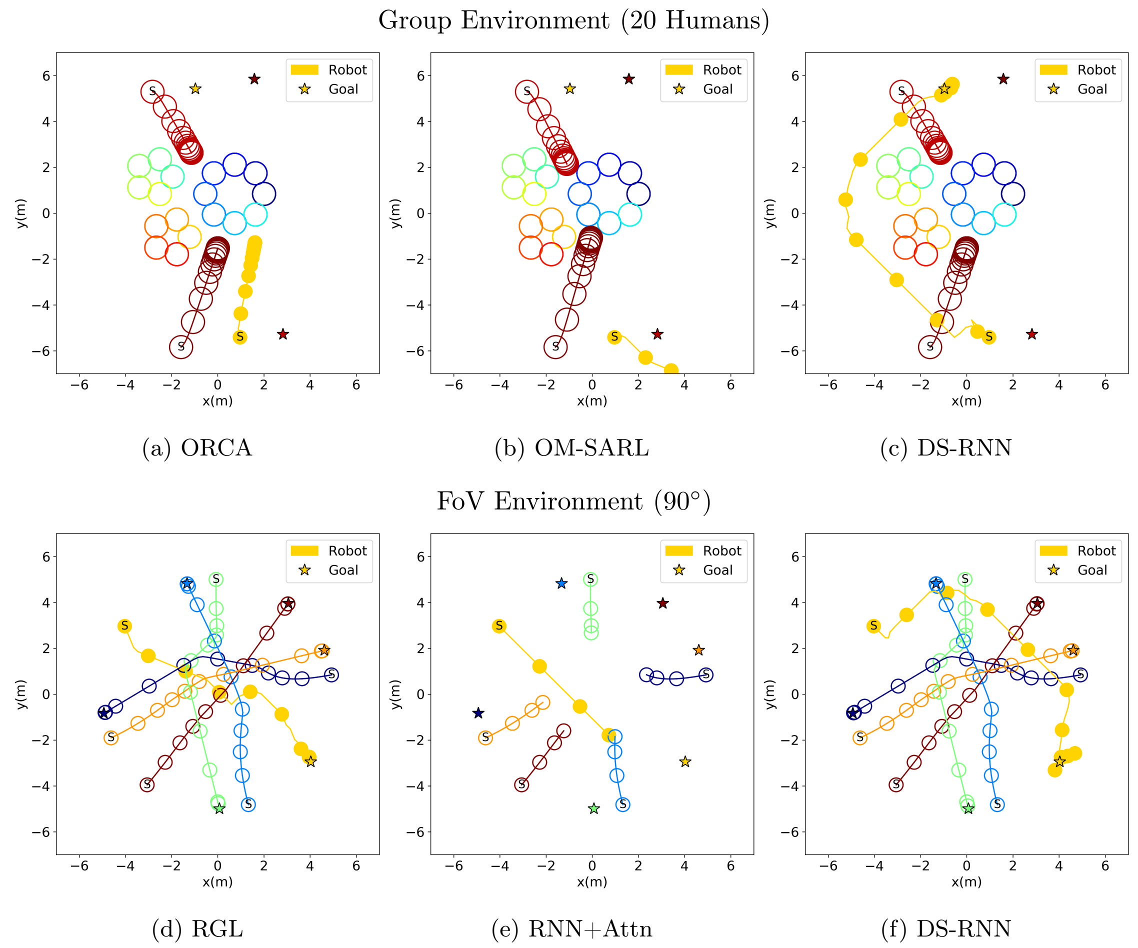


Figure 3.7: **Trajectory comparisons of different methods with the same test cases.** Letter "S" denotes moving agents' starting positions, and stars denote moving agents' goals. The yellow filled circle denotes the robot. For the Group Environment (top), static humans are grouped in three circles.

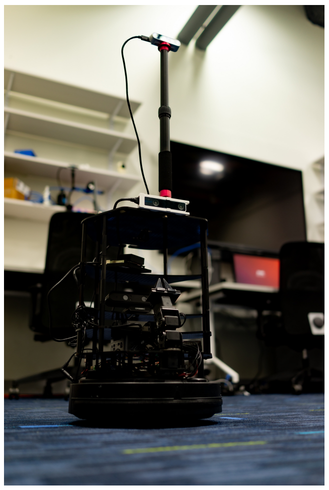


Figure 3.8: **The setup of the real world experiment.**

Comparison with Deep V-Learning

We compare model-free RL training with Deep V-Learning used by CADRL, OM-SARL, and RGL. In Fig. 3.6, all three baselines exhibit large timeout rates. The reason is that the value networks are initialized by a suboptimal expert (ORCA) in supervised learning and are insufficient to provide good state value estimates, resulting in policies that inherit OCRA's drawbacks. In contrast, model-free RL enables RNN+Attn and DS-RNN to learn from scratch and prevents the network from converging to a suboptimal policy too early. In addition, despite the unknown state transitions of all agents, RNN+Attn and our method still perform better in all metrics compared with Deep V-Learning.

As shown in Fig. 3.7d and 3.7f, RGL is competitive to our method in some cases, because the relational graph can perform spatial reasoning and human trajectory predictions make RGL long-sighted. However, in RGL, the relational graph and the robot planner are separated modules, while our network is trained end-to-end and jointly learns the robot-human interactions and decision making.

3.5 Real-world Experiments

We evaluate our trained model's performance on a TurtleBot 2i mobile platform. The hardware platform is shown in Fig. 3.8 and the pedestrian environment is shown in Fig. 3.1. An Intel RealSense depth camera D435 with an approximately 69.4° FoV is used to obtain human positions. We use YOLOv3 [94] for human detection and Deep SORT [95] for human tracking (our implementation is adopted from [96]). The human detection and tracking are combined with the camera depth information to calculate human positions. An Intel RealSense tracking camera T265 is used to localize the robot and obtain the robot orientation. We run the above perception algorithms and our decision-making model on a remote host computer. The communication between the robot and the host computer is established by Robot Operating System (ROS). A video demonstration is available at `https://youtu.be/bYO-1IAjzgY`, where the robot successfully reaches the goals with maneuvers to maintain a safe distance with humans in various scenarios.

3.6 Limitations

Our work encompasses the following limitations. First, our st-graph and DS-RNN only considers robot-human interactions, yet ignores human-human interactions which may also affect the robot's actions. Second, our framework only considers the past and current states of agents without explicitly predicting the future states. Thus, the robot may still exhibit some awkward behaviors such as intercepting in front of humans. Third, in the real world experiment, due to the sensor limitations, the detected human positions are noisy and thus cause differences in robot behaviors between the simulation and the real world. Finally, the total number of humans is fixed for each network model, which poses challenges to the generalization of our model to navigation scenarios with real human flows.

3.7 Summary

We propose a novel DS-RNN network that incorporates spatial and temporal reasoning into robot decision making for crowd navigation. We train our DS-RNN with model-free deep RL without any supervised learning or assumptions on agents' dynamics. Our experiments shows that our model outperforms various baselines in challenging simulation environments and show promising results in the real world.

Videos and simulation code are available at `https://sites.google.com/illinois.edu/crowdnav-dsrnn/home`.

Chapter 4

Intention Aware Robot Crowd Navigation with Attention-Based Interaction Graph

4.1 Introduction

In the Chapter 2, a partial st-graph and decentralized structural-RNN (DS-RNN) network is proposed as an RL-based crowd navigation robot controller [27]. DS-RNN models crowd navigation as a spatio-temporal graph (st-graph) to capture the interactions between the robot and the other agents through both space and time. However, as shown in Fig. 4.1a, DS-RNN and other previous methods such as ORCA [5] only consider the past and current state of humans without explicitly predicting their future trajectories. As a result, the robot sometimes exhibits unsafe or shortsighted behaviors. To deal with this problem, trajectory prediction-based methods plan optimal robot actions conditioned on the predicted future trajectories of other agents [84], [97], [98]. However, the prediction-based methods are usually limited to one-step prediction, discrete action space, or a small set of human intentions.

Another weakness of reaction-based and learning-based methods is that they only consider robot-human (RH) interactions but ignore human-human (HH) interactions. Since the previous works are usually evaluated in sparse crowds or dense crowds with mostly static agents, omitting the human-human interactions does not have a great impact. Yet in dense and highly interactive crowds, the performance of these methods degrades due to the lack of human-human interaction modeling. Some attempts have been made to consider HH interactions, but these works either use predefined values to hardcode HH attention scores or do not distinguish the difference between RH and HH interactions [76], [84]. Since we have control of the robot but not humans, RH and HH interactions have different effects on the robot's decision-making and thus need to be processed separately.

In this chapter, we seek to learn an intention-aware navigation policy which reasons about different interactions in the crowd, as shown in Fig. 4.1b. First, we propose spatio-temporal interaction graph (sti-graph), which captures all interactions in the crowd, and derive a novel neural network from the sti-graph to learn navigation policies. We use two separate multi-head attention networks to address the different effects of RH and HH interactions. The attention networks enable the robot to pay more attention to the important interactions, which ensures good performance when the number of humans increases and the graph becomes complex. Second, we enable longsighted robot behaviors by combining the predictions of other agents' intentions with planning. Given any pedestrian trajectory predictor, we design a reward function

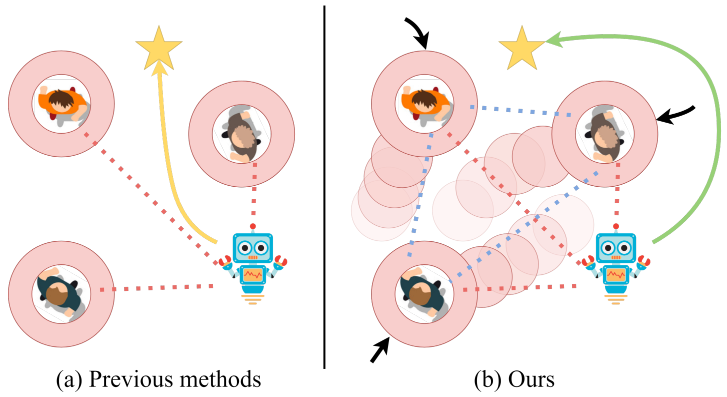


Figure 4.1: **Previous works and our method.** (a) Previous works consider only robot-human interactions (red-dotted lines) and the safety space of humans are circles centered at the pedestrian's position, which may result in unsafe or shortsighted robot behaviors. (b) Beyond robot-human interactions, our method considers human-human interactions (blue-dotted lines). Our safety space is a set of circles centered at the future positions of humans, which improves the performance and intention awareness of the robot.

which encourages the robot to keep away from both current and intended positions of humans, improving both safety and social awareness of the robot.

The main contributions of this chapter are as follows.

1. We propose a novel graph neural network that uses attention mechanism to effectively capture the spatial and temporal interactions among heterogeneous agents.

2. We propose a novel method to incorporate the predicted intentions of other agents into a model-free RL framework. Our method is compatible with any trajectory predictor.

3. The experiments demonstrate that our method outperforms previous works in terms of navigation performance and social awareness.

4. We create an open-source simulation benchmark for crowd navigation, and successfully transfer the learned policy to a real TurtleBot-2i.

This chapter is organized as follows: We review previous related works in Section 4.2. We formalize the problem and propose our network architecture in Section 4.3. Experiments and results in simulation and in the real world are discussed in Section 4.4 and Section 4.5, respectively. Finally, we conclude the chapter in Section 4.7.

4.2 Related Works

4.2.1 Trajectory prediction-based crowd navigation

Utilizing the recent advancements in pedestrian trajectory prediction [65], [90], [99], [100], one line of work uses the predicted human states to compute the transition probabilities of MDP and then plans optimal robot actions with search trees [84], [101], [102]. However, the computation cost of tree search grows exponentially with the size of robot action space. Thus, these methods usually choose discrete and small action spaces, which leads to less natural robot trajectories. Another line of work uses the one-step predictions as observations of model-free RL policies [97], [103]. Although the action space can be large and continuous, one-step predictions only capture the instantaneous velocities instead of intentions of pedestrians. Our method makes predictions for several timesteps to capture the long-term intents of pedestrians and modify the reward function based on predictions so that the robot is intention-aware.

Although some other works attempt to predict the long-term goals of pedestrians to aid planning, the set of all possible goals is finite and small, which limits the generalization of these methods [98], [104]. In contrast, for our method, the goals of pedestrians can be any position in a continuous two-dimensional space.

4.2.2 Attention mechanism for crowd interactions

Interactions amongst multiple agents contain essential information for both multi-pedestrian trajectory prediction and crowd navigation [16], [27], [90], [100]. To capture this information for each agent, self-attention computes attention scores between the agent and all other observed agents [92]. This better captures the pair-wise relationships in the crowd than combining the information of other agents with concatenation or an LSTM encoder [15], [75]. In crowd navigation of robots and autonomous vehicles, some works use attention to determine the relative importance of each human to the robot [16], [27], [105]. However, the interactions among humans can influence the robot but are not explicitly modeled. Other works attempt to model both robot-human (RH) and human-human (HH) attentions using graph convolutional networks [76], [84]. However, Chen and Liu et al. use predefined values to hardcode HH attention scores [76], while Chen and Hu et al. do not distinguish the difference between RH and HH interactions [84]. As a step further toward this direction, our method recognizes the heterogeneous nature of RH and HH interactions and learns the attention scores separately using different attention networks.

4.3 Methodology

In this section, we first formulate crowd navigation as an RL problem and introduce the prediction reward function. Then, we present our approach to model the crowd navigation scenario as a sti-graph, which leads to the derivation of our network architecture.

4.3.1 Preliminaries

MDP formulation

Consider a robot navigating and interacting with humans in a 2D space. We model the scenario as a Markov Decision Process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \mathcal{S}_0 \rangle$. Let $\mathbf{w}^t$ be the robot state which consists of the robot's position (p_x, p_y), velocity (v_x, v_y), goal position (g_x, g_y), maximum speed v_{max}, heading angle θ, and radius of the robot base ρ. Let $\mathbf{u}_i^t$ be the current state of the i-th human at time t, which consists of the human's position (p_x^i, p_y^i). Then, the K predicted future and the M previous positions of the i-th human are denoted as $\hat{\mathbf{u}}_i^{t+1:t+K}$ and $\mathbf{u}_i^{t-M:t-1}$, respectively. We define the state $s_t \in \mathcal{S}$ of the MDP as $s_t = [\mathbf{w}^t, \mathbf{u}_1^t, \hat{\mathbf{u}}_1^{t+1:t+K}, ..., \mathbf{u}_n^t, \hat{\mathbf{u}}_n^{t+1:t+K}]$ if a total number of n humans are observed at the timestep t, where n may change within a range in different timesteps. If

In each episode, the robot begins at an initial state $s_0 \in \mathcal{S}_0$. According to its policy $\pi(a_t|s_t)$, the robot takes an action $a_t \in \mathcal{A}$ at each timestep t. In return, the robot receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $\mathcal{P}(\cdot|s_t, a_t)$. Meanwhile, all other humans also take actions according to their policies and move to the next states with unknown state transition probabilities.

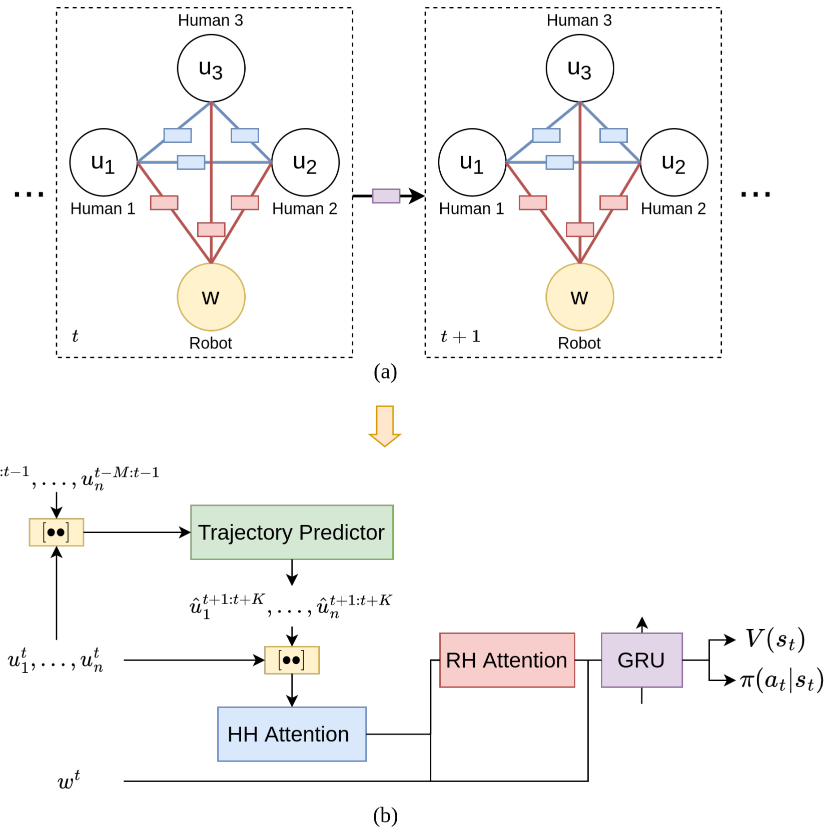


Figure 4.2: **The spatial-temporal interaction graph and the network architecture.** (a) Graph representation of crowd navigation. The robot node is denoted by w and the i-th human node is denoted by u_i. HH edges and HH interaction functions are in blue, while RH edges and RH interaction functions are in red. Temporal function that connects the graphs at adjacent timesteps is in purple. (b) Our network. A trajectory predictor is used to predict personal zones. Two attention mechanisms are used to model the human-human interactions and robot-human interactions. We use a GRU as the temporal function.

The process continues until the robot reaches its goal, t exceeds the maximum episode length T, or the robot collides with any humans.

Trajectory predictor

As shown in Fig. 4.2b, at each timestep t, a trajectory predictor takes the trajectories of observed humans from time $t - M$ to t as input and predicts their future trajectories from time $t + 1$ to $t + K$:

$$\hat{\mathbf{u}}_i^{t+1:t+K} = Predictor(\mathbf{u}_i^{t-M:t}), \quad i \in \{1, ..., n\} \tag{4.1}$$

We add a circle centered at each predicted position to approximate the intended paths of humans, as shown in Fig. 4.1b. If the predictor has learnable parameters, it needs to be pretrained and is only used for inference in our framework.

Reward function

To discourage the robot from intruding into the predicted zones of humans, we use a prediction reward r_{pred} to penalize such intrusions:

$$\begin{aligned} r^i_{pred}(s_t) &= \min_{k=1,\dots,K} \left(\mathbb{1}_i^{t+k} \frac{r_c}{2^k} \right) \\ r_{pred}(s_t) &= \min_{i=1,\dots,n} r^i_{pred}(s_t) \end{aligned} \tag{4.2}$$

where $\mathbb{1}_i^{t+k}$ indicates whether the robot collides with the predicted position of the human i at time $t+k$ and $r_c = -20$ is the penalty for collision. We assign different weights to the intrusions at different prediction timesteps, and thus the robot gets less penalty if it intrudes into the predicted social zone further in the future.

In addition, we add a potential-based reward $r_{pot} = 2(-d^t_{goal} + d^{t-1}_{goal})$ to guide the robot to approach the goal, where d^t_{goal} is the $L2$ distance between the robot position and goal position at time t. Let S_{goal} be the set of goal states, where the robot successfully reaches the goal, and S_{fail} be the set of failure states, where the robot collides with any human. Then, the whole reward function is defined as

$$r(s_t, a_t) = \begin{cases} 10, & \text{if } s_t \in S_{goal} \\ r_c, & \text{if } s_t \in S_{fail} \\ r_{pot}(s_t) + r_{pred}(s_t), & \text{otherwise.} \end{cases} \tag{4.3}$$

Intuitively, the robot gets a high reward when it approaches the goal and avoids intruding into the current and future positions of all humans.

The goal of the agent is to maximize the expected return, $R_t = \mathbb{E}[\sum_{i=t}^{T} \gamma^{i-t} r_i]$, where γ is a discount factor. The value function $V^\pi(s)$ is defined as the expected return starting from s, and successively following policy π.

4.3.2 Spatio-Temporal Interaction Graph

We formulate the crowd navigation scenario as a sti-graph. In Fig. 4.2a, at each timestep t, our sti-graph $\mathcal{G}_t = (\mathcal{V}_t, \mathcal{E}_t)$ consists of a set of nodes $\mathcal{V}_t$ and a set of edges $\mathcal{E}_t$. The nodes represent the detected agents. The edges connect two different detected agents and represent the spatial interactions between the agents at the same timestep. The invisible humans that fall out of the robot's field of view and their corresponding edges are not included in the graph $\mathcal{G}_t$, since the robot cannot predict or integrate their motions in planning. Since we have control of the robot but not the humans, the human-human interactions and the robot-human interactions have different effects on the robot decision-making. Thus, we divide the set of edges $\mathcal{E}_t$ into the human-human (HH) edges that connect two humans and the robot-human (RH) edges that connect the robot and a human. The two types of edges allow us to factorize the spatial interactions into HH interaction function and RH interaction function. In Fig. 4.2a, the HH and RH functions are denoted by the blue and red boxes respectively and have parameters that need to be learned. Compared with the previous works that ignore HH edges [5], [16], [27], [69], our method considers the pair-wise interactions among all visible agents and thus scales better in dense and highly interactive crowds.

Since the movements of all agents cause the visibility of each human to change dynamically, the set of nodes $\mathcal{V}_t$ and edges $\mathcal{E}_t$ and the parameters of the interaction functions may change correspondingly. To this end, we integrate the temporal correlations of the graph $\mathcal{G}_t$ at different timesteps using another function denoted by the purple box in Fig. 4.2a. The temporal function connects the graphs at adjacent timesteps, which overcomes the short-sightedness of reactive methods and enables long-term decision-making of the robot.

To reduce the number of parameters, the same type of edges share the same function parameters. This parameter sharing is important for the scalability of our sti-graph because the number of parameters is kept constant with an increasing number of humans [63].

4.3.3 Network Architecture

In Fig. 4.2b, we derive our network architecture from the sti-graph. In our network, a trajectory predictor predicts the personal zones of humans. We represent the HH and RH functions as feedforward networks with attention mechanisms, referred to as HH attn and RH attn respectively. We represent the temporal function as a gated recurrent unit (GRU). We use W and f to denote trainable weights and fully connected layers throughout this section.

Attention mechanisms

The HH and RH attention functions are similar to the scaled dot-product attention in [92], which computes attention score using a query Q and a key K, and applies the normalized score to a value V.

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{4.4}$$

where d is the dimension of the queries and keys.

In HH attention, the current states and the predicted future states of humans are concatenated and passed through linear layers to obtain $Q^t_{HH}, K^t_{HH}, V^t_{HH} \in \mathbb{R}^{n\times d_{HH}}$, where d_{HH} is the attention size for the HH attention.

$$\begin{aligned} Q^t_{HH} &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W^Q_{HH} \\ K^t_{HH} &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W^K_{HH} \\ V^t_{HH} &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W^V_{HH} \end{aligned} \tag{4.5}$$

We obtain the human embeddings $v^t_{HH} \in \mathbb{R}^{n\times d_{HH}}$ using a multi-head scaled dot-product attention, and the number of attention heads is 8.

In RH attention, $K^t_{RH} \in \mathbb{R}^{1\times d_{RH}}$ is the linear embedding of the robot states $\mathbf{w}^t$ and $Q^t_{RH}, V^t_{RH} \in \mathbb{R}^{n\times d_{RH}}$ are linear embeddings of the weighted human features from HH attention v^t_{HH}.

$$Q^t_{RH} = v^t_{HH}W^Q_{RH},\ K^t_{RH} = \mathbf{w}^t W^K_{RH},\ V^t_{RH} = v^t_{HH}W^V_{RH} \tag{4.6}$$

We compute the attention score from Q^t_{RH}, K^t_{RH}, and V^t_{RH} to obtain the twice weighted human features $v^t_{RH} \in \mathbb{R}^{1\times d_{RH}}$ as in Eq. 4.4. The number of attention heads is 1.

In HH and RH attention networks, we use binary masks that indicate the visibility of each human to prevent attention to invisible humans. Unlike DS-RNN that fills the invisible humans with dummy values [27], the masks provide unbiased gradients to the attention networks, which stabilizes and accelerates the training.

GRU

We embed the robot states $\mathbf{w}^t$ with linear layers f_R to obtain v_R^t, which are concatenated with the twice weighted human features v_{RH}^t and fed into the GRU:

$$v_R^t = f_R(\mathbf{w}^t), \quad h^t = \text{GRU}\left(h^{t-1}, ([v_{RH}^t, v_R^t])\right) \tag{4.7}$$

where h^t is the hidden state of GRU at time t. Finally, the h^t is input to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$.

Training

If the trajectory predictor has learnable parameters, we train the predictor with a dataset of human trajectories collected from our simulator. In RL training, we freeze the parameters of the trajectory predictor and only run inference. We train trajectory predictor and RL policy separately because the two tasks have different objectives, and thus joint training is unstable and less efficient. We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [56]. To accelerate and stabilize training, we run 16 instances of the environment in parallel to collect the robot's experiences. At each policy update, 30 steps of six episodes are used.

4.4 Simulation Experiments

In this section, we present our simulation environment, experiment setup, and experimental results in simulation.

4.4.1 Simulation environment

Our 2D environment simulates a scenario where a robot navigates through a dense crowd in a $12m \times 12m$ space, as shown in Fig. 4.3. Our simulation captures more realistic crowd navigation scenarios than previous works in two aspects [16], [27]. First, our robot sensor has a limited circular sensor range of $5m$, while the previous works unrealistically assume that the robot has an infinite detection range. Second, the maximum number of humans can reach up to 20, leading to a denser and more interactive human crowd.

In each episode, the starting and goal positions of the robot and the humans are randomly sampled on the 2D plane. To simulate a continuous human flow, humans will move to new random goals immediately after they arrive at their goal positions. All humans are controlled by ORCA and react only to other humans but not to the robot for two reasons: (1) This invisible setting prevents our model from learning an extremely aggressive policy in which the robot forces all humans to yield while achieving a high reward; (2) In a real-world dense crowd, the effect of the robot to humans can be ignored because there is little space for humans to react.

We use holonomic kinematics for simulated robot and humans, whose action at time t consists of the desired velocity along the x and y axis, $a_t = [v_x, v_y]$. The action space of the robot is continuous with a maximum speed of $1m/s$. We assume that all agents can achieve the desired velocities immediately, and they will keep moving with these velocities for the next Δt seconds. We define the update rule for an agent's

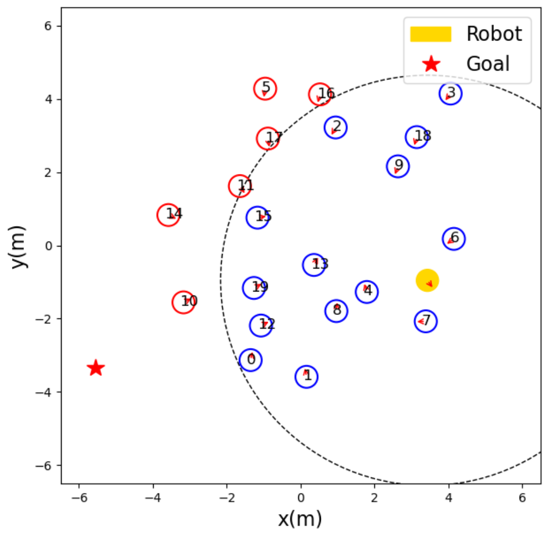


Figure 4.3: **Illustration of our simulation environment.** In a $12m \times 12m$ $2D$ plane, the humans are represented as circles, the orientation of an agent is indicated by a red arrow, the robot is the yellow disk, and the robot's goal is the red star. We outline the borders of the robot FoV with dashed lines. The humans in the robot's FoV are blue and the humans outside are red.

position p_x, p_y as follows:

$$\begin{aligned} p_x[t+1] &= p_x[t] + v_x[t]\Delta t \\ p_y[t+1] &= p_y[t] + v_y[t]\Delta t \end{aligned} \tag{4.8}$$

We use two simulation environments for experiments. The first environment simulates humans with constantly changing intents and different traits with the following randomizations: First, all humans occasionally change their goals to other random positions within an episode; Second, each human has a random maximum speed $v_{max} \in [0.5, 1.5]m/s$ and radius $\rho \in [0.3, 0.5]m$. In the second environment, all humans have the same maximum speed and radius with no random goal changes. The crowds in both environments are dense and interactive, while the randomizations in the first environment pose extra challenges for prediction and decision making.

4.4.2 Experiment setup

Baselines and Ablation Models

To show the benefits of incorporating predictions, we compare the performance of four models. The first model (No pred, HH attn) does not use any predictor. The second model (Const vel, HH attn) uses the constant velocity predictor which predicts linear future trajectories based on the current velocity of the agent. The third model (GST, HH attn) uses Gumbel Social Transformer (GST) predictor, which is a deep network and can predict non-linear trajectories when the robot has a limited field of view and the tracking of humans is imperfect [106]. The final model (Oracle, HH attn) is an oracle which has the access to the ground truth future trajectory of humans. All predictors predict human trajectories for 5 timesteps.

We also compare the performance of our model with the representative methods for crowd navigation. We choose ORCA [5] and SF [69] as the reactive-based baselines and DS-RNN [27] as the learning-based baseline[1]. To further show the benefits of including human-human interactions in RL policy, the ablation model (GST, no HH attn) uses GST as the trajectory predictor and replaces the HH attention with several fully-connected layers.

Training and Evaluation

For RL methods with prediction, we use the reward function as defined in Eq. 4.3. For RL methods without prediction, the reward function excludes r_{pred}. We train all RL methods for 2×10^7 timesteps with a learning rate 4×10^{-5}.

Table 4.1: Navigation results with randomized humans

Method	**SR↑**	**NT↓**	**PL↓**	**ITR↓**	**SD↑**
ORCA	69.0	14.77	17.67	19.61	0.38
SF	29.0	20.28	**15.93**	17.68	0.37
DS-RNN	64.0	16.31	19.63	23.91	0.34
Ours (No pred, HH attn)	67.0	16.82	20.19	16.13	0.37
Ours (GST, no HH attn)	77.0	**12.96**	18.43	8.24	0.40
Ours (Const vel, HH attn)	87.0	14.03	20.14	7.00	0.42
Ours (GST, HH attn)	**89.0**	15.03	21.31	**4.18**	**0.44**
Ours (Oracle, HH attn)	90.0	16.03	22.82	1.70	0.47

Table 4.2: Navigation results without randomized humans

Method	**SR↑**	**NT↓**	**PL↓**	**ITR↓**	**SD↑**
ORCA	78.0	15.87	18.53	26.04	0.36
SF	34.0	19.95	**17.75**	21.35	0.35
DS-RNN	67.0	20.06	25.42	13.31	0.37
Ours (No pred, HH attn)	82.0	19.15	22.82	14.87	0.37
Ours (GST, no HH attn)	82.0	**14.21**	19.35	7.22	0.40
Ours (Const vel, HH attn)	**94.0**	18.26	23.98	4.49	**0.43**
Ours (GST, HH attn)	**94.0**	17.64	22.51	**3.06**	**0.43**
Ours (Oracle, HH attn)	94.0	15.38	21.23	2.97	0.45

We test all methods with 500 random unseen test cases. Our metrics include navigation metrics and social metrics. The navigation metrics measure the quality of the navigation and include the percentage success rate (SR), average navigation time (NT) in seconds, and path length (PL) in meters of the successful episodes. The social metrics measure the social awareness of the robot, which include intrusion time ratio (ITR) and social distance during intrusions (SD) in meters. The intrusion time ratio per episode is defined as c/C, where c is the number of timesteps that the robot collides with any human's true future positions from $t + 1$ to $t + 5$ and C is the length of that episode. The ITR is the average ratio of all testing episodes. We

[1]Comparison to more baselines can be found in [27]

define SD as the average distance between the robot and its closest human when an intrusion occurs. For fair comparison, all intrusions are calculated by ground truth future positions of humans.

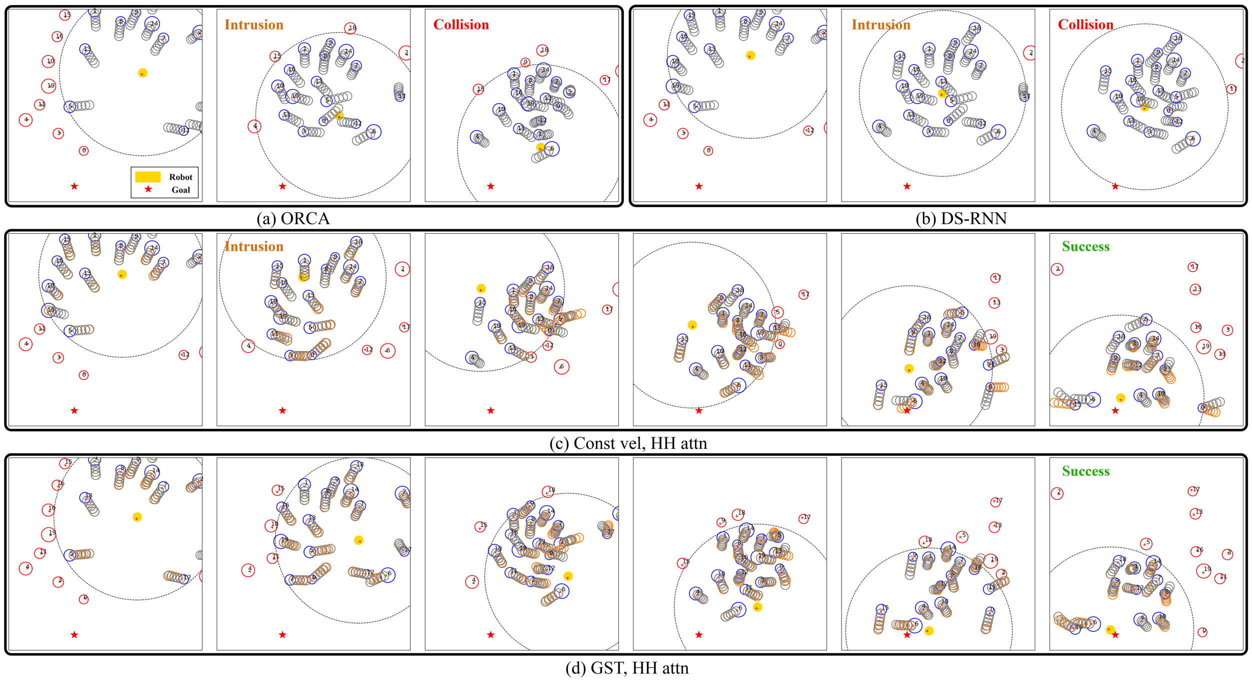


Figure 4.4: **Comparison of different methods in the same testing episode with randomized humans.** The orientation of an agent is indicated by a red arrow, the robot is the yellow disk, and the robot's goal is the red star. We outline the borders of the robot sensor range with dashed lines. Represented as empty circles, the humans in the robot's field of view are blue and those outside are red. The ground truth future trajectories and personal zones are in gray and are only used to visualize intrusions, and the predicted trajectories are in orange. More qualitative results can be found in the video attachment.

4.4.3 Results

Effectiveness of HH attention

The results of robot navigation in two simulation environments are shown in Table 4.1 and Table 4.2. In both tables, for methods with prediction, compared with the ablation model (GST, no HH attn), (GST, HH attn) has 12% higher success rates, lower ITR, and longer SD. For methods without prediction, (GST, no pred, HH attn) outperforms ORCA, SF, DS-RNN in most cases, especially in the more challenging environment in Table 4.1. Fig 4.4 provides an example episode where ORCA and DS-RNN end up with collisions. ORCA assumes that humans take half of the responsibility for collision avoidance, and thus fails immediately if the assumption no longer holds (Fig 4.4a). For DS-RNN, the robot enters instead of deviates from a clutter of humans (Fig 4.4b), indicating that the robot has difficulties reasoning about HH interactions. Since (GST, no HH attn) and the aforementioned three baselines only model RH interactions but ignore HH interactions, the HH attention and our graph network contribute to performance gain. These results suggest that the HH interactions are essential for dense crowd navigation and are successfully captured by our HH attention mechanism, regardless of the presence of predictor and prediction rewards.

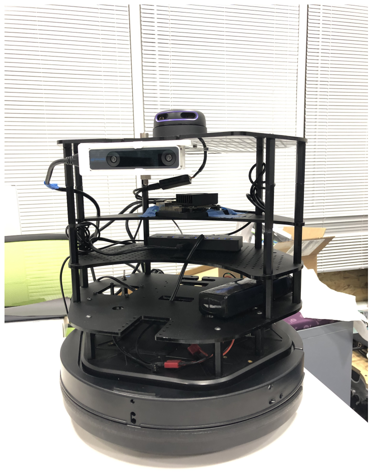


Figure 4.5: **The hardware configuration of TurtleBot 2i.**

Intention awareness

From both tables, we observe that the methods with predictions (last four rows) generally outperform those without predictions (first four rows). From the values of (No pred, HH attn), (Const vel, HH attn), and (GST, HH attn), we see that the trajectory predictors improve the success rate by around 20% with randomized scenarios (Table 4.1) and by around 12% without randomized scenarios (Table 4.2). In addition, the models aided by the trajectory predictors have at least 7% lower ITR and $0.02\,m$ larger SD. The longer NT and PL are caused by our reward function that penalizes the robot to take short yet impolite paths. Comparing the two tables, we observe that incorporating predictions gives a larger performance gain in the randomized scenario, where the crowd behaviors are more complicated and thus are more difficult to predict implicitly. Thus, we conclude that our prediction-aided pipeline enables proactive and long-sighted robot decision making especially in challenging crowd navigation tasks.

In both scenarios, the performance of (GST, HH attn) is closest to the oracle in terms of all metrics, followed by (Const vel, HH attn). Since GST can predict non-linear trajectories while constant velocity predictor cannot, this trend also indicates that a more expressive predictor leads to a better policy. However, we also notice that (GST, HH attn) only outperforms (Const vel, HH attn) with small margins, especially in Table 4.2. The reason is that the humans in our simulator are controlled by ORCA, which prefers linear motion if no other agents are nearby. Thus, the best choice of predictors depends on the complexity of crowd behaviors.

As an illustrative example, with predictions, the robot in Fig 4.4d always keeps a good social distance from the future paths of all humans, while intrusion only happens once in Fig 4.4c. In contrast, without predictions, the robots in Fig 4.4a and b are notably more aggressive and impolite. Therefore, our prediction-aided RL pipeline effectively prevents the robot from intruding into the intended paths of other agents and thus improves the robot's social awareness.

4.5 Real-world Experiments

To show the effectiveness of our method on real differential drive robots and real crowds, we deploy the model trained in the simulator to a TurtleBot 2i. We use an Nvidia Jetson Xavier with ROS as the on-board

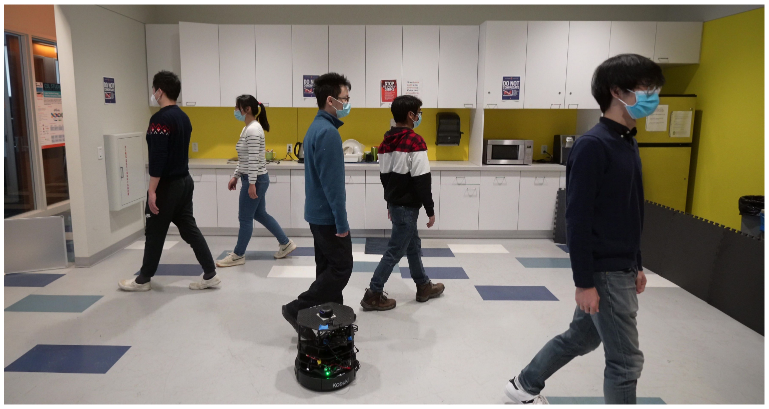


Figure 4.6: **The environment of the real world experiment.**

computer. We use an Intel RealSense tracking camera T265 to obtain the pose of the robot. Different from Sec. 3.5, we change sensor for human detection from an RGBD camera to a 2D LiDAR. The reason is that the camera field-of-view is too small and the mounting point of the camera is unstable, which causes errors in human detection. In Fig. 4.5b, with an RPLIDAR-A3 laser scanner, we first remove non-human obstacles on a map, and then use a 2D LIDAR people detector [107] to estimate the positions of humans and a Kalman filter for tracking.

As shown in Fig. 4.6, in a $5m \times 5m$ indoor space with 1 to 4 humans, we run 18 trials and the success rate is 83.33% (Qualitative results are in the video). In the failure cases, the turning angle of the robot is too large and it collides with the walls, which indicates that incorporating the environmental constraints is needed in future work. Nevertheless, we demonstrate that our navigation policy interacts with real humans proactively despite noises from the perception module and robot dynamics.

4.6 Limitations

Our work encompasses the following limitations. First, the invisible setting of our simulation environment is different from the reality where the motions of pedestrians and those of the robot mutually affect each other. Since the robot does not affect human behaviors, it is difficult to quantify the social awareness of the robot and incorporate it into our design. Second, in the real world experiment, due to the sensor limitations, the detected human positions are noisy and thus cause differences in robot behaviors between the simulation and the real world.

4.7 Summary

We propose a novel RL framework for intention-aware navigation in dense and interactive crowds. We capture the heterogeneous interactions in the crowd with self-attention mechanisms and propose a novel graph neural network to learn navigation policies. To incorporate the future intention of other agents, our reward function penalizes the robot if it intrudes into the predicted positions of other agents. The experiments show that our method outperforms various baselines in partially-observed dense crowds and shows promising results in the real world. Possible directions to explore in future work include (1) using datasets collected from real pedestrians to train our method, and (2) performing user studies to evaluate the social awareness of our model.

Website and videos are available at https://sites.google.com/view/intention-aware-crowdnav/home.

The simulation code and pretrained models are available at https://github.com/Shuijing725/CrowdNav_Prediction_AttnGraph.

The code and guidance for sim2real are available at https://github.com/Shuijing725/CrowdNav_Sim2Real_Turtlebot.

Chapter 5

Structured Graph Network for Constrained Robot Crowd Navigation with Low Fidelity Simulation

5.1 Introduction

Mobile robots are becoming increasingly prevalent in transportation, delivery, and people's homes. To co-exist and collaborate with people seamlessly, robots must navigate through dynamic environments with both moving agents and static obstacles. In dynamic environments, the traversable area is usually constrained. For example, in indoor environments, walls and furniture are common, while in outdoor environments, robots must stay on their lanes or sidewalks. To navigate, the robot must be able to understand the interactions among different agents and entities to make decisions that avoid collisions.

Robot crowd navigation has received much attention since last century [67], [78], [79], [108], [109]. Model-based approaches have explored various mathematical models, such as velocity obstacles [5], [68], dynamic windows [67], and attraction/repulsion forces [69], to optimize robot actions. Although these models consider both humans and obstacles, they are prone to failures such as the freezing problem due to unrealistic assumptions on human behaviors [7], [27]. In addition, the hyperparameters are sensitive to different environments and thus need to be hand-tuned to ensure good performance [110].

Another line of work trained more expressive crowd navigation policies using imitation learning and reinforcement learning [15], [16], [27], [111], [112]. However, these works focus on navigation in open spaces and ignore static obstacles and geometric constraints. To address this issue, some learning-based approaches use raw camera images or LiDAR scans to represent constrained and crowded environments [113], [114]. Other works use a combination of human detection and raw sensor information [108], [115]. These end-to-end (E2E) neural networks have made promising progress with very few assumptions about the environment and a simplified navigation pipeline. However, to deploy the learned policies to the real world, these E2E methods need either a large amount of demonstration data from the real world or a high-fidelity simulator, both of which are expensive to obtain and prone to domain shifts between training and testing scenarios [116].

In this chapter, we ask the following research question: Is it possible to deploy a reinforcement learning (RL) policy for constrained crowd navigation with a cheap low-fidelity simulator using no demonstrations?

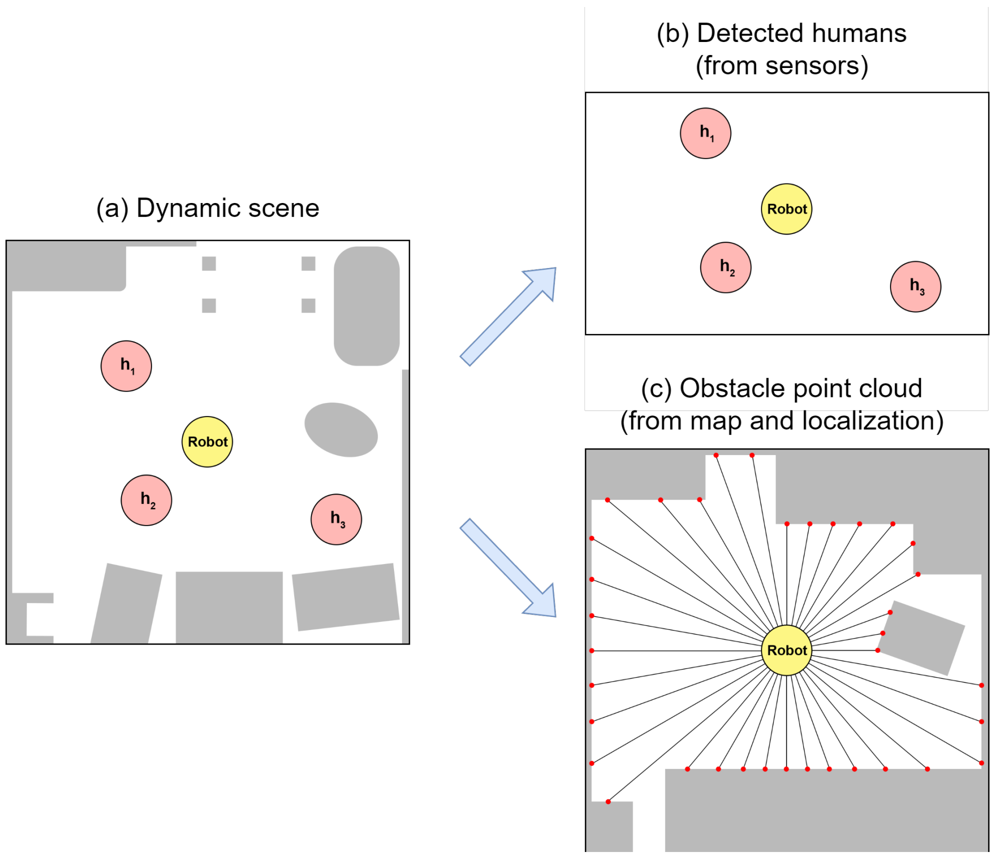

(a) Dynamic scene
h1
Robot
h2
h3
(b) Detected humans (from sensors)
h1
Robot
h2
h3
(c) Obstacle point cloud (from map and localization)
Robot

Figure 5.1: **A split representation of constrained navigation scenario.** In a dynamic scene, human information is obtained from detections by sensors. For obstacle information, we remove all humans and compute a point cloud from a known map and the robot's location. In this way, we can learn a robot policy with smaller sim2real gaps with a cheap low-fidelity simulator.

The first step is to come up with an environment representation that is robust to the sim2real gap from perception. As shown in Fig. 5.1, we propose to split the human and obstacle representation and leverage on processed inputs instead of raw sensor inputs. We represent humans with detected states and obstacles as computed point clouds from map and robot localization. With processed states of humans and objects, the split representation is less affected by inaccurate simulations of human gaits, visual appearance, and 3D shapes. As a result, we can train the robot policy in a low-fidelity simulator, such as the one in Fig. 5.3, with a much smaller sim2real gap during deployment compared with previous E2E methods.

After we separate the representations of each human from static objects, we seek to learn a navigation policy that reasons about interactions among different entities, as shown in Fig. 5.2. In constrained environments, agents have limited spaces to maneuver and thus interact with each other, as well as obstacles, frequently. To capture these subtle interactions, we propose spatio-temporal graph (st-graph) and derive a novel neural network from the st-graph to learn navigation policies. We use three separate attention networks to address the different effects of robot-human, human-human, and human-obstacle interactions. After training with RL, the attention networks enable the robot to pay more attention to the important interactions, which ensures good performance when the number of humans increases and the graph becomes complex.

The main contributions of this paper are as follows. (1) We propose a split representation of constrained and crowded environments, which enables RL policy learned in a low-fidelity simulator to deploy in the real world without additional finetuning. (2) We propose a graph-based policy network that uses attention mechanism to effectively capture the spatial and temporal interactions among agents and obstacles. (3) The experiments demonstrate the effectiveness of our method in various challenging navigation scenarios both simulation and real world.

5.2 Preliminaries

5.2.1 MDP formulation

Consider a robot navigating with humans and obstacles in a 2D space. We model the scenario as a Markov Decision Process (MDP), defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma, \mathcal{S}_0 \rangle$. Let $\mathbf{w}^t$ be the robot state which consists of the robot's position (p_x, p_y), velocity (v_x, v_y), goal position (g_x, g_y), and heading angle θ. Let $\mathbf{h}_i^t$ be the current state of the i-th human at time t, which consists of the human's position and velocity $(p_x^i, p_y^i, v_x^i, v_y^i)$. Let $\mathbf{o}^t$ be the current observations of the static obstacles and walls. We define the state $s_t \in \mathcal{S}$ of the MDP as $s_t = [\mathbf{w}^t, \mathbf{o}^t, \mathbf{h}_1^t, ..., \mathbf{h}_n^t]$ if a total number of n humans are observed at the timestep t, where n may change within a range in different timesteps.

In each episode, the robot begins at an initial state $s_0 \in \mathcal{S}_0$. According to its policy $\pi(a_t|s_t)$, the robot takes an action $a_t \in \mathcal{A}$ at each timestep t. In return, the robot receives a reward r_t and transits to the next state s_{t+1} according to an unknown state transition $\mathcal{P}(\cdot|s_t, a_t)$. Meanwhile, all other humans also take actions according to their policies and move to the next states with unknown state transition probabilities. The process continues until the robot reaches its goal, t exceeds the maximum episode length T, or the robot collides with any humans.

5.2.2 Reward function

The reward function awards the robot for reaching its goal and penalizes the robot for collisions with or getting too close to humans or obstacles. In addition, we add a potential-based reward shaping to guide the robot to approach the goal:

$$r(s_t, a_t) = \begin{cases} -20, & \text{if } d_{min}^t < 0 \\ 2(d_{min}^t - 0.25), & \text{if } 0 < d_{min}^t < 0.25 \\ 20, & \text{if } d_{goal}^t \leq \rho_{robot} \\ 2(-d_{goal}^t + d_{goal}^{t-1}), & \text{otherwise.} \end{cases} \tag{5.1}$$

where d_{min}^t is the minimum separation distance between the robot and any human or obstacle at time t, and d_{goal}^t is the $L2$ distance between the robot position and goal position at time t. Intuitively, the robot gets a high reward when it approaches the goal while maintaining a safe distance from all dynamic and static obstacles.

5.3 Methodology

In this section, we propose a structured representation of constrained and crowded scenes. Then, we formulate the scene as a spatio-temporal graph that allows the robot to reason about different interactions among entities, and derive a structured attention network to learn a robot policy.

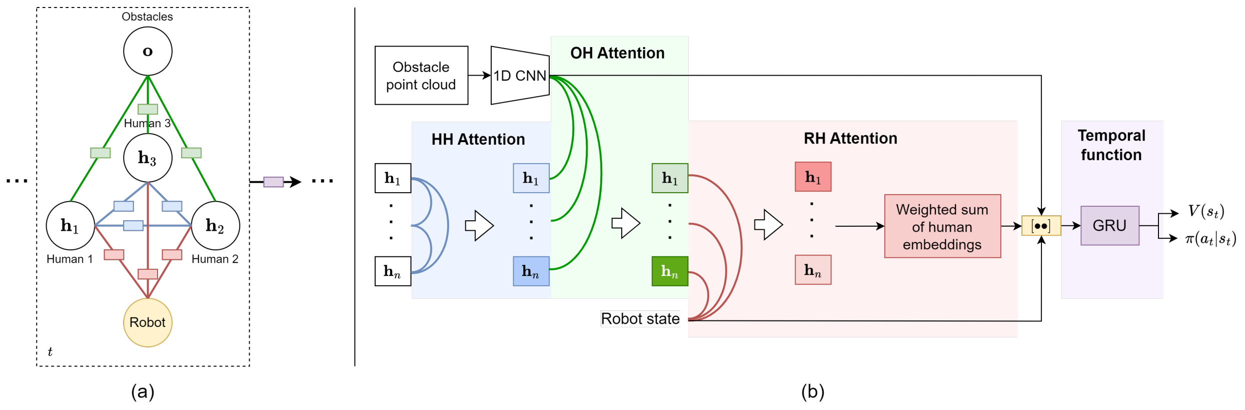


Figure 5.2: **The spatial-temporal interaction graph and the network architecture.** (a) Graph representation of crowd navigation. The robot node is denoted by w and the i-th human node is denoted by u_i. HH edges and HH functions are in blue, while RH edges and RH functions are in red. Temporal function that connects the graphs at adjacent timesteps is in purple. (b) Our network. A trajectory predictor is used to predict personal zones. Two attention mechanisms are used to model the human-human interactions and robot-human interactions. We use a GRU as the temporal function.

5.3.1 Scene representation

High-dimensional raw sensor representation suffer from large sim2real gaps due to the presence of humans, obstacles with different shapes, and complex landscapes. These gaps cause serious problems for robot deployment in real environments. Instead of investing in expensive high-fidelity simulators or laborious dataset collection, we use low-fidelity simulators. To circumvent the sim2real gap, our scene representation leverages processed information from perception modules, maps, and robot localization, which are relatively easier to obtain and robust to domain shifts.

As shown in Fig. 5.1, at each timestep t, we split a dynamic scene into a human representation denoted as $\mathbf{h}_1^t, ..., \mathbf{h}_n^t$, as well as an obstacle and constraint representation denoted as $\mathbf{o}^t$. In human representation, the position and velocity of each human is detected from off-the-shelf human detectors [94], [95], [107]. By representing each human as a low-dimentional state vector, we abstract away detailed information such as gaits and appearance, which are difficult to model accurately [104], [117]. To obtain obstacle and constraint representation, we first map the environment and process the map by combining close-by obstacles and approximating obstacle shapes as polygons. The map processing step is shown from Fig. 5.1(a) to Fig. 5.1(c). During navigation, assuming robot localization is available, we can compute a "fake" point cloud by performing a ray tracing algorithm centered at the robot location, as shown in Fig. 5.1(c). The "fake" point cloud contains approximate information on obstacles and constraints, which is sufficient for robot navigation. It is worth noting that compared to real point clouds from sensors, our obstacle representation is not affected by the presence of humans and is less sensitive to inaccuracies simulations of object appearance or shapes.

By decomposing the scene into dynamic and static parts, and obtaining them from perception algorithms and localization respectively, we avoid potential sources that cause sim2real gaps yet retain essential information for navigation.

5.3.2 Structured interaction graph

Interactions among different entities contain essential information for multi-agent problems including crowd navigation [16], [27], [90], [100], [118]. We formulate constrained crowd navigation as a spatio-temporal (st) graph, which breaks the problem into smaller components in a structured fashion [63]. In Fig. 5.2(a), the robot, all observed humans, and the observed static environment are nodes in the st-graph $\mathcal{G}_t$. At each timestep t, the edges that connect different nodes denote the spatial interactions among nodes. Since we have control of the robot but not the humans, robot-human interactions have direct effects while human-human interactions have indirect effects on the robot actions. Since the agents including humans and robots are movable but the obstacles are not movable, interactions among agents are mutual while the influence of static obstacles on agents is one-way. For these reasons, different interactions have different effects on robot decision-making. Thus, we categorize the spatial edges into three types: human-human (HH) edges (in blue color in Fig. 5.2), obstacle-human (OH) edges (green), and robot-human (RH) edges (red). The three types of edges allow us to factorize the spatial interactions into HH function, OH function, and RH function. Each function is a neural network that has parameters to be learned. Compared with the previous works that ignore some edges [16], [27], [111], our method allows the robot to reason about all observed spatial interactions that exist in constrained crowded environments.

Since the movements of all agents cause the visibility of humans and obstacles to change dynamically, the set of nodes and edges and the parameters of the interaction functions may change correspondingly. To this end, we integrate the temporal correlations of the graph $\mathcal{G}_t$ at different timesteps using another function denoted by the purple box in Fig. 5.2a. The temporal function connects the graphs at adjacent timesteps, which overcomes the short-sightedness of reactive methods and enables long-term decision-making of the robot.

To reduce the number of parameters, the same type of edges share the same function parameters. This parameter sharing is important for the scalability of our st-graph because the number of parameters is kept constant with an increasing number of humans [63].

5.3.3 Structured attention network

In Fig. 5.2b, we derive our network architecture from the st-graph. We represent the HH, OH, and RH functions as feedforward networks with attention mechanisms, referred as HH attn, OH attn, and RH attn respectively. We represent the temporal function as a gated recurrent unit (GRU). We use W and f to denote trainable weights and fully connected layers throughout this section.

Attention mechanisms

The attention modules assign weights to all edges that connect to a node, allowing the node to pay more attention to important edges or interactions. The 3 attention networks are similar to the scaled dot-product attention [92], which computes attention score using a query Q and a key K, and applies the normalized score to a value V.

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V \tag{5.2}$$

where d is the dimension of the queries and keys. Multi-head attention concatenate multiple single-head attention functions to jointly attend to information from h different representation embeddings of the agents:

$$\begin{aligned} \text{MultiHeadAttn}(Q, K, V) &= [\text{head}_i, ..., \text{head}_h]W^o \\ \text{where head}_i &= \text{Attn}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{5.3}$$

In HH attention, the current states of humans are concatenated and passed through linear layers to obtain $Q_{HH}^t, K_{HH}^t, V_{HH}^t \in \mathbb{R}^{n\times d_{HH}}$, where d_{HH} is the attention size for the HH attention.

$$\begin{aligned} Q_{HH}^t &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W_{HH}^Q \\ K_{HH}^t &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W_{HH}^K \\ V_{HH}^t &= [\mathbf{u}_1^{t:t+K}, ..., \mathbf{u}_n^{t:t+K}]^\top W_{HH}^V \end{aligned} \tag{5.4}$$

We obtain the human embeddings $v_{HH}^t \in \mathbb{R}^{n\times d_{HH}}$ using a multi-head scaled dot-product attention from Eq. 5.3, and the number of attention heads is 8.

In OH attention, the obstacle point cloud is fed into a 1D CNN network, which results in the obstacle embedding v_O^t:

$$v_O^t = f_{CNN}(\mathbf{o}^t) \tag{5.5}$$

Then, we use a linear layer to embed the encoded obstacle v_O^t, and the resulting output vector is denoted as $K_{OH}^t \in \mathbb{R}^{1\times d_{OH}}$. $Q_{RH}^t, V_{RH}^t \in \mathbb{R}^{n\times d_{RH}}$ are linear embeddings of the weighted human features from HH attention v_{HH}^t.

$$Q_{OH}^t = v_{HH}^t W_{OH}^Q,\ K_{OH}^t = v_O^t W_{OH}^K,\ V_{OH}^t = v_{HH}^t W_{OH}^V \tag{5.6}$$

We compute the attention score from Q_{OH}^t, K_{OH}^t, and V_{OH}^t to obtain the twice weighted human features $v_{OH}^t \in \mathbb{R}^{1\times d_{OH}}$ as in Eq. 5.2. The number of attention heads is 1.

Similarly, in RH attention, we first embed the robot with a linear layer

$$v_R^t = f_R(\mathbf{w}^t) \tag{5.7}$$

Then, we feed the robot embedding v_R^t to another linear layer, which results in an output embedding $K_{RH}^t \in \mathbb{R}^{1\times d_{RH}}$. $Q_{RH}^t, V_{RH}^t \in \mathbb{R}^{n\times d_{RH}}$ are linear embeddings of the weighted human features from OH attention v_{OH}^t.

$$Q_{RH}^t = v_{OH}^t W_{RH}^Q, \quad K_{RH}^t = \mathbf{w}^t W_{RH}^K, \quad V_{RH}^t = v_{OH}^t W_{RH}^V \tag{5.8}$$

We compute the attention score from Q_{RH}^t, K_{RH}^t, and V_{RH}^t to obtain the weighted human features for the third time $v_{RH}^t \in \mathbb{R}^{1\times d_{RH}}$ as in Eq. 5.2. The number of attention heads is 1.

In all three attention networks, we use binary masks that indicate the visibility of each human to prevent attention to invisible humans. The masks provide unbiased gradients to the attention networks, which stabilizes and accelerates the training.

GRU

We concatenate the robot embedding v_R^t, the obstacle embedding v_O^t, and the weighted human features v_{RH}^t and fed into the GRU:

$$h^t = \text{GRU}\left(h^{t-1}, ([v_{RH}^t, v_R^t, v_O^t])\right) \tag{5.9}$$

where h^t is the hidden state of GRU at time t. Finally, the h^t is input to a fully connected layer to obtain the value $V(s_t)$ and the policy $\pi(a_t|s_t)$.

We use Proximal Policy Optimization (PPO), a model-free policy gradient algorithm, for policy and value function learning [56]. The entire network, including 1D CNN, 3 attention networks, and the GRU are trained together.

5.4 Simulation Experiments

In this section, we present our simulation environment, experiment setup, and experimental results in simulation.

5.4.1 Simulation environment

Our simulator is developed with PyBullet [119]. The 3D environment simulates two scenarios where a TurtleBot 2i navigates through a crowd in a constrained space with static obstacles, as shown in Fig. 5.3. We conduct simulation experiments in *random environment* in Fig. 5.3(a). In each episode, obstacles are initialized with random shapes and random poses. The initial positions of the humans and the robot are also randomized. The human goals are set on the opposite side of their initial positions so that they cross each other in a circle. The number of humans varies from 2 to 4 and the number of obstacles varies from 7 to 9.

To simulate a continuous human flow, humans will move to new random goals immediately after they arrive at their goal positions. All humans are controlled by ORCA [5]. 80% of humans do not react to the robot and 20% of humans react to the robot. This mixed setting prevents our network from learning an extremely aggressive policy in which the robot forces all humans to yield while achieving a high reward, while maintaining a enough number of reactive humans to resemble the real crowd behaviors.

We use unicycle kinematics for the robot. The action of the robot consists of desired translational and rotational accelerations $a_t = [a_{trans}, a_{rot}]$. The robot action space is discrete: the translational acceleration $a_{trans} \in \{-0.05\,m/s^2, 0\,m/s^2, 0.05\,m/s^2\}$ and the rotational acceleration $a_{rot} \in \{-0.1\,rad/s^2, 0\,rad/s^2, 0.1\,rad/s^2\}$. At each timestep t, the robot policy outputs a desired acceleration command, and we assume a constant force corresponding to the acceleration is applied to the wheel motors for Δt seconds. The translational velocity is clipped within $[0\,m/s, 0.5\,m/s]$ and rotational velocity is within $[-1\,rad/s, 1\,rad/s]$. The robot motion is governed by the dynamics of TurtleBot 2i.

We use holonomic kinematics for humans. The human action space is the desired velocity along the x and y axis, $v_t = [v_x, v_y]$. The speed of humans is limited to 0.5 m/s to accommodate the speed of the robot. Due to the lack of a human dynamics model, we assume that humans can achieve the desired velocities immediately, and they will keep moving with these velocities for the next Δt seconds.

5.4.2 Experiment setup

Baselines and ablation models

To validate the effectiveness of the proposed scene representation, we compare our method with the following baselines:

- Object-centric: The human representation is the same as the proposed method, while obstacles are represented by the coordinates of their polygon vertices. The obstacle embedding network is changed into a multi-layer perception instead of 1D CNN.

- Raw point cloud: The humans and obstacles are represented as raw point clouds from a LiDAR. No human detection or artificial point cloud is available. The point cloud is fed into a 1D CNN and a GRU

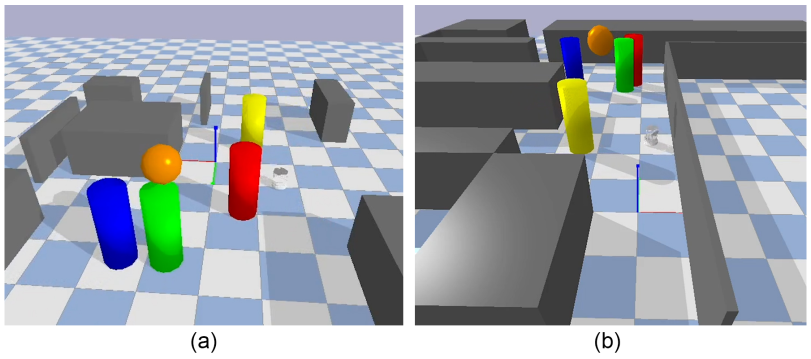


Figure 5.3: **Two PyBullet simulation scenarios.** (a) Random environment with random obstacles and circle-crossing humans. (b) Sim2real environment with fixed obstacles and the random human flow is designed based on the layout.

which outputs the robot action and the state value. No attention network is used since all entities are mixed as a single point cloud.

To validate the effectiveness of the structured graph network, we experiment with ablations of different attention models to justify the effect of 3 types of spatial interaction networks. Everything else except the presence of attention network(s) is kept the same as our method.

- Ours, RH attn: The network has only RH attention and does not have HH or OH attention. Policy networks with RH attention are widely used in previous works such as [16], [27], [105].
- Ours, RH+OH attn: The network has only RH and OH attention and does not have HH attention.
- Ours, RH+HH+OH attn: The full version of our proposed network with all 3 attention modules.

Training

The policy is trained for 6×10^7 steps in total. To accelerate and stabilize training, we run 16 parallel environments to collect the robot's experiences. The learning rate is 8×10^{-5} and decays linearly during training. At each policy update, 30 steps of six episodes are used.

Evaluation metrics

We test all methods with 500 random unseen test cases. Our metrics include success rate (Success), collision rate (Collision), timeout rate (Timeout), and average navigation time (Time) in seconds.

Table 5.1: Navigation results in random environment.

Method	**Success**$\uparrow$	**Collision**$\downarrow$	**Timeout**$\downarrow$	**Time**$\downarrow$
Object-centric	0.63	0.27	0.09	12.02
Raw point cloud	0.58	0.27	0.15	12.02
Ours, RH attn	0.56	0.28	0.16	12.02
Ours, RH+OH attn	0.68	0.26	0.06	**11.28**
Ours, RH+HH+OH attn	**0.75**	**0.15**	**0.10**	11.75

5.4.3 Results

In Table 5.1, among the baselines, raw point cloud performs the worst because the network needs to extract both human and obstacle features from raw sensor inputs, which slows down the convergence. Object-centric is better due to the presence of low-level state information. Nevertheless, object-centric is outperformed by our method, because the vertex representation is sparse and not always useful for navigation. For example, among all vertices of a long and thin wall, the occluded vertices or the vertices outside of the robot field-of-view are fed into the network, yet none of them affects navigation. In addition, the vertex representation is sensitive to the permutation order of vertices, which poses extra challenges for learning [120]. In contrast, our representation leverages low-level human states from detectors, which accelerates the convergence and thus achieves better performance with the same amount of training budget. For the obstacle representation, the "fake" point cloud is denser and more relevant to robot decision-making compared with vertex representation.

Among ablated models, we observe that if we remove HH attention, the success rate drops by 7% because the interactions among humans are dense due to the presence of obstacles and environmental constraints. As a result, a human constantly changes its trajectories due to other humans. If we further remove OH attention, the success rate drops by another 12% because the obstacles limit the traversable regions of agents, and thus human trajectories are also directly affected by obstacles. Thus, we conclude that reasoning about both human-human and human-robot interactions plays an important role in robot collision avoidance, in addition to robot-human interactions from previous works.

5.5 Real-world Experiments

We train our method in another simulation environment and transfer the policy to a TurtleBot 2i in a real constrained indoor environment with pedestrians. The *sim2real environment* in Fig. 5.3(b) mimics a real-world hallway with various obstacles in a university building. We define a set of human routes and robot routes based on the environment layout and randomly choose a route for each agent in each episode. In addition, some static humans are added to simulate a more realistic crowd. The poses obstacles are fixed and the number of obstacles is 11. The number of dynamic humans varies from 1 to 3 and the number of static humans varies from 1 to 3.

We use an Intel RealSense tracking camera T265 to obtain the pose of the robot. With an RPLIDAR-A3 laser scanner, we first remove non-human obstacles on a map, and then use a 2D LIDAR people detector [107] to estimate the positions of humans. The "fake" point cloud is obtained from the robot localization and the *sim2real* simulation environment. From results (see videos here), we observe that the robot is able to achieve goals without collisions no matter the pedestrians react to the robot or not. However, the robot fails if the pedestrians intentionally block its path, since this kind of adversarial behavior is not available in the simulator during training.

5.6 Summary

In conclusion, to enable robots to navigate in constrained crowded environments, we propose a structured scene representation from preprocessed information. Then, we formulate the scenario as a st-graph, which leads to the derivation of a robot policy network that reasons about different interactions during navigation. We train the network with RL and demonstrate good results in both simulation and the real world. However, our work encompasses the following limitations, which opens up opportunities for future work:

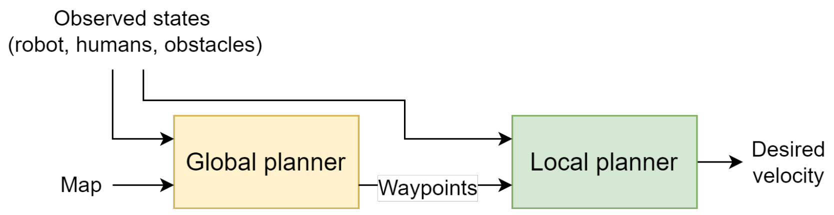


Figure 5.4: **A two-level hierarchical planner.** To enable long-horizon navigation, we can treat our method as a local planner and combine it with a global planner.

1. The robot only achieves a 75% success rate when it only navigates for 3 to 4 meters from start to goal. The task horizon and success rate are not enough for robot deployment in real applications, such as last-mile delivery. To address this issue, as shown in Fig. 5.4, we plan to adopt a hierarchical planner, which consists of a global planner that outputs waypoints, and a local planner that takes waypoints and performs low-level control. The current model can be used as the local planner. Possible options for the global planner include sample-based planners such as A^* and Rapidly exploring random tree (RRT) and learning-based policies.

2. As Sec. 5.5 discussed, the robot policy is overfitted to the simulated human behaviors. However, the simulated humans do not capture all the nuanced behavior patterns of real humans. A more realistic human motion model is necessary, which might need to be learned from real pedestrian data [121]. In addition, adversarial RL training may also improve the robustness of the robot policy with respect to changes in human behaviors [122], [123].

Chapter 6

A Dialogue-Based Robot for Assistive Navigation with Visual Language Grounding

6.1 Introduction

Wayfinding, defined as helping people orient themselves in an environment and guiding them from place to place, is a longstanding challenge for persons with visual impairments (PwVI) [17], [18]. To improve the quality of PwVI's lives, we aim to build a guiding robot that can connect language to the surrounding world to verbally interact with PwVI.

To pair wayfinding with communication, a line of previous works gives users signals such as navigation instructions [19], [124] and basic environment information [20], [125]. As a step further, other wayfinding technologies recognize and convey the semantic meaning of the surrounding environment such as naming the landmarks [126]–[128]. However, these methods require special environmental setups, such as multiple RFID tags and bluetooth beacons. To improve the aforementioned systems with recent advances in machine learning [10], [22], [129], we aim to remove dependence on these types of special infrastructure by integrating advances in visual-language grounding into conversational wayfinding.

More broadly, technologies in vision-language navigation and voice-controlled robots have made significant progress [10], [22], [129]. These navigation agents are able to perform various tasks according to natural language commands such as "Bring me a cup," with simple onboard sensors. This is usually achieved by encoding visual landmarks in a semantic map and associating language with these landmarks during navigation, which is referred to as visual-language grounding [21], [22]. However, these general-purpose frameworks assume that humans can provide step-by-step navigation instructions. These systems are not built for PwVI, who often need help perceiving the environment and planning paths. Thus, building a robot guide that can intuitively exchange semantic information with users remains an open challenge.

In this chapter, we propose DRAGON, a **D**ialogue-based **R**obot for **A**ssistive navigation with visual-language **G**r**o**u**n**ding. In Fig. 6.1, since PwVIs have limited vision, DRAGON uses speech to communicate with the user and a physical handle for fully autonomous navigation guidance. The dialogue and navigation can be executed simultaneously. When the user gives a speech command, Speech Recognition (SR) and

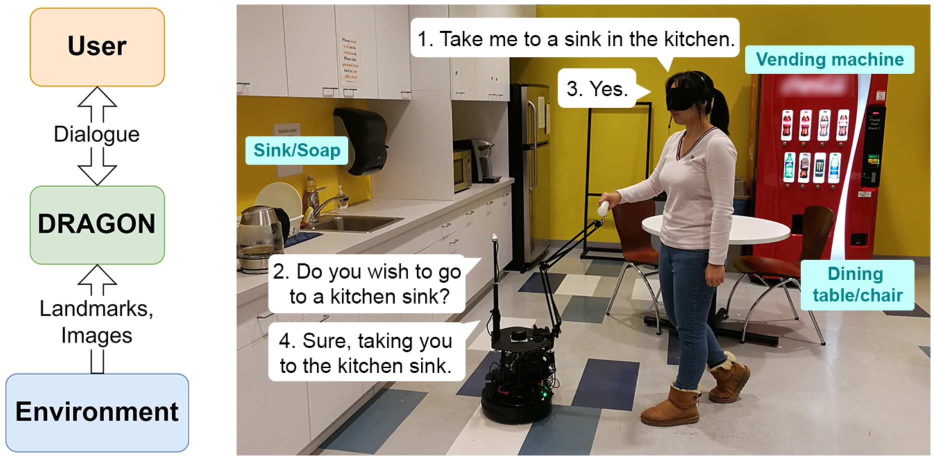

User
Dialogue
DRAGON
Landmarks, Images
Environment
1. Take me to a sink in the kitchen.
Vending machine
3. Yes.
Sink/Soap
2. Do you wish to go to a kitchen sink?
Dining table/chair
4. Sure, taking you to the kitchen sink.

Figure 6.1: **DRAGON identifies the intents of the user through dialogue, grounds language with the environment, and guides the user to their desired goal.**

Natural Language Understanding (NLU) modules first extract the user's intents and desired destinations. The user command does not have any templates or constraints on vocabulary or expressions. Based on the outputs of NLU, one of the following grounding functionalities is triggered: (1) finding users' desired destinations with a visual-language model [130] and guiding them to the destinations; (2) describing nearby objects; and (3) answering questions from users. With (2) and (3), DRAGON can help users gain awareness of their surroundings during navigation.

To find users' intended goals on a map, we propose a novel landmark recognition module based on Contrastive Language-Image Pre-training (CLIP) [130]. After a straightforward mapping process, the landmark recognizer is able to select the landmark whose image best matches the user descriptions. Our landmark recognizer is able to associate flexible and open-vocabulary commands with few constraints on user expressions. If the description is ambiguous, our system will disambiguate user intents through additional dialogue. Then, the corresponding goal location is passed to the path planners for navigation guidance. Combined with the robot's navigation module, the powerful and reliable landmark recognizer is essential to ensure the success and user experience of DRAGON.

Our main contributions are as follows: (1) As an interactive navigation guide for PwVI, DRAGON enables voice-based dialogue, which carries rich information and has grounding capabilities; (2) We propose a novel landmark mapping and recognition method that can associate free-form language commands with image landmarks. Our method can be easily plugged into the standard navigation module of mobile robots; (3) A user study with five blindfolded participants (N=5) demonstrates that DRAGON is able to understand user intents through dialogue and guide them to desired destinations in an intuitive manner. To the best of our knowledge, our work is the first to show that visual-language grounding via dialogue benefits robotic assistive navigation.

Table 6.1: Benchmark for conversational wayfinding technologies.

Method	**User-chosen**	**Speech dialogue**		**Environment**	**VQA**	**Form**	**Environmental**
	semantic goals	Input	Output	**description**			**Instrumentation**
GuideBeacon [126]	✓	✓	✓			Phone application	Bluetooth beacons
NavCog3 [127]	✓	✓	✓	❍	❍	Phone application	Bluetooth beacons
LandmarkAI [131]	✓	✓	✓	✓	❍	Phone application	GPS
SeeWay [132]	✓	✓	✓			Phone application	WiFi
Robotic Shopping cart [133]	❍		✓			Robot	RFID tags
CaBot [20]			✓	✓		Robot	Remote joystick
Ballbot [19]	✓	✓	✓			Robot	WiFi + Remote computer
Ours	✓	✓	✓	✓	✓	Robot	WiFi + Remote computer

A ✓ means that the functionality is implemented. A ❍ means partial implementation. A blank cell means the functionality is absent. (In [133], the users have to enter a number sequence into a keypad to specify their destinations. [131] can only describe a fixed set of pre-mapped landmarks and can only answer two fixed questions.)

6.2 Related Works

6.2.1 Wayfinding robots and technologies

Navigation guidance:

To guide PwVI from point A to point B following a planned path, unactuated devices, such as smartphones and wearables, rely on haptic or audio feedback to give instructions such as going straight and turning right [124], [126], [127], [132]. However, delays and misunderstandings might lead to inevitable deviations, which take time and effort to recover from [125], [126]. On the other hand, robots provide a physical holding point, which offers kinesthetic feedback to minimize deviations and reduce the mental load of users [133]–[135]. Such physical guidance can be combined with aforementioned verbal or haptic navigation instructions to further improve performance at the cost of a more expensive system [19], [20]. To ensure both efficiency and low cost, we mount a handle on our robot to give intuitive real-time steering feedback in navigation.

Semantic communication:

A large part of blind navigation technologies ignores exchanging environmental information with users [19], [134], [136]. To deal with this issue, CaBot applies object recognition to describe the user's neighborhood, yet the user cannot hold conversations with the robot or choose their destinations [20]. To enable users to choose a semantic goal (*e.g.* a restroom), some works mark points of interest using bluetooth beacons [126], [127] or Radio-Frequency Identification (RFID) tags [128], [133], which requires heavy instrumentation. As an alternative, extracting semantic information from ego-centric camera images is much cheaper and easier. For example, SeeWay uses skybox images to represent landmarks [132]. Similarly, Landmark AI offers semantic-related functionalities including describing the environment, reading road signs, and recognizing landmarks using a phone camera [131]. However, these phone applications are not robots and thus cannot physically guide users or provide a stable mounting point for cameras. In contrast, Table 6.1 shows that DRAGON brings conversational wayfinding to the next level: A robot can simultaneously offer physical guidance and enable users to trigger a variety of functionalities through dialogue.

6.2.2 Command following navigation

Tremendous efforts have been made in understanding and grounding human language instructions for various robotic tasks [10], [22], [129]. In command following navigation, a modular pipeline usually consists of three modules: (1) an NLU system to map instructions to speaker intent; (2) a grounding module to associate the intent with physical entities; and (3) a simultaneous localization and mapping (SLAM) and a planner to generate feasible trajectories [21], [137], [138]. Other works attempt to learn end-to-end policies from simulated environments or datasets [62], [139]–[141]. However, due to sim-to-real gaps in perception, language, and planning, deploying these policies to the real world remains an open challenge for applications in the low data regime such as wayfinding [142]. Therefore, we adopt the modular pipeline to ensure performance in the real world.

6.2.3 Semantic landmark recognition

Understanding the semantic meanings of a scene is a vital step towards interactive navigation [21], [22]. Some works reconstruct volumetric maps for the environment, where each grid is associated with a semantic label [22], [138], [143]. Other works build more abstract scene graphs [144], [145]. However, implementing these methods on a real robot is expensive, as they require accurately calibrated depth cameras and high-performing instance segmentation models.

Another line of work collects images as landmarks to create topological graphs [146]–[148]. In navigation, the goal location is retrieved by computing the similarity between a goal image and all stored landmarks. However, the above works only consider image goals, which are less natural than language in human-centered applications. Inspired by Shah *et al.* [21] and Huang *et al.* [22], we use CLIP [130] to associate image landmarks with users' language commands. Compared with previous works that use closed vocabulary object detectors, which are limited to a predefined set of semantic classes [143]–[145], our method can handle more flexible and open-vocabulary commands. We use CLIP to select landmarks and keep traditional cost maps for planning, enabling easy integration of our method into the navigation stack of mobile robots.

6.3 System Overview

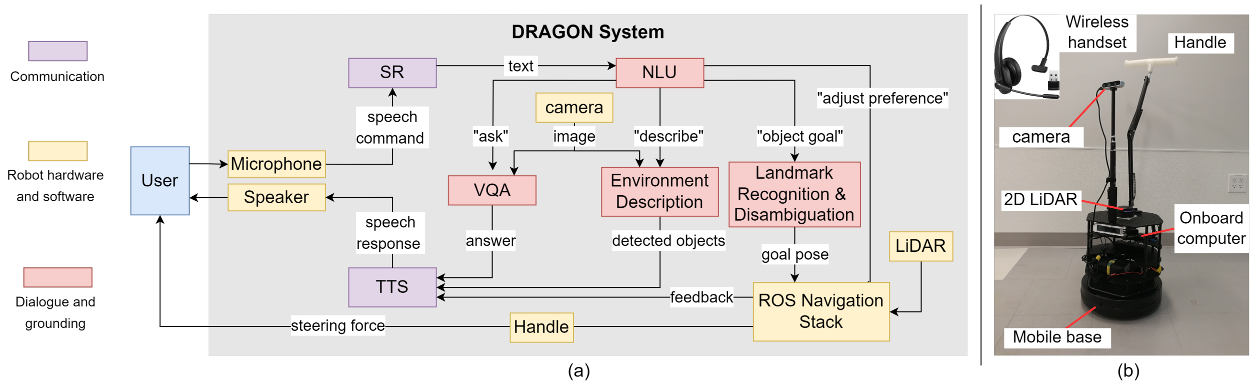


Figure 6.2: **An overview of the system and platform of DRAGON**. (a) Submodules, message passing, and user interface. (b) The robot platform.

In this section, we describe the setup and configuration of our robot guide with special considerations for PwVI users. Fig. 6.2(a) shows an overview of our proposed system with three main components: (1) The TurtleBot platform (yellow); (2) Audio communication interface (purple); (3) Dialogue and grounding modules (red). The modules communicate with each other through ROS. We expand part (1) and (2) in this section and part (3) in Sec. 6.4.

6.3.1 Robot platform

Overview

We use the Turtlebot2i as our robot platform. As shown in Fig. 6.2(b), the robot is fitted with the following sensors and equipment: (1) An RP-Lidar A3 laser range finder is mounted on the top of the robot structure for SLAM; (2) An Intel RealSense D435i camera is mounted on the top of a monopod facing forward for scene description and question answering; (3) A wireless headset is used to communicate with the user. The headset is lightweight and maximally protects the users' privacy, while the absence of wires avoids tripping hazards; (4) A T-shaped handle is attached to the top rear side of the robot as a holding point for the user's arm. The handle allows users to choose their preferred holding configurations such as one hand or two hands. The robot is connected to a desktop computer which provides more computation resources through WiFi.

Planning and Navigation

The robot operations are managed by the ROS `move_base` navigation stack, which is a standard package to autonomously navigate a mobile robot to a given goal pose. Before navigation, we create a 2D occupancy map of the environment using laser-based SLAM and mark the semantic landmarks at the same time (see Fig. 6.3 and Sec. 6.4.2 for details). At the beginning of each trial, the goal pose is obtained from the dialogue with the user (further specified in Sec. 6.4). During navigation, adaptive Monte Carlo localization is used to localize the robot on the map. We use the dynamic window approach (DWA) [67] and A^* as local and global planners, respectively. The minimum translational velocity is restricted to be non-negative to prevent the robot from moving backward and colliding with the user. The maximum velocity of the robot can be adjusted by the user (see Sec. 6.4.4).

6.3.2 Audio Communication Interface

Speech is a natural choice for human-robot communication, particularly in cases where the human has limited vision [18], [149], [150]. To this end, as shown in purple in Fig. 6.2(a), we develop an audio communication interface that consists of: (1) Input: When audio is captured by the `audio_capture` package, the OpenAI Whisper speech recognition model [151] transcribes speech commands to text, which are passed to the NLU module. The SR module continuously transcribes the audio from the microphone and publishes the text transcriptions to a ROS topic in real time; (2) Output: We use the Google text-to-speech (TTS) service to convert the text output from the visual-language modules and navigation module to speech, which is then narrated to the user via the headset. The TTS is another ROS topic that converts and plays the synthesized sound constantly.

6.4 Dialogue And Grounding

The goal of DRAGON is to connect the user with the environment through conversation. In this section, we describe how our dialogue system understands user language (Sec. 6.4.1), maps and localizes semantic landmarks (Sec. 6.4.2), provides information about the environment (Sec. 6.4.3), and adjusts the navigation preference of the user (Sec. 6.4.4). Our grounding system is visualized in the red parts of Fig. 6.2(a). The inputs to the subsystem are the transcribed texts from SR and the outputs are synthetic speech from TTS.

6.4.1 Natural language understanding (NLU)

The NLU takes a transcribed sentence as input and outputs user intents and entities of interest. The intent recognizer is a multi-label classifier with all classes shown in Table 6.2. The intents are designed based on the needs of our tasks. The entities are locations, objects, and object attributes which include the material and functionalities of an object. We use Dual Intent and Entity Transformer for intent classification and entity recognition [152]. We train the model using a custom dataset with 1092 sentences collected by ourselves. For each intent, we collect various expressions including misspelled and phonetically similar phrases, which makes our NLU robust to the nuances of human language and the errors caused by the SR. For example, “a think” and “a sink” both refer to the kitchen sink. We also collected expressions for multi-intents and unknown

Table 6.2: All user intents and their descriptions.

Intents	**Descriptions**
Greet	Wake up the robot and begin an interaction.
Object goal	Go to a specific object landmark. May contain entities including objects and attributes.
Location goal	Go to a rough goal location (kitchen, lounge, etc) without mentioning a specific object. May contain location entities.
Affirm	Confirm the goal.
Deny	Deny the goal.
Describe	Ask for a description of the surrounding environment.
Ask	Ask a question about the surrounding environment.
Pause	Pause the current navigation.
Resume	Resume the current navigation.
Accelerate	Increment velocities, up to a limit.
Decelerate	Decrement velocities, down to a limit.
Unknown	The text does not belong to any intents above (i.e. be noise, chitchat, etc) and is ignored by the robot.

intents so that the NLU can fulfill a request containing multiple intents and ignore noise input. For instance, "Hello robot, can you take me to a sofa?" will both activate the robot and set an object goal. Once the intent and entities are extracted, the corresponding downstream module is activated. The NLU may pass additional input arguments to modules such as extracted entities or the whole sentence. Different downsteam models are triggers based on extracted intents and entities.

During navigation, the landmark recognition is triggered if the user intent is *Object goal* or *Location goal* and the NLU extracts a goal object from the input sentence. The extracted information of the goal is kept in memory throughout the conversation. If the user mentions additional information about the landmark, we use simple prompt engineering to make the description more specific. For example, locations and attributes of objects, such as "a chair in the kitchen," can be added to the memorized description. In addition, the robot uses clarification dialogue to disambiguate the desired landmark if the input description does not contain any object. If the user only provided the location or attributes without mentioning the object name (*e.g.* "Take me to the kitchen"), our system provides hints to encourage the user to provide more specific descriptions (*e.g.* "What object are you looking for in the kitchen?"). If there are multiple similar objects in different landmarks, our system disambiguates the user's preferred landmark (*e.g.* "What kind of chair are you looking for? A dining chair, an office chair, or a sofa?"). After choosing a unique landmark, our system confirms the memorized goal description with the user (*e.g.* "Do you wish to go to a dining chair?"). No further action is taken until the user affirms the goal. The memorized goal information is cleared after the confirmation to prepare for the next goal.

With the disambiguation and confirmation dialogue, the NLU is able to precisely capture the user's desired destination with minimal constraints on the user's phrasing, which is crucial for the whole navigation experience. Using better language models for the NLU is left for future work.

6.4.2 Landmark mapping and recognition

To guide the user to their object goals, we first record the images and locations of landmarks during SLAM. Then, we use a fine-tuned CLIP model to match the user's description with goal images, whose corresponding location and orientation are sent to the navigation stack for navigation guidance. The CLIP model version is ViT-B/32 [130].

The landmark mapping process is performed simultaneously with SLAM. During SLAM, when the robot is at a landmark that might be a point of interest, we simply save the current robot pose in the map frame and an RGB image of the landmark to the disk with a single key press. No labels or text descriptions are needed at this stage. The resulting landmark map is shown in Fig. 6.3.

During navigation, this module is activated when the intent is *Object goal* or *Location goal*. After the goal is confirmed by the NLU, the CLIP model selects the landmark whose image has the highest similarity score with the descriptions of landmarks. To obtain the image-text similarity score, a text encoder and an image encoder first convert the input text and all images to vector embeddings. Then, the text and image similarity score is computed by the cosine similarity between the pairwise text and image embeddings. The image with the highest similarity score is selected as the goal. Finally, the corresponding location of the chosen landmark on the map is sent to an action client, which sets the goal for the robot.

The zero-shot performance of pre-trained CLIP models is not satisfactory in our environment due to distribution shifts. As shown in Fig. 6.3, the objects in the images are frequently cropped due to the low mounting point of our camera and the close distance between the camera and the objects. In addition, the descriptions of landmarks from a PwVI might be vaguer than those in public datasets (*e.g.* "a chair" v.s. "a blue chair in front of a white wall"). To this end, we fine-tune the CLIP model with a custom dataset containing 544 image and text description pairs with a 8×10^{-6} learning rate for 35 epochs. The images are taken by the robot camera in our environment and the text is provided by the authors. By using an open-vocabulary model to recognize landmarks, DRAGON can handle free-form language and is not limited to a fixed set of object classes. Thus, the user expressions are less restricted, making the grounding module easier for non-experts to use.

6.4.3 Environment understanding modules

To help the user gain awareness of their surroundings, we use an object detector [153] to describe the objects (activated if the intent is *Describe*) and a visual-question-answering (VQA) model [154] to answer the user's questions (activated if the intent is *Ask*). Both models take the current camera image as input.

The output of the object detector consists of a list of detected instances, their object classes, confidence scores, and bounding boxes. To avoid narrating a long list and to keep the description concise, we post-process the output as follows. We first apply non-maximum suppression and filter out the detected instances with low confidence scores. Then, for the remaining instances, we keep the top three classes with the largest average bounding boxes, and list the object class names together with the numbers of objects (*e.g.* "2 chairs, 1 person, and 1 table").

The VQA model takes the current camera image and the user's question from the SR and outputs a short answer to the question. Since healthy people and PwVI would ask different questions to the same images [155], we collect a dataset of 10252 (image, question, answer) triplets to fine-tune the VQA model for 20 epochs. Again, images are taken by the robot camera in our environment and the text is provided by the authors. To handle free-form user expressions, the dataset contains cases where multiple questions have the same meaning but different phrasing (*e.g.* "Is any person in front of me?" and "Anyone here?").

Finally, the outputs of the object detector and VQA are narrated to the user in real time. Since both models can only take an RGB image, our system cannot provide depth-based information or detect anything out of the camera view.

6.4.4 Navigation preference customization

To accommodate the different walking paces of users and to avoid tiring the user during navigation, the robot can change its speed (activated if the intent is *Accelerate* or *Decelerate*), take a pause (*Pause*), and resume (*Resume*). To pause the robot, our system stores and cancels the current goal from the action client in the navigation stack. To resume, the stored goal is sent to the action client again. To update the speed, we change the maximum translational and rotational velocities of the DWA local planner in real-time.

6.5 Experiments

6.5.1 Baseline

We compare the CLIP-based landmark recognizer with a closed-vocabulary object detector as the baseline [153] [1]. The vocabulary size, or the number of classes, of the detector is more than 1200 and it is fine-tuned with the same amount of data as CLIP. In the baseline, the landmark images are passed into the object detector, which outputs the class names of detected objects. During navigation, the baseline chooses the landmark with the highest number of objects mentioned by the user. Since the vocabulary of object detectors is fixed, the baseline is unable to incorporate an object's attributes or locations obtained from disambiguation. All other modules are the same for our system and the baseline.

6.5.2 User Study

Environment

All experiments were conducted in an everyday indoor environment in a university building. Three routes were created with furniture obstacles. Fig. 6.3 provides a layout of the environment, all landmarks, and three routes highlighted with orange curves. The routes were designed to have varying levels of difficulties for the system to correctly interpret the destination. Specifically, landmark A of Route 1 contains simple objects, landmark B of Route 2 contains more complicated objects, and landmark C of Route 3 contains a transparent door that is hard for object recognition.

Participants

The user study was conducted with N=5 participants (mean age=26; 3 males; 2 females; all participants were university students). All participants have full (corrected) vision and are asked to wear a blindfold to simulate a visual impairment. While our true target population are PwVI, the purpose of this pilot study is to validate the capabilities of DRAGON. A user study with PwVI is left for future work.

[1]Open-vocabulary object detectors exist [153]. We choose a closed-vocabulary detector to represent a closed-vocabulary grounding model.

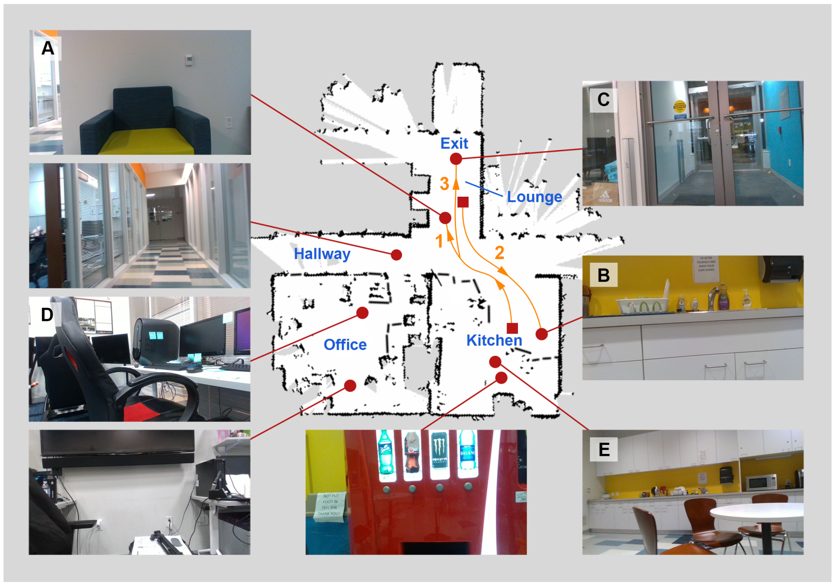


Figure 6.3: **The map of our environment with semantic landmarks.** The images are landmarks with locations marked by red dots. The orange lines are the three routes in the user study. The red squares are the starting locations of routes.

Table 6.3: Example expressions and their corresponding landmarks from CLIP v.s. the detector. The landmark labels are from Fig. 6.3. Underlined expressions are collected from the user study.

Landmark	CLIP	Detector
A	door, exit, entrance, gate glass door, automatic door	poster
B	sofa, couch, coach, fabric chair, relaxing chair thermostat, climate control	sofa, thermostat
C	sink, think, sync, faucet soap, hand wash, water pipe paper towel dispenser, bowls kitchen countertop, drying rack	faucet bottle dispenser bowl

Procedure

Participants were first familiarized with the goals of the study and requested to fill a demographic and robot technology survey. Then, participants were provided with a test run to get familiar with the system and its intricate navigation feedback mechanism. To begin the trial, the users were asked to command the robot to take them to a predetermined goal destination. Participants were not constrained in either the vocabulary or the sentence structure of their speech commands. The users were also informed that they could interact with the robot (*e.g.* ask for a description of their surroundings) at any point of the navigation. After each route, we used a short questionnaire to measure the participant's perception of the system. A strictly structured post-survey interview was conducted after participants finished all three routes to collect their feedback with the system. The same procedure was performed for CLIP and the detector, resulting in a total of 15 trials per method (3 routes and 5 users). The order of which method was tested first was randomized for each

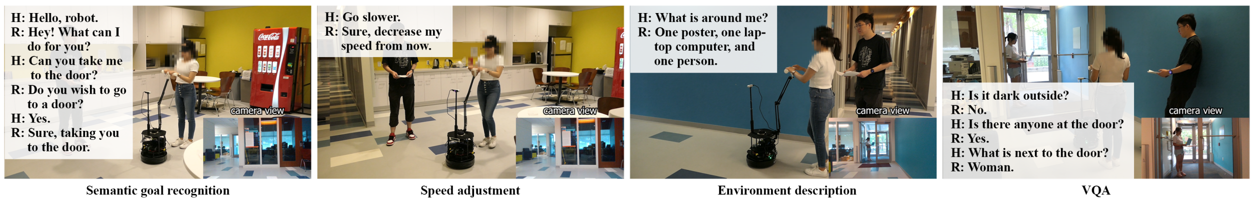

H: Hello, robot.
R: Hey! What can I do for you?
H: Can you take me to the door?
R: Do you wish to go to a door?
H: Yes.
R: Sure, taking you to the door.
camera view
Semantic goal recognition
H: Go slower.
R: Sure, decrease my speed from now.
camera view
Speed adjustment
H: What is around me?
R: One poster, one laptop computer, and one person.
camera view
Environment description
H: Is it dark outside?
R: No.
H: Is there anyone at the door?
R: Yes.
H: What is next to the door?
R: Woman.
camera view
VQA

Figure 6.4: **An example navigation trial with human-robot dialogue in the user study.** In the dialogue boxes, "H" denotes the human and "R" denotes the robot. The camera view is shown in the lower right corner.

participant to minimize the bias introduced due to the order of testing. All materials included in the user study, including a full walkthrough of the whole study for a participant and all questionnaires can be found here: https://drive.google.com/file/d/15KNR6C82mUrKSPMFRCnAJZ1C2NGX7dXJ/view.

6.5.3 Metrics

Objective Metrics

We measure the accuracy of all interactions during the user study, including 312 NLU, 30 landmark recognition (LR) and navigation trials, 15 environment description (EnvDes), 74 VQA, and 15 navigation preference adjustment (NavAdj). The NLU is correct when the extracted intent and entities (if any) are both correct. We also measure the accuracy of the NLU by taking the correctness of SR into account to analyze the effect of wrong SR. The effect of wrong NLU outputs is ignored when evaluating its downstream modules. An LR is considered correct if the robot chooses the correct landmark. A navigation trial is successful if the robot guides the user to the correct landmark without any delays or collisions along the route. An EnvDes is considered fully correct if all named objects exist in the camera image and the number of all objects is correct. It is considered partially correct if all named objects exist but the number of some objects is wrong. The correctness of answers from VQA is based on the camera images, not on the information out of the camera view. A NavAdj is successful if the change in robot speed is consistent with the user commands.

Subjective metrics

For both methods, we compare the scores for categories from the short questionnaire in Table 6.6. The difference in scores for each participant was aggregated and analyzed to discount individual biases. We evaluate user preferences for the other modules through a simple Likert scale analysis on the responses from the post-survey interview. Additionally, participants' feedback is summarized for qualitative analysis.

Table 6.4: Success rates (%) of LR and navigation (including overall success rate, and success rate if LR is correct), and the average number of dialogue rounds for a successful LR.

Method	**LR**		**Navigation**	
	Overall	# rounds	Overall	Correct LR
Ours	100	2.4	100	100
Baseline	46.67	3.71	46.67	100

Table 6.5: Accuracies (%) of the SR, NLU (including overall accuracy, accuracy if SR is correct and if SR is wrong), EnvDes with fully correct and partially correct number of objects, VQA, and navigation adjustment modules.

SR	**NLU**			**EnvDes**		**VQA**	**NavAdj**
	Overall	Correct SR	Wrong SR	Full	Partial		
70.19	85.26	93.61	65.59	45.45	75.76	82.43	100

6.6 Results

In this section, we discuss the results of our user study. Example navigation trials, as well as demonstrations of each module during the user study, are in this video and Fig. 6.4.

6.6.1 Quantitative Evaluation

LR and navigation

CLIP and the baseline only differ in LR and its resulting navigation. As seen in Table 6.4, the success rate of navigation is 100% if LR succeeds, because ROS navigation stack can navigate the robot to any desired goal pose robustly in our environment. This dependency indicates that the performance of LR is the key factor for navigation in the DRAGON system.

For LR, as shown in Table 6.4, our CLIP model with disambiguation outperforms the detector baseline by achieving 100% success rate in LR and navigation with fewer rounds of dialogue on average. We attribute this result to the fact that CLIP is an open vocabulary model that can take free-form query text, which is essential for our task because the user may use different expressions to refer to the same landmark. On the contrary, a closed vocabulary object detector can only handle a fixed set of object classes with limited expressions. For example, in Table 6.3, although both models can handle different objects that belong to the same landmark, CLIP can associate synonyms, such as “sofa” and “couch,” and wrong transcriptions, such as “coach,” to the correct landmark. In contrast, the closed-vocabulary detector can only handle strictly fixed expressions. The detector misidentifies some objects such as the transparent door in Landmark C after fine-tuning. Since our target users are usually non-experts, the baseline sometimes needs the user to rephrase multiple times to recognize the goal, which causes the user to run out of patience, and results in failure or more rounds of dialogue.

Besides CLIP, the disambiguation dialogue also contributes to the performance. With disambiguation, additional information such as the material and functionality of objects can be merged into the query text, such as “fabric chair” and “relaxing chair,” as shown in Table 6.3. These rich descriptions are helpful in distinguishing landmarks that have the same objects with different attributes, such as the different types of chairs in Landmark A, D, and E in Fig. 6.3 with fewer rounds of user rephrasing.

NLU

In Table 6.5, the overall accuracy of NLU is over 15% higher than SR, as the NLU is trained with incorrectly transcribed text and thus can work even when SR is incorrect. However, we do notice that NLU performs better with correct SR. The common failure cases of NLU occur when (1) The SR mistakenly breaks a sentence into two halves (*e.g.* “Is there anything?” and “To my right.” are treated as two sentences); and (2)

The NLU does not correctly extract intents from noisy transcriptions and chitchat, which can happen during the user study. Thus, we believe that a better SR engine would vastly benefit the performance of the whole system. However, since DRAGON will not begin navigation until the user confirms the goal in the dialogue, the wrong SR and NLU have little effect on navigation.

Other Modules

The system's environment descriptions are sometimes inaccurate due to errors in the object detector such as: (1) detecting incorrect number of objects (*e.g.* 3 wall sockets, when there was only 1 present); and (2) incorrect object classifications of rare or uncommon objects (*e.g.* a building information tablet was classified as a poster). Although we use non-maximum suppression and confidence score threshold to reduce the errors, they are hard to entirely eliminate due to the data distribution shift and the blurry images caused by the robot motion. Nevertheless, in Table 6.5, the model is able to output a list of objects with correct class names in 75.76% of the cases, which might be more important to the user than a correct number of objects.

The VQA module accurately answers the user's questions in 82.43% of the cases. The model fails in cases where the user asks questions that the robot cannot answer based on a single RGB image. For example, without precise depth information the VQA model only answers "far" or "close" if the question is "How far is the person from me?". Without a wider field of view, the model outputs objects on the front side if the question is "What is on my right?".

6.6.2 Qualitative Evaluation

In Table 6.6, participants showed an increasing preference for DRAGON with CLIP over the detector in all user experience categories across all routes. Specifically, participants reported a 32% improvement with a mean score difference of 1.60 ± 0.89 in the overall experience and a mean score difference of 1.40 ± 0.89 in the communication experience. The difference increases as the goal landmark contains more complicated objects in Route 2, and objects that are difficult to detect in Route 3, where the failures in LR significantly lower the user score for the detector based system. Particularly, participants noted that DRAGON with CLIP understood their intent, asked good follow-up questions, and correctly guided them to their destination. In contrast, the closed-vocabulary detector failed at these aspects and occasionally was unable to recognize destinations even though they existed. Participants also noted that the failures in intent understanding led to a frustrating communication experience with the detector.

One user in particular mentioned that the CLIP based model "... was able to actually understand me, so it accurately took me to the location and correctly answer [sic] my questions." while the detector based model "... would confirm the location I wanted to go to but could not find [sic; participant meant understand] the right location". However, users also mentioned potential improvements for DRAGON including more detailed environment descriptions, a quicker response time, and warnings of potential dangers such as "We're going through a door."

For the user experience categories that are the same for both LR methods, such as the 'intuitiveness of communication interface' and the ability of the system to aid in 'gaining awareness of the environment', participants reported average scores of 7.07 ± 2.17 and 6.07 ± 3.21, respectively. As evidenced by these scores, the users' opinions regarding these two categories were positive, due to the inclusion of the dialogue and grounding modules. However, participants highlighted minor inaccuracies in the environment descriptions and the slow pace of communication due to processing times and network delays as potential issues.

Table 6.6: Mean user experience scores on a scale of 1 to 10.

Use experience category	**Route 1**		**Route 2**		**Route 3**	
	CLIP	Detector	CLIP	Detector	CLIP	Detector
Ease of following	**8.8**	8.6	**8.8**	5.6	**9.2**	1.0
Navigational Experience	**8.4**	7.4	**7.6**	4.8	**8.8**	1.0
Intent Understanding	7.6	**8**	**7.6**	4.6	**8.4**	3.4

6.7 Limitations

DRAGON encompasses the following limitations, which opens up directions for future work. First, the current dialogue system is rule-based with fixed behaviors for each intent. Replacing hard-coded rules with adaptive learning-based policies, such as large language models, should generalize to more complex user behaviors and more subtasks. Second, the environment understanding modules provide limited information. The environment understanding module can only enlist detected objects, and the VQA module can only answer simple questions with a single-word answer. Future informative descriptions should include object relationships in images, incorporate information from the map and other sensors, and inform users about the planned path and potential dangers. Finally, the physical interface of the platform should be redesigned to improve ergonomics.

6.8 Summary

In conclusion, we present DRAGON, a first-of-its-kind guide robot that fulfills user intents and familiarizes the user with their surroundings through interactive dialogue. We use CLIP to retrieve landmark destinations from commands and provide visual information through language. The user study shows promising communication, grounding, and navigation performance of DRAGON. Our work suggests that visual-language grounding and dialogue can greatly improve human-robot interaction.

Videos and more information are available at `https://sites.google.com/view/dragon-wayfinding`.

Code implementation in ROS and Python is available at `https://github.com/Shuijing725/Dragon_Wayfinding`.

Chapter 7

Conclusion

7.1 Summary of contributions

This thesis presents systems and algorithms for robot navigation in interactive human environments by answering the following questions: (1) How to predict unspoken intentions of humans and reason about implicit interactions when navigating among humans? (2) How to infer the human intentions through dialogue and exchange semantic information about the surrounding environment?

Chapter 2 presented an unsupervised method for inferring driver traits such as driving styles from observed vehicle trajectories. We use a variational autoencoder with recurrent neural networks to learn a latent representation of traits without any ground truth trait labels. Then, we use this trait representation to learn a policy for an autonomous vehicle to navigate through a T-intersection with deep reinforcement learning. Our pipeline enables the autonomous vehicle to adjust its actions when dealing with drivers of different traits to ensure safety and efficiency. Our method demonstrates promising performance and outperforms state-of-the-art baselines in the T-intersection scenario.

Chapter 3 presented decentralized structural-Recurrent Neural Network (DS-RNN), a novel network that reasons about spatial and temporal relationships for robot decision making in crowd navigation. DS-RNN is trained with model-free deep reinforcement learning without any expert supervision. Simulation experiments demonstrate that our model outperforms previous methods in challenging crowd navigation scenarios. We successfully transfer the policy learned in the simulator to a real-world TurtleBot 2i.

Chapter 4 presented a novel recurrent graph neural network with attention mechanisms to capture heterogeneous interactions among agents through space and time in dense crowds. To encourage longsighted robot behaviors, we infer the intentions of dynamic agents by predicting their future trajectories for several timesteps. The predictions are incorporated into a model-free RL framework to prevent the robot from intruding into the intended paths of other agents. We demonstrate that our method enables the robot to achieve good navigation performance and non-invasiveness in challenging crowd navigation scenarios. We successfully transfer the policy learned in simulation to a real-world TurtleBot 2i.

Chapter 5 presents a scene representation and structured graph attention network for navigation in dynamic and constrained environments. The scene representation and network enable smooth deployment of the robot in real domains with a policy trained using reinforcement learning (RL) with a low-fidelity simulator and no demonstration data. In the representation, we separate human and obstacles and leverage on processed state information. Additionally, we extend our previous spatio-temporal graph attention network to capture interactions among humans and obstacles. We demonstrate the effectiveness of this method through various challenging constrained crowd navigation scenarios in both simulation and real-world.

Chapter 6 presented a guiding robot that interacts with blind people through language and physical forces. The robot is equipped with a dialogue system and can ground objects and landmarks in the environment with natural language. The main functionalities of the robot include understanding the commands from the user, guiding the user to the desired landmarks on the map, describing the environment, and answering questions from visual observations. We conduct a user study with blindfolded participants in an everyday indoor environment. Our results demonstrate the effectiveness and ease of use of our system.

Overall, these contributions address key challenges in deploying robots in human environments. However, remaining challenges still exist before deploying these works in real-world human environments. Practical insights and possible directions for future work are described below.

7.1.1 Practical insights

Translating the systems and algorithms presented in this thesis into real-world applications involves addressing the following considerations. First, improving human models is needed to minimize sim2real gaps caused by inaccurate human behaviors. Besides ORCA and IDM, data-driven methods might be beneficial to improving social-awareness and naturalness of human models [121]. Second, integrating these advanced algorithms into existing robotic hardware requires ensuring compatibility and optimizing performance to operate in real-time. This might involve hardware upgrades or modifications to enhance sensor capabilities, perception modules, and computational efficiency. Third, developing comprehensive safety protocols and fail-safe mechanisms is essential to guarantee the safety of both humans and robots in interactive environments. Lastly, regulatory and ethical considerations must be addressed to ensure compliance with legal standards and societal acceptance. Addressing the above practical insights can enhance the deployment and mass production of the proposed systems in real life.

7.2 Future work

For robots to serve humans in all types of everyday scenarios, we need to iterate between training the robot and deploying it in real human environments. To facilitate this training and deployment loop, we point out future work in the following three directions: 1) Developing a unified model for both implicit and explicit interactions to unlock more robot capabilities; 2) Learning social behaviors from foundational models to improve the finetuning efficiency during deployment; 3) Developing intuitive lifelong learning methods from non-expert humans.

7.2.1 Unified interaction models

For robots to collaborate with humans on tasks such as preparing a meal, they must interact with humans both implicitly and explicitly. For this reason, I aim to expand the spatio-temporal interaction graph to capture interactions beyond movement, such as motions, dialogue, gaze, and so on. However, as we expand the graph, we must consider computational constraints and better understand the important factors in different interaction settings. To achieve this, inspired by the successful adaptation of st-graph in complex prediction tasks [63], I plan to design st-graphs with edges that encode various types of interactions.

7.2.2 Interaction learning from foundational models

My previous works train predictors and planners using a separate dataset or simulator for each task. However, this “tabula rasa” paradigm can be data inefficient. The trained models also have limited generalization across different tasks [156]. On the other hand, large language models (LLMs) encompass common sense knowledge, such as social norms [157]. To improve the data efficiency and generalization of my method, I plan to incorporate visual language foundation models into the intention predictor and robot planner. The proposed approach takes observed video frames and task-specific prompts as inputs and outputs human intentions or robot motion primitives. Since my previous framework makes minimal assumptions on the form of submodules, it has promising compatibility for LLMs.

7.2.3 Data efficient and intuitive learning from non-experts

When a trained robot is deployed in everyday environments, its performance drops inevitably due to domain shifts. Ideally, we need data-efficient and intuitive fine-tuning algorithms that allow non-experts with little domain expertise to customize and improve the robot. Building on my previous work on planning with representations [129], I aim to propose 1) intuitive user interfaces for non-experts to provide finetuning data from their own devices, such as phone cameras and microphones, and from direct physical teleoperation; 2) data-efficient algorithms for robot to self-improve with minimal data collected from non-experts [158], [159]. With an intuitive and data-efficient finetuning mechanism, the robot can continually improve itself in target environments and customize toward user preferences.

References

[1] K. Conger, *Driver charged in uber's fatal 2018 autonomous car crash*, https://www.nytimes.com/2020/09/15/technology/uber-autonomous-crash-driver-charged.html, Accessed: May 2024, 2020.

[2] T. Thadani, *Cruise recalls all its driverless cars after pedestrian hit and dragged*, https://www.washingtonpost.com/technology/2023/11/08/cruise-crash-driverless-recall, Accessed: May 2024, 2023.

[3] D. Elinav, *Delivery robot deployment models: Autonomous vs. remote operation*, https://driveu.auto/blog/last-mile-delivery-robots-deployment-how-automotive-startups-can-adopt-a-minimal-viable-product-mindset/, Accessed: May 2024.

[4] J. P. Tuohy, *Amazon astro review: Too much alexa, not enough arms*, https://www.theverge.com/23141966/amazon-astro-robot-review, Accessed: May 2024, 2022.

[5] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*, Springer, 2011, pp. 3–19.

[6] Y. Yang, X. Lou, and C. Choi, "Interactive robotic grasping with attribute-guided disambiguation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 8914–8920.

[7] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010, pp. 797–803.

[8] V. Vlasov, A. Drissner-Schmid, and A. Nichol, "Few-shot generalization across dialogue tasks," *arXiv preprint arXiv:1811.11707*, 2018.

[9] Y. Hu, J. Yang, L. Chen, *et al.*, "Planning-oriented autonomous driving," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 17 853–17 862.

[10] M. Ahn, A. Brohan, N. Brown, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on Robot Learning (CoRL)*, 2022.

[11] W. Zhan, L. Sun, D. Wang, *et al.*, "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps," *arXiv preprint arXiv: 1910.03088*, 2019.

[12] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[13] J. Morton and M. J. Kochenderfer, "Simultaneous policy learning and latent state inference for imitating driver behavior," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[14] X. Ma, J. Li, M. J. Kochenderfer, D. Isele, and K. Fujimura, "Reinforcement learning for autonomous driving with latent state inference and spatial-temporal relationships," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

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

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

[17] V. Kulyukin, C. Gharpure, J. Nicholson, and S. Pavithran, "Rfid in robot-assisted indoor navigation for the visually impaired," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, 2004, 1979–1984 vol.2.

[18] M. A. Bayles, T. Kadylak, S. Liu, *et al.*, "An interdisciplinary approach: Potential for robotic support to address wayfinding barriers among persons with visual impairments," in *Human Factors and Ergonomics Society Annual Meeting (HFES)*, vol. 66, 2022, pp. 1164–1168.

[19] Z. Li and R. Hollis, "Toward a ballbot for physically leading people: A human-centered approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4827–4833.

[20] J. Guerreiro, D. Sato, S. Asakawa, H. Dong, K. M. Kitani, and C. Asakawa, "Cabot: Designing and evaluating an autonomous navigation robot for blind people," in *International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2019, pp. 68–82.

[21] D. Shah, B. Osiński, brian ichter, and S. Levine, "LM-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on Robot Learning (CoRL)*, 2022.

[22] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual language maps for robot navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[23] Z. N. Sunberg, C. J. Ho, and M. J. Kochenderfer, "The value of inferring the internal state of traffic participants for autonomous freeway driving," in *American Control Conference (ACC)*, 2017, pp. 3004–3010.

[24] "Traffic safety facts 2019," National Highway Traffic Safety Administration, Tech. Rep. DOT HS 813 141, 2021, p. 84.

[25] B. Qian, H. Zhou, F. Lyu, J. Li, T. Ma, and F. Hou, "Toward collision-free and efficient coordination for automated vehicles at unsignalized intersection," *IEEE Internet of Things Journal*, vol. 6, no. 6, pp. 10 408–10 420, 2019.

[26] W. Song, G. Xiong, and H. Chen, "Intention-aware autonomous driving decision-making in an uncontrolled intersection," *Mathematical Problems in Engineering*, vol. 2016, 2016.

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

[28] K. Brown, K. Driggs-Campbell, and M. J. Kochenderfer, "A taxonomy and review of algorithms for modeling and predicting human driver behavior," *arXiv preprint arXiv:2006.08832*, 2020.

[29] C. Dong, J. M. Dolan, and B. Litkouhi, "Intention estimation for ramp merging control in autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1584–1589.

[30] H. Bai, S. Cai, N. Ye, D. Hsu, and W. S. Lee, "Intention-aware online pomdp planning for autonomous driving in a crowd," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 454–460.

[31] K. Driggs-Campbell, V. Govindarajan, and R. Bajcsy, "Integrating intuitive driver models in autonomous planning for interactive maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 12, pp. 3461–3472, 2017.

[32] K. Driggs-Campbell, V. Shia, and R. Bajcsy, "Improved driver modeling for human-in-the-loop vehicular control," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 1654–1661.

[33] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: Leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, 2018.

[34] C. M. Martinez, M. Heucke, F.-Y. Wang, B. Gao, and D. Cao, "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2017.

[35] W. Dong, T. Yuan, K. Yang, C. Li, and S. Zhang, "Autoencoder regularized network for driving style representation learning," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2017, pp. 1603–1609.

[36] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *International Conference on Computer Vision (ICCV)*, 2015, pp. 2794–2802.

[37] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and learn: Unsupervised learning using temporal order verification," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 527–544.

[38] Y. Liu, Q. Yan, and A. Alahi, "Social nce: Contrastive learning of socially-aware motion representations," in *International Conference on Computer Vision (ICCV)*, 2021.

[39] J.-B. Grill, F. Strub, F. Altché, *et al.*, "Bootstrap your own latent - a new approach to self-supervised learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020, pp. 21 271–21 284.

[40] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *International Conference on Learning Representations (ICLR)*, 2014.

[41] K. Sohn, H. Lee, and X. Yan, "Learning structured output representation using deep conditional generative models," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015, pp. 3483–3491.

[42] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio, "Generating sentences from a continuous space," *arXiv preprint arXiv:1511.06349*, 2015.

[43] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 683–700.

[44] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2020.

[45] X. Feng, Z. Cen, J. Hu, and Y. Zhang, "Vehicle trajectory prediction using intention-based conditional variational autoencoder," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2019, pp. 3514–3519.

[46] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3399–3406.

[47] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2034–2039.

[48] A. Cosgun, L. Ma, J. Chiu, *et al.*, "Towards full automated drive in urban environments: A demonstration in gomentum station, california," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1811–1818.

[49] M. M. Minderhoud and P. H. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, no. 1, pp. 89–97, 2001.

[50] M. Bouton, A. Cosgun, and M. J. Kochenderfer, "Belief state planning for autonomously navigating urban intersections," in *IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 825–830.

[51] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 392–399.

[52] K.-H. Lee, "A study on distance measurement module for driving vehicle velocity estimation in multi-lanes using drones," *Applied Sciences*, vol. 11, no. 9, p. 3884, 2021.

[53] K. Han, E. Lee, M. Choi, and S. B. Choi, "Adaptive scheme for the real-time estimation of tire-road friction coefficient and vehicle velocity," *IEEE/ASME Transactions on mechatronics*, vol. 22, no. 4, pp. 1508–1518, 2017.

[54] A. Kesting, M. Treiber, and D. Helbing, "Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 368, no. 1928, pp. 4585–4605, 2010.

[55] E. Grefenstette, K. M. Hermann, M. Suleyman, and P. Blunsom, "Learning to transduce with unbounded memory," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, pp. 1828–1836, 2015.

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

[57] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1726–1743, 2013.

[58] P. Du, Z. Huang, T. Liu, *et al.*, "Online monitoring for safe pedestrian-vehicle interactions," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020.

[59] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[60] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[61] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2616–2625.

[62] P. Chang, S. Liu, H. Chen, and K. Driggs-Campbell, "Robot sound interpretation: Combining sight and sound in learning-based control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[63] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 5308–5317.

[64] Z. Huang, A. Hasan, and K. Driggs-Campbell, "Intention-aware residual bidirectional lstm for long-term pedestrian trajectory prediction," *arXiv:2007.00113*, 2020.

[65] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2255–2264.

[66] R. P. Bhattacharyya, D. J. Phillips, C. Liu, J. K. Gupta, K. Driggs-Campbell, and M. J. Kochenderfer, "Simulating emergent properties of human driving behavior using multi-agent reward augmented imitation learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 789–795.

[67] 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.

[68] J. Van den 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, pp. 1928–1935.

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

[70] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, "Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns," *Autonomous Robots*, vol. 35, no. 1, pp. 51–76, 2013.

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

[72] P. Trautman, J. Ma, R. Murray, and A. Krause, "Robot navigation in dense human crowds: The case for cooperation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 2153–2160.

[73] 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.

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

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

[76] 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.

[77] N. Roy, W. Burgard, D. Fox, and S. Thrun, "Coastal navigation-mobile robot navigation with uncertainty in dynamic environments," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, 1999, pp. 35–40.

[78] M. Hoy, 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.

[79] A. V. Savkin and C. Wang, "Seeking a path through the crowd: Robot navigation in unknown dynamic environments with moving obstacles based on an integrated environment representation," *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1568–1580, 2014.

[80] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[81] J. Snape, J. Van Den Berg, S. J. Guy, and D. Manocha, "The hybrid reciprocal velocity obstacle," *IEEE Transactions on Robotics*, vol. 27, no. 4, pp. 696–706, 2011.

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

[83] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, 2017.

[84] 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.

[85] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[86] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2018.

[87] L. Fan, W. Wang, S. Huang, X. Tang, and S.-C. Zhu, "Understanding human gaze communication by spatio-temporal graph reasoning," in *International Conference on Computer Vision (ICCV)*, 2019, pp. 5724–5733.

[88] M. Khodayar and J. Wang, "Spatio-temporal graph deep neural network for short-term wind speed forecasting," *IEEE Transactions on Sustainable Energy*, vol. 10, no. 2, pp. 670–681, 2018.

[89] A. Sadeghian, A. Alahi, and S. Savarese, “Tracking the untrackable: Learning to track multiple cues with long-term dependencies,” in *IEEE International Conference on Computer Vision (CVPR)*, 2017, pp. 300–311.

[90] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 1–7.

[91] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” in *International Conference on Learning Representations (ICLR)*, 2017.

[92] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.

[93] I. Kostrikov, *Pytorch implementations of reinforcement learning algorithms*, https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

[94] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[95] N. Wojke, A. Bewley, and D. Paulus, “Simple Online and Realtime Tracking with a Deep Association Metric,” in *2017 IEEE international conference on image processing (ICIP)*, 2017, pp. 3645–3649.

[96] Z. Pei, *Deep Sort with PyTorch*, 2020. [Online]. Available: https://github.com/ZQPei/deep_sort_pytorch.

[97] K. Li, M. Shan, K. Narula, S. Worrall, and E. Nebot, “Socially aware crowd navigation with multimodal pedestrian trajectory prediction for autonomous vehicles,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–8.

[98] 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, pp. 3277–3283.

[99] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, “Social lstm: Human trajectory prediction in crowded spaces,” in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016, pp. 961–971.

[100] Y. Huang, H. Bi, Z. Li, T. Mao, and Z. Wang, “Stgat: Modeling spatial-temporal interactions for human trajectory prediction,” in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[101] S. Eiffert, H. Kong, N. Pirmarzdashti, and S. Sukkarieh, “Path planning in dynamic environments using generative rnns and monte carlo tree search,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[102] K. Matsumoto, A. Kawamura, Q. An, and R. Kurazume, “Mobile robot navigation using learning-based method based on predictive state representation in a dynamic environment,” in *IEEE/SICE International Symposium on System Integration (SII)*, 2022, pp. 499–504.

[103] 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, pp. 11 345–11 352.

[104] A. Vemula, K. Muelling, and J. Oh, “Modeling cooperative navigation in dense human crowds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1685–1692.

[105] E. Leurent and J. Mercat, “Social attention for autonomous decision-making in dense traffic,” in *Machine Learning for Autonomous Driving Workshop at Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[106] Z. Huang, R. Li, K. Shin, and K. Driggs-Campbell, “Learning sparse interaction graphs of partially detected pedestrians for trajectory prediction,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 1198–1205, 2022.

[107] D. Jia, A. Hermans, and B. Leibe, “DR-SPAAM: A Spatial-Attention and Auto-regressive Model for Person Detection in 2D Range Data,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020, pp. 10 270–10 277.

[108] Z. Xie and P. Dames, “Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles,” *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2700–2719, 2023.

[109] C. Mavrogiannis, K. Balasubramanian, S. Poddar, A. Gandra, and S. S. Srinivasa, “Winding through: Crowd navigation via topological invariance,” *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 121–128, 2023.

[110] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, and J. Pan, “Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 6252–6259.

[111] S. Liu, P. Chang, Z. Huang, *et al.*, “Intention aware robot crowd navigation with attention-based interaction graph,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 015–12 021.

[112] Y.-J. Mun, M. Itkina, S. Liu, and K. Driggs-Campbell, “Occlusion-aware crowd navigation using people as sensors,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 12 031–12 037.

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

[114] D. Dugas, O. Andersson, R. Siegwart, and J. J. Chung, “Navdreams: Towards camera-only rl navigation among humans,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2504–2511.

[115] A. Pokle, R. Martın-Martın, P. Goebel, *et al.*, “Deep local trajectory replanning and control for robot navigation,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 5815–5822.

[116] A. H. Raj, Z. Hu, H. Karnan, *et al.*, “Rethinking social robot navigation: Leveraging the best of two worlds,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[117] N. Tsoi, A. Xiang, P. Yu, *et al.*, “Sean 2.0: Formalizing and generating social situations for robot navigation,” *IEEE Robotics and Automation Letters*, pp. 1–8, 2022.

[118] N. Chakraborty, A. Hasan, S. Liu, *et al.*, “Structural attention-based recurrent variational autoencoder for highway vehicle anomaly detection,” in *IFAAMAS International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2023.

[119] E. Coumans and Y. Bai, *Pybullet, a python module for physics simulation for games, robotics and machine learning*, http://pybullet.org, 2016–2019.

[120] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[121] H. Chen, T. Ji, S. Liu, and K. Driggs-Campbell, "Combining model-based controllers and generative adversarial imitation learning for traffic simulation," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2022, pp. 1698–1704.

[122] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[123] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning (ICML)*, 2017, pp. 2817–2826.

[124] L. Jin, H. Zhang, and C. Ye, "A wearable robotic device for assistive navigation and object manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 765–770.

[125] J. Wilson, B. N. Walker, J. Lindsay, C. Cambias, and F. Dellaert, "Swan: System for wearable audio navigation," in *IEEE International Symposium on Wearable Computers*, 2007, pp. 91–98.

[126] S. A. Cheraghi, V. Namboodiri, and L. Walker, "Guidebeacon: Beacon-based indoor wayfinding for the blind, visually impaired, and disoriented," in *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2017, pp. 121–130.

[127] D. Sato, U. Oh, K. Naito, H. Takagi, K. Kitani, and C. Asakawa, "Navcog3: An evaluation of a smartphone-based blind indoor navigation assistant with semantic features in a large-scale environment," in *International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2017, pp. 270–279.

[128] V. Kulyukin, C. Gharpure, J. Nicholson, and G. Osborne, "Robot-assisted wayfinding for the visually impaired in structured indoor environments," *Autonomous robots*, vol. 21, pp. 29–41, 2006.

[129] P. Chang, S. Liu, and K. Driggs-Campbell, "Learning visual-audio representations for voice-controlled robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[130] A. Radford, J. W. Kim, C. Hallacy, *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning (ICML)*, 2021, pp. 8748–8763.

[131] M. Saha, A. J. Fiannaca, M. Kneisel, E. Cutrell, and M. R. Morris, "Closing the gap: Designing for the last-few-meters wayfinding problem for people with visual impairments," in *International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS)*, 2019, pp. 222–235.

[132] Z. Yang, L. Yang, L. Kong, *et al.*, "Seeway: Vision-language assistive navigation for the visually impaired," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 52–58.

[133] V. A. Kulyukin and C. Gharpure, "Ergonomics-for-one in a robotic shopping cart for the blind," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2006, pp. 142–149.

[134] A. Nanavati, X. Z. Tan, J. Connolly, and A. Steinfeld, "Follow the robot: Modeling coupled human-robot dyads during navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 3836–3843.

[135] Y. Zhang, Z. Li, H. Guo, *et al.*, "'i am the follower, also the boss': Exploring different levels of autonomy and machine forms of guiding robots for the visually impaired," in *ACM Conference on Human Factors in Computing Systems (CHI)*, 2023.

[136] A. Xiao, W. Tong, L. Yang, J. Zeng, Z. Li, and K. Sreenath, "Robotic guide dog: Leading a human with leash-guided hybrid physical interaction," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11 470–11 476.

[137] R. Liu and X. Zhang, "A review of methodologies for natural-language-facilitated human-robot cooperation," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, 2019.

[138] S. Y. Min, D. S. Chaplot, P. Ravikumar, Y. Bisk, and R. Salakhutdinov, "Film: Following instructions in language with modular methods," in *International Conference on Learning Representations (ICLR)*, 2022.

[139] P. Anderson, Q. Wu, D. Teney, *et al.*, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 3674–3683.

[140] C. Chen, U. Jain, C. Schissler, *et al.*, "Soundspaces: Audio-visual navigation in 3d environments," in *European Conference on Computer Vision (ECCV)*, 2020, pp. 17–36.

[141] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation," *arXiv preprint arXiv:2203.10421*, 2022.

[142] H. Zhu, J. Yu, A. Gupta, *et al.*, "The ingredients of real world robotic reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2020.

[143] J. McCormac, A. Handa, A. Davison, and S. Leutenegger, "Semanticfusion: Dense 3d semantic mapping with convolutional neural networks," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 4628–4635.

[144] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3D scene graph construction and optimization," in *Robotics: Science and Systems*, 2022.

[145] S.-C. Wu, J. Wald, K. Tateno, N. Navab, and F. Tombari, "Scenegraphfusion: Incremental 3d scene graph prediction from rgb-d sequences," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 7515–7525.

[146] N. Savinov, A. Dosovitskiy, and V. Koltun, "Semi-parametric topological memory for navigation," in *International Conference on Learning Representations (ICLR)*, 2018.

[147] X. Meng, N. Ratliff, Y. Xiang, and D. Fox, "Scaling local control to large-scale topological navigation," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 672–678.

[148] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12 875–12 884.

[149] S. Azenkot, C. Feng, and M. Cakmak, "Enabling building service robots to guide blind people a participatory design approach," in *ACM/IEEE International Conference on Human-Robot Interaction*, 2016, pp. 3–10.

[150] P. Bhat and Y. Zhao, "'i was confused by it; it was confused by me:' exploring the experiences of people with visual impairments around mobile service robots," *ACM on Human-Computer Interaction*, vol. 6, no. CSCW2, pp. 1–26, 2022.

[151] A. Radford, J. W. Kim, T. Xu, G. Brockman, C. McLeavey, and I. Sutskever, *Robust speech recognition via large-scale weak supervision*, 2022.

[152] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, "Diet: Lightweight language understanding for dialogue systems," *arXiv preprint arXiv:2004.09936*, 2020.

[153] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," in *European Conference on Computer Vision (ECCV)*, 2022.

[154] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International Conference on Machine Learning (ICML)*, vol. 139, 2021, pp. 5583–5594.

[155] S. Antol, A. Agrawal, J. Lu, *et al.*, "Vqa: Visual question answering," in *IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2425–2433.

[156] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta, "R3m: A universal visual representation for robot manipulation," in *Conference on Robot Learning (CoRL)*, 2022.

[157] P. Schramowski, C. Turan, N. Andersen, C. A. Rothkopf, and K. Kersting, "Large pre-trained language models contain human-like biases of what is right and wrong to do," *Nature Machine Intelligence*, vol. 4, no. 3, pp. 258–268, 2022.

[158] P. Chang, S. Liu, T. Ji, N. Chakraborty, K. Hong, and K. R. Driggs-Campbell, "A data-efficient visual-audio representation with intuitive fine-tuning for voice-controlled robots," in *Conference on Robot Learning (CoRL)*, 2023.

[159] C. Wang, L. Fan, J. Sun, *et al.*, "Mimicplay: Long-horizon imitation learning by watching human play," in *Conference on Robot Learning (CoRL)*, 2022.