UNIVERSITÄT BONN

Social-aware Robot Navigation based on Deep Reinforcement Learning

DISSERTATION

zur Erlangung des Doktorgrades (*Dr. rer. nat.*)

der Mathematisch-Naturwissenschaftlichen Fakultät

der Rheinischen Friedrich–Wilhelms–Universität, Bonn

vorgelegt von

YANYING ZHOU

aus Liaoning, China

Bonn, 2024

Angefertigt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Rheinischen Friedrich–Wilhelms–Universität Bonn

1. Gutachter / 1st Advisor: Prof. Dr. Jochen Garcke
2. Gutachter / 2nd Advisor: Prof. Dr. Ira Neitzel
Tag der Promotion / Day of Promotion: April 15th 2024
Erscheinungsjahr / Year of Publication: 2024

Abstract

by Yanying Zhou

for the degree of

Doctor rerum naturalium

With the increase of applications involving autonomous mobile robots, there is a growing need for them to navigate safely and effectively in environments shared with humans. In recent years, social-aware robot navigation has received a lot of attention as it enables robots to understand and follow human social norms, thereby avoiding potential conflicts and dangers. Although several methods have been proposed for environment modeling and motion planning to guide robot behavior, these methods often tend to ignore social rules and focus mainly on motion control and path planning. To address this issue and enhance robot navigation efficiency and safety in dense crowds, this thesis introduces social-aware robot navigation algorithms based on Deep Reinforcement Learning (DRL) to capture crowd interactions and group motion characteristics.

To this end, we first propose a novel Foresighted Social-Aware Reinforcement Learning (FSRL) framework aimed at enabling mobile robots to achieve collision-free navigation. Due to the sparsity of traditional reward signals, it is difficult for robots to learn effective strategies from complex environments. Such sparse rewards may lead to extremely inefficient learning for the robot, requiring a significant amount of time and attempts to learn useful strategies. To address this problem, we employ reward shaping techniques to provide additional reward signals to guide the robot's learning process. Compared to previous learning-based methods, our approach, supported by rich reward mechanisms, is foresighted. It considers not only the current human-robot interactions to avoid immediate collisions but also estimates upcoming social interactions to maintain an appropriate distance. Additionally, our method introduces efficiency constraints, significantly reducing navigation time. Comparative experiments are conducted to validate the effectiveness and efficiency of our proposed method in more realistic and challenging simulated environments.

To further enhance the generalization performance of the navigation method, we then present a novel deep graph learning architecture based on the attention mechanism. While previous works have demonstrated the effectiveness of using reinforcement learning frameworks to train efficient navigation strategies, their performance deteriorates when crowd configurations change (i.e., become larger or more complex). Therefore, it is crucial to fully understand the complex, dynamic interactions of the crowd in order to bring proactive and foresighted behaviors to robot navigation. Our method utilizes spatial-temporal graphs to augment robot navigation, using spatial graphs to capture current spatial interactions, and integrating with RNNs, temporal graphs employ past trajectory information to infer the future intentions of each agent. The reasoning capability of spatial-temporal graphs enables robots to better understand and interpret the relationships between agents over time and space, thereby making wiser decisions. Compared to previous state-of-the-art methods, our approach exhibits exceptional robustness in safety, efficiency, and generalization across var-

ious challenging scenarios.

Lastly, this paper considers the challenge posed by the limited perception range of sensors for robot navigation, which leads to incomplete and uncertain information about the observed environment. To achieve collision avoidance in crowded and partially observable environments, we propose a novel deep reinforcement learning architecture. The architecture combines spatial graphs and attention reasoning to enhance the modeling of relationships among moving robots, static obstacles, and surrounding individuals. In this way, our method significantly outperforms state-of-the-art methods in crowded scenarios with limited robot sensor range, particularly in reducing collisions and improving navigation efficiency. Additionally, the adoption of parallel double deep Q-learning significantly reduces training time.

In conclusion, by employing advanced deep learning techniques and effective model design, this thesis has made significant advancements in robot navigation. It provides valuable insights for real-time navigation and interaction of robots in complex crowd environments.

Keywords: deep reinforcement learning, social-aware robot navigation, crowd interaction, graph neural network

Acknowledgements

This work would not have been possible without the love, support, and companionship of my friends, family, and professors over these four years.

During my Ph.D. journey, I have often said how fortunate I am. I was lucky to meet my professor, Prof. Dr. Jochen Garcke, fortunate to work with many outstanding individuals, blessed to pursue my Ph.D. with my loved ones in the same city and even the same university, and lucky to have met so many kind and wonderful people in my life.

First and foremost, I would like to express my deepest gratitude to my professor, Prof. Dr. Jochen Garcke, for his continuous support and guidance throughout my Ph.D. studies. Over these past years, he has provided me with opportunities to explore and develop my ideas and offered guidance and motivation when I needed it most. I am thankful for his assistance and crucial support at key moments. I will always cherish the good times after completing work, as well as the countless valuable lessons in scientific writing and presentations. I am grateful to the entire Numerical Simulation team; I am thrilled to have had the opportunity to come to Bonn and work with so many distinguished individuals. I would like to thank Jannik Schurg for familiarizing me with the environment on my first visit to the office, allowing me to quickly integrate into the team. I would like to express my gratitude to Karen Petersen and Stephanie Zacharias. It's always been delightful to have conversations with them. I would also like to thank Paolo, who has solved many difficult problems for me and provided a lot of help. Working in the same office with him has been a pleasure. I'm also thankful to other colleagues, such as Arno Feiden, Sara Hahner, and David Ebert, for their insightful discussions and enjoyable conversations.

Additionally, I am grateful to my friends, who have given me tremendous support and encouragement in both life and work. I would like to express my heartfelt thanks to my best friend, He Li. When I reminisce about the days of laughter and tears we shared during our Ph.D. journey, my heart is filled with nostalgia. I am also grateful for the close companionship of my colleague Liu Le, who is like a sister to me. My gratitude also extends to Xieyuanli Chen, an amazing friend from whom I have learned a lot and who has been immensely helpful. Special thanks to my best friend in Japan pursuing a medical Ph.D., Yaojia Ma, who has always been there to offer advice and comfort whenever I fell ill, I appreciate your patient companionship throughout. Additionally, I would like to thank Jinhui Yi, Yanan Luo, and Jingjing Li for their support and encouragement during my Ph.D. studies.

Lastly, I wish to mention my family. I am grateful for my parents' unwavering support and patience. Although far from home, they have always kept me in their thoughts, providing the greatest emotional support and being my strongest pillar. Most importantly, I want to thank my husband, Shijie Li. His kindness, patience, and love have been my driving force. He is my world, making everything I do meaningful.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

An alphabetically sorted list of abbreviations used in the thesis:

APF	Artificial Potential Field
CE	Cross Entropy
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DDQN	Double Deep Q Network
DL	Deep Learning
DNN	Deep Neural Network
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
FCN	Fully Connected Network
FRP	Freezing Robot Problem
GAN	Generative Adversarial Network
GAT	Graph Attention Network
GCN	Graph Convolutional Network
GD	Gradient Descent
GNN	Graph Neural Network
GRU	Gated Recurrent Unit
HRVO	Hybrid Reciprocal Velocity Obstacle
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MLP	Multi-Layer perceptron
MSE	Mean Square Error
NN	Neural Network
ORCA	Optimal Reciprocal Collision Avoidance
ReLU	Rectified Linear Unit
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RVO	Reciprocal Velocity Obstacle
SFM	Social Force Model
SGD	Stochastic Gradient Descent
TD	Temporal Difference
VFH	Vector Field Histogram
VO	Velocity Obstacle

List of Publications

The thesis is based on the following publications:

- **Foresight Social-aware Reinforcement Learning for Robot Navigation**
  Yanying Zhou, Shijie Li, Jochen Garcke
  Chinese Control and Decision Conference (CCDC), 2023.

- **Learning Crowd Behaviors in Navigation with Attention-based Spatial-Temporal Graphs**
  Yanying Zhou and Jochen Garcke
  IEEE International Conference on Robotics and Automation (ICRA), 2024, accepted.

- **Enhanced Spatial Attention Graph for Motion Planning in Crowded, Partially Observable Environments**
  Weixian Shi, Yanying Zhou, Xiangyu Zeng, Shijie Li, Maren Bennewitz
  IEEE International Conference on Robotics and Automation (ICRA), 2021.

The following publications are not covered by this thesis:

- **Spatial-Temporal Consistency Network for Low-Latency Trajectory Forecasting**
  ShiJie Li, Yanying Zhou, Jinhui Yi, Juergen Gall
  International Conference on Computer Vision (ICCV), 2021.

- **Aggregation-Interaction Transformer for Efficient Trajectory Forecasting**
  Shijie Li, Yanying Zhou, and Juergen Gall
  Submitted

CHAPTER 1

Introduction

Motivation

Upon observing our surroundings, it becomes evident that robots have become an integral part of our households and everyday lives, as illustrated in Figure 1.1. Home assistant devices such as automatic vacuum cleaners and other smart home gadgets have become common companions, aiding individuals in daily cleaning and maintenance tasks. Healthcare robots are engineered to assist the elderly and individuals with disabilities, enhancing their quality of life by providing physical assistance or monitoring health conditions. In modern society, the role of robots is dramatically evolving and expanding, no longer confined to repetitive tasks in industrial settings. Moreover, industrial robots continue to play a pivotal role in the manufacturing sector, performing a range of tasks from simple assembly line work to complex welding and painting assignments, thus boosting efficiency and reducing hazards.

In recent years, we have witnessed the rapid advancement of autonomous vehicle technology, marking a significant breakthrough in robotics for transportation and road safety. These autonomous

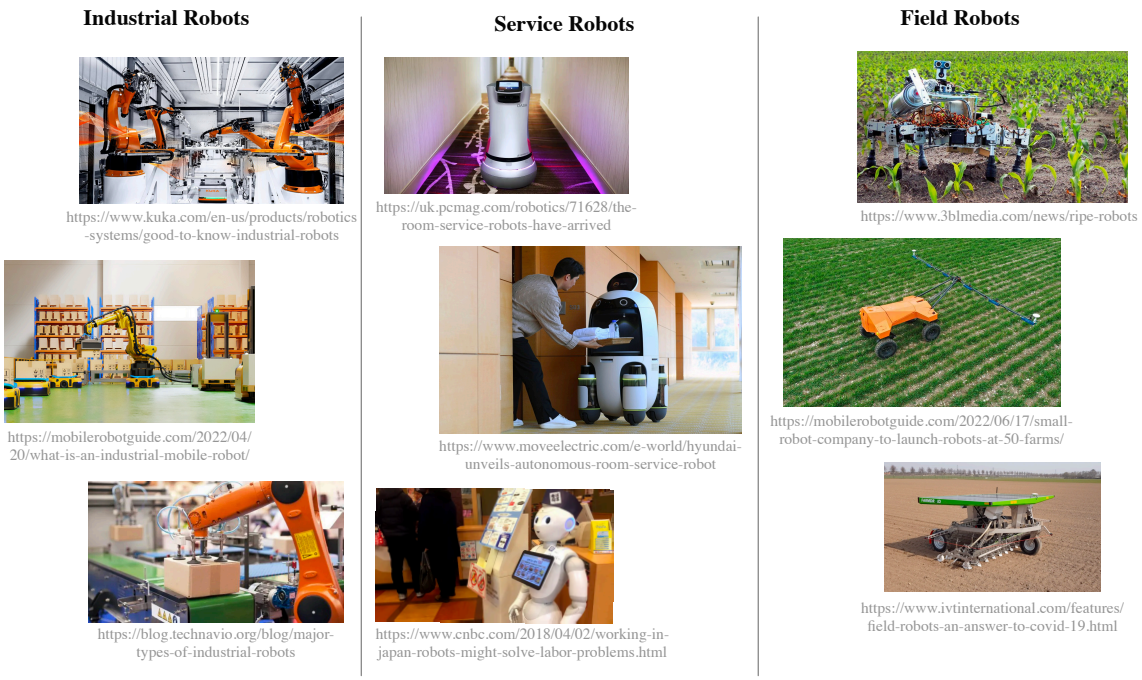


Figure 1.1: Classification for the autonomous mobile robots.

systems, leveraging sophisticated sensors and AI algorithms, have not only addressed issues of traffic congestion and safety but have also pioneered a brand-new self-driving experience. Concurrently, we have observed robots becoming increasingly autonomous, with their interactions with humans growing more intimate and complex. They have evolved not just into tools for performing tasks, but also into entities capable of sensing environment changes and, to a certain extent, interpreting human needs and emotions. It's not hard to envision that interactions between robots and humans will become more familiar, becoming an indispensable part of our lives. This transition has not only propelled the advancement of robotics but also opened a new dimension of interaction and collaboration for humanity.

Meanwhile, Deep Reinforcement Learning (DRL) has emerged as a hot topic in the machine learning domain in recent years, amalgamating the representation learning capabilities of deep learning with the decision-making mechanisms of reinforcement learning. Following the tremendous success of deep learning in tasks such as image recognition and natural language processing, researchers began exploring its potential in more complex decision-making tasks, where reinforcement learning became a natural choice. The advent of DRL has provided an effective methodology for handling high-dimensional, continuous state and action space problems, overcoming the limitations traditionally associated with reinforcement learning in these realms.

Against the backdrop of continuous advancements in robotic technology, the issues faced by autonomous mobile robots navigating in environments shared with humans have become particularly significant. Navigation refers to the ability of mobile robots to perceive their environment and their own state through sensors, achieving autonomous movement toward a target amidst obstacles. Whether on bustling streets, in shopping centers, or on public transportation, robots need to be able to move safely, smoothly, and in accordance with social conventions. This not only entails technical challenges, such as how to identify, predict, and avoid obstacles but also encompasses understanding and adapting to human behavioral patterns and expectations.

Traditional robotic navigation systems often rely on fixed rules and simple sensor feedback to avoid obstacles or reach specific targets. However, when robots enter human social spaces, these rules become inapplicable. For instance, robots might "freeze" in crowded places or appear overly rigid and unnatural when interacting with humans. Such mistakes and failures can not only lead to physical collisions and accidents but may also evoke distrust and fear towards robots among people. Moreover, the challenges of social navigation go far beyond physical obstacle avoidance. People's behaviors in public places are complex, dynamic, and variable. They might suddenly stop, turn, or change speed. Furthermore, their behaviors are influenced by culture, customs, and the current context. For example, in a culture where people are accustomed to yielding, robots might need to take actions more assertively; whereas in another culture, robots might need to be more passive. Deep Reinforcement Learning (DRL), with its ability to automatically extract environmental features, offers a new direction for tackling navigation issues. Through interaction with the environment, DRL can autonomously learn effective navigation strategies without the need for explicit programming. More importantly, DRL can help robots learn these complex social interaction rules and make appropriate decisions in practical applications. For instance, robots can learn how to adjust their speed and direction in crowded scenarios to avoid collisions with humans while not appearing too abrupt or unnatural.

Addressing the aforementioned issues, designing and implementing an advanced robotic navigation algorithm to enable robots to reach their destinations safely and efficiently in crowds and

complex social environments is crucial. To achieve this goal, this first needs to delve into the interaction patterns among individuals in groups. Only with a genuine understanding of the dynamics and interactions within crowds can robots truly blend in, avoiding misunderstandings of human behavior and possible collisions. Additionally, to ensure the effectiveness of robotic navigation in social environments, it is essential to capture and simulate the core features of group movements. The patterns and laws of group behavior are the cornerstone of navigation decisions.

In this thesis, by employing Deep Reinforcement Learning (DRL) methods combined with Graph Learning Networks and attention mechanisms, I have significantly enhanced the capability of robots to navigate with social awareness in crowds, especially in partially observable environments, effectively addressing navigation performance issues stemming from sensor perception limitations through optimizing foresighted predictions of group movements and processing of local spatial information.

Problem Statement

In recent years, Deep Reinforcement Learning (DRL) has emerged in the field of machine learning, allowing new research directions and possible solutions for numerous complex problems. As a fusion of deep learning and reinforcement learning, DRL allows models to automatically extract and learn meaningful features from raw data, and combine trial-and-error approaches in decision-making to find the optimal strategy in a given task.

The objective of Socially Aware Robot Navigation based on DRL is to amalgamate the robust representational learning capabilities of deep learning with the decision optimization characteristics of reinforcement learning. This fusion enables robots to autonomously navigate in crowds or social settings, while understanding and anticipating the behaviors and intentions of the people around them. This ensures safe, efficient, and socially pleasant interactions and maneuvers.

As illustrated in Figure 1.2, a state s_t depicts the robot and its surrounding environment, an action a_t represents the possible behaviors of the robot, and a reward function r_t reflects the social benefits and costs of the robot interacting with the crowd. By training a deep neural network, I aim to obtain a policy π^* that guides the robot's actions in a way that maximizes an expected cumulative reward, thereby ensuring efficient and socially-compliant navigation through crowds. I will discuss the specific details of the socially-aware robot navigation method based on deep reinforcement learning in Chapter 4.

Challenges

Social-aware robot navigation based on deep reinforcement learning faces many challenges. In this section, I discuss some of these challenges and open questions that need to be addressed for building models that can effectively navigate through human crowds while exhibiting a social understanding.

Learning from Interaction

When required to navigate through crowds, robots need to understand human social signals and behaviors. Through interaction, robots can collect data, extract useful information, and then update their knowledge or strategies based on this information. The purpose of learning is to enhance the

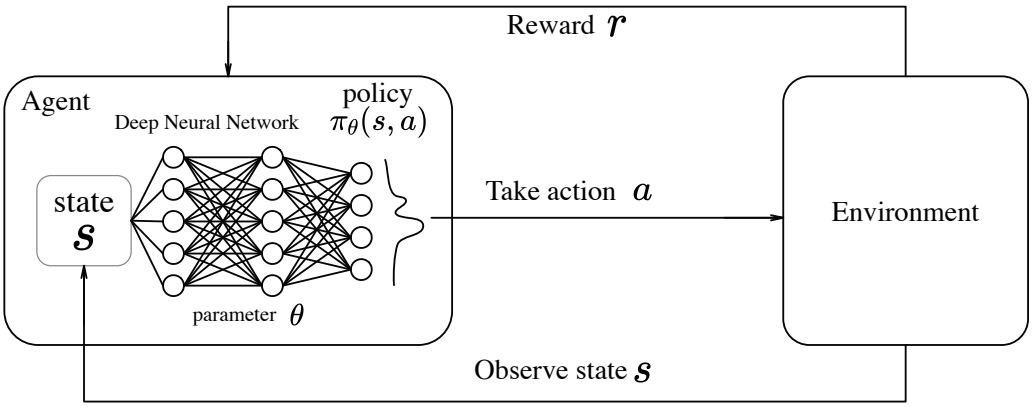


Figure 1.2: Deep reinforcement learning structure.

robot's performance, enabling it to better accomplish specific tasks, such as navigating safely and efficiently through crowds. If the robot cannot accurately understand and adapt to human social norms, it may lead to low navigation efficiency or even collide accidents. Therefore, utilizing technologies of deep learning and reinforcement learning to assist robots in learning and understanding human social behavior from a vast amount of interaction data is crucial.

Human Behavior Understanding

Human behavior is diverse and complex, potentially influenced by various factors such as individual emotions, experiences, and cultural backgrounds. In crowds, the interactions between individuals and collective behavioral patterns escalate the difficulty of understanding human behavior. If robots cannot accurately comprehend human behavior and intentions, they might fail to make optimal navigation decisions, not only compromising the safety and efficiency of navigation but also affecting people's trust in robots. Therefore, it is crucial to integrate deep learning models with reinforcement learning to process and analyze unstructured data, capture the features and patterns of human behavior, and enable robots to continuously learn and enhance their understanding of human behavior through interactions with humans and the environment.

Sparse Reward

In many practical scenarios, a robot may only receive reward signals upon reaching a destination or completing a specific task, while for most of the time, it receives no reward signals. This sparsity of reward signals makes it challenging for the robot to learn effective strategies from the environment. Such sparse rewards may result in extremely low learning efficiency for the robot, necessitating a substantial amount of time and trials to learn useful strategies. Therefore, employing reward shaping techniques to provide additional reward signals for guiding the robot's learning process is crucial.

Generalization to Different Environments

The real-world environment exhibits extremely high diversity and uncertainty, encompassing different spatial layouts, dynamic obstacles, and human behavioral patterns. If a robot navigation system cannot generalize well to new environments, its usability and reliability will be significantly reduced. Therefore, effectively capturing and understanding the structure and relational information in data is crucial to assist robots in better comprehending and adapting to new environments.

Partial Observation

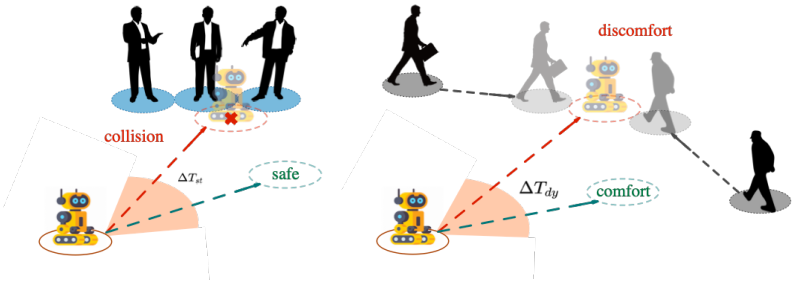
In complex social settings, robots, limited by the range of their sensors, typically can only observe local information. For instance, they might only be able to see the dynamics of a small area of the crowd around them, without access to the global environmental information. This local observability hinders the robots from making accurate decisions. Since robots cannot obtain complete environmental information, they may need more time and attempts to learn effective navigation strategies. More importantly, this may degrade the navigation performance of robots, making it difficult for them to respond to environmental changes in a timely manner, which might lead to collisions or going off route. Therefore, enhancing the understanding and capture of environmental dynamics is crucial.

Contributions

In this thesis, I propose approaches to address the challenges discussed above. First, I propose a model to enhance robot navigation by assessing real-time and estimated future interactions. Then, I propose a method to capture and aggregate spatial-temporal interactions to enhance the robot's reasoning capabilities and robustness in navigation across diverse challenging scenarios. Finally, I propose a method to model the interactions between the robot and humans by combining spatial graphs and attention reasoning in partially observed environments.

Reward Reshaping for Sparse Reward

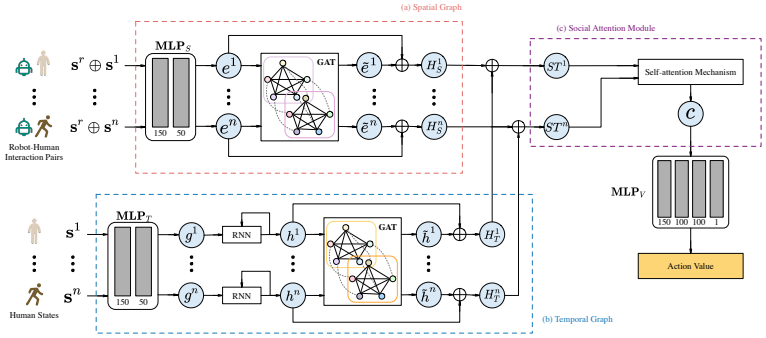
The first contribution is a reward reshaping structure for improving sparse functions. In this structure, I enrich the reward mechanism, granting the framework foresight, which allows not only for consideration of current human-machine interactions to prevent immediate collisions but also the prediction of future social interactions to maintain a proper distance. This is particularly crucial for navigation within dynamic crowds as the robot needs to predict the movements of individuals to avoid collisions. To capture the varying kinematics of humans, different constraints are applied to individuals based on their states. To enhance navigation efficiency, I introduce an efficiency parameter that can significantly shorten navigation time, which is especially critical for navigating in environments with



(a) A scenario with standing humans (b) A scenario with dynamic humans

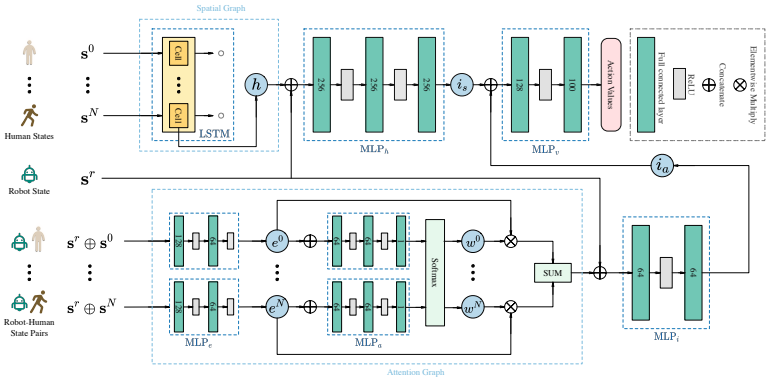
both dynamic and static crowds. By optimizing navigation paths and reducing collisions, the robot can reach its destination faster while maintaining social awareness. I demonstrate that the proposed method reduces navigation collisions and navigation time in complex simulated scenarios encompassing both static and dynamic obstacles, showcasing efficiency and effectiveness.

Graph Neural Network for Navigation Generalization



Although the reward shaping framework discussed above is particularly successful in enhancing the foresight abilities of robot navigation, the training strategies rely on the training environment. Given the dynamics and uncertainties within crowds, re-training the robot every time it enters a new environment would significantly increase time and costs. Recent approaches suggest that Graph Neural Networks (GNNs) can capture and understand structural and relational information in data, with knowledge being transferable and contributable across different parts within the graph, aiding robots in better understanding and adapting to new environments. As the second contribution, I propose employing Graph Neural Networks to overcome the limitations of previous methods. For this purpose, I suggest introducing Graph Attention Networks to capture spatial and temporal interactions within crowds. The attention weights can be adaptively adjusted based on the features of nodes within the graph, capturing multi-scale features and relationships. Simultaneously, to enhance predictive reasoning capabilities, I integrate Recurrent Neural Networks to capture past states and interactions, which helps the robot obtain temporal dependencies, thereby better understanding the current scenario and predicting future situations. The proposed method offers a comprehensive understanding of crowds and enhances reasoning abilities. I demonstrate that employing Graph Attention Neural Networks contributes to improved navigation performance in varying crowd configurations.

Graph-Attention Reasoning for Partial Observation



Considering further the limited perception range of sensors, robots can only observe partial and incomplete information locally. As the final contribution, I propose a framework to understand and capture the dynamics of the surrounding environment based on

partially observed information. I perform spatial and attention reasoning on the partially observable environmental information separately. Spatial reasoning is employed to understand the spatial relationships between the robot and other objects in the environment (such as people or other obstacles), encouraging prediction and interpretation of dynamic changes in its surrounding environment. Attention reasoning allows the robot to focus on relatively crucial objects in the partially observed environment. Subsequently, I adopt a parallel double DQN for network training, which accelerates the training process. I demonstrate that even in highly crowded environments, under the application of a restricted sensor perception range, the proposed method possesses a higher success rate and a lower collision rate, significantly reducing the navigation time.

Thesis Structure

The rest of this thesis is organized as follows:

Chapter 2 provides an overview of the related work on social-aware robot navigation and deep reinforcement learning techniques.

Chapter 3 gives a brief overview of the concepts that are used throughout the thesis. It starts with a quick introduction to reinforcement learning. Then, it describes the deep learning methods, including convolutional, recurrent and graph neural networks. Finally, it introduces deep reinforcement learning techniques, including value-based learning and policy-based learning methods. provides a formal definition of the evaluation metrics that are used to evaluate the performance of the proposed approaches.

Chapter 4 provides an overview of the fundamental aspects of research on DRL-based social-aware robot navigation. First, it introduces the basic framework of DRL-based social-aware robot navigation. Then, it formulates the problem and discusses its essential components. Finally, it briefly described the current simulation experiments' environmental settings, experimental setup, and performance metrics for evaluation.

Chapter 5 proposes a reward-reshaping-based method, called Foresight Socially-aware Reinforcement Learning (FSRL), which is very efficient and accurate. This chapter is based on *Zhou et al.* (2023). Shijie Li contributed to the writing. Jochen Garcke supervised the project and he contributed with discussion and writing.

Chapter 6 proposes a novel deep graph learning framework, called attention-based spatial-temporal graph learning framework (ASTG). It combines graph attention networks and recurrent neural networks to capture and integrate high-order spatial and temporal interactions between entities in crowds, facilitating socially-aware robot navigation in complex environments. Jochen Garcke supervised the project and he contributed with discussion and writing.

Chapter 7 proposes a deep-learning-based robotic navigation method, called Enhanced Spatial Attention (ESA). It integrates LSTM to encode spatial information and a social attention mechanism to analyze and learn from the motion patterns of surrounding individuals, aiming to

predict their future positions and facilitate collision-free navigation in crowded environments. This work is based on *Shi et al.* (2022). I presented the main idea of the paper and developed the whole model architecture. Weixian Shi performed the experiments with different reinforcement learning methods and contributed to the implementation of the spatial graph with social LSTM. Shijie Li contributed to participate in the discussion.

Finally, conclusions are given in Chapter 8 along with suggested directions for future work.

CHAPTER 2

Related Work

In this chapter, I discuss the related work for this thesis. First, I provide an overview of robot navigation. Second, I discuss approaches for social-aware robot navigation. Finally, I review the development of deep reinforcement learning methods.

Contents

2.1 Traditional Robot Navigation

Robot technology has undergone a significant development from early attempts to create humanoid robots to modern intelligent robots. Initially, robots were dependent on human assistance to carry out simple tasks. However, with the rapid advancement of technology, modern robots have evolved to possess increasingly sophisticated structures and powerful processing capabilities, featuring autonomous navigation, speech recognition, and visual perception. For example, Boston Dynamics's robots (*BostonDynamics*), including "Atlas", "Spot" and "Handle", can move and manipulate objects in various complex environments. Besides, *Laboratories* has developed humanoid robots, such as "Geminoid HI" and "Erica", which have realistic appearances and limb movements that allow them to explore interaction and communication between humans and robots. Without the need for human control, intelligent robots can improve their performance and adaptability by learning and optimizing algorithms, thereby increasing efficiency and productivity. Therefore, intelligent robots are increasingly used in various industries, including search and rescue, transportation, medical surgery, and other fields.

Navigation is a basic skill for autonomous intelligent robots. The development of robot navigation technology enables robots to independently identify and avoid obstacles, plan suitable routes, and reach their destinations in unmanned environments. For example, in airports, intelligent robots guide pedestrians to specific locations such as security checkpoints and assist them in carrying luggage; in logistics and warehousing such as JD in China and Amazon in the United States, parcel delivery intelligent robots can complete package delivery tasks through autonomous navigation; in rescue scenarios, rescue robots can autonomously search for and rescue trapped people at disaster sites. In the last decade, intelligent robot navigation has been one of the hot research topics, which is key for achieving robot autonomous movement and completing tasks. Therefore, I provide a brief overview of the advances in robot navigation technology research.

Traditional navigation systems were mainly based on sensory perceptions and computer vision technology (*Mur-Artal et al.*, 2015; *Zhang and Singh*, 2014; *Engel et al.*, 2014; *Levinson and Thrun*, 2010). These robots typically used cameras and laser sensors (LiDAR), such as LSD-SLAM (*Engel et al.*, 2014), ORB-SLAM (*Mur-Artal et al.*, 2015) and LOAM (*Zhang and Singh*, 2014) to detect their surrounding environment and used algorithms to determine their position and direction. LSD-SLAM and ORB-SLAM are notable instances of visual SLAM (Simultaneous Localization and Mapping) systems in the scope of camera-based sensing. Upon establishing an understanding of their environment, these navigation systems could then use pre-built maps or real-time created maps to plan their actions and navigate to the specified target. Additionally, various algorithms were deployed for the dual purposes of map construction and path planning. Map construction primarily adopts two approaches: grid maps (*Thrun and Bücken*, 1996; *Batalin et al.*, 2004) and probabilistic maps (*Levinson and Thrun*, 2010; *Wurm et al.*, 2010). Grid maps divide the environment into a grid of cells, each representing whether the region is occupied, free, or unknown. Probabilistic maps, conversely, are commonly utilized in Simultaneous Localization and Mapping (SLAM) techniques, where they estimate the likelihood of occupancy for each cell, thereby offering a more nuanced depiction of the environment.

Although traditional approaches for motion planning and control in mobile robot navigation have been successful, reliable navigation systems require extensive engineering effort, such as manually adjusting parameters when modeling the environment to meet specific scene requirements (*Xiao et al.*, 2022). Recent research indicates that by employing machine learning techniques, particularly neural networks and deep learning, the engineering effort required for robot navigation in complex environments can be alleviated. Neural networks (*Rosenblatt*, 1958) have the capacity to automatically learn and infer effective navigation strategies from a vast amount of robot data. This ability for automatic learning and inference reduces the need for manual design and adjustment of navigation algorithms, thereby lightening the engineering workload. *Thrun* (1995) is one of the pioneering efforts to utilize end-to-end machine learning for robot navigation, serving as an initial proof-of-concept for completely replacing the traditional architecture "sense-plan-act" with a single learned policy. *Pfeiffer et al.* (2017) presents a model capable of learning the complex mapping from raw 2D-laser range findings and a target position to the required steering commands for robot collision avoidance navigation. *Chen et al.* (2022) presents a neural network-based algorithm trained on human decisions for navigating a mobile robot in scenarios with various obstacle features, achieving close to 90% accuracy in replicating human decision-making process for navigation

In addition to the evolution of navigation algorithms, the focus of robot navigation is continuously changing and evolving. As robots become more prevalent and share physical space with humans, it is crucial to establish rules for social order. Therefore, while focusing on issues such as motion control and path planning, robot navigation technology has also started to pay attention to the interaction and integration of robots and human society. This includes considering factors such as human comfort (*Sisbot et al.*, 2010), naturalness (*Shi et al.*, 2011), and sociality (*Pacchierotti et al.*, 2007) during the navigation process, namely *social awareness* (*Rios-Martinez et al.*, 2015). It refers to a robot's ability to recognize and understand the comfort and social rules of surrounding humans (*Ge*, 2007) and adjust its behavior and performance accordingly (*Lindner and Eschenbach*, 2011). Social-aware robot navigation preserves a comfortable interaction with humans, resulting in behavior predictable, adaptable, and easily understood by humans (*Kuderer et al.*, 2012).

2.2 Social-aware Robot Navigation

The work of C.L. Breazeal pioneered the concept of social robots and this concept has been extended to other areas, which aim to social learning and interaction (*Breazeal*, 2004). The American Association for Artificial Intelligence (AAAI) has proposed a series of robotics challenges that have led to several large-scale projects for robot navigation in human environments, such as *Maxwell* (2007), *Michaud et al.* (2007). These studies attempted to consider situations where robots encounter humans and navigate around them, but largely still modeled humans as obstacles or targets and had only basic concern for social interactions (*Thomaz et al.*, 2016). However, as the field of social robotics evolved, so did the understanding of the importance of socially-aware navigation. Robots operating in human-centric environments need to adhere to social norms and exhibit behavior perceived as natural and intuitive. This includes understanding personal spaces, navigating in a predictable and non-intrusive manner, and being able to interpret and respond to human social cues (*Gao and Huang*, 2022). Existing works on social-aware robot navigation can be roughly grouped into three categories (*Guillén-Ruiz et al.*, 2023): (1) reaction-based methods; (2) prediction-based methods; (3) learning-based methods.

2.2.1 Reaction-based Navigation

In dynamic or unknown environments, navigating robots need to respond quickly to avoid dynamic obstacles while performing path planning. This kind of reactive method has become the most basic approach in social-aware robot navigation. In reaction-based methods, robots react to other mobile agents using one-step interaction strategies (*Chen et al.*, 2021). They may not consider prediction or learning and are mostly based on rules such as Velocity Obstacles (VOs) (*Fiorini and Shiller*, 1993; *Shiller et al.*, 2001; *Van Den Berg et al.*, 2011; *Van den Berg et al.*, 2008; *Snape et al.*, 2011), Social Force Models (SFMs) (*Helbing and Molnar*, 1995; *Reddy et al.*, 2021; *Ferrer et al.*, 2017; *Truong and Ngo*, 2017), Vector Field Histograms (VFHs) (*Borenstein et al.*, 1991; *Babinec et al.*, 2018), or Artificial Potential Fields (APFs) (*Khatib*, 1985; *Yao et al.*, 2020).

The concept of VO was originally proposed by *Fiorini and Shiller* (1993) and has been widely utilized in navigation, such as in the works of *Shiller et al.* (2001); *Van Den Berg et al.* (2011); *Van den Berg et al.* (2008); *Snape et al.* (2011), and others. Considering a moving agent A within the motion range of a robot R, VO means the set of all velocities of R that can collide with A. Thus, the robot can select a velocity vector different from the VO to ensure collision-free navigation. In the context of multiple agents operating independently but in the same environment, under the assumption that all agents use the same navigation technique, Reciprocal Velocity Obstacle (RVO) *Van den Berg et al.* (2008) avoids potential oscillations that may arise when approaching or crossing paths with one other. Optimal Reciprocal Collision Avoidance (ORCA) (*Van Den Berg et al.*, 2011) improved RVO by introducing a cost function and an optimization method to select an optimal velocity. The Hybrid Reciprocal Velocity Obstacle (HRVO) (*Snape et al.*, 2011) extends RVO by set priorities in the interaction between robots.

APFs were first introduced by *Khatib* (1985) based on virtual potential fields. It views the environment around the robot as an energy field, in which the potential energy between the robot and obstacles, attraction and repulsion forces with the robot generated by the goal point and obstacles, define the direction of the robot's motion. *Yao et al.* (2020) introduced reinforcement learning on

dealing with dynamic scenes to improve traditional APFs. VFHs (*Borenstein et al.*, 1991) construct a two-dimensional histogram to model environmental information. *Babinec et al.* (2018) propose the VFH* to handle dynamic obstacles.

In addition, SFMs (*Helbing and Molnar*, 1995) is a physics-based method that infers the direction and speed of mobile agents' movement for collision-avoidance behavior by simulating the interaction forces between individuals, such as attraction and forces. *Ferrer et al.* (2017) extends SFM to present a robot companion and uses interactive learning to adjust the parameters of the model. By taking advantage of both the extended SFM and the HRVO, *Truong and Ngo* (2017) takes the socio-spatio-temporal into account and proposes Proactive SFM (PSFM) to achieve efficient and proactive collision-avoidance navigation. It considers not only the human states relative to the robot (position, motion, and hand pose) but also the social interaction information of human obstacles and human group interactions.

However, the above methods heavily rely on the accuracy of deterministic motion models and can only capture simple interactions, making it difficult to extend to complex scenarios. Moreover, because robots only take actions for the next step based on the one-step rule and the current state, the planning path is shortsighted and unnatural. This may lead to a failure to adhere to social norms, especially in complex or crowded environments.

2.2.2 Trajectory-based Navigation

Unlike reaction-based methods, rather than using pre-defined rules, trajectory-based methods first attempt to forecast other humans' trajectories and behaviors from large-scale datasets (*Prediction Motion Model*) and then plan a proper path for the robot accordingly (*Path Planning*). By anticipating human actions, these methods aim to achieve smoother interactions and more socially compliant navigation.

Approaches such as the Kalman filter can predict the positions of moving agents around the robot by accounting for various uncertainties and noise in the measurements. However, when there are multiple moving agents, handling uncertainty becomes very challenging, possibly leading to an uncertainty explosion (*Trautman et al.*, 2015) which might prevent the robot from safely navigating to its destination. Thus, various human motion models have been proposed to try to control the growth of uncertainty. Mixing Dirichlet Process (DP) (*Teh et al.*, 2010) and Gaussian Process (GP) (*Wang et al.*, 2005) were used to simulate target motion patterns in *Joseph et al.* (2011). *Aoude et al.* (2013) combined Rapidly-exploring Random Trees (RRT) (*LaValle et al.*, 2001) with GP (RR-GP) to improve Gaussian predictions and identify probabilistically feasible paths.

In scenarios where a robot shares the environment with multiple moving agents, it is crucial for the robot to avoid collisions. Additionally, the robot should maintain a safe distance to prevent interference with the agents' activities, taking into account their interactions within the model. Therefore, the robot must be able to predict how each agent's behavior evolves over time, understanding and adapting to human social norms. Common methods for predicting trajectories include Kalman filters (*Kalman*, 1960; *Choset et al.*, 2005; *Welch et al.*, 1995) or Particle filters (*Doucet et al.*, 2001; *Liu and Chen*, 1998; *Pitt and Shephard*, 1999), but goal-based policy methods might provide more suitable and effective solutions for social navigation, achieving more natural and socially compliant navigation behavior. They assume that the behaviors of agents are captured in previously observed trajectories, determining which trajectory group the current trajectory belongs to by simulating hu-

man goal-oriented trajectories. *Bennewitz et al.* (2002) applied the expectation maximization (EM) algorithm (*Moon*, 1996) to cluster similar behaviors into single motion patterns to learn motion models. *Ziebart et al.* (2009) introduced maximum entropy inverse optimal control to simulate human future trajectories. These predictive models incorporated dimensions of uncertainty to predict the positions of moving agents.

Furthermore, considering interactions with static and dynamic obstacles, data-driven methods have also been proposed to capture agent motion. Inspired by the success of Long Short-Term Memory networks (LSTM) (*Hochreiter and Schmidhuber*, 1997), a data-driven architecture called Social-LSTM is proposed by *Alahi et al.* (2016) to predict future human trajectories. It introduces a social pooling layer to capture the dependencies between multiple related sequences. Considering implicit cooperations between different agents in collision-free navigation, *Vemula et al.* (2018) proposed Social Attention to capture the relative importance of each mobile agent while navigating, without emphasizing the distance between them.

After establishing the human motion model, trajectory-based methods define a planner that can find the optimal navigation policy. Based on consolidated and discretized human model, Co-MDP (*Smith et al.*, 2021) is proposed to generate obstacle avoidance behavior, but also to be robust and natural. *Svenstrup et al.* (2010) proposes a path planner based on RRT to avoid unforeseen pedestrians. Moreover, in order to respect people's personal space while avoiding collisions, *Rios-Martinez et al.* (2011) designed the Risk-RRT method, which uses Gaussian Process learning to estimate the O-space of people.

In dense crowds, the computational cost and time required for explicitly calculating the evolution of joint paths are incredibly high, especially if the state space expands rapidly as the crowd group becomes large. Additionally, due to the adoption of overly conservative strategies by robots, there is limited navigation space available for planning. Consequently, the robots are more prone to experiencing the "freezing robot problem" (FRP) (*Trautman and Krause*, 2010), where the planner predicts that all feasible routes are deemed unsafe, resulting in the robot being stuck and unable to move.

2.2.3 Learning-based Robot Navigation

In recent years, there has been a growing trend in combining machine learning or deep learning algorithms with Reinforcement Learning (RL), using these learning-based methods to learn how to achieve improved navigation performance in complex environments. Deep Reinforcement Learning (DRL) is most classically used to learn excellent navigation strategies, which introduce deep neural networks to solve reinforcement learning problems. DRL methods explore efficient interaction rules through pre-training a value function, which enables the robot to determine the optimal action based on the currently observed states. As a result, the robot can avoid collisions while trajectory planning.

In the last decade, a number of DRL-based collision avoidance training methods have been proposed, some of which take social awareness into account. CADRL (Collision Avoidance with DRL) (*Chen et al.*, 2017b) was the earliest work replacing handcraft techniques with DRL and achieving success in multi-agent scenarios. It is based on the reciprocal assumption, which means that when two or more agents attempt to avoid collisions, they assume that others are also taking collision avoidance actions. Considering humans and social norms, this work was extended in *Chen et al.* (2017a) (Social Aware-CADRL, SA-CADRL). Specifically, SA-CADRL learns socially compliant behaviors, such as passing on the right, by incorporating a reward function that depends on the dynamics

of the scenarios. However, an over-reliance on reciprocal assumption can lead to overly conservative behavior, and their effectiveness is limited when dealing with complex decentralized scenarios. *Everett et al.* (2018) (LSTM-RL) improved it by using Long-Short Term Memory (LSTM) cells to encode other agents and model the collective impact of the crowd. LSTM-RL processes the states of each neighbor sequentially in reverse order of the distance to the robot. To improve the comfort of people sharing the crowd with a robot, *Hu et al.* (2022) introduced the social stress index in a deep reinforcement learning framework, which extracts local features and calculates social-attention scores through a multilayer perceptron. *Gil et al.* (2021) introduced a method that combines robot velocities learned from an RL model (AutoRL (*Francis et al.*, 2020)) with robot velocities calculated using an SFM to determine robot actions.

Although these methods have successfully addressed multi-agent navigation, they fall short in accounting for the complex interactions among humans (*Samsani and Muhammad*, 2021). Robots primarily rely on their perception and reasoning capabilities to interpret human behavior and predict future human actions, yet lack feedback from the crowd. These methods often treat robot navigation in crowded environments as a one-way human-robot interaction problem, which could pose challenges as the size of the crowd increases. To overcome these limitations, *Chen et al.* (2019) proposed a Social Attention Reinforcement Learning (SARL) approach, explicitly modeling the interactions between the robot and the crowd. By considering both human-robot and human-human interactions, self-attention is utilized to discover the collective influence of the crowd. Furthermore, SOADRL (*Liu et al.*, 2020) extended SARL by separately handling information associated with static and dynamic objects, addressing the challenge of navigating with sensors providing only a limited field of view in crowds. Additionally, *Chen et al.* (2020b) suggested enabling the system to identify the most critical individuals for navigation within a crowd. *Gao et al.* (2019) considers partial observability issues (e.g., due to sensor limitations, occlusions, or perceptual uncertainties). To achieve this, it leverages Recurrent Neural Networks (RNNs), specifically Gated Recurrent Units (GRUs), to infer unobservable states. For the real-time response to human behavior, Samsani and Muhammad (*Samsani and Muhammad*, 2021) proposed modeling dangerous zones for the robot. These zones are defined by considering real-time human behavior and then encoding all possible actions that people could take at a given time. The robot is trained to avoid these dangerous zones, aiming for safe and reliable navigation.

To address the challenge of handling large-scale crowds, some methods are proposed to identify the most critical individuals for navigation within the crowd. They utilize graph representations to learn the optimal policy, encoding information about the crowd to predict human attention scores during the navigation tasks. *Chen et al.* (2020b) employs a method based on human gaze data to train a Graph Convolutional Network (GCN), accurately predicting human attention towards different agents within the crowd during navigation. They then integrated the learned attention into a graph-based reinforcement learning framework (Gaze-GCN based RL, G-GCNRL). Since the dynamic relations in a crowd produce non-Euclidean data, Graph Neural Networks (GNNs) can be used to extract efficient representations in crowd navigation. GNNs inherit the complementary strengths of graphs, which possess powerful representation capabilities, and neural networks, which have end-to-end learning power. In GNNs, problems are formulated by representing components as nodes and relationships as edges in a graph. These models typically propagate local information throughout the graph, explicitly capturing the relationships between nodes. This has been proven effective in dealing with a variety of structured tasks, especially where the structured nature of the problem

plays a significant role. There are two typical types of GNNs: Graph Attention Networks (GATs) *Zhou et al.* (2022) and Graph Convolutional Networks (GCNs) *Chen et al.* (2020a,b). *Chen et al.* (2020a) introduces two-layer GCNs to model a Relation Graph Learning (RGL) structure to learn the agents' interactions. In contrast to GCNs, which utilize fixed weights in convolution operations, GATs employ learnable attention mechanisms to dynamically weigh the importance of each neighbor during the information aggregation process. *Zhou et al.* (2022) improves two-layer GATs to extract efficient graph representation, modeling Social Graph-based Double Dueling Deep Q-Network (SG-D3QN). The social attention mechanism in GATs allows them to dynamically emphasize the crucial neighbors of each node, thereby granting these networks higher flexibility to handle graphs with various structures.

Additionally, some methods also employ supervised learning schemes to mimic expert demonstrations for enhancing learning efficiency, such as Imitation Learning (IL) (*Hussein et al.*, 2017; *Osa et al.*, 2018). IL is sample-efficient and can quickly find a navigation model based on the training data (*Pfeiffer et al.*, 2017). *Tai et al.* (2018); *Long et al.* (2017); *Liu et al.* (2018) developed navigation strategies that map various inputs (such as depth images, LiDAR measurements, and local maps) to control actions by directly mimicking expert demonstrations. *Tai et al.* (2018) proposes a real-time, socially compliant navigation method for mobile robots among pedestrians using raw depth inputs and Generative Adversarial Imitation Learning (GAIL). This approach improves safety and efficiency in real-world deployments, also offering a simulation plugin for modeling pedestrian behaviors. *Pfeiffer et al.* (2018) combined IL and Reinforcement Learning (RL) for single-goal-driven navigation in mapless scenarios. The Reinforced Imitation Learning (R-IL) utilizes expert demonstrations to pre-train navigation strategies and then applies Constrained Policy Optimization (CPO) (*Achiam et al.*, 2017) to incorporate constraints during the RL training phase. It demonstrated that this method could reduce training time to achieve performance levels comparable to standard RL. *Pfeiffer et al.* (2017, 2018) tested their solutions in static environments.

CHAPTER 3

Preliminaries

In this chapter, I briefly discuss essential concepts that are relevant to this thesis. I start with an introduction to reinforcement learning, markov decision process, policy and value function, temporal difference learning and Q learning. Then I introduce the knowledge related to deep learning, such as neural networks, recurrent neural networks, and graph neural networks. Finally, I present some deep reinforcement learning methods, which are classified into value-based methods and policy-based methods.

Contents

3.1 Reinforcement Learning (RL)

Reinforcement Learning (RL) (*Sutton and Barto*, 2018; *Wiering and Van Otterlo*, 2012), which is inspired by behaviorist psychology, interacts with the environment in a trial-and-error mechanism, learning from these experiences to optimize strategies.

In reinforcement learning, the decision-maker is defined as an agent that receives state information from the environment and performs specific actions based on the current state, thereby influencing the environment and receiving corresponding reward or punishment signals. The goal of the agent is to learn a behavioral policy by iteratively adjusting strategies through trial and error in the interaction with the environment, aiming to maximize the long-term cumulative reward. For example, in the context of autonomous driving, the agent represents the vehicle itself, while the environment encompasses the physical world beyond the vehicle, including roads, other vehicles, and pedestrians.

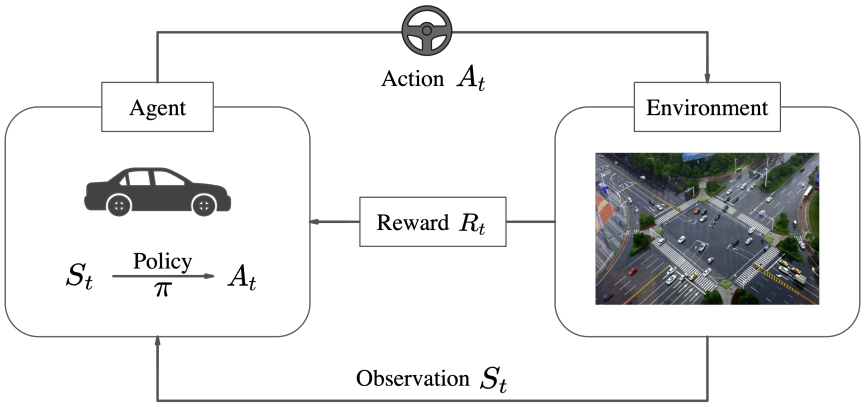


Figure 3.1: Reinforcement learning model for autonomous driving.

The reward serves as feedback from the environment to evaluate the quality of the agent's actions. The autonomous vehicle (agent) needs to select appropriate actions, such as deceleration, accelerating, and steering, based on the current state information like speed and position. The environment then provides corresponding punishments or rewards based on the vehicle's actions, guiding the vehicle to drive safely, smoothly, and efficiently toward its destination. More specifically, when the autonomous vehicle reaches the destination safely and smoothly, the agent receives positive rewards, indicating that the actions taken by the agent align with expectations and provide positive feedback. Conversely, if the agent encounters accidents or violates traffic rules, it receives punishments, which is negative feedback. Similarly, in the case of a Go-playing algorithm, the robot (agent) has to decide how to move to win the game based on the current game state (environment).

3.1.1 Markov Decision Process (MDP)

The sequential decision problem in the RL framework can be modeled as a ***Markov Decision Process (MDP)*** (*Sutton and Barto*, 2018; *Bellman*, 1966; *Howard*, 1960), which are defined by $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$ comprising by the key elements states $\mathcal{S}$, actions $\mathcal{A}$, transition probability $\mathcal{P}$, rewards $\mathcal{R}$ and discount factor γ. $\mathcal{S}$ represents the environment's state , $\mathcal{A}$ is the action taken by the agent. $\mathcal{P}(S_{t+1}|S_t, A_t)$ describes the probability that the agent transfer from one state to another after performing a particular action. $\mathcal{R}(S_t, A_t, S_{t+1})$ is the immediate numerical reward that the agent receives at each state. The discount factor γ determines the importance of future rewards when calculating the total return of a current decision. A lower discount factor indicates less emphasis on future rewards, whereas a higher discount factor implies that future rewards are almost as significant as immediate rewards. The agent's policy $\pi(A_t|S_t)$ can be viewed as a mapping function that maps the current state $S_t \in \mathcal{S}$ to the corresponding action $A_t \in \mathcal{A}$. When the current state is S_t, the agent takes an action A_t according to the policy π and then move transfers to the next state S_{t+1} based on $\mathcal{P}$ and receives a reward signal $R_t \in \mathcal{R}$ from the environment. The generated sequence is

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3... \tag{3.1}$$

The expected cumulative reward G_t is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{T} \gamma^k R_{t+k+1} \tag{3.2}$$

where $\gamma \in [0, 1]$ and T is ∞ or a finite value for infinite horizon or finite problems, respectively.

The objective of RL is to find the opitmal policy that maximizes the expected cumulative reward:

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, \mathbb{E}[G_t] \tag{3.3}$$

3.1.2 Policy and Value Functions

The value function $V_\pi(s)$ and the action value function $Q_\pi(s, a)$ are utilized as a measure of the long-term cumulative reward expectation for each state or state-action pair. These values can evaluate the strategy and guide the agent's decision-making under different states.

$$V_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s], \tag{3.4}$$
$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a], \tag{3.5}$$

can be shown to satisfy the recursive relationship (*Sutton*, 1988):

$$V_\pi(s) = \sum_a \pi(s, a) \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma V_\pi(s')], \tag{3.6}$$
$$Q_\pi(s, a) = \sum_{s'} P^a_{ss'}[R^a_{ss'} + \gamma \sum_{a'} Q_\pi(s', a')], \tag{3.7}$$

where

$$s = S_t, s' = S_{t+1}, a = A_t, a' = A_{t+1},$$
$$R^a_{ss'} = \mathbb{E}[R_{t+1}|S_t = s, S_{t+1} = s', A_t = a],$$
$$P^a_{ss'} = P(S_{t+1} = s'|S_t = s, A_t = a),$$

Eq. 3.6 and Eq. 3.7 can be calculated by the value function or action-value function of the successor state. They are known as ***Bellman Equations***, where approximate solutions are obtained through ***Dynamic Programming (DP)*** (*Bellman*, 1966). Through iterative calculations, the optimal value function of the states can be gradually updated. This process begins with an initial estimate of the value for all states, followed by adjustments using the Bellman equation to more accurately reflect expected returns. In each iteration, the value of each state is adjusted based on the potential rewards from all possible subsequent states. Once the value function stabilizes, the optimal policy π^* that provides the maximum expected reward for each state can be identified:

$$\pi^* = \underset{a}{\operatorname{argmax}}\, V^*(s), \tag{3.8}$$
$$V^*(s) = \max_a R^a_s + \gamma \sum_{s'} P^a_{ss'} V^*(s'), \tag{3.9}$$

or

$$\pi^* = \underset{a}{\operatorname{argmax}}\, Q^*(s,a), \tag{3.10}$$

$$Q^*(s,a) = R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} Q^*(s',a'), \tag{3.11}$$

Due to the unavailability of state transfer probabilities for all states required for DP, researchers have proposed two learning algorithms: *Monte Carlo* (*Hastings*, 1970; *Sutton and Barto*, 2018) and *Temporal Difference (TD) learning* (*Sutton*, 1988).

3.1.3 Temporal Difference (TD) Learning

Temporal Difference (TD) (*Sutton*, 1988; *Sutton and Barto*, 2018) is a central role in RL. It is used to estimate the value function and is capable of evaluating the value function after performing an action. *TD learning* is based on Temporal Difference (TD) error, which is the difference between the estimated value of the current state and the estimated value of the next state. This error is utilized to update the state value function in order to better reflect the actual returns. Through continuous iterations, *TD learning* gradually converges to the optimal value function, enabling the agent to make optimal decisions. Its update rule is:

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \tag{3.12}$$

where $r + \gamma V(s') - V(s)$ is the TD error, and α is a learning rate. Its pseudo-code is presented in Algorithm 1 to clearly describe and share the implementation method of the *TD learning* algorithm.

Algorithm 1 TD learning

Input: the policy π to be evaluated
Output: value function V
  Initialize V(s) arbitrarily
  for each episode **do**
    initialize state s
    for each step of episode, state s is not terminal **do**
      $a \leftarrow$ action given by π for s
      take action a, observe r, s'
      $V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
      $s \leftarrow s'$
    end for
  end for

3.1.4 Q Learning

Q-learning (*Watkins and Dayan*, 1992) and ***SARSA*** (*Rummery and Niranjan*, 1994) are the two typical *TD learning* methods. This thesis is primarily based on *Q-learning* which learns the action value function with the following update rule to find the optimal policy:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)] \tag{3.13}$$

Algorithm 2 Q-learning

Input: the policy π to be evaluated
Output: action value function Q
  Initialize Q(s,a) arbitrarily
  for each episode **do**
    initialize state s
    for each step of episode, state s is not terminal **do**
      $a \leftarrow$ action for s derived by Q, e.g., ε-greedy
      take action a, observe r, s'
      $Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
      $s \leftarrow s'$
    end for
  end for

It estimates the maximum value of the value function of each action, and the optimal strategy can be determined by iteratively finding the extreme value of the Q-function directly. The pseudo code of Q-learning is presented in Algorithm 2. Here, ε is the exploration rate, determining the frequency of taking unknown actions during the learning process. A higher exploration rate means trying new actions more frequently, while a lower exploration rate tends to repeat actions known to yield better results. In general, in the early stages of learning, a higher exploration rate helps the algorithm to understand the environment more comprehensively. As learning progresses, the exploration rate is typically reduced gradually to make more use of the knowledge already acquired.

3.1.5 Monte Carlo

The Monte Carlo method plays a significant role in reinforcement learning as well, utilizing sampling to estimate value functions. Unlike Temporal Difference (TD) learning methods, the Monte Carlo approach does not rely on estimating the value of the next state but rather updates the value function based on actual returns.

In Monte Carlo learning, the update of the value function occurs at the end of a complete episode. The algorithm records all states, actions, and rewards during the episode, and at the end of the episode, it uses the accumulated rewards obtained to update the value estimates of all previously experienced states. This method is particularly suitable for episodic tasks, which have a clear beginning and end.

The update rule for the Monte Carlo method can be represented as follows:

$$V(s) \leftarrow V(s) + \alpha[G_t - V(s)] \tag{3.14}$$

where G_t is the cumulative discounted reward from time t to the end of the episode, α is the learning rate, and $V(s)$ is the current value estimate of state s. In this algorithm, the value of each state is calculated based on the average of the accumulated rewards in all episodes that pass through that state. As more episodes are completed, the value estimates gradually stabilize. A major advantage of this method is that it does not require a model of the environment, meaning it does not need to know the specific probabilities of state transitions and rewards.

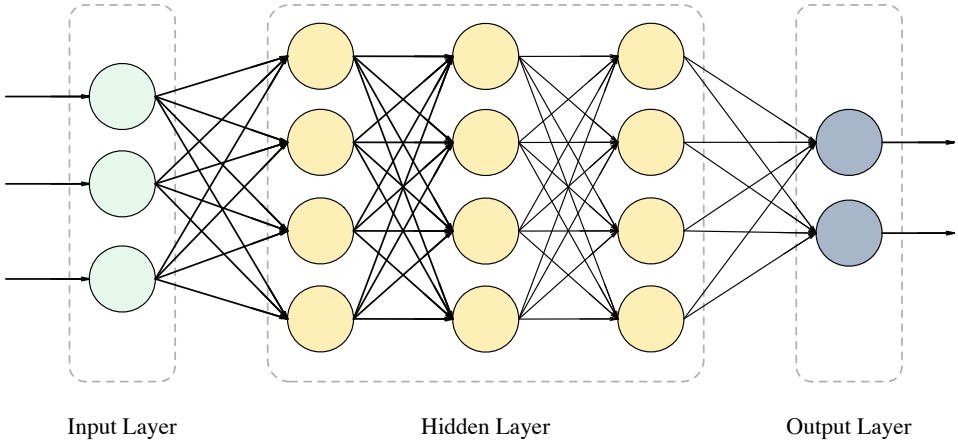


Figure 3.2: A feed-forward neural network with three hidden layers. Each layer consists of multiple neurons, which are represented by circular nodes. The arrow between neurons represents the flow direction of information.

3.2 Deep Learning (DL)

Combining *Bellman's equation* and *TD learning* has led to significant breakthroughs in reinforcement learning research. However, the above methods rely on storing value functions or policies in a 2D tabular form, which cannot be stored in an array when the state and/or action space is large or continuous. Even traditional reinforcement learning algorithms face the curse of dimensionality, where the amount of computation increases dramatically with the number of dimensions of the state space. Here the powerful representation capabilities of ***Deep Learning (DL)*** methods (*LeCun et al.*, 2015; *Goodfellow et al.*, 2016) make RL advance again. DL approximates non-linear functions by training ***Deep Neural Networks (DNNs)*** and learns the intrinsic laws and essential features of the input data.

3.2.1 Neural Networks (NNs)

A neural network is a computational model composed of artificial neurons that simulates information processing and learning capabilities through layer-by-layer linking and weight adjustment. ***Neural networks (NNs)*** (*Schmidhuber*, 2015) usually consist of multiple layers, including an input layer, hidden layers, and an output layer. The input and output layer are the first and the last layer, respectively, and there are one or more hidden layers in between. Each layer uses the output of the previous layer as its input. If the information in a neural network flows in a forward manner without forming cyclic connections, then this neural network model is called a feed-forward neural network or ***Multilayer Perceptron (MLP)***. Feed-forward neural network (or MLP) is a very widely used neural network model.

NNs are a powerful tool for approximating functions due to their highly flexible struture and parameter tuning capabilities. ***Deep NNs (DNNs)*** (with multiple hidden layers) are particularly good at approximating complex nonlinear functional relationships. In the following, I will give a brief

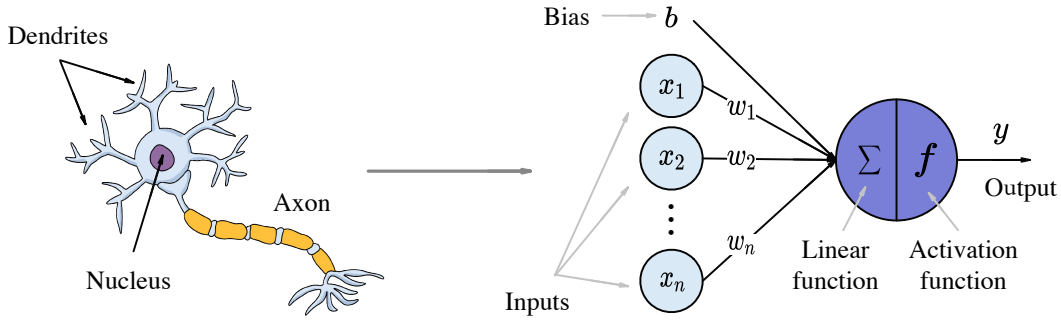


Figure 3.3: The model of biological neuron and artificial neuron. **Left:** A biological neuron (for inspiration). **Right:** An artificial neuron. It has n inputs $(x_1, x_2, ..., x_n)$, weights $(w_1, w_2, ..., w_n)$, a bias b and an activation function f.

description of the different components that make up a neural network.

Neuron

Artificial neurons are the basic processing units of neural networks. Inspired by biological neurons, the mathematical model of artificial neurons was proposed in 1943 (*McCulloch and Pitts*, 1943). An artificial neuron, similar in function to its biological counterpart, takes inputs, performs computations, and generates outputs. Unlike biological neurons, which consist of dendrites, axons, and a nucleus, artificial neurons are simplified computational units designed to process and transmit information. Figure 3.3 illustrates the basic model of both biological neurons (for inspiration) and artificial neurons, emphasizing the fundamental idea of information processing in artificial neural networks.

The neuron computes a weighted sum of the inputs Its mathematical function is as shown:

$$y = f(b + \sum w_i x_i) \tag{3.15}$$

where f is an activation function. For MLP, the neural network consists of multiple layers of neurons with unidirectional signalling. Each neuron takes the output of the previous layer as input to compute the weighted sum, and applies some non-transformation to the computed result to produce the output of this layer as follows:

$$s_j^{(l)} = b_j^{(l)} + \sum_{i=1}^{k} w_{ij}^{(l)} y_i^{(l-1)}, \tag{3.16}$$

$$y_j^{(l)} = f(s_j^{(l)}), \tag{3.17}$$

where k is the total number of neurons at layer $l-1$. $y_i^{(l)}$ is the output of neuron at layer l. $w_{ij}^{(l)}$ is the weight value from ith neuron at layer $l-1$ to jth neuron at layer l. $b_j^{(l)}$ is the bias. $s_j^{(l)}$ is the weighted sum of inputs from layer $l-1$. f is a nonlinear activation function to add nonlinearity in the output. Otherwise, the results we get are just a linear combination of the inputs.

Output Layer

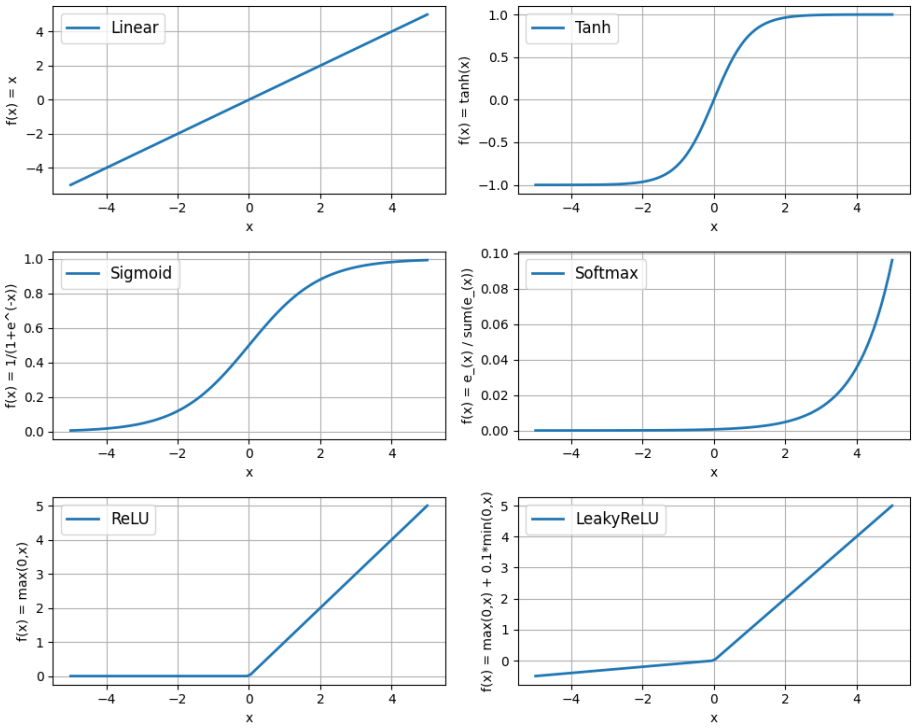


Figure 3.4: Different activation functions.

The output layer represents the final layer in the network, responsible for producing the ultimate prediction corresponding to the given input. Activation functions are utilized in neural networks to accept input signals and generate output signals in output layers. These functions can be classified into two types: linear and nonlinear activation functions, with the choice depending on the specific task at hand. Figure 3.4 shows some of the most commonly used activation functions which are also applied in this thesis.

When dealing with linear regression problems, linear activation functions are the always choice.

$$f_{linear}(x) = x, \tag{3.18}$$

where a linear activation function directly allows for straightforward prediction results. However, it's important to note that the input-output relationship of linear activation functions is limited to simple linear transformations. As a result, this can restrict the expressive capacity of deep neural networks.

In contrast, for the majority of tasks, nonlinear activation functions are favored, such as Sigmoid, ReLU, Softmax and so on. These functions enable more complex input-output mappings, thereby allowing deep neural networks to learn and represent intricate patterns and relationships effectively.

In the binary classification tasks, the sigmoid function is commonly used:

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \tag{3.19}$$

that restricts the output within the range of 0 to 1, representing the corresponding probability.

Similarly, the softmax function is well-suited for multi-class classification tasks to assign probabilities to multiple classes:

$$f_{softmax}(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \tag{3.20}$$

where s_i is the output of the i-th neuron. The softmax function takes the input values and transforms them into probability distributions for various classes, ensuring that the sum of all its output values is equal to 1.

In addition, to alleviate the problem of vanishing gradient, ReLU is used:

$$f_{ReLU}(x) = \begin{cases} x & , x \geqslant 0 \\ 0 & , x < 0, \end{cases} \tag{3.21}$$

that is able to keep the gradient from decaying when $x > 0$. Further, LeakyReLU is used to mitigate the problem of dead ReLU:

$$f_{LeakyReLU}(x) = \begin{cases} x & , x > 0 \\ \gamma x & , x \leqslant 0, \end{cases} \tag{3.22}$$

where γ is a very small postive value.

Furthermore, a tanh activation could restricts the output in the range of -1 to 1:

$$f_{tanh}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{3.23}$$

Objective Function

The objective function, also known as the loss function or cost function, is used to measure the discrepancy or error between the output of a neural network and the desired value. It guides the network's parameter learning and updates by back-propagating the measurement error, aiming to minimize the objective function. As the neural network's parameters gradually adjust, the network can better fit the training data, thereby improving model performance and generalization ability.

A widely used objective function is the mean squared error (MSE):

$$\mathcal{L}_{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}(x_i, \theta))^2, \tag{3.24}$$

where the objective function $\mathcal{L}_{MSE}$ is parameterized with the network parameters θ, which denotes the biases and weights of the network. N is the number of training samples. y_i is the desired value for i-th training sample. $\hat{y}$ is the predicted value of the sample x_i given by the neural network.

For multi-class classification tasks with C classes, cross-entropy (CE) is often employed:

$$\mathcal{L}_{CE}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{C} y_{i,j} log(\hat{y}_j(x_i, \theta)), \tag{3.25}$$

where $y_{i,j}$ denotes the desired value of the sample x_i belonging to the class j. $\hat{y}_j(x_i, \theta)$ is predicted probability of the sample x_i belonging to the class j.

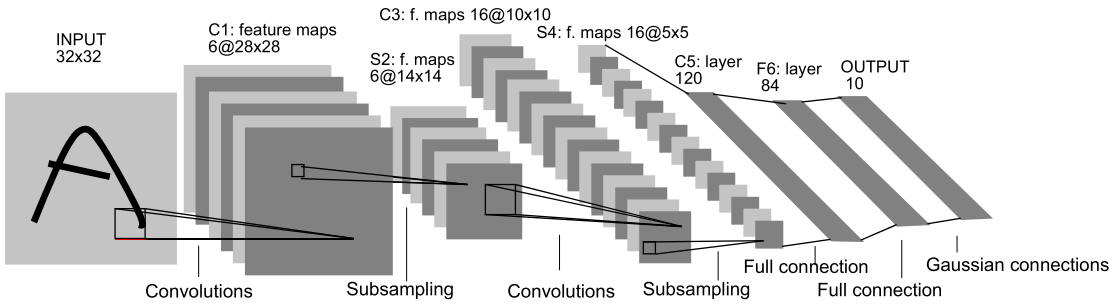


Figure 3.5: The network architecture of LeNet-5 (*LeCun et al.*, 1998).

Generally, any function capable of quantifying the discrepancy between the network's output and a target value can serve as an objective function for training a neural network, provided that it exhibits differentiability concerning the network parameters.

Network Training

Training of a neural network is the process of tuning the parameters of the network so that it can learn from the input data and gradually fit the desired output. The main common neural network training methods are ***Gradient Descent (GD)***, Stochastic Gradient Descent (SGD), Adam (*Kingma and Ba*, 2014), etc.

GD is one of the most common optimisation algorithms. It gradually decreases the objective function by calculating the gradient of the objective function with respect to the parameters and updating the values of the parameters along the descent direction of the gradient

$$\theta_{t+1} = \theta_t - \alpha \nabla \mathcal{L}(\theta_t) \tag{3.26}$$

where θ_t is the parameters at the t-th iteration. $\mathcal{L}$ is the objective function and $\nabla \mathcal{L}(\theta_t)$ is the gradient of objective function $\mathcal{L}$ with respect to parameter θ_t. α is the learning rate that controls the step size of each parameter update in the negative gradient direction. The size of the learning rate directly affects the effectiveness and speed of training. If α is too small, the step size of each parameter update is small, the training process will become very slow and may take a long time to converge to the optimal solution. On the contrary, if α is set too large, the training may become unstable, and may even lead to the objective function constantly jumping or even diverging.

In addition to this, GD has several variants such as Batch Gradient Descent (BGD), Stochastic Gradient Descent (SGD) and Mini-Batch Gradient Descent (MBGD). Among them, SGD applies the gradient of only one sample at each update, which can lead to faster convergence. Further, Adam is an optimization algorithm that combines momentum and adaptive learning rate, which adaptively adjusts on the first-order moment estimates and second-order moment estimates of the gradient.

3.2.2 Convolutional Neural Network (CNN)

Traditional neural networks are structured with fully connected layers, wherein every neuron is connected to all neurons from the preceding layer. In this setup, each layer's representation is derived from the matrix multiplication between a weight matrix and an activation vector from the earlier

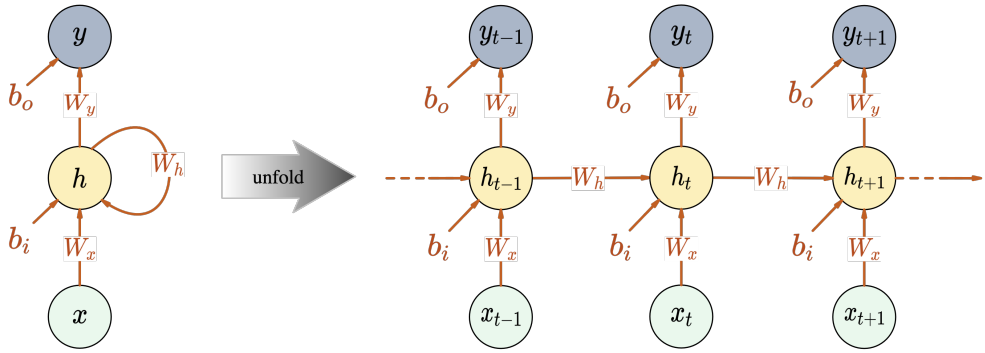


Figure 3.6: The structure of a vanilla RNN unfolded in time. The hidden state h_t relies on the current input x_t and the previous hidden state h_{t-1}. W_x, W_y, W_h, b_i and b_o are shared over time.

layer. On the other hand, convolutional networks distinguish themselves by having at least one convolutional layer. Unlike the dense connectivity in fully connected layers, convolutional layers feature sparse connectivity, where each neuron is linked to only a localized region of the previous layer. Additionally, a hallmark of convolutional layers is the parameter sharing among neurons within the same layer, which significantly trims down the number of parameters, making them more parameter-efficient compared to fully connected layers.

Convolutional Neural Networks (CNNs) (*LeCun et al.*, 1998; *He et al.*, 2016) have found extensive application in image processing tasks, demonstrating remarkable success in image classification and recognition. Each layer in a CNN can be envisioned as a learned filter or kernel that sifts through the input image to extract local features. Thanks to parameter sharing among neurons working on different regions, multiple kernels can be learned with a relatively modest parameter count. The core operation in a convolutional layer boils down to a convolution operation between the input image I and the learned kernel W. The formula for a two-dimensional convolution is described as follows:

$$(I * W)(i, j) = \sum_{l,m} I(i - l, j - m)W(l, m). \tag{3.27}$$

Typically, convolutional networks comprise a sequence of convolutional layers, which is followed by fully connected layers towards the end. A non-linear transformation is employed after the convolutional layer using activation functions such as ReLU to introduce non-linearity. In some cases, a pooling layer is introduced where the activations of multiple neurons are formed into a single value. Utilizing layers like max pooling offers the benefit of imparting invariance to minor translations, making the network more robust to slight variations in the input.

3.2.3 Recurrent Neural Networks (RNNs)

Recurrent Neural Networks (RNNs) are a class of neural network models designed for processing sequential data. Unlike traditional feedforward neural networks, which have one-to-one correspondence between inputs and outputs (i.e., one input corresponds to one output with no connection between different inputs), RNNs consist of artificial neurons and one or more feedback loops. By introducing state variables to store past information and using them along with the current input to

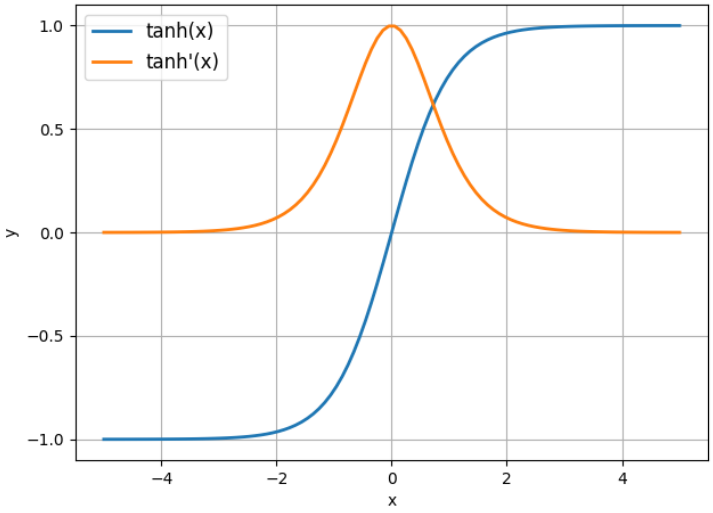


Figure 3.7: The $\tanh$ function and its derivation function $\tanh'$.

determine the current output, information can be passed within the network, and weights can be shared as each sequence element is processed. The left part of Figure 3.6 shows the structure of RNN, where x is the input layer, y is the output layer and h is the hidden layer with the loop. For an input sequence ..., x_{t-1}, x_t, x_{t+1}, ..., the right part of Figure 3.6 shows the network structure when the loop is unfolded. RNN updates its hidden state h_t based on the current input x_t and the previous state h_{t-1}. A vanilla RNN is shown as follows:

$$y_t = g(W_y h_t + b_o), \tag{3.28}$$

$$h_t = f(W_x x_t + W_h h_{t-1} + b_i), \tag{3.29}$$

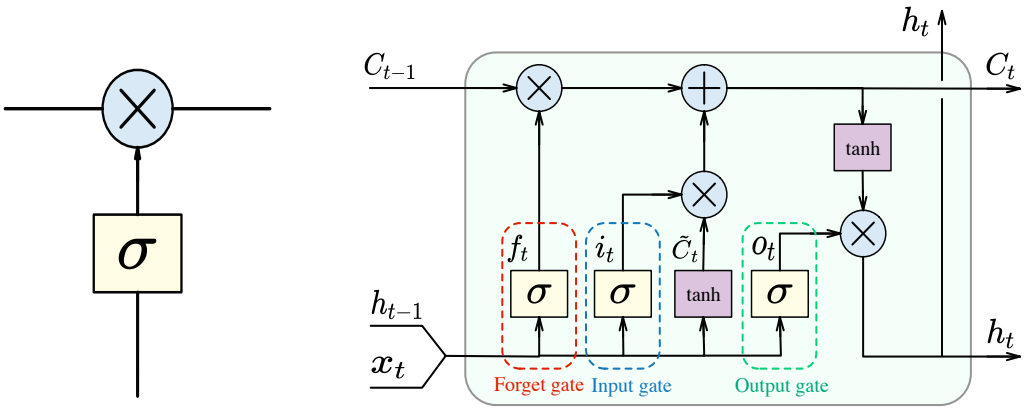
where b_* is a bias vector. W_x, W_y and W_h are weight matrices. f and g are non-linear activation functions. RNNs share the weight parameters and bias vectors on the time axis, which allows models to handle sequence data of variable length. A common activation function f in RNNs is the hyperbolic tangent function.

In general, the training of neural networks is achieved through back-propagation. For RNN, its back-propagation becomes Back-Propagation Through Time (BPTT) due to its special structure. Since the output at each time step t is iteratively calculated based on the computations from time step 0 to $t-1$, a large concatenated multiplication term actually arises in the back-propagation. Let $\mathcal{L}_t$ represent the loss function, the formula for calculating the partial derivative of $\mathcal{L}_t$ for W_x at time t is as follows:

$$\frac{\partial \mathcal{L}_t}{\partial W_x} = \sum_{k=0}^{t} \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W_x}, \tag{3.30}$$

Similarly, $\frac{\partial \mathcal{L}_t}{\partial W_h}$ is shown as:

$$\frac{\partial \mathcal{L}_t}{\partial W_h} = \sum_{k=0}^{t} \frac{\partial \mathcal{L}_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \left(\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} \right) \frac{\partial h_k}{\partial W_h}, \tag{3.31}$$



(a) (b)

Figure 3.8: The structure of (a) gate and (b) LSTM.

In this case, when an activation function is present, the following equation is satisfied:

$$\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}} = \prod_{j=k+1}^{t} f'(h_{j-1}) W_h \tag{3.32}$$

If the activation function of f is the Tanh function, it has $0 \leqslant \tanh' \leqslant 1$. From Figure 3.7 which shows the Tanh function and its derivative function Tanh', we can see: When t is large, $\prod_{j=k+1}^{t} f'(h_{j-1}) W_h$ tends to 0, resulting in a vanishing gradient problem. In addition, when W_h is large resulting $\prod_{j=k+1}^{t} f'(h_{j-1}) W_h$ tends to infinity, the gradient explosion problem arises (*Bengio et al.*, 1994). In order to overcome these problems, researchers have proposed many improved models based on RNN, two typical models are ***Long-Short Term Memory (LSTM)*** (*Hochreiter and Schmidhuber*, 1997) and ***Gated Recurrent Unit (GRU)*** (*Cho et al.*, 2014).

3.2.3.1 Long-Short Term Memory (LSTM)

In vanilla RNNs, the long temporal dependency problem often leads to gradient vanishing or gradient explosion, making it difficult for the network to capture long temporal sequence dependencies. LSTM's improvement lies in the introduction of gate mechanisms, allowing the network to selectively pass or forget information, thereby better handling long sequential data. A gate is a component that selectively allows information to pass through, composed of a layer with Sigmoid activation function and an element-wise multiplication operation, as shown in Figure 3.8(a).

LSTM introduces three gate functions: the input gate, the forget gate, and the output gate, to

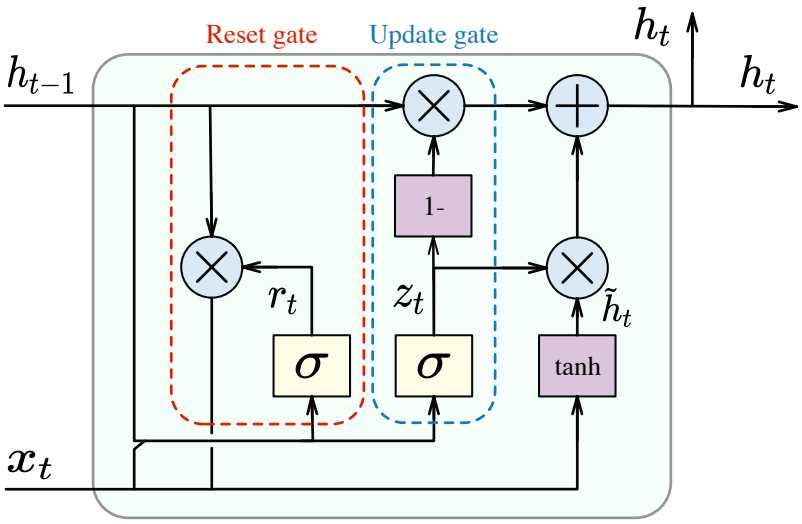


Figure 3.9: The structure of GRU.

control input values, memory values, and output values. The updated rules are as follows:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f), \tag{3.33}$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i), \tag{3.34}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o), \tag{3.35}$$
$$\tilde{c}_t = \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c), \tag{3.36}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{3.37}$$
$$h_t = o_t \tanh(c_t), \tag{3.38}$$

where $\odot$ is the Hadamard product, and σ is the sigmoid function. W_{**} are weights of LSTM, b_* are bias vectors. x_t and h_t are the input and hidden state at time t, respectively. c_t is the memory cell. i_t is the input gate, f_t is the forget gate, and o_t is the output gate.

At each time step, the forget gate determines which information to be forgotten from the memory cell, while the input gate selects which information to be added and updates the current information in the memory cell. Finally, the output gate decides which information from the memory will be output as the current time step's output by multiplying it with the memory cell.

3.2.3.2 Gated Recurrent Unit (GRU)

GRU is a variant of LSTM with a simpler structure and fewer gate mechanisms. It retains the advantages of LSTM, such as effectively capturing long temporal dependencies. However, it is easier to optimize during the training process, and it is less prone to issues like vanishing or exploding gradients. The main feature of GRU is the introduction of two gate mechanisms: the update gate and the reset gate. These gate mechanisms help GRU determine how to process input data and the previous

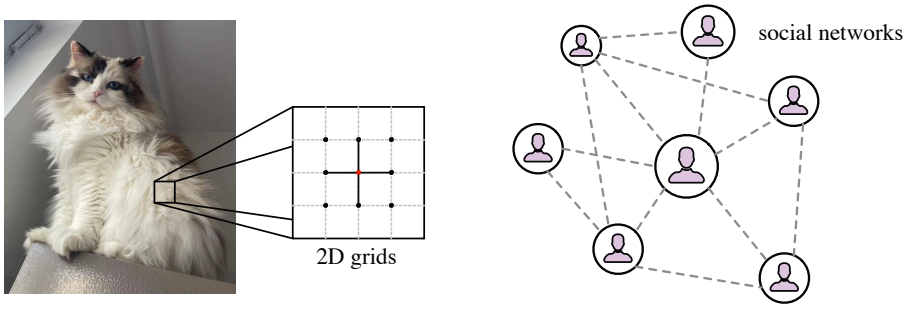


Figure 3.10: **Left:** Image in Euclidean space. **Right:** Graph in non-Euclidean space.

hidden state. The update rules for GRU at each time step are as follows:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \tag{3.39}$$
$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{3.40}$$
$$\tilde{h}_t = \tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h), \tag{3.41}$$
$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t, \tag{3.42}$$

where W_* and U_* are weight matrices of GRU, b_* are bias vectors. x_t and h_t are the input and hidden state at time t, respectively. $\tilde{h}_t$ is denoted as the candidate state at time t. $\odot$ is the Hadamard product. σ is the sigmoid function. z_t, r_t are the update gate and reset gate, respectively.

In GRU, the reset gate controls how much past information should be forgotten, while the update gate decides how much information from the previous time step and the current time step should be propagated to the next time step.

3.2.4 Graph Neural Networks (GNNs)

Graph Neural Networks (GNNs) (*Kipf and Welling*, 2016; *Velickovic et al.*, 2017) are a class of deep learning models specifically designed for handling graph-structured data. They can handle data with irregular spatial structures commonly found in real-life scenarios, i.e., data in non-Euclidean spaces. For example, abstracted graphs from electronic transactions and recommendation systems, where each node is connected to others without fixed patterns. Figure 3.10 shows examples of the data in Euclidean space and the data in non-Euclidean space. In order to facilitate understanding, I have summarized some definitions related to graphs.

Definition 1 *A graph is composed of nodes and edges, usually denoted as $\mathcal{G} = (V, E, A)$, where V represents the set of nodes, E represents the set of edges, and A represents the adjacency matrix of the graph. $v_i \in V$ is the ith node, and $e_{ij} \in E$ is the edge between ith node and jth node. If there exists $e_{ij} \in E$, then $A_{ij} = w_{ij} > 0$, otherwise $A_{ij} = 0$ if $e_{ij} \notin E$.*

Definition 2 *An undirected graph is a graph in which all edges have no direction, and $A_{ij} = A_{ji}$. For the directed graph, all edges point from one node to another one, and $A_{ij} \neq A_{ji}$.*

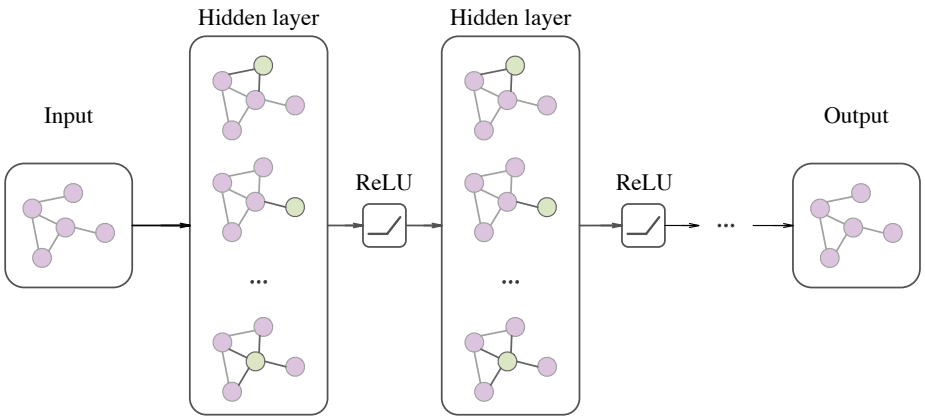


Figure 3.11: Structure of Graph Convolution Networks (GCNs) with multi-layer.

The goal of GNNs is to learn feature representations between nodes and graphs. For example, given a graph $\mathcal{G}$, where x_v represents the feature of node v, and $x_{v,u}$ represents the feature of the edge between nodes v and u. The v node's hidden state h_v is calculated based on its own feature x_v, the features of connected edges $x_{co[v]}$, the features of its neighboring nodes $x_{ne[v]}$ and the hidden states of its neighbors $h_{ne[v]}$:

$$h_v = f(x_v, x_{co[v]}, h_{ne[v]}, x_{ne[v]}) \tag{3.43}$$

where f is a parametric function that is called local transition function and shared through all nodes. Graph neural networks model data in non-Euclidean spaces by propagating information between nodes in the network, capturing internal dependencies within the graph data. They can update and integrate features for nodes and edges, effectively utilizing the global information within the graph. Two typical models of GNNs are Graph Convolutional Graphs (GCNs) and Graph Attention Networks (GATs).

3.2.4.1 Graph Convolution Networks (GCNs)

Graph Convolution Networks (GCNs) (*Kipf and Welling*, 2016) generalize convolutional operations from traditional data to graph data. Specifically, for each node, GCN computes the weighted average of its neighboring node features and integrates this average feature with the node's own feature, resulting in a new feature representation for the node. This aggregation process enables nodes to propagate and share information on the graph, thereby acquiring more enriched feature representations. As shown in Figure 3.11, the general equation in GCN used to propagate feature representations to the next layer is as follows:

$$H^{l+1} = f(H^l, A) \tag{3.44}$$

where L is the number of layers, A is the adjacency matrix. H is the set of hidden state of each layer, $H^0 = X$ which X is the input, and $H^l = Z$ which Z is the output. In *Kipf and Welling* (2016), it is defined as:

$$H^{l+1} = \sigma\left(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^l W^l\right) \tag{3.45}$$

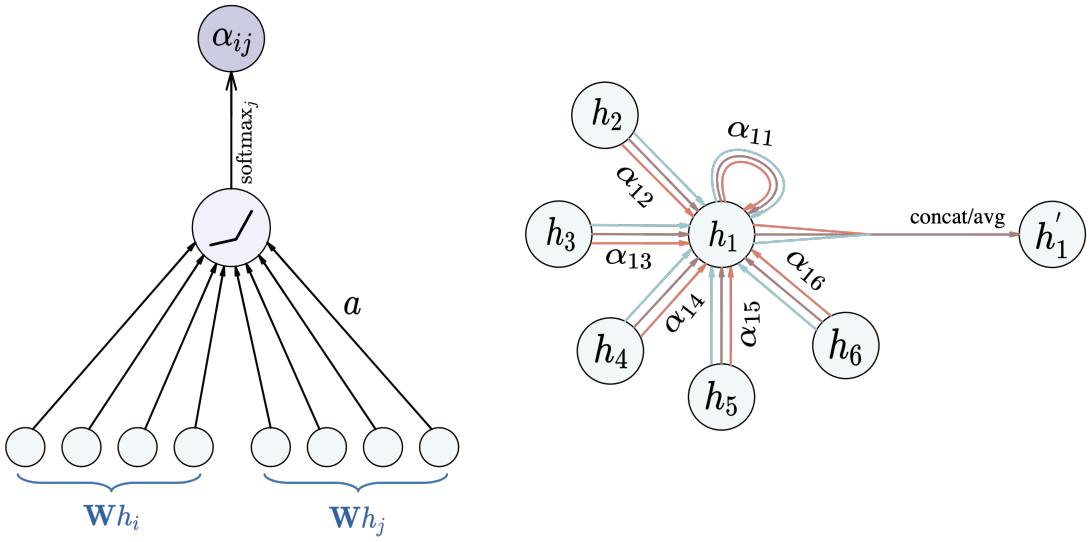


Figure 3.12: **Left:** Attention mechanism. **Right:** Illustration of multi-head attention with head(k) = 3. Different colors of lines indicate independent attention computations. (*Velickovic et al.*, 2017)

where $\hat{A} = A + I$ is an adjacency matrix after adding self-loops and and I is an identity matrix. $\hat{D}_{ii} = \sum_j \hat{A}_{ij}$ is a diagonal degree matrix. $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$ is a normalized adjacency matrix that represents the structure of the graph. W^l is a weight matrix for l-th neural network layer. $H^l W^l$ gives a linear transformation to the embedding of all nodes in layer l. $\sigma(.)$ is a non-linear activation function.

3.2.4.2 Graph Attention Network (GAT)

Graph Attention Networks (GATs) (*Velickovic et al.*, 2017) introduce the attention mechanism for handling graph-structured data. Specifically, GATs dynamically aggregate neighboring node features by calculating attention weights between nodes and their neighbors as shown in Figure 3.12, allowing for node feature updates and learning. Unlike fixed neighbor weights in traditional GNNs, this attention function considers the relationship and feature similarity between nodes, assigning appropriate weights to different neighbors. Through the calculation of attention weights, GATs can adaptively learn the importance of nodes, leading to a more effective aggregation of neighboring node information.

First, in order to enhance the expressive power and transform the input features into higher-level representations, GAT performs self-attention on the given set of node feature vectors. For vertex i, the attention weights (or the similarity coefficients) are calculated between itself and each of its neighbors ($j \in N_i$) one by one:

$$e_{ij} = a([\mathbf{W}h_i || \mathbf{W}h_j]), j \in N_i \tag{3.46}$$

where N indicates the number of the neighborhood of node i, $\mathbf{W}$ is the weight matrix and $||$ is the concatenation operation. e_{ij} indicates the importance of the features of j-th node to i-th node. $a(\cdot)$ is

a mapping function that projects the concatenated high-dimensional features to a real number, such as a feedforward neural network. It allocates attention only to the neighboring node set of node i.

Then, the softmax function is used to normalize all the attention weights:

$$\alpha_{ij} = \frac{\exp(\text{LeakyReLU}(e_{ij}))}{\sum_{k \in N_i} \exp(\text{LeakyReLU}(e_{ik}))} \tag{3.47}$$

where α_{ij} is called normalized attention coefficients. LeakyReLU denotes the LeakyReLU nonlinearity function.

Finally, the attention weights are applied to weight and average the neighboring nodes' features, resulting in a new aggregated feature representation $h_i^{'}$ for node i:

$$h_i^{'} = \sigma\left(\sum_{j \in N_i} \alpha_{ij} \mathbf{W} h_j\right), \tag{3.48}$$

where σ is the activation function. In addition, GATs utilize a multi-head attention mechanism with K heads to further enhance the model's capacity for fitting:

$$h_i^{'}(k) = ||_{k=1}^{K} \sigma\left(\sum_{j \in N_i} \alpha_{ij}^{k} \mathbf{W}^{k} h_j\right), \tag{3.49}$$

where $||$ indicates the concatenation operation. The multi-head attention enables parallel computation since the computation for each head is independent.

3.3 Deep Reinforcement Learning (DRL)

As described earlier, in order to overcome the limitations of classical reinforcement learning methods due to the curse of dimensionality and poor training performance on large datasets, deep reinforcement learning methods train deep neural networks to approximate any nonlinear function, thereby extracting the inherent features of input data and generalizing to unknown states. DRL integrates the powerful understanding capabilities of deep learning in perceptual tasks like vision with the decision-making capabilities of reinforcement learning, enabling end-to-end learning. DRL can be classified into two categories: value-based methods and policy-based methods.

3.3.1 Value-based Learning

Building on the classical temporal-difference methods discussed in section 3.1.3, value-based DRL, unlike querying value tables, approximates values using deep neural networks. Value-based methods then indirectly obtain the agent's policy by iteratively updating the value function. When the value function reaches its optimal value, the agent's optimal policy is obtained through the optimal value function.

3.3.1.1 Deep Q-Network (DQN)

Value-based deep reinforcement learning, exemplified by ***Deep Q-Network (DQN)*** (*Mnih et al.*, 2013, 2015) introduced by DeepMind in 2015. DQN extends the principles of Q-learning to handle

high-dimensional raw scene data, such as game images, as its input, by using deep neural networks to approximate the Q-function. The Q-network's output represents the Q-function values for various actions in the given state, enabling the selection of the action with the maximum Q-value at each time step for decision-making and action. Additionally, DQN introduces two mechanisms, "experience replay" and "target network."

Experience replay is used to store the agent's previous experiences. At each time step, the experiences (s, a, r, s') are stored in an experience replay buffer, and a random sample of experiences from the experience replay buffer is used for training. Experience replay helps remove the correlation between data sequences and smooths out the variation in data distribution, thus enhancing training stability.

The target network in DQN, is an independent neural network used to calculate the target Q-values and stabilize the training process. The parameter θ_i^- of the target Q-network $Q(s', a'; \theta_i^-)$ is periodically copied from the current Q-network $Q(s, a; \theta_i)$ with a certain frequency. Because the target Q-network's parameter θ_i^- is fixed for a certain period, the training network's update targets remain relatively stable. This helps avoid the problem of value estimation bias caused by continuous target updates and improves the stability and efficiency of DQN training.

In DQN, the network parameters are updated based on Gradient Descent (GD), and the loss function is defined as follows:

$$\mathcal{L}_i = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{3.50}$$

If we choose to optimize V-value rather than the more common choice Q-value, it is called Deep V-Learning (Deep Value Learning). Similar to the classical DQN algorithm, Deep V-Learning also performs incremental updates to the values.

3.3.1.2 Double Deep Q Network (DDQN)

In DQN, we use Q-values to estimate the value of each action, where higher Q-values imply a higher likelihood of yielding greater rewards. However, when we employ deep neural networks to estimate Q-values, a problem known as overestimation arises. This means that the network may overestimate the Q-values for certain actions, leading to training instability and convergence difficulties.

Definition 3 *"Overestimate" refers to the process of first taking the maximum value from a series of numbers $\{X_1, X_2, ..., X_N\}$ and then calculating the average, typically resulting in a value that is greater than or equal to the result obtained by first calculating the average and then finding the maximum value. This can be mathematically expressed as:* $\mathbb{E}(\max(X_1, X_2, ..., X_N)) \geqslant \max(\mathbb{E}(X_1), \mathbb{E}(X_2), ..., \mathbb{E}(X_N))$.

To address the issue of overestimation of Q-values in DQN, Double Deep Q Network (DDQN) was introduced in *Van Hasselt et al.* (2016) as an improvement. DDQN utilizes two separate Q-networks, where one is used for action selection, and the other is used for action evaluation. Specifically, DQN is to directly select the maximum Q-value among the Q values corresponding to the next state's various actions in the target Q-network, which is shown as:

$$Y_{target}^{\text{DQN}} = r + \gamma \max_{a'} Q(s', a'; \theta_i^-) \tag{3.51}$$

In contrast, DDQN uses the estimated Q-network to select the optimal action corresponding to the maximum Q-value and then uses the target Q-network to evaluate the selected optimal action, which can be shown as follows:

$$\text{selection:} \quad a^{max}(s', \theta_i) = \arg\max_{a'} Q(s', a'; \theta_i), \tag{3.52}$$

$$\text{evaluation:} \quad Y_{target}^{\text{DDQN}} = r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta_i), \tag{3.53}$$

Through this approach, even if one network overestimates the value of certain actions, the other network provides a more accurate evaluation, reducing the impact of overestimation. DDQN then does a GD step with a loss function which is defined as follows:

$$\mathcal{L}_i = \mathbb{E}\left[\left(r + \gamma Q(s', \arg\max_{a'} Q(s', a'; \theta_i); \theta_i^-) - Q(s, a; \theta_i)\right)^2\right] \tag{3.54}$$

By minimizing the loss function, the model learns how to adjust its policy to maximize the cumulative rewards in reinforcement learning tasks. Meanwhile, the parameters of the target Q-network θ_i^- are periodically copied from the estimated Q-network θ_i, aiming at maintaining the stability of the training process, as having a fixed target prevents the instability caused by continuously changing targets during training. This strategy effectively resolves the overestimation issue found in DQN, making DDQN more effective and stable across various reinforcement learning tasks.

3.3.2 Policy-based Learning

Compared to value-based methods, which are primarily designed for discrete and low-dimensional action spaces, policy-based algorithms are typically more suitable for tasks with continuous and high-dimensional state spaces, and they perform better in exploring complex environments.

In Policy-based algorithms, neural networks are commonly used to represent the policy function, which maps environmental states to probability distributions over actions, shown as $\pi(a|s, \theta)$. Such networks are called policy networks, and their parameters are trained to maximize the cumulative reward. Unlike value-function methods (such as DQN), Policy-based algorithms learn how to choose optimal actions directly by optimizing the policy, rather than learning a value function and selecting actions based on it (e.g., e-greedy strategy).

Depending on whether they rely on Gradient Descent (GD) method, Policy-based algorithms can be divided into two categories. The ones based on GD are known as policy gradient methods. In general, such methods update the parameters of the policy network using the GD method. Defining the policy objective function as $J(\theta)$, the goal of policy gradient methods is to find the maximum of the objective function:

$$\theta_* = \arg\max_{\theta} J(\theta) \tag{3.55}$$

Policy gradient methods perform parameter updates to update the parameter:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{3.56}$$

where α is the learning rate and $\nabla_\theta J(\theta)$ is the policy gradient.

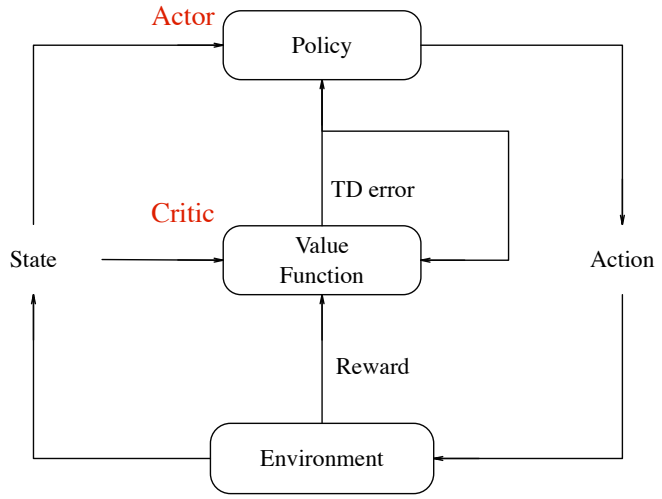


Figure 3.13: Actor-Critic (AC) architecture shown in *Sutton and Barto* (2018).

3.3.2.1 Actor-Critic (AC) Architectures

Actor-Critic (AC) Architecture (*Sutton et al.*, 1999; *Mnih et al.*, 2016) combines the Actor (policy network) from the Policy Gradient method and the Critic (value function network) from the Value-based method. The Actor is used to learn the policy in the environment and find the optimal actions of given states. The Critic is used to estimate the value of states or state-action pairs and provides feedback to the Actor, helping it update the policy for better exploration and exploitation of the environment.

3.3.2.2 Deep Deterministic Policy Gradient (DDPG)

Deep Deterministic Policy Gradient (DDPG) method (*Lillicrap et al.*, 2015) is an Actor-Critic based approach suitable for continuous action spaces, where the obtained policy is a deterministic policy (i.e., $a_t = \mu(s_t|\theta^\mu)$). Additionally, it utilizes two important mechanisms from DQN, namely Experience Replay and Target Network, to enhance training stability and convergence. N samples are randomly selected in the experience replay buffer, and the Q-network $Q(s, a|\theta^Q)$ is updated using gradient descent. The Q-network's loss function is shown as follows:

$$\mathcal{L} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^Q))^2 \tag{3.57}$$

where y_i indicates the Critic target network with $y_i = r_i + \gamma Q^{'}(s_{i+1}, \mu^{'}(s_{i+1}|\theta^{\mu^{'}})|\theta^{Q^{'}})$. And the parameters of target networks are updated by:

$$\theta^{*^{'}} \leftarrow \tau\theta^* + (1-\tau)\theta^{*^{'}} \tag{3.58}$$

CHAPTER 4

DRL-based Social-aware Robot Navigation

In this chapter, I provide an overview of the fundamental aspects of research on DRL-based social-aware robot navigation. First, I introduce the basic framework of DRL-based social-aware robot navigation. Next, I formulate the problem and discuss its essential components. Finally, I briefly describe the current simulation experiments' environmental settings, experimental setup, and performance metrics for evaluation.

Contents

4.1 Framework

Mobile robots move in two-dimensional or three-dimensional space, such as autonomous vehicles navigating city roads and aircraft flying in two-dimensional space. Many real-world applications only require navigation in two dimensions. The purpose of their navigation is to find an optimal path from the starting point to the destination that is safe and efficient. These robots only need to handle motion and obstacle avoidance in a two-dimensional plane, which significantly simplifies the model and problem complexity. As a result, the majority of current research focuses on navigation problems in two dimensions, making the problems easier to model and address.

As mentioned above, mobile robot navigation can be seen as a combination of Point-to-Point (P2P) movement and obstacle avoidance behavior. P2P requires obtaining the position of the target point relative to the starting point. The mobile robot can achieve this by using a global positioning system such as GPS or ultra-wideband localization system to obtain its absolute position on the map,

or by knowing the absolute coordinates of the target point in advance and calculating the coordinate difference between the target point and its current position, thus obtaining the relative position information. Additionally, the robot can also use visual sensors or other sensors to perceive the surrounding environment and estimate relative position information. For obstacles in the environment, they can be classified into static, dynamic, and structurally continuous or discontinuous obstacles, which can be sensed using cameras or various comprehensive perception sensors. In this work, I mainly focus on obstacles composed of static or dynamic agents, as their positions and shapes may vary with time and changes in the environment. These obstacles may require the mobile robot to make real-time decisions based on real-time perception information to avoid collisions or choose the optimal path.

Currently, navigation based on DRL has been used to replace or integrate into traditional navigation frameworks, as shown in Figure 4.1 depicting the interaction between the DRL-based agent and the environment. DRL-based agents often apply and extend various well-known DRL algorithms, such as DQN and DDPG, by modeling the navigation process as an MDP. They typically utilize sensor observations as states in an attempt to maximize the expected rewards, although it's important to note that the suitability of the MDP model may vary depending on the specific navigation environment and its characteristics. Overall, DRL-based navigation algorithms exhibit strong adaptability, high learning capability, and efficiency. In particular, when mobile agents (e.g., robots or intelligent vehicles) navigate in crowded environments like airports or shopping malls, they not only need to interact socially with other pedestrians but also efficiently reach their destination while adhering to social norms. This is known as social-aware robot navigation. For mobile agents in social navigation, the challenges are even greater: (1) the higher density of dynamic obstacles in crowded environments, and (2) the randomness of human behavior makes adhering to social norms in navigation more difficult. The purpose of using DRL algorithms for social navigation is to enable agents to autonomously learn and optimize navigation strategies, empowering robots with enhanced decision-making capabilities and adaptability to cope with the complexity and uncertainty in crowded environments with human presence.

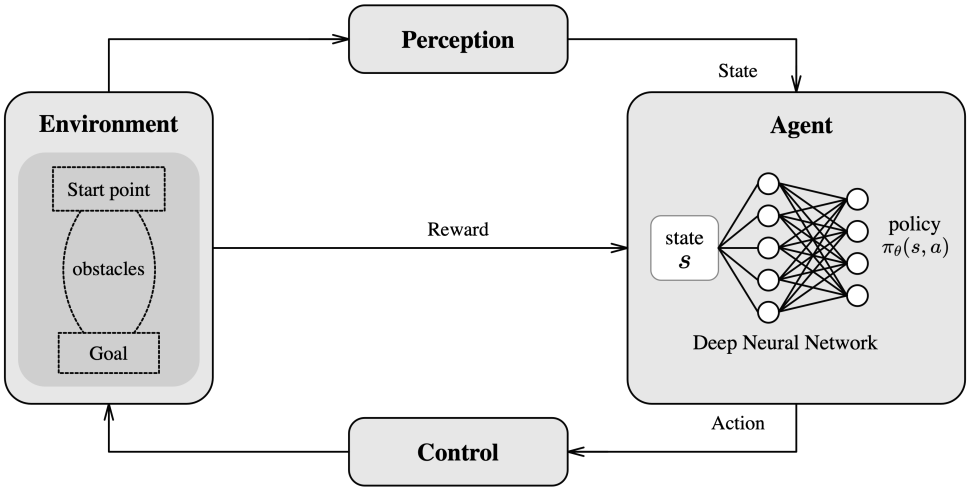


Figure 4.1: DRL-based navigation system

4.2 Problem Formulation

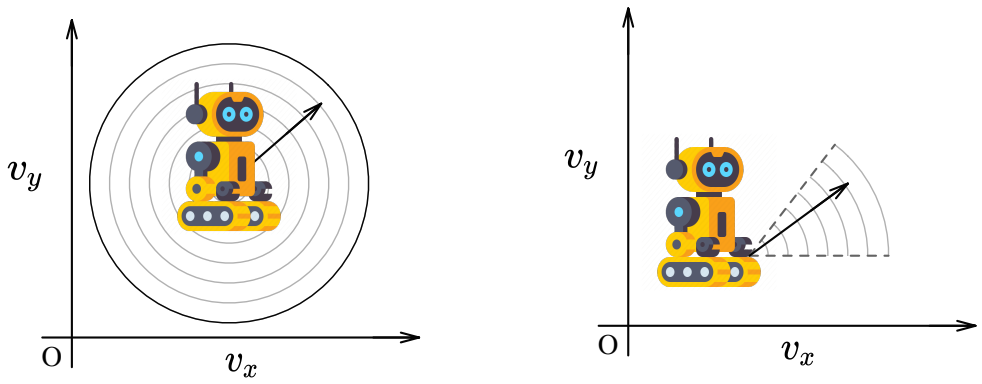
Based on existing papers like *Chen et al.* (2019, 2017b); *Everett et al.* (2018), social-aware robot navigation in a 2D Euclidean space can be formulated as a sequential decision-making problem within an RL framework, wherein the robot learns to make optimal actions at each time step to reach a goal while navigating through a crowd of n humans. I model the interactions between the robot and other humans as an MDP, defined by the tuple $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$. $\mathcal{P}$ is the probability transition function and γ is the discount factor. The state space $\mathcal{S}$, the action space $\mathcal{A}$, and the reward function $\mathcal{R}$ are the three key elements. These elements together form the foundation for the agent's decision-making and learning within the environment, directly influencing the performance and application scenarios of the DRL algorithm. By optimizing the agent to select the most advantageous actions in a given state, aiming to maximize cumulative rewards, the navigation system steadily improves its performance and becomes applicable to various navigation tasks.

4.2.1 States Space and Parametrization

The state space in robot navigation tasks consists of two main components: the robot's self-state and the obstacle (e.g., dynamic or static humans) state. The robot's self-state s includes its current position $\mathbf{p}^r = [p_x^r, p_y^r]$, target position $\mathbf{p_g}^r = [p_{gx}^r, p_{gy}^r]$, speed $\mathbf{v}^r = [v_x^r, v_y^r]$, orientation θ, and radius r, which aid in perceiving its own movement and location. On the other hand, the obstacle states comprise information about obstacles in the environment, such as the current position $\mathbf{p}^i = [p_x^i, p_y^i]$, speed $\mathbf{v}^i = [v_x^i, v_y^i]$, and radius r^i of the i-th human, which influence the robot's navigation decisions. Notably, for an agent (robot or human), its position $\mathbf{p}^r/\mathbf{p}^i$, speed $\mathbf{v}^r/\mathbf{v}^i$, and radius r/r^i can be observed by others. However, the robot's target position $\mathbf{p_g}$, preferred speed v_{pref}, and orientation θ are unobservable and known only to the robot itself. Let $\mathbf{s}_t^r$ denote the robot state and $\mathbf{s}_t^h = [\mathbf{s}_t^1, \mathbf{s}_t^2, ..., \mathbf{s}_t^n]$ be the observable state of humans at timestep t. Thus, the joint state for robot navigation is $\mathbf{s}_t^{jn} = [\mathbf{s}_t^r, \mathbf{s}_t^h]$.

To simplify calculations, researchers transformed the global Cartesian coordinates into local polar coordinates and introduced a new coordinate system that sets the robot as the center of the coordinate system. In the original global coordinate system, the representation of the robot's position and target point is absolute, which might lead to coordinate transformations or rotations that do not affect the optimal decisions. For example, the robot's facing direction in the global coordinate system may vary, but its decision goals and strategies remain the same, as such directional information is unnecessary. By introducing the agent-centric local coordinate system, the robot's current position becomes the origin, and the x-axis points toward the robot's target. The representation of the target point is then converted into relative coordinates with respect to the robot's position. This transformation eliminates uncertainty and redundancy in the global coordinate system, simplifying and enhancing the robot's decision-making process. Furthermore, agent-centric parameterization streamlines the representation of the robot's state and action selection, reducing computational complexity and enhancing efficiency. For simplicity, after omitting the timestep t, the transformed state of the robot and pedestrians is represented as follows:

$$\begin{aligned} s^r &= [d_g, v_x^r, v_y^r, v_{pref}, r], \\ s^i &= [p_x^i, p_y^i, v_x^i, v_y^i, r^i, d^i, r^i + r], \end{aligned} \tag{4.1}$$



(a) Holonomic kinematics (b) Non-holonomic kinematics

Figure 4.2: Holonomic kinematics and non-holonomic kinematics. (a) Left: The robot that has holonomic kinematics can take any possible action (velocity vector) without motion constraints. (b) Right: The set of permissible velocity vectors for the robot that has non-holonomic kinematics.

where $d_g = ||\mathbf{p}^r - \mathbf{p}_g||_2$ is the distance from the current position of the robot to its goal position and $d^i = ||\mathbf{p}^r - \mathbf{p}^i||_2$ denotes the distance from the robot's current position to the position of i-th human.

4.2.2 Action Space

The action space contains all possible actions that a robot can take, which depends on the robot's motion ability and the requirements of the navigation task. The robot's motion can be classified into holonomic and non-holonomic motion, as shown in the Figure 4.2.

The robot with holonomic kinematics refers to its ability to move freely on a plane and in any direction without motion constraints. In this case, the action space is typically continuous, allowing the robot to select any combination of speed and direction for smooth and unrestricted movement. On the other hand, non-holonomic motion imposes certain constraints on the robot's motion, limiting its movement to specific directions while facing constraints in other directions. When the robot has non-holonomic kinematics, its action space is usually discrete, restricting the robot's action choices based on specific motion constraints. These constraints may lead to more restricted turning and movement, resulting in a larger turning radius or requiring more action combinations to complete turns, which could affect the smoothness of motion. To achieve smoother motion, for robots with non-holonomic motion, appropriate action planning and control strategies may be needed to enable smooth turning and movement in feasible directions, avoiding sudden changes in direction or jitter.

In social-aware robot navigation, the action space consists of allowed velocity vectors. If we assume that the robot's velocity can be immediately reached after receiving the action command, then $\mathbf{v}^r = \mathbf{a}$. When the robot has holonomic kinematics, $||\mathbf{v}^r||2 < v_{pref}$. When the robot has non-holonomic kinematics, it means it has rotation constraints:

$$\mathbf{a} = [v, \omega], \text{for } v < v_{pref}, |\omega - \theta| < \phi, \tag{4.2}$$

$$|\theta_{t+1} - \theta_t| < \Delta t \cdot v_{pref}, \tag{4.3}$$

where v is the linear velocity and $v_{pref} = 1m/s$. ω is the angular velocity and ϕ is the angular value of the rotation constraint. Eq. 4.2 restricts the agent's possible directions of motion. In Eq. 4.3, the relationship is defined between the maximum turning rate and the minimum turning radius of 1.0m. This equation establishes the maximum rate at which the robot can turn while ensuring that it achieves a minimum turning radius of 1.0m.

Overall, the size of the action space impacts the flexibility and efficiency of robot navigation. A larger action space may increase computational complexity, but the robot can have more flexible action choices. A smaller action space may reduce the computational burden, but the robot's action choices may be more limited.

4.2.3 Reward Function

The reward function in DRL-based social-aware robot navigation is used to evaluate the effectiveness of specific actions taken by the agent (robot) in a given state. The design of the reward function plays a crucial role in RL, as it encourages the agent to take beneficial actions by providing positive rewards and punishes harmful actions with negative rewards. Consequently, the agent aims to maximize the long-term accumulated rewards through learning and optimizing its strategies, thereby achieving improved navigation behavior.

In general, in social-aware robot navigation, the reward function typically encourages the robot to move toward the target direction and reach the designated goal. For instance, significant positive rewards may be given upon reaching the goal, motivating the robot to successfully complete the navigation task. Furthermore, to avoid collisions with obstacles, the reward function may offer positive rewards for actions that avoid collisions, and negative rewards for actions that result in collisions or getting close to obstacles. Additionally, in certain scenarios, energy conservation becomes a crucial objective. The reward function may encourage the robot to navigate in an energy-efficient manner, such as by choosing a more energy-saving path. In conclusion, the design of the reward function needs to consider various factors, including task objectives, environmental characteristics, and the robot's motion capabilities.

A well-designed reward function can enhance the learning efficiency and navigation performance of the reinforcement learning algorithm. However, designing the reward function may also present challenges, as it requires balancing trade-offs between different objectives and avoiding adverse reward signals that could lead to unstable training or getting stuck in local optima. In most of the social-aware navigation works, the common reward function is sparse which only rewards when reaching the goal position and penalizes when having a collision or being discomfort, while most of the time in navigation, it does not receive any reward signals. As shown in *Chen et al.* (2017b), the common sparse reward function is given:

$$R(\mathbf{s}_t^{jn}, \mathbf{a}_t) = \begin{cases} -0.25 & \text{if } d_{min} < 0 \\ -0.1 + d_{min}/2 & \text{else if } d_{min} < r_c \\ 1 & \text{else if } \mathbf{p} = \mathbf{p}_g \\ 0 & \text{otherwise,} \end{cases} \tag{4.4}$$

where d_{min} represents the minimum distance between the robot and other humans within a duration of Δt. r_c represents the comfort distance, which indicates the minimum distance at which the agent prefers to keep from others to maintain comfort and safety. It serves as a measure of the personal

space or safety buffer that the agent desires to have while navigating or interacting with its surrounding environment. The comfort distance may vary depending on the specific agent and its operating context, which is set 0.2m in *Chen et al.* (2017b).

4.2.4 State Transition Model

The state transition model $\mathcal{P}(\mathbf{s}_{t+1}^{jn}, \mathbf{s}_t^{jn}|\mathbf{a}_t)$ defines the possible next states and their corresponding transition probabilities given the current state and a specific action taken by the robot. In social-aware robot navigation, the state transition model is typically determined by the environment's dynamics or motion rules, which may depend on factors such as the robot's motion capabilities, the distribution of obstacles in the environment, and the positions of other robots. The state transition model is often unknown in practice, primarily due to the complexity and dynamic nature of the environment, which makes accurate modeling challenging. However, it's important to note that in simulated environments, although the state transition model may start as unknown, it often becomes known over time as the training progresses. Additionally, interactions with other agents, such as pedestrians or other robots, introduce uncertainty into the state transitions.

As a result, model-free reinforcement learning methods, such as Deep Q Network (DQN), are commonly used. These methods do not rely on prior knowledge of the state transition model. Instead, they collect sample data through interactions with the environment and use this data for policy optimization and value function estimation. This enables the robot to learn and optimize navigation strategies, adapting to unknown state transition models and complex navigation environments.

4.2.5 Value Function

The value function V_s or action value function $Q(s, a)$ represents the expected cumulative reward of an agent taking specific actions in a given state. It measures the desirability of various actions in the current state to assist the agent in making optimal decisions. Therefore, in social-aware crowd navigation, the objective is to find the optimal value function $V^*(\mathbf{a}_t^{jn})$ or action-value function $Q^*(\mathbf{a}_t^{jn}, \mathbf{a}_t)$. Based on Section 3.1.2, the optimal value function $V^*(s_t^{jn})$ is:

$$V^*(s_t^{jn}) = \sum_{t'=t}^{T} \gamma^{t' \cdot v_{pref}} R(\mathbf{s}_{t'}^{jn}, \pi^*(\mathbf{s}_{t'}^{jn})), \tag{4.5}$$

where $\gamma \in (0, 1)$ is a discount factor and v_{pref} here is utilized as a normalization factor according to numerical reasons, as otherwise, the value function of slowly moving agents may become very small. Then, the optimal policy $\pi^*(\mathbf{s}_t^{jn})$ can be determined from the optimal value function $V^*(s_t^{jn})$:

$$\pi^*(\mathbf{s}_t^{jn}) = \underset{\mathbf{a}_t}{\operatorname{argmax}}\, R(\mathbf{s}_t^{jn}, \mathbf{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{s}_{t+\Delta t}^{jn}} P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) V^*(\mathbf{s}_{t+\Delta t}^{jn})\, d\mathbf{s}_{t+\Delta t}^{jn}, \tag{4.6}$$

4.3 Simulation Environment

4.3.1 Simulation Setup

In the real world, conducting experiments on social-aware robot navigation may involve complex environments and dynamic interactions, which can lead to collisions or other safety issues among

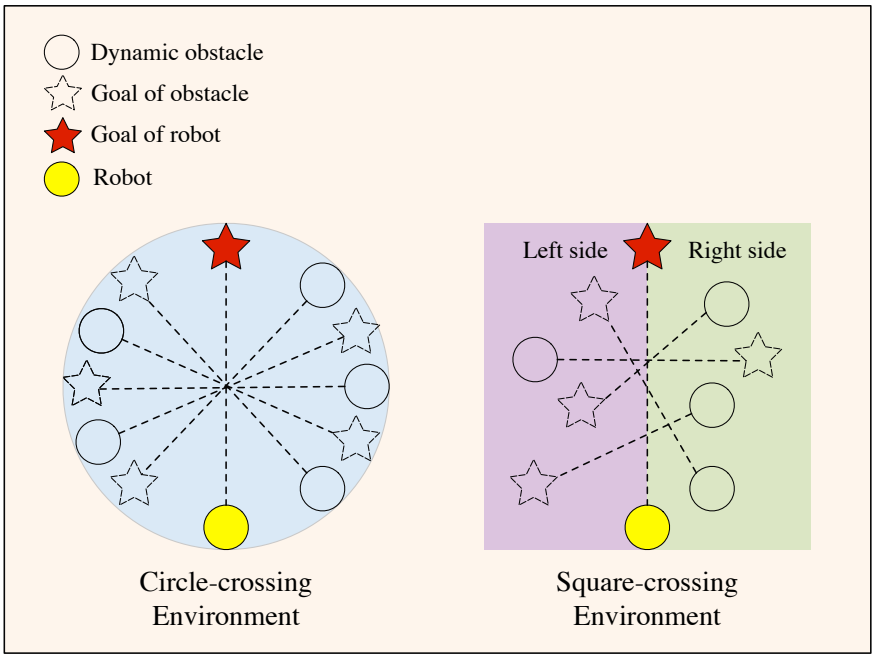


Figure 4.3: Circle-crossing scenario and square-crossing scenario.

robots. Using a simulation platform provides a safe virtual environment, allowing researchers to conduct experiments without real risks. In the simulation platform, researchers have full control over the environment parameters and settings, such as the number and speed of robots, the positions of obstacles, etc., enabling better observation and analysis of the effectiveness of navigation strategies. Moreover, experiments based on the simulation platform can be easily repeated multiple times for the same scenario, verifying the stability and consistency of navigation algorithms, and ensuring the reliability of research results. For diverse navigation scenarios, researchers can quickly test different navigation algorithms and strategies to find optimal navigation solutions. In summary, the simulation platform is designed to offer a safe, controllable, and flexible research environment to accelerate the development and optimization of navigation algorithms and provide valuable guidance and reference for practical applications.

Training and testing of socially aware robot crowd navigation algorithms based on simulation platforms typically involve two key components: the simulation environment and collision avoidance algorithms. The simulation environment creates a virtual setting with robots and other agents (like pedestrians) and simulates sensors (such as LIDAR, cameras, or sonar) that a robot might use to gather information about its surroundings. This environment provides state information ($s^{r/i}$) of the agents (like their positions $p^{r/i}$, speeds $v^{r/i}$) and a standardized data interface, enabling algorithms to read the environmental state (such as the positions $p^{r/i}$ and speeds $v^{r/i}$ of agents) and make decisions (like movement commands $\mathbf{a}$). Meanwhile, the collision avoidance algorithms continuously receive and analyze the state information of agents ($s^{r/i}$) from the simulation environment, calculating and updating decisions to plan the best paths or speeds to avoid collisions, and adapting to the dynamic changes in the environment.

As demonstrated in the work of *Chen et al.* (2019), a simulation environment for social-aware robot navigation can be built based on Python, incorporating Gym (*Brockman et al.*, 2016) and RVO (*Van den Berg et al.*, 2008). These environments simulate complex robot crowd navigation scenarios in the real world, challenging the robot's intelligent decision-making and avoidance abilities in crowded environments. Typically, the environment consists of $n + 1$ agents, including one robot and n humans controlled by certain unknown policies. The humans' behaviors in the environment usually follow the ORCA policy (*Van Den Berg et al.*, 2011), with parameters sampled from a Gaussian distribution to introduce behavioral diversity. The circle-crossing scenarios and square-crossing scenarios are two common simulation scenarios for robot crowd navigation, shown in Figure 4.3. these circle-crossing scenarios are usually set with a radius of 4m for both training and testing, and square-crossing scenarios are with a width of 10m. In the circle-crossing scenarios (square-crossing scenarios) scene, the robot's starting position is set to $(0, -4)$ and its goal position to $(0, 4)$. The initial positions of dynamic humans are randomly placed on the circle (square), and their goals are on the opposite side of the same circle (square). Conversely, static humans have their initial positions randomly scattered within the circle (square) and maintain a velocity $v^i = 0m/s$ during training or testing, ensuring their positions remain unchanged. This configuration can lead to paths where agents converge towards the center, resulting in strong interactions near the center. Additionally, random perturbations are added to the x, y coordinates of the initial and target positions to make the RL algorithm more robust and capable of generalization.

Human groups are typically composed of individuals who are simultaneously walking and stationary. To simulate and understand human group behavior more realistically and facilitate better interaction and navigation with humans, the simulation environment can include both dynamic and stationary humans. Dynamic humans refer to individuals in the simulation environment who exhibit motion behaviors with $v^i \leqslant 1m/s$. These dynamic humans may move over time, such as walking or running, and can influence the navigation behavior of the robot. On the other hand, stationary humans are individuals in the simulation environment who do not exhibit motion behaviors ($v^i = 0m/s$); they are typically fixed at specific positions or remain stationary. Although these stationary humans do not move during navigation, they can still impact the robot's path planning and obstacle avoidance behavior, as the robot needs to avoid collisions with them. By considering both dynamic and stationary humans, the simulation environment can better simulate human group behaviors in the real world, leading to more realistic and effective robot navigation.

In addition, to thoroughly evaluate the effectiveness of the proposed model, existing work usually explores two simulation settings: invisible and visible. In the former, the robot is set not visible to humans. This means that the simulated humans could not react to the robot's actions, such as giving way. This setting allows studying the robot's ability to interact with both humans and other robots without affecting human behavior. In the latter setting, known as the visible setting, human-robot interactions introduce more uncertainty in the environment, resembling real-world situations where robots and humans interact with each other.

4.3.2 Training and Testing

During the training stage, RL algorithms such as DRL are utilized to train the robot's navigation policy. Throughout this process, the robot interacts with the simulation environment, collecting sample data, and continuously updating value functions, policy networks, or other learning parameters

through this interaction. As a result, the robot gradually learns the optimal strategies for different actions in various states to maximize long-term cumulative rewards.

In the testing stage, the trained robot navigation policy is evaluated and validated. During this phase, a set of new or previously unseen test scenarios is commonly employed to assess the robot's navigation performance in unknown environments. These test scenarios may involve different environmental parameters, initial positions, and target locations, allowing for a comprehensive evaluation of the robot's generalization capability and adaptability.

The benefit of conducting both training and testing in simulations is the significant reduction in experiment costs and risks. Within the simulation environment, extensive training and testing can be swiftly performed, parameter adjustments can be made, and various strategies can be explored, thereby accelerating the optimization process of the algorithms

4.3.3 Metrics

Evaluating the protocol for social-aware robot navigation typically involves multiple metrics to assess navigation performance and social interaction effects. Based on *Wang et al.* (2022), some common evaluation metrics include both basic performance metrics and socially conformity-focused metrics.

Basic performance metrics include success rate ("Succ.(%)"), collision rate ("Coll.(%)"), timeout rate ("Timeout(%)"), and average navigation time of the robot ("t_{nav}").

(1) "Succ.(%)": the rate of success cases that the robot reaches the specified target position without collision within the maximum allowed time t_{limit}.

(2) "Coll.(%)": the rate of collision cases that the robot collides with other agents.

(3) "Timeout(%)": the rate of timeout cases that the robot neither reaches the target nor collides with others within the maximum allowed time t_{limit}.

(4) "t_{nav}": the average navigation time for the above success cases which is measured in seconds.

Regarding socially conformity-focused metrics, the typical one is comfort which is defined as the absence of annoyance and stress for humans during interaction with the robot. Specifically, to increase the comfort of surrounding humans, the robot must create a sense of safety. To provide this sense of safety, the robot should avoid entering the personal space of pedestrians, steer clear of their anticipated paths, and plan paths with minimal impact on pedestrians. Therefore, social comfort can be measured by the following metric, which is called discomfort frequency ("Disc. (%)"):

(5) "Disc.(%)": average frequency of duration when the robot's distance to any other agents is less than the comfort distance.

By incorporating these metrics, a more comprehensive evaluation and objective comparison of crowd navigation algorithms can be facilitated.

4.4 Challenges in Social-aware Robot Navigation

In the following chapters, I conduct an in-depth study of the challenges in crowd navigation methods based on reinforcement learning with social awareness. The architecture of the following chapters are shown in Figure 4.4.

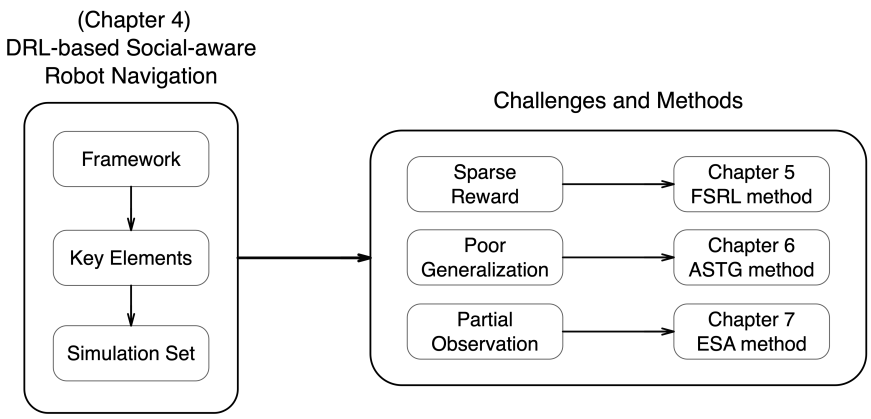


Figure 4.4: Architecture of main content of this thesis.

Firstly, I noticed that the reward functions in these navigation methods are usually sparse. This is due to the complexity and unpredictability of the scenarios, making it difficult to obtain continuous and clear feedback, resulting in a lack of reward signals at most decision points. For instance, successfully navigating to a destination in a crowd without collisions might be the only significant reward event, and before reaching the destination, the robot might receive little or no reward. This leads to the robot's behavior lacking direct feedback to guide its learning most of the time. To address this challenge, in Chapter 5, I used reward shaping techniques to make the reward function denser, thus providing more continuous and intuitive learning signals to help the robot more effectively learn complex crowd navigation strategies, which is Foresight Socially-aware Reinforcement Learning (FSRL) framework. The FSRL introduces additional rewards and penalties for more continuous learning signals. This enhancement allows the robot to adapt more quickly to complex environments and social rules, improving navigation efficiency in crowded settings. The FSRL method predicts future interactions and potential collisions, enabling smoother and more efficient navigation in dynamically changing and challenging environments. This foresight in navigation, combined with enhanced learning signals from the enriched reward function, significantly improves the robot's ability to navigate safely and effectively in socially complex environments.

Secondly, considering the lack of diversity in environmental and crowd configurations in training data, I found that existing DRL-based social-aware methods exhibit limited generalization capability in social-aware crowd navigation problems. This shortfall in generalization primarily stems from the fact that the datasets used during training often fail to comprehensively cover all possible environments and crowd interaction scenarios. For instance, certain specific crowd densities or behavioral patterns might be underrepresented in training data, leading to suboptimal performance when the learned strategies encounter unseen scenarios. To address this issue, Chapter 6 introduces the concept of Attention-based Spatial-Temporal Graphs (ASTG). The core idea of ASTG is to utilize DL technology to capture the complex dynamic relationships between robots, crowds, and the environment. By applying attention mechanisms to spatial-temporal graphs, the model can focus more on important social dynamics and environmental changes. The advantage of this approach lies in its enhanced ability to understand and predict human behavior, thereby improving the robot's navigation strategies in complex environments. Moreover, the introduction of ASTG not only enhances the

model's understanding of the current environmental state but also enables it to predict future crowd dynamics based on past experiences. This capability of remembering past behaviors and predicting future situations significantly boosts the model's generalizability. In this way, ASTG enhances the robot's adaptability and decision-making ability in various scenarios, including those not encountered in the training set.

Furthermore, I particularly emphasize the importance of observation data. In social-aware navigation tasks, the information observed includes not only the physical attributes of the environment but also the behaviors, intentions, and actions of other static/dynamic entities, such as humans. This information is crucial for understanding and predicting the dynamic changes within a crowd. Since navigation in a crowd is decentralized, it means that agents cannot fully understand the strategies and goals of other entities. Thus, there is a significant reliance on observed information to predict and comprehend the trajectories of other agents, enabling safe and efficient navigation in social environments. In the context of DRL, these depth-based observations are key inputs for the policy-learning neural networks. Depth information aids in training models to encapsulate complex navigation patterns, integrate social behaviors, and make context-aware decisions. This is especially important in crowded or dynamic environments, where accurate depth-based observations are critical for adapting to unforeseen circumstances, ensuring safe navigation, and facilitating smooth interactions with human counterparts. Considering the special cases of partially observable environments, Chapter 7 introduces the method of Enhanced Spatial Attention Graph (ESA). The ESA method is designed to address the limitations that traditional observation methods may encounter in complex environments. In partially observable settings, a robot's perception range is limited, and it might not be able to acquire comprehensive information about the environment and crowd. ESA, by conducting more in-depth analysis and utilization of the limited sensory information, enables the robot to make effective navigation decisions even when its field of view is restricted. This not only enhances the robot's navigation performance in partially observable environments but also improves its adaptability and flexibility in the face of constantly changing social dynamics. For instance, ESA can help the robot better identify and predict sudden events or changes in behavioral patterns within a crowd, allowing for swift adjustments when necessary to ensure safety and smooth navigation. Through this approach, ESA greatly enhances the robot's capability to navigate in complex and uncertain environments, opening new possibilities for more intelligent and flexible robot navigation.

CHAPTER 5

Foresight Reinforcement Learning for Social-Aware Robot Navigation

In this chapter, based on *Chen et al.* (2019), I introduce an innovative Foresight Socially-aware Reinforcement Learning (FSRL) framework for achieving collision-free navigation in mobile robots. Prior learning-based methods have shown subpar performance when robots navigate congested and intricate environments, where they must simultaneously evade collisions and accomplish navigation tasks. This contrasts with their better performance in stable and uniform settings. Such discrepancies often lead to reduced success rates and inefficiencies. However, the proposed FSRL approach, empowered by an enriched reward mechanism, possesses foresight capabilities. It takes into account not just the present human-robot interactions to prevent immediate collisions, but also anticipates future social interactions to sustain appropriate distancing. Moreover, the proposed FSRL method introduces efficiency parameters, which notably curtail navigation time. To validate the efficiency and effectiveness of the FSRL method in more lifelike and challenging simulated scenarios, I conduct comparative experiments with three methods across three progressively demanding environments.

Contents

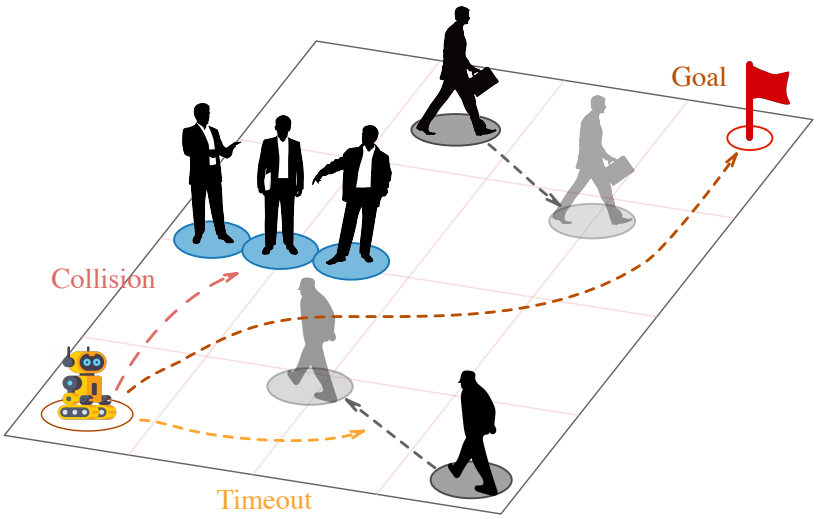


Figure 5.1: In a realistic scenario, a mixture of dynamic (grey) and stationary (blue) objects coexist. Unlike the overly simplified scenarios in prior methods, which consider only dynamic objects, the presence of both types results in the more frequent formation of traps and blind spots that persist over time. Consequently, the robot is more prone to being ensnared within the crowd.

5.1 Introduction

Designing navigation algorithms for mobile robots is of significant importance for enabling robots to autonomously navigate and move in human environments, especially in complex and unknown settings. While there have been successful efforts in the field of mobile robot navigation, the challenge remains in designing navigation algorithms that consider social interactions, such as following social norms, giving way, and maintaining comfortable distances, when robots interact with humans.

As mentioned in Section 2.2, traditional robot navigation methods often treat moving agents as static obstacles (*Borenstein and Koren*, 1989, 1990; *Borenstein et al.*, 1991) or react to them one step ahead (*Van den Berg et al.*, 2008; *Van Den Berg et al.*, 2011; *Snape et al.*, 2011), leading to shortsighted, unsafe, and unnatural behaviors. To navigate densely populated environments in a socially normative way, robots need to understand human behaviors and follow cooperative rules (*Fong et al.*, 2003; *Kruse et al.*, 2013; *Kretzschmar et al.*, 2016). As a result, some works (*Everett et al.*, 2018; *Chen et al.*, 2019; *Liu et al.*, 2021) combine deep learning (DL) and reinforcement learning (RL) for socially aware robot navigation. In these methods, an effective policy is learned that implicitly models agents' complex interactions and cooperation. Specifically, deep neural networks (DNN) are used to approximate value functions, and optimal actions are chosen based on these value functions. Despite the superiority of learning-based methods over non-learning approaches, these methods still face limitations when navigating mobile robots in more realistic and complex environments.

A major issue in previous robot navigation works is the oversimplification of the environment by considering only avoidance of dynamic objects. In such cases, robots tend to adopt a simple strategy

of avoiding all dynamic objects to reach their goals. Firstly, these previous methods can be shortsighted, as they make decisions based solely on current interactions, ignoring future possibilities, and limiting their navigation capabilities in complex crowds. Additionally, these methods implicitly penalize inefficient strategies. Hence, they often lack the ability to navigate effectively in complex crowds, often inefficiently bypassing the crowd to avoid collisions. In more realistic scenarios with stationary objects, these issues are exacerbated, as shown in Figure 5.1.

Another limitation in many existing studies is the assumption that robots have holonomic kinematics. This simplifies robot navigation tasks as robots can move freely in any direction at any time. Holonomic autonomous robots can execute large-angle rotations to smoothly reach their destinations along non-smooth navigation paths. However, most robots in real-life scenarios are non-holonomic (*Gao et al.*, 2021), such as delivery and service robots. This restricts robots to move smoothly, exacerbating the aforementioned challenges. For example, a trapped non-holonomic robot may struggle to escape.

Given these limitations, I propose a novel deep reinforcement learning (DRL) approach for collision-avoiding robot navigation. Building upon the framework proposed in *Chen et al.* (2019), the proposed FSRL approach enhances the RL method by predicting potential future interactions based on the current state, endowing the robot with foresight, as depicted in Figure 5.2. The robot can make decisions by considering both current and predicted interactions, allowing it to react earlier before potential collisions occur. Therefore, smooth movement between consecutive time steps is feasible regardless of the environment's complexity. Furthermore, the proposed FSRL approach explicitly accounts for stationary objects. Specifically, unlike methods that neglect different kinematics of objects, the proposed FSRL approach applies different constraints to obstacles based on their motion status. The FSRL approach can detect potential unsafe situations and take action preemptively, as depicted in Figure 5.2. Additionally, the FSRL approach incorporates constraints on navigation time, enabling the learning of more efficient strategies. Due to its ability to predict future collisions (foresight), I term the FSRL approach Foresight Socially Aware Reinforcement Learning (FSRL) algorithm.

In summary, the core of the FSRL approach lies in its ability to predict potential future collisions and take actions to avoid them, significantly enhancing the quality and efficiency of navigation. Furthermore, by explicitly considering the constraints on navigation time, the FSRL method improves the efficiency of strategy learning. Lastly, I conducted extensive experimental validation of this method in complex environments, including dynamic and stationary humans. It achieves optimal performance across different environments, demonstrating its effectiveness and robustness.

5.2 Foresight Socially Aware Reinforcement Learning

Based on the problem formulation outlined in Section 4.2, where the joint state is denoted as $\mathbf{s}_t^{jn} = [\mathbf{s}_t^r, \mathbf{s}_t^h]$ and $\mathbf{s}_t^r$ and $\mathbf{s}_t^h$ represent the robot state and the observable state of humans at time t respectively. The objective is to find the optimal navigation policy $\pi^* : \mathbf{s}_t^{jn} \mapsto \mathbf{a}_t$, which assigns actions to each state, aiming to maximize the expected cumulative reward of future actions until the

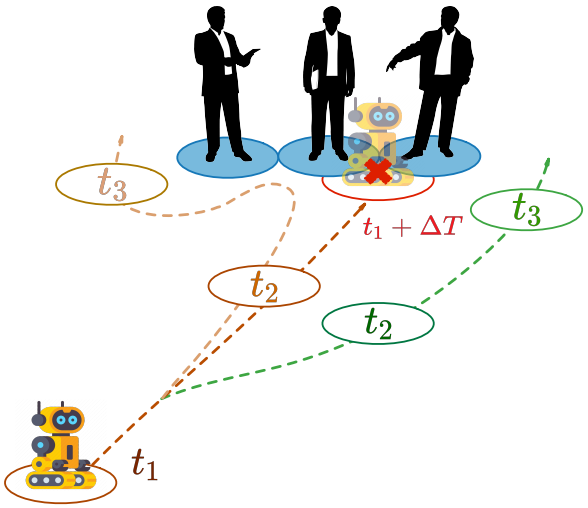


Figure 5.2: Previous methods (depicted in brown) only identify collisions at time t_2, resulting in abrupt maneuvers to evade collisions, a shortsighted strategy. Conversely, the FSRL approach (illustrated in green) can anticipate potential collisions as early as time t_1 within the time window Δt (highlighted in red), allowing for preemptive actions to smoothly avoid collisions, reflecting a foresighted approach. This foresight is especially critical for nonholonomic robots, as they encounter difficulties extricating themselves from traps.

goal is achieved. The value function network and optimal policy are formulated as follows:

$$V^*(s_t^{jn}) = \sum_{t'=t}^{T} \gamma^{t' \cdot v_{pref}} R(\mathbf{s}_{t'}^{jn}, \pi^*(\mathbf{s}_{t'}^{jn})), \tag{5.1}$$

$$\pi^*(\mathbf{s}_t^{jn}) = \underset{\mathbf{a}_t}{\operatorname{argmax}}\, R(\mathbf{s}_t^{jn}, \mathbf{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{s}_{t+\Delta t}^{jn}} P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) V^*(\mathbf{s}_{t+\Delta t}^{jn})\, d\mathbf{s}_{t+\Delta t}^{jn}, \tag{5.2}$$

Building upon the SARL framework from *Chen et al.* (2019), I advance the enhancement of safety and efficiency in navigation by refining the sparse reward function. Firstly, I provide an overview of the SARL framework in Section 5.2.1, followed by the introduction of the fundamental sparse reward function in Section 5.2.2. In Section 5.2.3, I present the predictive constraint reward function. Distinct foresight penalties are devised for diverse obstacle motion statuses, encompassing dynamic and stationary scenarios. These penalties serve to penalize potential future collisions or discomfort effectively. Moreover, in Section 5.2.4, I introduce a reward function that operates under time constraints, contributing to the enhancement of navigation efficiency.

5.2.1 Social Attention-based Deep Reinforcement Learning method (SARL)

SARL achieves socially compliant navigation by computing relative importance and encoding the collective impact of neighboring agents. SARL consists of three modules: the Interaction Module, the Pooling Module, and the Planning Module.

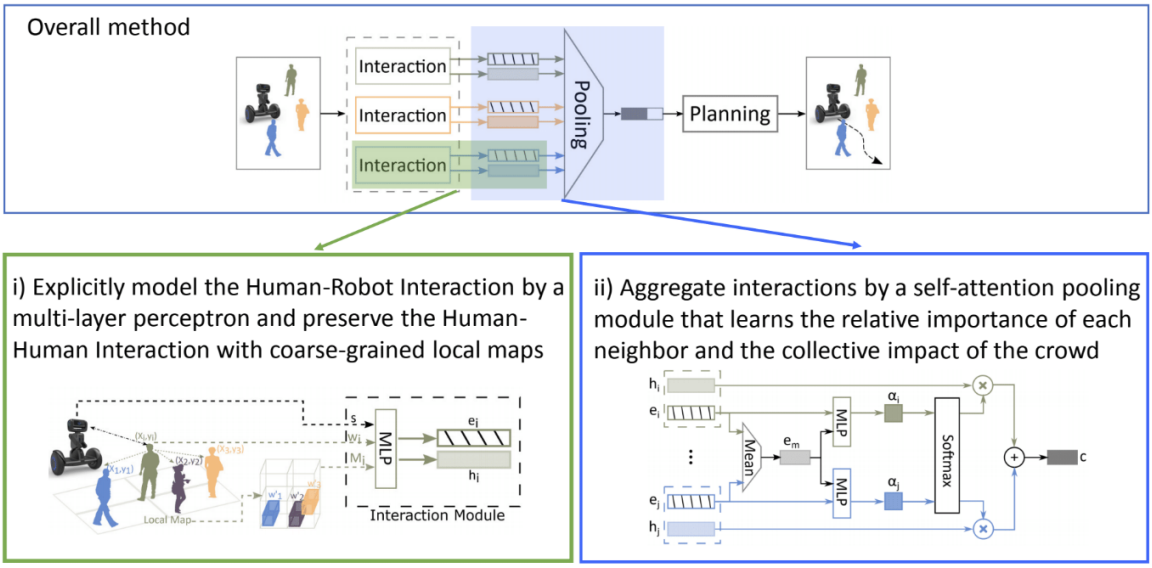


Figure 5.3: Overview of SARL method. (*Chen et al.*, 2019)

Based on the state space shown in Section 4.2.1, the Interaction Module (Figure 5.3 i)) explicitly models human-robot interactions h_i by a multi-layer perceptron (MLP) and encodes these interactions through coarse-grained local maps:

$$e^i = \phi_e(s^r, s^i, M^i; W_e), \tag{5.3}$$

$$h^i = \psi_h(e^i; W_h), \tag{5.4}$$

where e^i is an embedding vector for the states of agents (s^r and s^i) and the map tensor M^i. h^i is the pairwise human-robot interaction feature. ϕ_e, ψ_h are different MLP with ReLU nonlinear activations and W_e, W_h are the network weights. $M^i(a, b, :)$ is a $L \times L \times 3$ map tensor centered at each human i with a neighborhood of size L, which is used to encode the presence and velocities of neighbors.

$$M^i(a, b, :) = \sum_{j \in N^i} \delta_{ab}[x^j - x^i, y^j - y^i] w'^j \tag{5.5}$$

where $w'^j = (v_{x^j}, v_{y^j}, 1)$ is a local state vector for human j, and $\delta_{mn}[x^j - x^i, y^j - y^i]$ is an indicator function which equals to 1 only if the relative position $(\Delta x, \Delta y)$ is located in the cell (a, b), N^i is the set of neighboring humans around the ith person. The Interaction Module jointly models both human-robot and human-human interactions within the deep reinforcement learning framework.

Subsequently, the Pooling Module aggregates interactions into fixed-length embedding vectors c through self-attention mechanisms. Specifically, the Pooling Module employs MLPs to compute attention scores α^i for each individual based on their embedding vectors e^i and the average embedding vector e^m.

$$e^m = \frac{1}{n} \sum_{k=1}^{n} e^k, \tag{5.6}$$

$$\alpha^i = \psi_\alpha(e^i, e^m; W_a), \tag{5.7}$$

where ψ_α is the MLP with nonlinear ReLU activations and weights W_a. The final joint representation c is a weighted sum of pairwise interactions.

$$c = \sum_{i=1}^{n} \text{softmax}(\alpha^i) h^i, \tag{5.8}$$

Lastly, the Planning Module is utilized to present the value v of joint states (s^r, c) between the robot and the crowd for social navigation as shown:

$$v = f_v(s^r, c; W_v) \tag{5.9}$$

where $f_v(\cdot)$ is an MLP which has ReLU nonlinear activations and the network weights are W_v. The value network is trained by the temporal-difference method (Section 3.1.3) with standard experience replay and fixed target network techniques (Section 3.3.1.1) to accurately approximate the optimal value function V^*.

5.2.2 Sparse Reward

As illustrated in Section 4.2.3, the common sparse reward function, as introduced in prior works such as *Chen et al.* (2019, 2017b), relies on the decision at each time step t, which is only determined by the present state. The formulation of the sparse reward function $R_c(\mathbf{s}_t^{jn}, \mathbf{a}_t)$ is as follows:

$$R_c(\mathbf{s}_t^{jn}, \mathbf{a}_t) = \begin{cases} -0.25 & \text{if } d_{min} < 0 \\ 0.5 * (d_{min} - r_c) & \text{else if } d_{min} < r_c \\ 1 & \text{else if } \mathbf{p} = \mathbf{p}_g \\ 0 & \text{otherwise,} \end{cases} \tag{5.10}$$

where $d_{min} = min\{d^i - r - r^i\}$ represents the shortest distance between the robot and the other humans at time t. The term $r_c = 0.2m$ signifies the comfort distance, defined as the minimal distance ensuring comfort between humans and the robot.

5.2.3 Foresight Reward Augmentation

One reason for the poor performance of prior approaches in intricate environments is often attributed to their susceptibility to entrapment, largely due to their inability to anticipate collisions in the future. This issue is exacerbated when adopting more realistic nonholonomic kinematics, which restricts sharp large-angle movements. To address this, I propose a foresighted approach that anticipates actions in advance.

First, the prediction involves projecting the movement of both the robot and the humans forward in time, considering their current velocities and directions $v^{r/i} = [v_x^{r/i}, v_y^{r/i}]$ to anticipate the future locations of the robot and the i-th human

$$p_{\Delta T_{future}}^{r/i} = p^{r/i} + v^{r/i} * \Delta T_{future} \tag{5.11}$$

where ΔT_{future} is a timeframe. Essentially, the robot uses its current state information to estimate where it and the dynamic humans will be positioned after several seconds have elapsed. This is due to

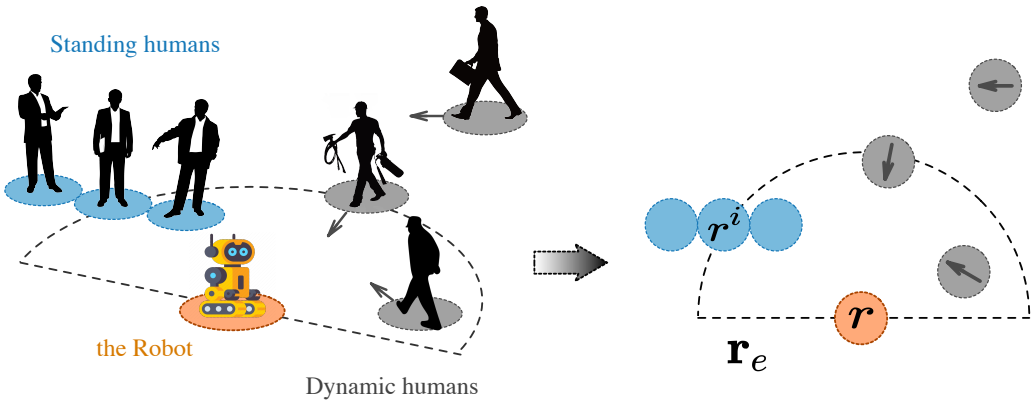


Figure 5.4: An illustration of the effective range $\mathbf{r}_e$ of the robot. The grey and blue circles respectively represent the dynamic and stationary humans, each having a radius of r^i, with arrows indicating their movement direction.

that although the constrained rotational capacity of nonholonomic kinematics, the robot's movements within a limited timeframe exhibit smoothness and can be approximated as linear motion. Thus, this characteristic enables us to estimate the positions of agents at a certain time in the future based on their current state to gain insights into future scenarios. Then, I introduce a foresight penalty, denoted as R_f, that discourages potential future collisions. Moreover, in a realistic environment, both dynamic and stationary objects coexist, necessitating differentiation – unlike previous methods. The FSRL method categorizes them according to their velocities and handles them distinctively, treating objects with non-zero velocities as dynamic and those with zero velocity as static. This allows for a tailored approach where dynamic objects, which may require real-time responsiveness due to their changing state, are handled differently from static objects, which remain constant over time. By doing so, FSRL optimizes the learning strategy.

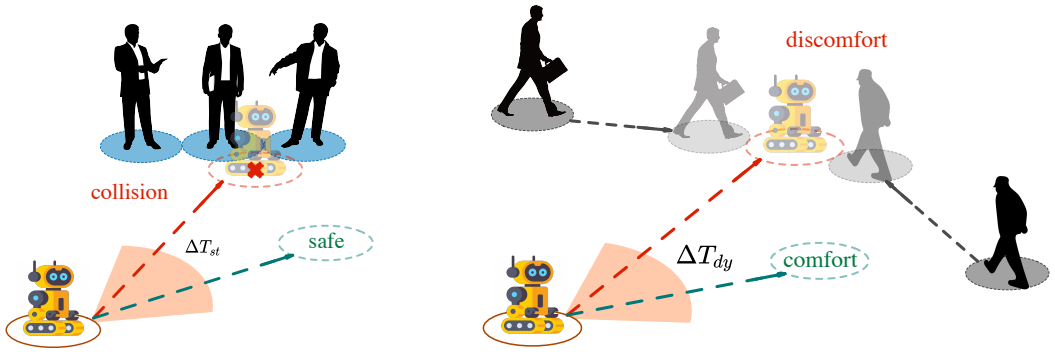
The foresight penalty R_f is now formulated as:

$$R_f(\mathbf{s}_t^{jn}, \mathbf{a}_t) = R_{dy}(\mathbf{s}_t^{jn}, \mathbf{a}_t) + R_{st}(\mathbf{s}_t^{jn}, \mathbf{a}_t), \tag{5.12}$$

where R_{dy} and R_{st} correspond to dynamic and stationary objects, respectively, and are introduced in the subsequent context.

Taking into account that objects located farther away have negligible influence on the current decisions, I restrict the focus to objects within the effective range of $\mathbf{r}_e = 2m$, as illustrated in Figure 5.4.

Regarding stationary objects within the effective range $\mathbf{r}_e$, the robot's viable option is to navigate to bypass them, given that these objects remain stationary and will not disappear over time. If the robot identifies the likelihood of potential collisions with stationary objects within a timeframe of $\Delta T_{st} = 2s$ based on its current decision, a penalty is enforced. This penalty is directly proportional to the count of potential collisions with stationary objects, revealing the degree of congestion, and is



(a) A scenario with stationary humans (b) A scenario with dynamic humans

Figure 5.5: (a) It shows a scenario with stationary humans. The robot can anticipate and prevent collisions by identifying potential interactions with stationary objects within ΔT_{st} shown in the green line. This proactive behavior is incentivized by the reward factor R_{st}. (b) It shows a scenario with dynamic humans. According to the comfortable distance r_c within ΔT_{dy}, the robot chooses to detour the crowd (the green line) instead of going through it (the red line), which would annoy humans. This is achieved by the reward component R_{dy}.

mathematically represented as follows:

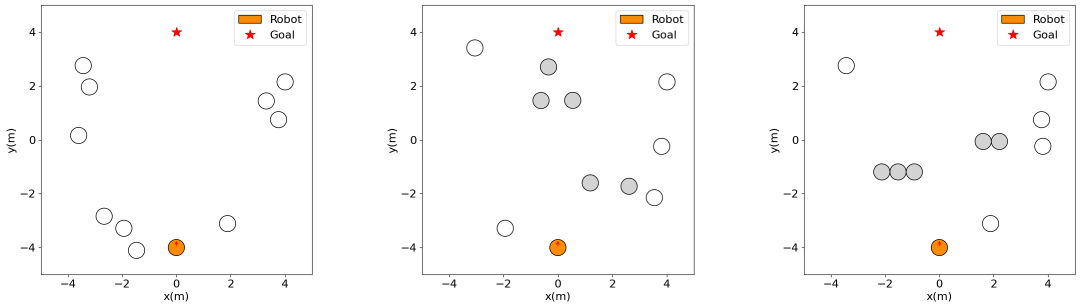
$$R_{st}(\mathbf{s}_t^{jn}, \mathbf{a}_t) = -\alpha * \frac{N_{col}}{N_{static}}, \tag{5.13}$$

where $\alpha = 0.15$. Here, N_{static} refers to the count of stationary humans located within the robot's effective range $\mathbf{r}_e$. Within the timeframe ΔT_{st}, a potential collision is considered to have occurred if the shortest distance $d_{\Delta T_{st}}$ between the robot and any stationary objects is less than zero, and N_{col} denotes the number of such potential collisions detected within the effective range $\mathbf{r}_e$, where $d_{\Delta T_{st}} = min\left\{||p^r_{\Delta T_{st}} - p^i_{\Delta T_{st}}||_2 - r - r^i\right\}$ with setting ΔT_{future} as ΔT_{st} in Equation 5.11. The implementation of this reward penalty encourages the avoidance of potential collisions, while the forward-looking nature of this penalty encourages the robot to take proactive actions in order to prevent sharp movements. This contributes to a smoother navigation path, as depicted in Figure 5.5(a).

Regarding dynamic objects within the effective range $\mathbf{r}_e$, they easily tend to cluster together, leading to collisions. Unlike the previous scenario with stationary objects, these clusters disperse over time. Consequently, the robot faces the decision of whether to wait and then proceed to go through or bypass the clusters. Moreover, in real-world applications, the robot and humans should not disrupt each other. Hence, beyond achieving successful navigation, the perceived navigation quality is also a concern. Given that dynamic humans adjust their positions for comfort, an additional penalty is imposed if the robot disrupts people by breaching the comfortable distance. This can be formulated as:

$$R_{dy}(\mathbf{s}_t^{jn}, \mathbf{a}_t) = \beta * (d_{\Delta T_{dy}} - r_c), \tag{5.14}$$

where $\beta = 0.5$. Here, $d_{\Delta T_{dy}}$ represents the minimum distance between the robot and dynamic humans after a time period of $\Delta T_{dy} = 1s$, where $d_{\Delta T_{dy}} = min\left\{||p^r_{\Delta T_{dy}} - p^i_{\Delta T_{dy}}||_2 - r - r^i\right\}$.



(a) Env.1 (b) Env.2 (c) Env.3

Figure 5.6: Three increasingly challenging simulation environments. Solid grey circles depict static stationary agents, while hollow circles represent dynamic objects. (a) Environment 1 (Env.1) consists of 10 dynamic objects. (b) Environment 2 (Env.2) features 5 randomly positioned stationary objects, while the remaining are dynamic objects. (c) Environment 3 (Env.3) is characterized by two clusters of barriers, each containing 2 and 3 stationary objects respectively, alongside other dynamic objects. In comparison to the preceding environment, the subsequent two environments present scenarios where robots are more prone to entrapment and face increased difficulty in maneuvering around different obstacles.

$\Delta T_{dy} = 1s$ is half the duration of ΔT_{st} in R_{st} due to the mobility of dynamic objects, necessitating a shorter time window. Through this reward penalty, the robot is encouraged to avoid aggressive navigation, effectively reducing instances of disturbing people, as illustrated in Fig. 5.5(b).

5.2.4 Efficiency Reward Augmentation

In addition to assessing the navigation's success rate, I also incorporate efficiency into the reward framework, a factor often overlooked in prior approaches. To accomplish this, I introduce a navigation time constraint and encourage the adoption of efficient strategies by:

$$R_t(\mathbf{s}_t^{jn}, \mathbf{a}_t) = \begin{cases} -0.1 * \frac{t}{t_{limit}} & \text{if } \mathbf{p} = \mathbf{p}_g \\ -0.2 & \text{else if } t >= t_{limit}, \end{cases} \tag{5.15}$$

where t_{limit} is the maximum navigation time, which is set as $t_{limit} = 25s$ in experiments. By adding this term, detours will be discouraged.

5.2.5 Augmented Reward Function

Together, the whole reward function R is defined as:

$$R(\mathbf{s}_t^{jn}, \mathbf{a}_t) = R_c(\mathbf{s}_t^{jn}, \mathbf{a}_t) + R_f(\mathbf{s}_t^{jn}, \mathbf{a}_t) + R_t(\mathbf{s}_t^{jn}, \mathbf{a}_t). \tag{5.16}$$

5.3 Experiments

5.3.1 Simulation Setup

As mentioned in Section 4.3.1, I construct the simulation environment using Gym (*Brockman et al.*, 2016) and RVO (*Van den Berg et al.*, 2008) as outlined in *Chen et al.* (2019). The environment consists of 1 single robot and 10 dynamic/stationary humans. The objective is to develop an optimal strategy for the robot to navigate to its destination efficiently and safely. In experiments, I set circle-crossing scenarios where the robot's starting position is set to $(0, -4)$ and its goal position to $(0, 4)$. However, this kind of scenario is overly simplistic, allowing easy bypassing of obstacles, which deviates from real-world complexity. Consequently, it fails to comprehensively assess the navigation capabilities of algorithms.

To comprehensively demonstrate the navigation capabilities, I train and test all methods on *three increasingly challenging environments*, as depicted in Figure 5.6.

- **Environment 1 (Env.1)** replicates the conditions of prior studies, featuring only dynamic agents.
- **Environment 2 (Env.2)** introduces 5 dynamic and 5 stationary agents. The positions of the stationary agents are randomly distributed throughout the entire environment. Notably, this setup more easily gives rise to the formation of traps and blind spots.
- **Environment 3 (Env.3)** represents an intensified version of Environment 2. The only difference lies in the fixed positioning of the stationary agents, resulting in two permanent traps and blind spots that persist over time.

I conduct a comparative analysis involving the proposed FSRL method and three state-of-the-art approaches: ORCA (*Van Den Berg et al.*, 2011), a representative reaction-based method; LSTM-RL (*Everett et al.*, 2018) and SARL (*Chen et al.*, 2019), both being DRL methods. Meanwhile, I also perform experiments in two settings: invisible and visible setting, which are discussed in Section 4.3.1. Thus, for the three environments and the two settings for each environment (6 environments in total), I train the unicycle rotation-constrained robot with all four algorithms separately and then obtain the corresponding test results. To reduce the influence of randomness, 3 experiments are conducted for each method on each environment and report the average performance.

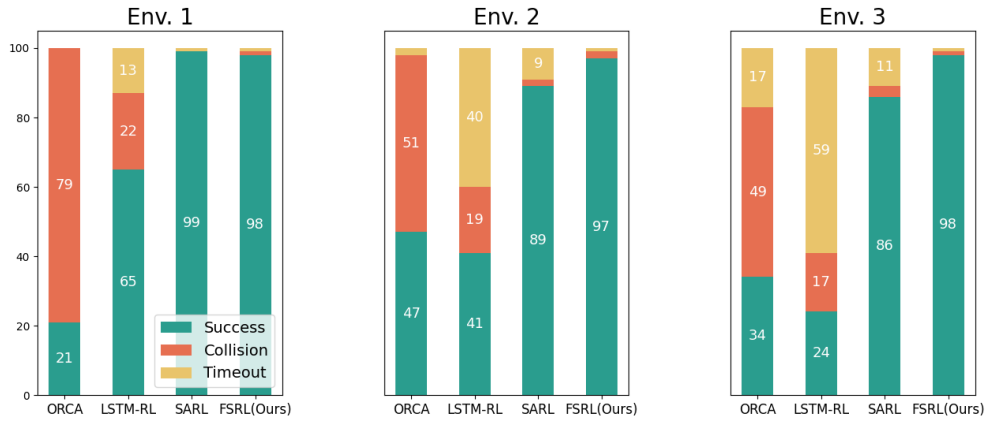
5.3.2 Training and Testing

Training:

To ensure a fair comparison, all other parameters are set the same as in *Chen et al.* (2019), except for the augmented reward function. This can reveal FSRL's capacity for performance enhancement.

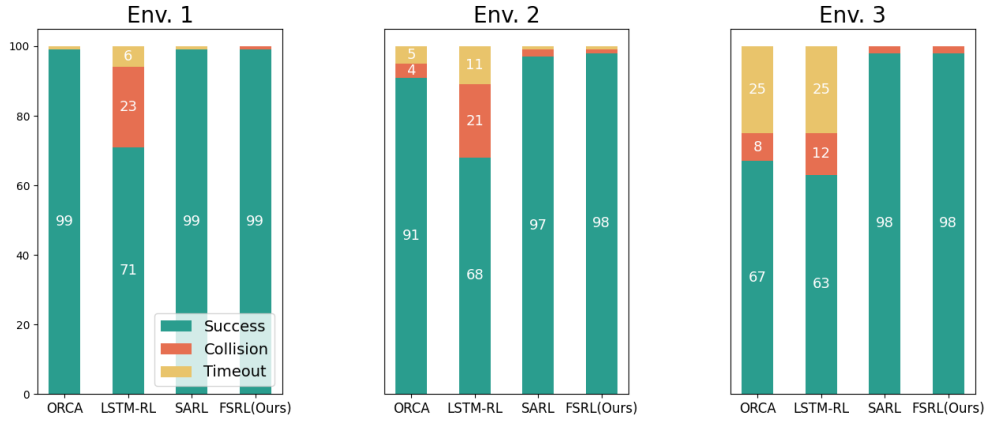
Testing:

For each scenario, model evaluation involves 500 random test cases. Distinct random seeds are assigned for various episodes. Consequently, in a given test episode, the scenarios for all methods feature the same environment's state, like the start and goal positions of robots and humans. However, there are variations in the humans' start and end positions due to re-randomization across episodes. The metrics used in the quantitative evaluation are presented in Section 4.3.3.



(a) Env.1 (b) Env.2 (c) Env.3

Figure 5.7: Quantitative evaluation on three environments under the **invisible** setting. Out of the four methods, FSRL consistently exhibits the highest success rate and the lowest collision rate. In comparison to the other three approaches, LSTM-RL (*Everett et al.*, 2018) displays the highest rate of timeout cases. Meanwhile, ORCA (*Van Den Berg et al.*, 2011) has the most collision cases. For SARL (*Chen et al.*, 2019), the rate of timeout cases increases significantly with the growing challenge of the environment (from Env. 1 to Env. 3).



(a) Env.1 (b) Env.2 (c) Env.3

Figure 5.8: Quantitative evaluation on three environments under the **visible** setting. FSRL performs as well as the SARL (*Chen et al.*, 2019) method. Meanwhile, it outperforms the other two methods, namely ORCA (*Van Den Berg et al.*, 2011) and LSTM-RL (*Everett et al.*, 2018).

5.3.3 Comparison with State-of-the-art Methods

I present a performance comparison between FSRL and state-of-the-art approaches in Figures 5.7 and 5.8, and Tables 5.1 and 5.2, considering both invisible and visible settings. It's evident that FSRL outperforms other methods in almost all scenarios.

Methods	Nav Time (invisible)				Nav Time (visible)			
	Env.1	Env.2	Env.3	AVG	Env.1	Env.2	Env.3	AVG
ORCA (*Van Den Berg et al.*, 2011)	12.49	11.98	12.04	12.14	11.99	10.97	10.74	11.23
LSTM-RL (*Everett et al.*, 2018)	15.52	14.12	12.81	14.15	11.01	10.96	12.32	11.29
SARL (*Chen et al.*, 2019)	12.25	11.39	11.15	11.60	9.85	10.33	10.01	10.06
FSRL(*ours*)	10.90	10.92	11.69	11.04	11.08	10.32	11.60	11.00

Table 5.1: "Nav. Time" quantitative results in 3 environments under invisible and visible settings. Under the invisible setting, FSRL exhibits the shortest average navigation times, whereas, under the visible setting, it requires slightly longer navigation times, averaging less than 1 second.

Methods	Disc(%) (invisible)				Disc(%) (visible)			
	Env.1	Env.2	Env.3	AVG	Env.1	Env.2	Env.3	AVG
ORCA (*Van Den Berg et al.*, 2011)	0.39	0.38	0.43	0.40	0.41	0.41	0.45	0.42
LSTM-RL (*Everett et al.*, 2018)	0.07	0.28	0.16	0.17	0.11	0.35	0.39	0.20
SARL (*Chen et al.*, 2019)	0.01	0.06	0.06	0.04	0.09	0.11	0.10	0.10
FSRL(*ours*)	0.01	0.07	0.04	0.04	0.06	0.08	0.07	0.07

Table 5.2: "Disc. (%)" quantitative results in 3 environments under invisible and visible settings. Under both settings, FSRL has the lowest discomfort rate on average.

The success, collision, and timeout rates are visualized in Figure 5.7 and Figure 5.8. Whether in the visible or invisible setting, FSRL consistently attains the highest success rates and the lowest collision and timeout rates. The learning-based methods, like LSTM-RL or SARL, typically rely only on the present states for making optimal decisions, rendering them short-sighted. This limitation becomes pronounced with nonholonomic robots that are constrained in rotations. As they are unable to rotate widely when encountering obstacles, such robots might experience prolonged navigation times, a greater likelihood of entrapment, or even mission failure. As depicted in Figure 5.7 and Figure 5.8, these methods exhibit more instances of timeouts and collisions. As anticipated, FSRL outperforms the others due to its foresight capability, which not only takes into account the present state but also predicts future situations. Consequently, the nonholonomic robot can proactively take appropriate actions to ensure smooth and safe movement.

In each environment, the deep V-learning algorithm (*Chen et al.*, 2019) referred in Section 3.3.1.1 is utilized along with the proposed augmented reward function (Equation 5.16) to train the FSRL method. The training process comprises two stages. Firstly, the policy is initialized via imitation learning (Section 2.2.3), involving the collection of 3000 demonstration episodes based on the ORCA (*Van Den Berg et al.*, 2011) policy. This initial policy is trained by employing the Adam optimizer (*Kingma and Ba*, 2014) for 50 epochs, with a learning rate of 0.01. Subsequently, the policy is enhanced through the utilization of the proposed augmented reward functions, employing a learning rate of 0.0001. Additionally, the discount factor γ (Section 3.1.1) is set to 0.9. In terms

of exploration, greedily, the exploration rate (Section 3.1.4) decreases linearly from 0.5 to 0.1 during the initial 4000 episodes and then maintains a level of 0.1 for the following 6000 episodes. FSRL assumes nonholonomic kinematics for the robot, implying constraints on the rotation angle. The robot's velocity is discretized into 5 exponential speeds within the range of (0, V_{pref}], accompanied by 10 headings distributed evenly between $[-\pi/8, \pi/8]$.

5.3.3.1 Invisible Robot

In the **invisible** setting, as expected, the ORCA method experiences significant failures due to the violation of the reciprocal assumption which is introduced in Section 2.2.3. Interestingly, the ORCA robot exhibits notably improved performance under Env.2 and Env.3 compared to Env.1. This observation can be attributed to the fact that non-learning-based approaches like ORCA are more sensitive to dynamic humans. Consequently, a reduced presence of dynamic humans correlates with improved performance.

In contrast, there is a noticeable decline in the performance of learning-based methods (LSTM-RL, SARL) when transitioning from Env.1 to Env.2 and Env.3. This can be attributed to their struggle in acquiring a robust representation for a complex environment containing both static and dynamic objects, in contrast to a simpler scenario where only one type of object exists. This highlights that prior learning-based methods are better suited for straightforward scenarios but struggle in more complex environments. Furthermore, the LSTM-RL method exhibits a more cautious behavior, resulting in a sharp increase in timeout cases (Figure 5.7) and longer navigation times in successful scenarios (Table 5.1). In contrast, FSRL demonstrates the shortest navigation times and the lowest discomfort rates. These outcomes underscore the foresighted nature of FSRL, enabling a more optimal balance between safety and efficiency.

5.3.3.2 Visible robot

Under the **visible** setting, even though the robot can navigate through the crowd more easily due to the cooperative behaviors of humans, it needs to be more adept at understanding human interactions to deal with the increased uncertainty. This includes scenarios where humans exhibiting cooperative behavior, like give-way behavior, might enter the robot's comfort zone. Moreover, due to the bidirectional interaction between humans and the robot, the robot might take riskier actions in order to achieve better navigation outcomes. Consequently, in contrast to the invisible setting, the performance of all methods generally improves except for the discomfort rate.

For LSTM-RL and ORCA methods, their shortsight leads them to disregard stationary agents in Emv.2 and Env.3. Consequently, their navigation tends to be both unsafe and inefficient, resulting in lower success rates, longer navigation times, and higher discomfort rates. As illustrated in Figure 5.8, FSRL achieves comparable performance to the SARL method. While the navigation time of FSRL takes slightly longer (averaging less than 1 second), it effectively reduces the discomfort rate by 3% through safer robot movements (Table 5.2) in comparison to SARL.

The performance metrics of FSRL remain consistent in both visible and invisible scenarios. In a comprehensive comparison to previous methods, FSRL emerges as more effective, efficient, and robust.

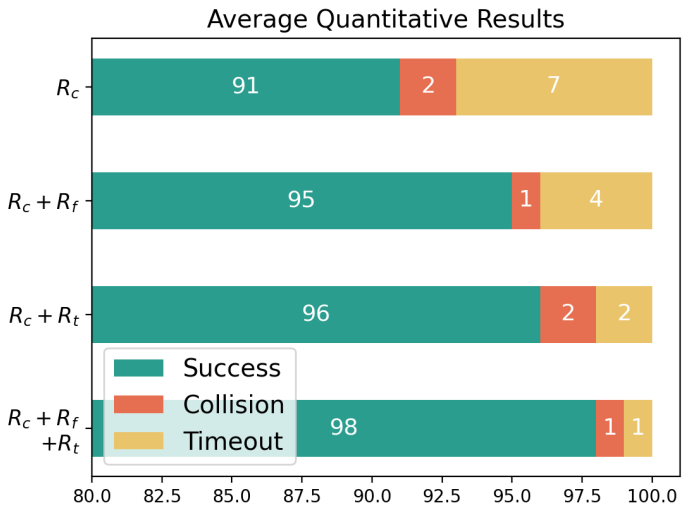


Figure 5.9: Average quantitative results under three environments of ablation experiments. (Invisible setting)

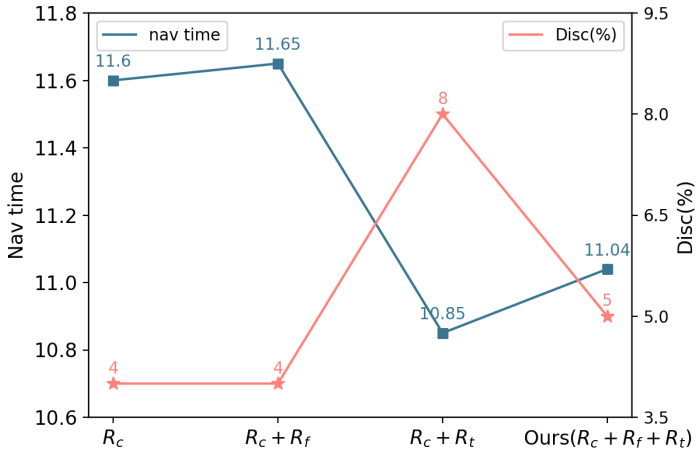
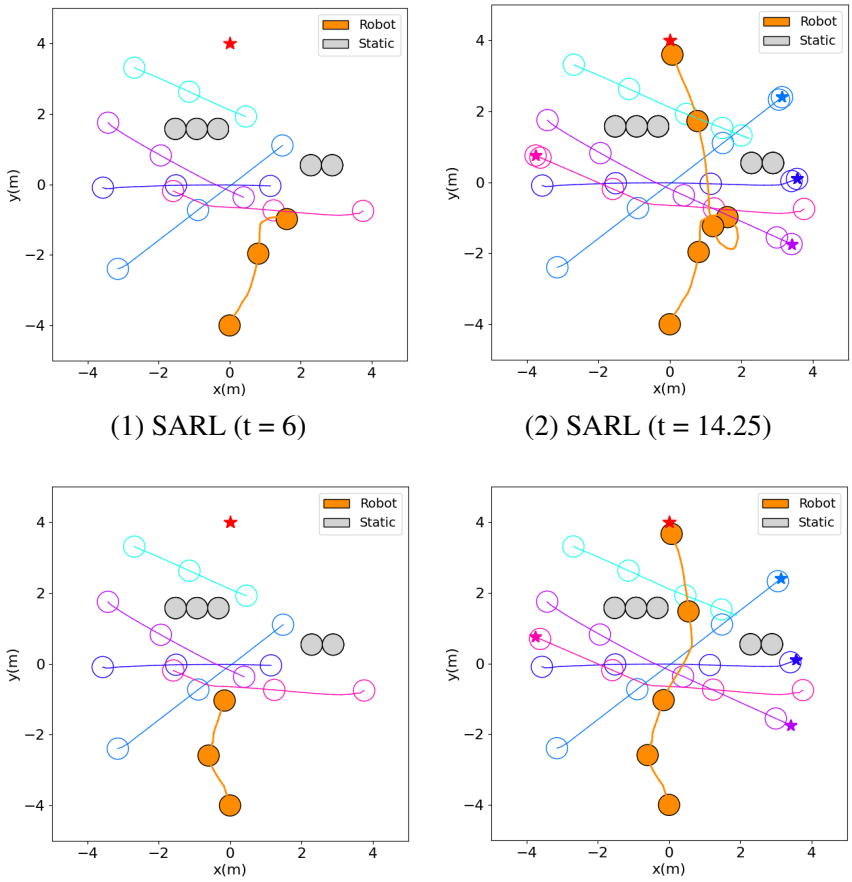


Figure 5.10: Average "Nav. Time" and "Disc(%)" results under three environments of ablation experiments. (Invisible setting)

5.3.4 Ablation Study

I present the impact of each component of proposed novel reward R function in Figure 5.9 and 5.10. These components include R_c, the reward that considers only the current state (Equation 5.10); R_f, the foresight penalty (Equation 5.12); and R_t, the efficiency constraint (Equation 5.15). For clarity, I average the quantitative results across three environments for comparison.

In Figure 5.9, it is evident that both the foresight penalty (R_f) and the efficiency constraint (R_t) significantly enhance the success rate (Succ.(%)). Notably, the improvement is more pronounced



(1) SARL (t = 6) (2) SARL (t = 14.25)

(3) FSRL (t = 6) (4) FSRL (t = 11.50)

Figure 5.11: Qualitative results. In a complex environment containing both stationary (grey solid circles) and dynamic (hollow circles of other colors) objects, the proposed method (FSRL) enables the robot (orange solid circle) to navigate more smoothly and efficiently compared to SARL*Chen et al.* (2019).

in complex environments (Env.2 and Env.3). Additionally, the efficiency constraint (R_t) primarily contributes to a reduction in navigation time, as shown in Figure 5.10. This implies that improved efficiency leads to the avoidance of some timeout cases, resulting in shorter navigation times. However, this efficiency enhancement comes at the cost of more aggressive navigation, leading to a higher discomfort rate (Disc.(%)). This indicates that the robot may disturb people more frequently to save navigation time. On the other hand, the foresight penalty (R_f) enhances the success rate (Succ.(%)) by focusing on maintaining a comfortable distance and thus causing less disturbance. By combining both the foresight penalty (R_f) and efficiency constraints (R_t), FSRL achieves the highest success rate (Succ.(%)), effectively balancing efficiency and user comfort (less disturbance). This showcases the efficacy of the designed reward components.

5.3.5 Qualitative Evaluation

In Figure 5.11, I present qualitative results for Env.2 under the invisible setting. At time $t = 6$, SARL executes a sharp turn while being close to individuals. This is a consequence that SARL only relies on the present state, disregarding potential future collisions. Consequently, SARL takes more navigation time (14.25s) to reach the goal position. In contrast, the navigation trajectory of FSRL is notably smoother. This can be attributed to the capability of FSRL to anticipate potential collisions ahead of time and take preemptive measures. Furthermore, FSRL achieves a shorter navigation time ($t = 11.50$), underscoring its efficiency.

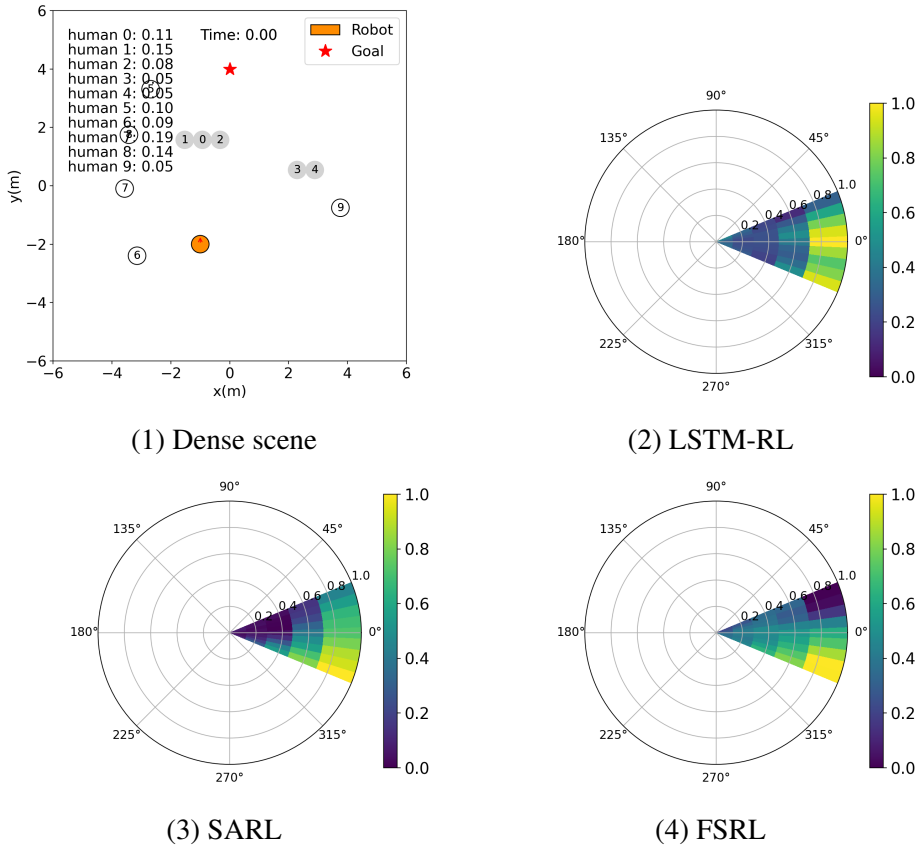
In summary, FSRL's ability to forecast future interactions and proactively respond results in a more effective and efficient navigation strategy compared to SARL.

Given that the ultimate goal of our model is the accurate estimation of the state value, I compare the state values estimated by different methods under the same environment in Figure 5.12. Considering that dynamic humans #7 and #8 are likely to cross the robot's straight path to the target, the robot should either move to the side or slow down to avoid them. At the same time, the robot also needs to consider a row of stationary humans #1, #0, and #2 obstructing the robot's straight path to the goal; unlike dynamic humans, they will not disperse over time, so the robot needs to anticipate danger and take actions in advance. A more effective approach would be to smoothly navigate around the right side of the stationary humans.

LSTM-RL model prefers to move full speed straight towards the destination, but its shortsightedness leads to a disregard for the potential collision risk with stationary humans #0 and #2 directly ahead at 0°. SARL model considers the repulsion of dynamic humans, predicting significantly lower full-speed values in the dangerous direction from 0° to 22.5°, leading to a rightward movement to avoid collisions. However, unlike the SARL model, our FSRL predicts the inevitable potential collisions in the dangerous direction from 0° to 22.5° and strictly avoids danger in this direction. It takes a smart action at -20°, laying the foundation for subsequent smooth navigation. This demonstrates our FSRL's ability to predict and avoid potential collisions.

Furthermore, I compared the value estimations and environment states at different time steps ($t = 3s, 4s$ and $5s$) within the same episode for different methods, to gain a clearer understanding of the reasons behind the trajectory formation of different approaches and to compare their performances.

From the LSTM-RL trajectory in the first row and (a) of Figure 5.13, it is evident that due to the shortsightedness of LSTM-RL, the robot initially opts to deviate to the right to avoid the dynamic crowd, only considering evading when it is about to encounter stationary humans #3 and #4. However, at this point, the limitation on the rotation angle means that while the robot successfully avoids obstacles, it is unable to make immediate large-angle adjustments towards the destination and even deviates at a greater angle away from the destination, resulting in a timeout case. Similarly, SARL takes a risky behavior towards the destination and, due to its shortsightedness, is forced to make a complete right turn and circle around when it is about to encounter an obstacle formed by dynamic and static humans. SARL waits for the gathered crowd to disperse, consequently requiring more time to reach the destination. In contrast, our FSRL predicts potential future collisions and opts to initially slow down and wait safely, thereby navigating to the destination safely and efficiently.



(1) Dense scene (2) LSTM-RL

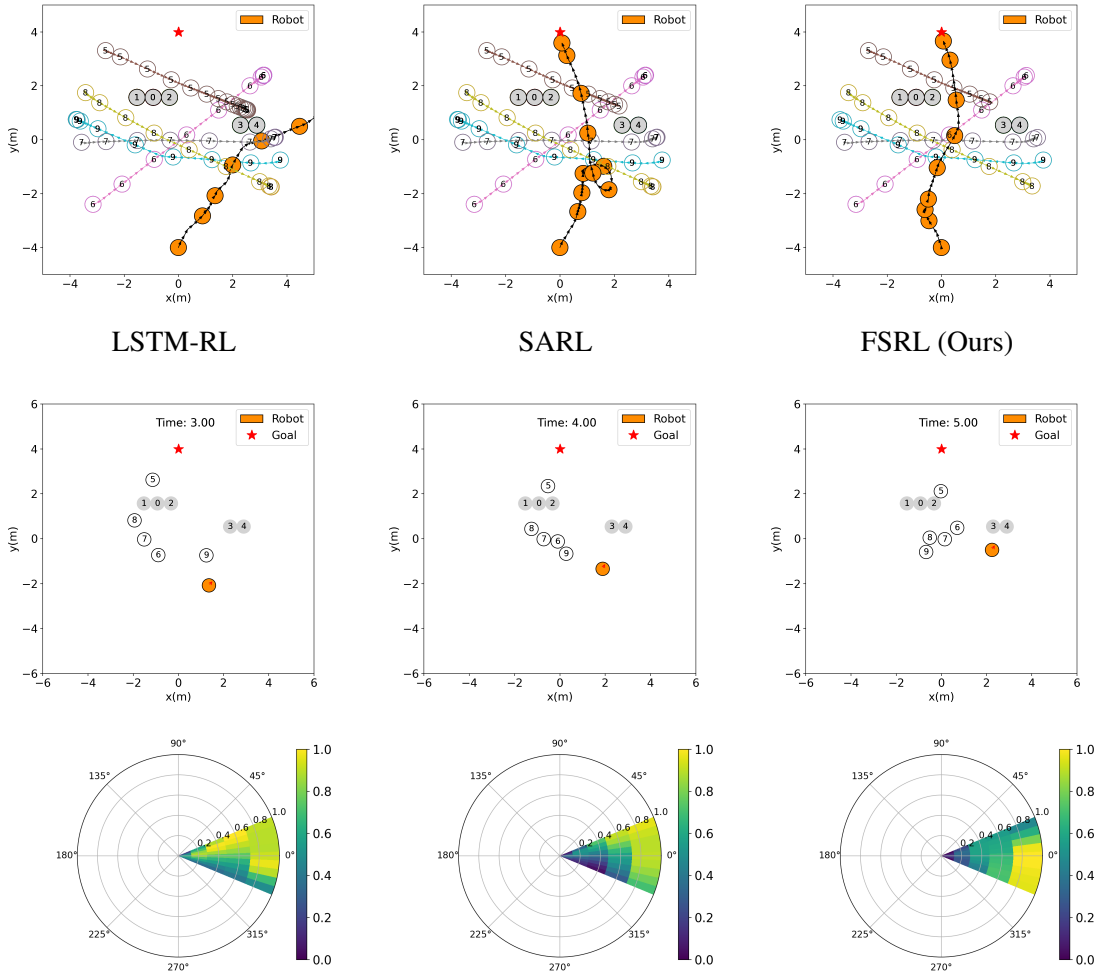
(3) SARL (4) FSRL

Figure 5.12: Value estimations by different methods for the dense scene (1). (1) shows attention scores in a complex scene (Env.3). Our FSRL assigns high important scores to dynamic human #7 and gives sub-high important scores to dynamic human #8 and #5 and static human #1 and #3. Above humans are most likely to get close to the robot and have potential interaction in the future, so the robot needs to pay more attention to them. However, our FSRL assigns the lowest weight to dynamic #9 and stationary #3 and #4 are far from the robot. (2) shows that LSTM-RL predicts high values for high speeds toward the goal, which is dangerous because of the existence of stationary humans #0 and #2. (3) and (4) show that our FSRL, compared to SARL, not only prefers to turn right to -25° to avoid dynamic humans but also strictly avoids the possibility of potential collisions with stationary humans #1 and #0.

5.3.6 Parameters Choosen

I trained the system with different values set for the parameters of the reward function term R_{st} and R_{dy} to analyze the impact of parameter selection on performance, and to show the reason behind our chosen parameters. The comparison of the success rate, navigation time, and cumulative rewards are shown in Figure 5.14.

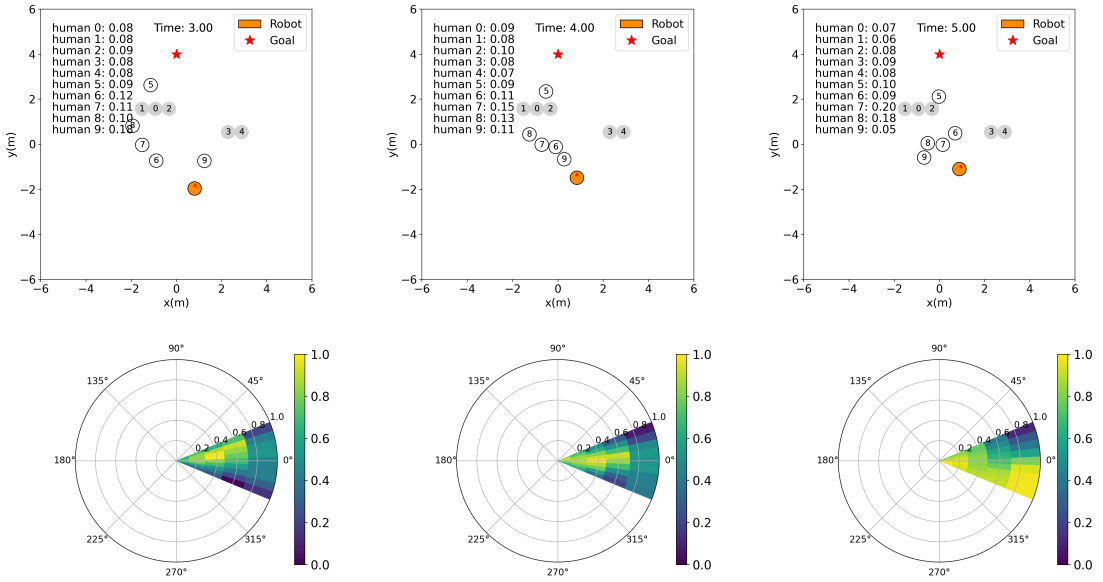
On one hand, I fixed the value of $\alpha = 0.15$ in R_{st} and examined training with $\beta = 0.10, 0.25, 0.35, 0.40, 0.45, 0.50, 0.60, 1.00$ separately. Figure 5.14(a) shows that our chosen value of $\beta = 0.50$ converges fastest in terms of navigation time and cumulative rewards, yielding the short-



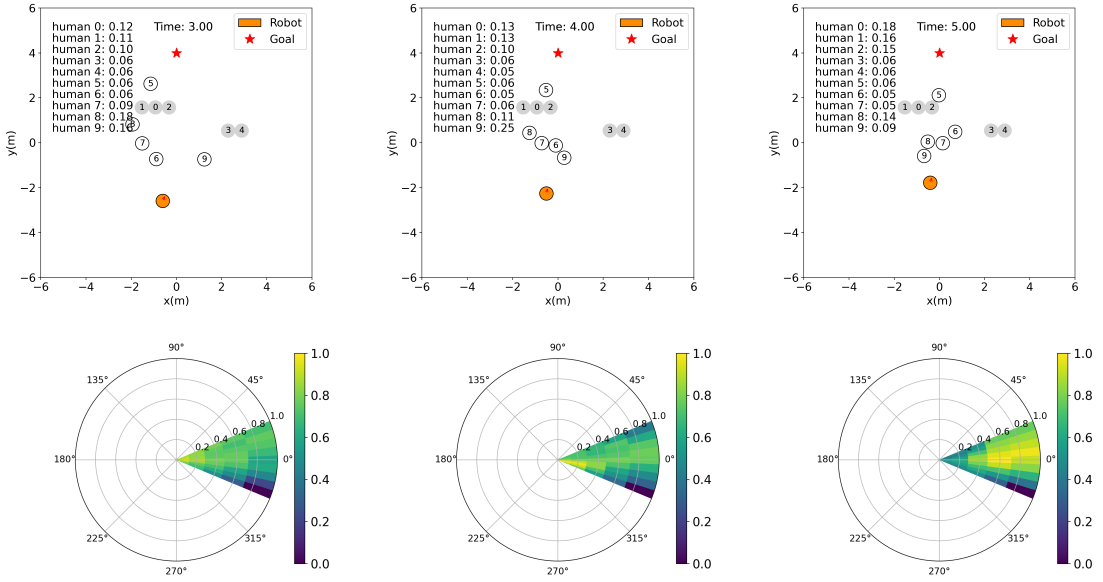
LSTM-RL SARL FSRL (Ours)

(a) LSTM-RL: Value estimations and Environments at $t = 3s, 4s$ and $5s$, separately.

Figure 5.13: Value estimations and Environments for SARL and our FSRL with the same episode at different timestep. The first row shows the trajectories of different methods. (a) shows that LSTM is shortsighted and more inclined to choose right-turning actions to bypass dynamic and stationary humans with a large angle, ultimately resulting in overtime outcomes. (Continued on next page)

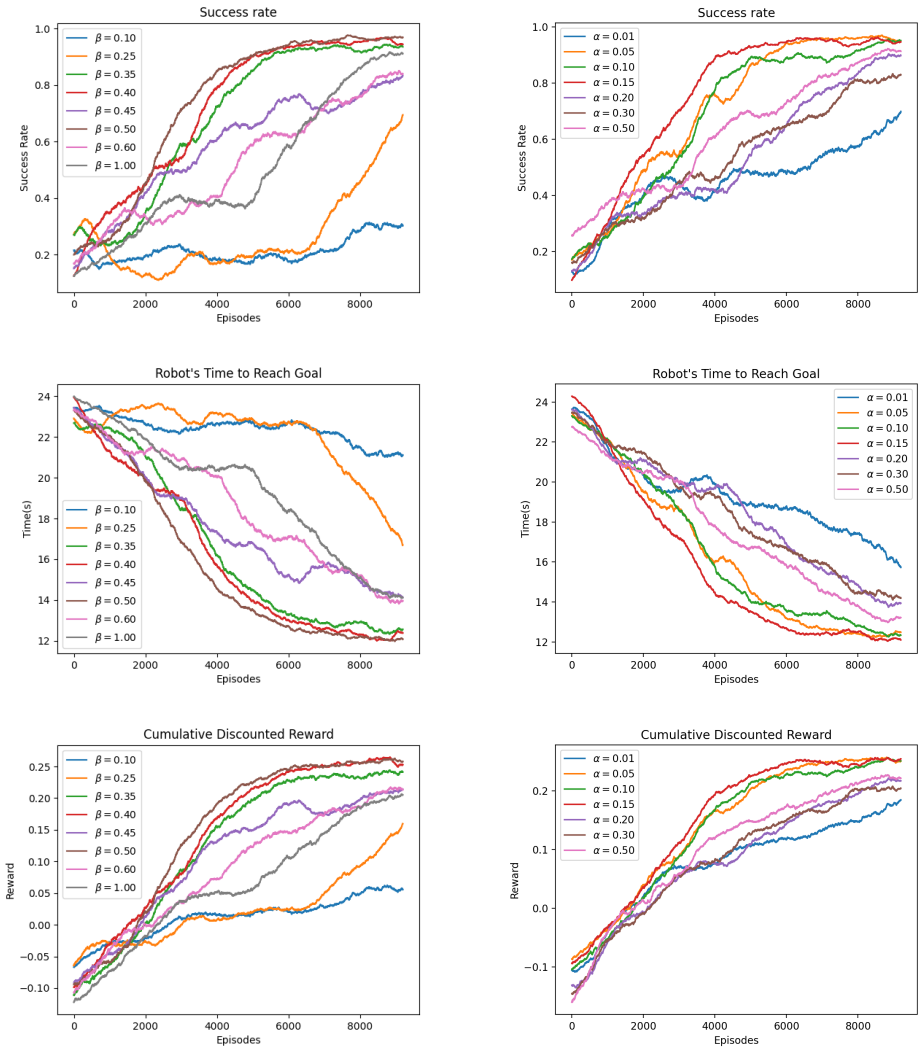


(b) SARL: Value estimations and Environments at $t = 3s, 4s$ and $5s$, separately.



(c) FSRL (Ours): Value estimations and Environments at $t = 3s, 4s$ and $5s$, separately.

Figure 5.13: (Continued from previous page) Value estimations and Environments for SARL and our FSRL with the same episode at different timestep. SARL, also shortsighted, initially navigates riskily towards the destination, but then encounters a convergence of dynamic crowds and stationary humans forming an obstacle, forcing it to opt for a full right rotation to wait for the dynamic crowd to disperse, thereby taking more time to reach the destination. In contrast, our FSRL opts to initially slow down and wait safely.



(a) Dynamic Parameter (b) Static Parameter

Figure 5.14: Performance Comparison under Different Settings of Reward Function Parameters.

est navigation time and the highest cumulative rewards. Although the convergence speed is slightly slower than $\beta = 1.00$ regarding the success rate, the final convergence value is the highest. Additionally, I observed that when the values are too small, like $\beta = 0.10, 0.20$, the performance is poor, and as the values increase, the larger the difference between the chosen β and 0.5, the worse the performance will become.

On the other hand, I fixed the value of $\beta = 0.50$ in R_{dy}, and examined training with $\alpha = 0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30$ separately. Figure 5.14(b) demonstrates that our chosen value of $\alpha = 0.15$ converges noticeably faster than other values during training. It also achieves the shortest navigation time and the highest cumulative reward. Additionally, it shows that when the values are too low, such as $\alpha = 0.01$, the performance is the poorest. When chosen around $\alpha = 0.15$, whether the value of α increases or decreases, the performance declines within a certain range, and the convergence rate slows down.

5.4 Summary

In this chapter, I introduce a novel Foresight Social-Aware Reinforcement Learning (FSRL) framework designed for navigating robotic systems. Unlike conventional methods that solely rely on the current state for decision-making, FSRL approach excels by anticipating potential collisions in the future and acting preemptively to avert them. FSRL method can estimate potential collisions in the future and hence takes action in advance to avoid collisions. In contrast, previous methods only make the decision according to the current state and thus perform not well when the environment becomes complex. Furthermore, I address the efficiency aspect by incorporating an efficiency constraint into FSRL methodology, leading to a substantial reduction in navigation time. The experimental study encompassed three progressively challenging environments, and the results indicate that FSRL approach enhances the effectiveness and efficiency of navigation methods.

CHAPTER 6

Generalization on Social-Aware Robot Navigation Behaviors

In the previous chapter, I introduced a complex environment comprising dynamic and static humans. The robot is enhanced with foresight capabilities through an augmented reward function. However, the trained policies by existing methods are heavily dependent on the training environment. When crowd configurations change, such as variations in crowd size or more intricate human state compositions, navigation performance deteriorates. In this chapter, I present a novel deep graph learning architecture to capture the relationships between different entities. Specifically, I (1) introduce a graph attention network to extract higher-order spatial and temporal interactions within the crowd based on the agent's state, (2) incorporate an RNN to encode historical information, and (3) jointly aggregate paired spatial-temporal interactions into a social attention mechanism to capture crowd representation. By capturing direct and indirect spatial and temporal interactions, the proposed method offers a comprehensive understanding of the crowd and enhances reasoning capabilities. When compared to state-of-the-art methods, my approach demonstrates remarkable robustness in terms of safety, efficiency, and generalization across various challenging scenarios.

Contents

6.1 Introduction

In social-aware robot navigation, it is a significant challenge for robots to generalize or transfer learned navigation strategies or knowledge from training to unknown environments. One of the goals of social-aware robot navigation is to cooperate and coordinate in various social environments,

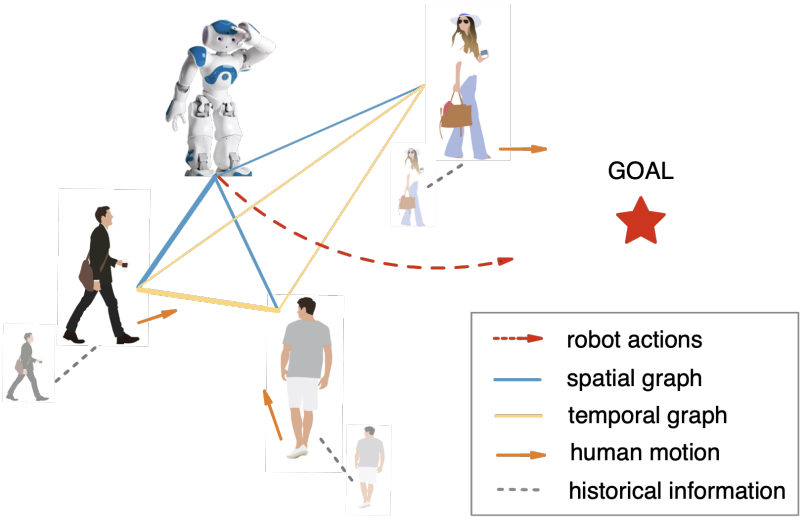


Figure 6.1: Illustration of our work. When navigating, I consider each spatial information as a node in the spatial graph and each human's historical information as a node in the temporal graph. By aggregating spatial-temporal pairwise relationships with social attention weights, the robot evaluates the values of all actions with deep V-learning and generates the best decisions that are robust and predictive.

such as densely populated cities, shopping centers, hospitals, etc. If robots are learned only in limited simulated scenarios, they might overly adapt to those scenarios, leading to poor performance in new scenarios. Enhancing generalization enables robots to avoid overfitting issues, and interact more universally with diverse types of crowds, not just limited to training scenarios, thereby better achieving social interaction goals. Additionally, in crowd navigation, robots need to predict and adapt to human behaviors. Robots with strong generalization capabilities can reliably predict human behaviors in different situations, reducing collision risks and enhancing navigation safety. Particularly, when facing novel, unfamiliar scenarios, robots can draw from prior experiences to adapt effectively. Thus, enhancing model generalization is crucial for social-aware robot navigation.

While our proposed FSRL (Foresight Socially Aware Reinforcement Learning) methods achieved success shown in the previous chapter, like most learning-based methods, they heavily rely on simulated environments. It leads to the algorithm overfitting to specific conditions in simulation and hinders its ability to adapt to new situations or changes in the real world, thereby reducing the algorithm's generalization capability. Navigation methods based on Deep Reinforcement Learning (DRL) essentially train policies to select actions with maximum cumulative rewards in their training simulated environments, which implicitly encode interactions between agents, capturing the collective impact of the crowd by incorporating pairwise interactions through LSTMs (*Graves and Graves*, 2012) or the maximin operator. However, interactions vary significantly across different simulation environments, rendering navigation policies trained in one environment unsuitable for another. This is because there exist uncertainties in the dynamic crowd, which affects the way and impact of interactions in it. Specifically, as the density of crowds increases, it leads to increased mutual influence and interference among individuals, which in turn increases the difficulty and complexity of human-robot interaction in such environments. Therefore, robots need to timely adjust their behavior

strategies according to changes in crowd behavior, in order to better adapt to different scenarios and improve the quality of human-robot interaction.

From another perspective, robots need to encode interaction information more effectively to comprehensively perceive and understand social interactions within the crowd, accurately capturing crucial features and patterns, such as dynamic changes in crowd behavior. By learning these patterns from interactions, robots can generalize these patterns to unfamiliar scenarios, thus better adapting to new social interactions. Graph Neural Networks (GNNs) offer an effective means of capturing intricate interaction patterns. GNNs achieve this by propagating and aggregating local information, resulting in richer node representations. This aids models in better comprehending distinct individuals within the crowd and their relationships, ultimately facilitating navigation decisions. Furthermore, when facing various crowd environments, even those previously unseen, GNNs can enhance generalization by adapting to different graph topologies (different crowd simulation environments).

As introduced in the last paragraph and Section 3.2.4, graph networks, by learning the complex relationships between nodes and the global distribution of graph structures, can capture the deep features of different graph structures, allowing them to effectively generalize and predict when faced with different and unseen graph data. Moreover, in dealing with diverse and dynamically changing crowd environments, the Graph Attention Network (GAT) can adaptively determine node feature representations by learning the attention weights between nodes. Therefore, in this chapter, I propose an attention-based spatial-temporal graph learning framework for crowd navigation trained with Deep Reinforcement Learning (DRL), named "ASTG". First, leveraging the strong generalization ability of graph networks, the Graph Attention Network is introduced to fully model both the spatial relations (such as spatial positioning) and temporal relations (such as trajectory intent inference) of agents in crowd navigation scenarios separately. Furthermore, before modeling the temporal relation interactions, Recurrent Neural Networks (RNNs) (*Medsker and Jain*, 2001) are combined to encode each agent's historical trajectory information as input for the temporal graph, thus implicitly inferring the intentions of all humans. Then, I aggregate the pairwise spatial-temporal features of each agent into a social attention mechanism to capture the relative importance of each individual. Together, a comprehensive understanding of crowd behaviors can facilitate efficient robot navigation strategies, adaptable to various crowded scenarios.

In summary, the core of the ASTG method lies in utilizing the spatial-temporal graph's inferential capabilities to enable robots to better understand and interpret the relationships between agents in both time and space, thereby making wiser decisions in dynamically changing crowd environments. Here, the spatial graph is used to capture current spatial interactions, and through integration with RNNs, the temporal graph utilizes past trajectory information to infer the future intentions of each agent. Finally, I conducted extensive experimental validations of this method in various unknown and challenging scenarios. It achieved better performance in dynamically changing environments compared with other methods, demonstrating its robustness and generalizability.

6.2 A General Graph Learning Navigation Method

6.2.1 Problem Formulation

As outlined in Section 4.2, the joint state $\mathbf{s}_t^{jn}$ of the robot state $\mathbf{s}_t^r$ and the humans' states $\mathbf{s}_t^h$ is denoted as $\mathbf{s}_t^{jn} = [\mathbf{s}_t^r, \mathbf{s}_t^h]$. As before in Section 5.2, the objective is to maximize the cumulative reward for

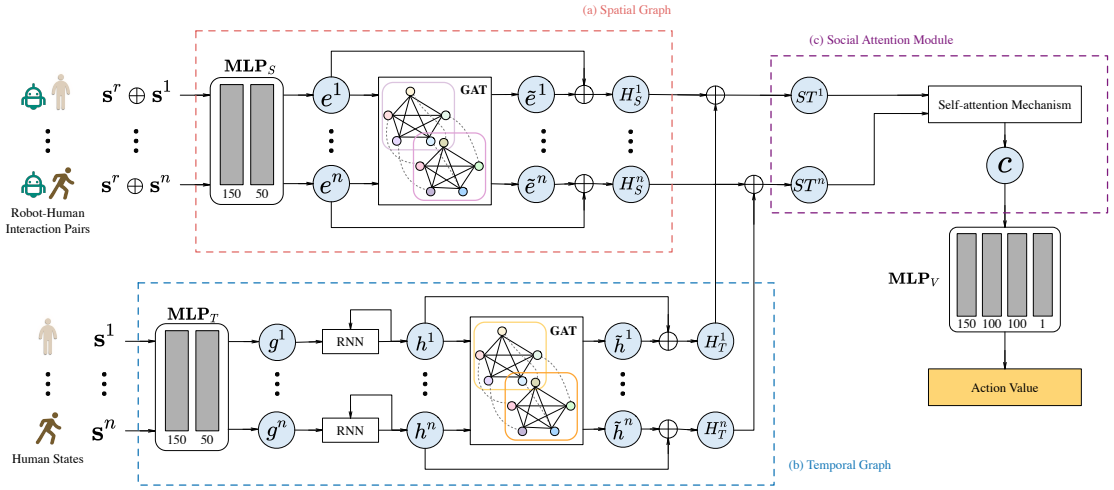


Figure 6.2: Network architecture. (a) The spatial graph utilizes the GAT to encode direct and indirect spatial interactions between agents. (b) The temporal graph incorporates an RNN to reason about the temporal interactions based on historical information. (c) The social attention mechanism jointly aggregates the pairwise spatial-temporal interactions to capture the crowd representation in the crowd feature, which is then used to estimate the action values.

each state and thereby to find the optimal policy $\pi^* : \mathbf{s}_t^{jn} \mapsto \mathbf{a}_t$:

$$V^*(s_t^{jn}) = \sum_{t'=t}^{T} \gamma^{t' \cdot v_{pref}} R(\mathbf{s}_{t'}^{jn}, \pi^*(\mathbf{s}_{t'}^{jn})), \tag{6.1}$$

$$\pi^*(\mathbf{s}_t^{jn}) = \underset{\mathbf{a}_t}{\operatorname{argmax}}\, R(\mathbf{s}_t^{jn}, \mathbf{a}_t) + \gamma^{\Delta t \cdot v_{pref}} \int_{\mathbf{s}_{t+\Delta t}^{jn}} P(\mathbf{s}_t^{jn}, \mathbf{a}_t, \mathbf{s}_{t+\Delta t}^{jn}) V^*(\mathbf{s}_{t+\Delta t}^{jn})\, d\mathbf{s}_{t+\Delta t}^{jn}, \tag{6.2}$$

where Δt is the time step and $\gamma \in (0, 1)$ is a discount factor that is normalized by the preferred velocity v_{pref}.

I utilize the sparse reward function (Equation 5.10) formulated as in *Chen et al.* (2019) which is also outlined in Section 4.2.3. It provides rewards for task accomplishment while simultaneously imposing penalties for collisions or uncomfortable distances. Unlike the augmented reward function (Equation 5.16) in Chapter 5, I chose the sparse reward function (Equation 5.10) in this chapter because inferring the intentions of agents through the adoption of graph structures can also endow the algorithm with foresight, which can replace the forward-looking aspect of FSRL methods. Furthermore, for a better explanation, I conducted experiments to verify the effectiveness and generalizability of FSRL and/or ASTG methods in different dynamic crowd environments, which is shown in Section 6.3.6.

6.2.2 Attention-based Spatial-Temporal Graph Learning (ASTG)

I propose a novel framework for social-aware robot navigation tasks, depicted in Figure 6.2. Graph structures are more flexible for environments with varying numbers of agents, as they can adapt to

graphs of varying sizes through information exchange between nodes. Especially compared to using only a feed-forward neural network to reason about crowd interactions, graph neural networks do not require input data of a fixed size, but instead operate iteratively by exchanging information among the nodes of the graph, handling an arbitrary number of nodes with shared parameters like the weights of the adjacency matrix.

Thus, here, I construct the spatial graph and temporal graph both are fully-connected graphs. The number of nodes is the number of humans n and the edges are weighted by attention mechanism in GAT. It allows me could use GATs to efficiently extract not only the spatial graph representation but also the temporal graph representation. Unlike in *Chen et al.* (2019), attention weights in GATs are computed for all agents, and the interactions in the graph can be modeled using the same graph convolutional operation. Then, a social attention mechanism is utilized to encode the collective influence of neighbors based on the spatial and temporal graphs. Finally, the value function is estimated.

6.2.2.1 Spatial Graph Representation

The spatial-temporal graph network calculates the spatial attention and temporal attention separately from the environment state matrix, which is used to represent the edges of the spatio-temporal graph. In the spatial graph, I input rough combined spatial interactions to calculate agents' spatial dependencies. For each human, I initially aggregate its state at the current moment with that of the robot to compute rough spatial interactions. Then, I calculate spatial attention maps through graph attention network layers, representing the importance of adjacent interactions, and encode spatial features such as distance and relative direction between agents in pairwise interactions. For instance, for the ith interaction (node), its influence with other interactions (nodes) makes them connected, and the greater the influence, the higher the attention weight. Simultaneously, the ith (node) interaction aggregates spatial information from its surroundings (all connected (nodes) interactions) through the influence of spatial edges, capturing the impact of interactions within the crowd on ith node. This impact not only considers explicitly modeled pairwise human-robot interactions but also implicitly captures indirect human-human interactions and crowd-robot interactions through edge weights.

As shown in the upper part of Figure 6.2(a), in the spatial graph, I use the states of the robot and the ith human at time t to derive a node feature via a multilayer perception (MLP):

$$e_t^i = f_{spatial}(\mathbf{s}_t^r, \mathbf{s}_t^i; W_e), \tag{6.3}$$

where W_e are the network weights and $f_{spatial}(\cdot)$ is an MLP with a ReLU activation function. The rectified linear unit (ReLU) activation function used in the framework aims to capture non-linear features in the feedforward network.

I then use the graph convolution operation in a GAT (Section 3.2.4.2) to model spatial interactions in a crowd. The input at time t is denoted by $E = [e_t^1, ..., e_t^n]$, where n is the number of nodes and $e^i = e_t^i$ for short. The normalized spatial coefficients α_{ij}^S in the attention mechanism can be computed by:

$$\alpha_S^{ij} = \frac{\exp(\text{LeakyRelu}(a[\mathbf{W}e^i || \mathbf{W}e^j]))}{\sum_{k \in \mathcal{N}^i} \exp(\text{LeakyRelu}(a[\mathbf{W}e^i || \mathbf{W}e^k]))}, \tag{6.4}$$

where α_S^{ij} represents the importance of node j to node i. $\mathbf{W}(\cdot)$ is the weight matrix and $||$ is the concatenation operation, while $\mathcal{N}^i$ indicates the neighborhood of node i in the graph. After passing

through a fully connected network (FCN) $a(\cdot)$, a LeakyReLU activation follows. Finally, a softmax function is used to normalize the spatial attention coefficients α_{ij}^{S}.

Thus, the output node features of the graph attention layer at time t are:

$$\tilde{e}^{\,i} = \sigma(\sum_{j \in \mathcal{N}^i} \alpha_S^{ij}\, \mathbf{W} e^{\,j}), \quad \tilde{E} = [\tilde{e}^{\,1}, ..., \tilde{e}^{\,n}], \tag{6.5}$$

In this work, through the graph attention layer, the ith interaction feature is transformed into a higher-level spatial feature that models indirect robot-human and human-human interactions in a crowd. Additionally, residual connections are employed in the spatial and temporal graph networks to accelerate convergence and stabilize the framework (*He et al.*, 2016). Thus, the final spatial features $H_{spatial}$ is described by:

$$H_S^i = e^i + \tilde{e}^{\,i}, \quad H_{spatial} = [H_S^1, ..., H_S^n]. \tag{6.6}$$

6.2.2.2 Temporal Graph Representation

Due to the high motion-dependency of the temporal dimension, I introduced an RNN for each agent to capture the dynamics of its trajectory. Then, in the temporal graph, I treat the motion feature of each human as a node of the graph, and the edges signify the relationships between humans' motions. The weight of each edge quantifies the significance of the relationship it represents. Since motion is continuous, I can predict future actions based on the current motion information. To achieve this, I utilize graph attention network layers to compute a temporal attention map, estimating the mutual influence of neighboring agents' motion behaviors and aggregating these influences to form temporal features. These temporal features encompass both the current trajectory information of humans and predictions of future trajectories, allowing the robot to better understand human behavior and intentions.

First, I take the i-th human's state $\mathbf{s}_t^i$ as an input through an MLP $f_{temporal}$ to obtain the temporal latent state g_i^t:

$$g_t^i = f_{temporal}(\mathbf{s}_t^i; W_g), \tag{6.7}$$

where W_g are the embedding weights. As before, $f_{temporal}(\cdot)$ is an MLP with ReLU activation function.

Then, I process the temporal latent state g_i^t with an RNN cell:

$$h_t^i = \text{RNN}(h_{t-1}^i, g_t^i), \tag{6.8}$$

where h_t^i is the hidden state at time t, which changes over time, reflecting the evolution of ith agent's state.

The hidden states h_t^i are then fed into the graph attention layer to model temporal interactions. GAT learns the attention distribution between nodes adaptively, enabling it to weigh the information from different nodes based on their relationship. This implies that GAT facilitates the propagation of temporal interaction information between nodes. Similar to the spatial graph, the input at time t is $H = [h_t^1, ..., h_t^n]$, where N is the number of humans and $h^i = h_t^i$ for short. The normalized temporal coefficients α_{ij}^{T} in the attention mechanism can be described by:

$$\alpha_T^{ij} = \frac{\exp(\text{LeakyRelu}(\alpha[\mathbf{W}h^i || \mathbf{W}h^j]))}{\sum_{k \in \mathcal{N}^i} \exp(\text{LeakyRelu}(\alpha[\mathbf{W}h^i || \mathbf{W}h^k]))}, \tag{6.9}$$

Combining RNN and GAT allows for a more accurate capture of the evolving states and interactive changes of agents in a social environment over time. While RNN captures the temporal variations in agents' states, GAT, through its attention mechanism, considers interactions with other nodes, leading to a finer-grained temporal dynamic modeling. By integrating the state of each agent and the evolution of their social interactions, robots are enabled to more accurately predict future human behaviors and intentions, and even assist in analyzing collective behavior patterns within groups. Hence, the final temporal graph feature with a one-layer GAT can be described as follows:

$$\tilde{h}^i = \sigma(\sum_{j \in \mathcal{N}^i} \alpha_T^{ij} \mathbf{W} h^j), \quad \tilde{H} = [\tilde{h}^1, ..., \tilde{h}^n], \tag{6.10}$$

Similar to the spatial graph network, a residual connection structure is also deployed in the framework to obtain the final temporal features $H^{temporal}$:

$$H_T^i = h^i + \tilde{h}^i, \quad H_{temporal} = [H_T^1, ..., H_T^n]. \tag{6.11}$$

6.2.2.3 Social Attention Mechanism

In order to capture the uncertainty of crowd movements, I build a social attention module that fuses the heterogeneous spatial and temporal features of each agent and captures dependencies among agents. Inspired by (*Chen et al.*, 2019), I build a self-attention network to assign attention weights w^i for each agent's aggregated spatial-temporal combined feature $ST^i = [H_S^i, H_T^i]$ and encode the collective impact of a crowd.

First, our combined feature ST^i is transformed into an attention score w^i:

$$ST^m = \frac{1}{n} \sum_{k=1}^{n} ST^k, \quad w^i = f_\alpha(ST^i, ST^m; W_\alpha), \tag{6.12}$$

where W_α is the weight and f_α is nonlinear transformation.

I then obtain the final crowd representation c by the product sum of each spatial-temporal pairwise feature ST^i and attention weights w_i:

$$c = \sum_{i=1}^{n} softmax(w^i) ST^i \tag{6.13}$$

6.2.2.4 Graph-based Planning

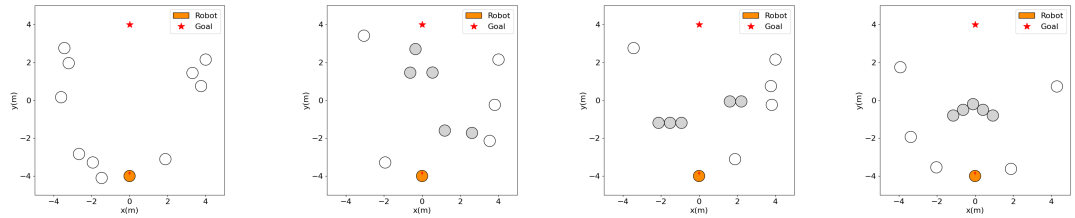
The crowd representation c is the input to an MLP f_v to estimate the state value V for planning:

$$V = f_v(\mathbf{s}^r, c; W_v) \tag{6.14}$$

where W_v denotes the weights.

Similar to Chapter 5, I also use Deep V-Learning which is referred in Section 3.3.1.1for policy and value function learning. The training process is also the same as in Chapter 5.

Based on the above, when the trained algorithm is tested in environments with varying numbers of people, our algorithm demonstrates exceptional adaptability and generalization capability. This is because, although the number of nodes changes in the temporal and spatial graphs, GAT adapts



(a) BASE (10 dynamic) (b) DS (distributed) (c) RO (row3and2) (d) CO (concave5)

Figure 6.3: Simple and complex scenarios for testing phases. Solid grey circles depict static standing agents, while hollow circles represent dynamic objects. I show the example of 10 humans in the environment: (a) BASE are with all dynamic humans. (b) DS (distributed) features 5 scattered static humans, while the remaining are dynamic humans. (c) RO (row3and2) is with two clusters of barriers, each containing 2 and 3 standing humans respectively, alongside other dynamic humans. (d) CO (concave5) is characterized by a concave composed of five static humans, while other humans are dynamic.

to different graph sizes through its dynamic attention mechanism. It assigns appropriate importance weights to each node, ensuring efficient and relevant information exchange even with changes in crowd size. Unlike MLPs, which typically require a fixed structure for input data and need adjustments for different data sizes, GAT dynamically adjusts. Furthermore, in the social attention mechanism stage, the algorithm aggregates features from the spatial and temporal graphs to capture complex interactions among agents, forming a comprehensive representation of the crowd. This method adapts to varying numbers of agents, maintaining the algorithm's efficiency and accuracy, and showcasing its strong generalization capability in diverse environments.

6.3 Experiments

6.3.1 Simulation Setup

I build our 2D simulation environment for crowd robot navigation following *Chen et al.* (2019). I keep the invisible setting for the robot to prevent humans from overreacting to the robot and thus avoid learning an aggressive navigation policy that moves too directly to the goal position and gets too close to humans. I use circle crossing scenarios with a radius of 4m in our simulation. All humans are controlled by the ORCA policy. Dynamic humans are randomly positioned on the circle and will move to opposite positions on the same circle, while static humans are randomly located in the same circle and do not move. To fully analyse the effectiveness of the proposed model, I evaluate all models for 1000 random test cases in both simple and complex scenarios.

6.3.2 Training and Testing

I compare the performance of our model with three state-of-the-art methods: SARL (*Chen et al.*, 2019), RGL (*Chen et al.*, 2020a) and SG-DQN (*Zhou et al.*, 2022). SARL is a baseline for Deep V-learning. In addition, RGL and SG-DQN are used as baselines for graph-based learning methods.

6.3.2.1 Training

I train all methods in an environment that consists of 5 dynamic and 2 scattered static humans using data from 10k episodes. For a fair comparison, the network architectures of all baselines are trained as stated in the original papers.

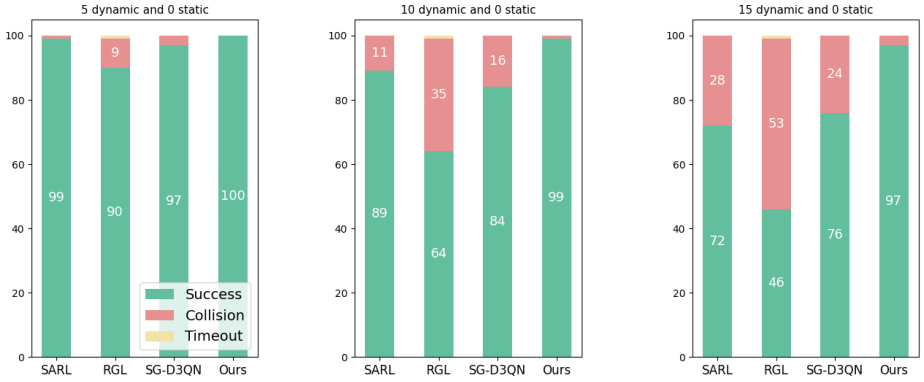
6.3.2.2 Testing

I evaluate the generalization capability of the policy of each method, learned in the above training scenario, in both simple and complex scenarios with 1k episodes. To vary the start and end positions of agents over the episodes I use different random seeds, where each (random) episode configuration is evaluated for all methods. As evaluation metrics, I use the percentage of success, collision, and timeout cases, respectively, which is defined in Section 4.3.3. I also present navigation time-related metrics for better comparison.

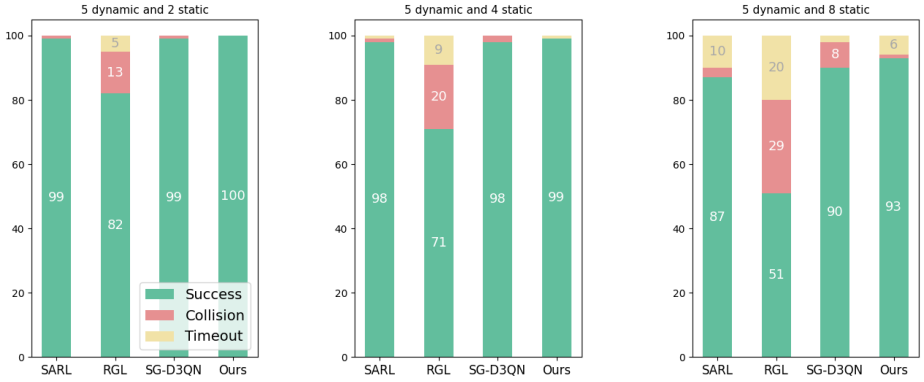
Simple scenarios are set up with different numbers of distributed dynamic and/or static humans (like Figure 6.3(a) and (b)), such as scenarios with only $N = 5, 10, 15$ dynamic humans, and scenarios with 5 dynamic humans or $M = 2, 4, 8$ scattered static humans. In addition, based on 5 dynamic humans, the complex scenario has 5 static humans with different group combinations (distributed, line group or concave group, like Figure 6.3(b),(c) and (d)), whose positions are randomly set in the circle.

6.3.3 Quantitative Evaluation

To show the effectiveness of our method, I compare our method with SARL (*Chen et al.*, 2019), RGL (*Chen et al.*, 2020a) and SG-DQN (*Zhou et al.*, 2022). Figure 6.4, Figure 6.5, Table 6.3, Table 6.1 and Table 6.2 show the results of all methods under simple and complex environments. Due to the difference in the reward function used by the SG-DQN method compared to other methods, I have decided to retain its original reward function design to ensure SG-DQN performs at its best. Consequently, in Table 6.3, I do not directly compare the reward function values of SG-DQN with those of other methods, to ensure fairness and validity in the experimental results. No matter whether evaluated in simple scenarios or complex scenarios, it can be seen from the data that my model is the best one with the highest success rate (Figure 6.4, Figure 6.5), the lowest comfort rates (Table 6.1, Table 6.2) and the highest reward (Table 6.3).



(a) Scenarios with different numbers of dynamic humans and no static humans.



(b) Scenarios with 5 dynamic humans plus the different numbers of static humans.

Figure 6.4: Quantitative evaluation on simple scenarios with different numbers of humans. Among the four methods, our ASTG always has the highest success rate and lowest collision rate.

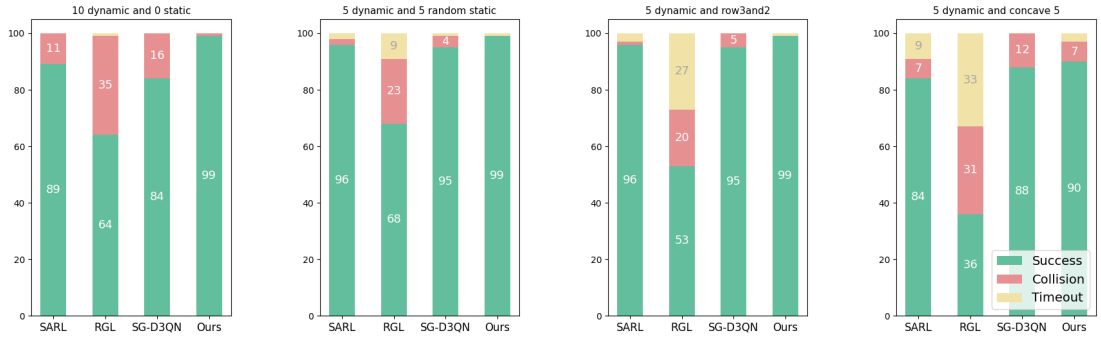


Figure 6.5: Quantitative evaluation on complex scenarios with 5 dynamic humans plus other 5 dynamic humans or different groups comprised by 5 static humans.

Metrics	$Disc.(\%)$			$t_{succ.nav}$			$t_{weighted.nav}$		
N dynamic humans without static humans									
N(dynamic)	5	10	15	5	10	15	5	10	15
SARL	0.02	0.11	0.22	10.58	11.64	**12.33**	10.91	14.18	17.80
RGL	0.21	0.42	0.53	10.23	11.43	12.51	12.57	17.65	21.60
SG-DQN	0.05	0.08	0.09	**10.03**	**11.24**	13.41	**10.63**	13.71	16.56
Ours	**0.01**	**0.04**	**0.08**	11.40	12.65	13.28	11.45	**12.97**	**14.13**
N static humans with 5 dynamic humans									
N(static)	2	4	8	2	4	8	2	4	8
SARL	0.05	0.09	0.15	10.51	**10.64**	**11.16**	11.10	11.71	13.52
RGL	0.34	0.43	0.57	10.59	10.97	12.06	13.88	15.53	18.41
SG-DQN	0.04	0.06	**0.09**	**10.07**	11.24	12.07	**10.42**	11.79	13.62
Ours	**0.03**	**0.06**	0.10	11.09	11.22	11.97	11.25	**11.56**	**12.83**

Table 6.1: Evaluation performance comparison in the simple scenarios. "Disc. (%)" is the average frequency of duration that the human invades the comfort area of the robot. "$t_{succ.nav}$" is the average navigation time of success cases. "$t_{weighted.nav}$" (Equation 6.15 is a novel weighted navigation time metric considering the impact of the discomfort steps and collision cases.

Metrics	$Disc.(\%)$			$t_{succ.nav}$			$t_{weighted.nav}$		
5 dynamic humans with different static groups									
Groups	DS	RO	CO	DS	RO	CO	DS	RO	CO
SARL	0.11	0.10	0.05	**10.83**	**11.08**	11.54	12.18	12.31	13.21
RGL	0.44	0.56	0.59	11.25	11.37	11.27	16.22	16.50	18.58
SG-DQN	0.07	0.07	0.06	10.96	11.25	11.15	11.93	12.20	13.06
Ours	**0.06**	**0.06**	**0.03**	11.35	11.34	**11.05**	**11.75**	**11.76**	**12.21**

Table 6.2: Evaluation performance comparison in the complex scenarios with 5 dynamic humans and different static groups (DS (distributed), RO (row3and2), and CO (concave)).

Scenarios	***N Dynamic Humans***			***M Static Humans***			***Group Static***		
	5	10	15	2	4	8	DS	RO	CO
SARL	**0.3319**	0.2390	0.1205	**0.3324**	0.3171	0.2521	**0.3041**	0.2951	0.2414
RGL	0.2420	0.0425	0.0015	0.1664	0.0957	0.0006	0.0667	0.0294	0.0034
Ours	0.3101	**0.2665**	**0.2352**	0.3179	**0.3174**	**0.2599**	0.3017	**0.3001**	**0.2714**

Table 6.3: Average reward in simple and complex scenarios.

In Figure 6.4 and Figure 6.5, as a previous learning-based social-aware navigation method, SARL (*Chen et al.*, 2019) has exhibited limited performance and social compliance, as it fails to capture deeper and comprehensive social interactions. Furthermore, compared to other baselines that also utilize GNN to model human-robot interactions, RGL (*Chen et al.*, 2020a) has not demonstrated strong generalization performance. This may be attributed to the fact that GCN employed in RGL performs simple convolutional operations on interaction/human features encoded with fixed rule-based weights, lacking the flexibility to generalize to different environment structures. This leads to a sharp increase in collision rates for RGL (*Chen et al.*, 2020a) as the number of obstacles increases. In contrast, GAT can assign different weights based on the importance of these features, thereby better capturing critical environmental characteristics. Therefore, as shown in Figure 6.4 and Figure 6.5, SG-DQN (*Zhou et al.*, 2022), also based on GAT, is competitive with our ASTG. However, SG-DQN (*Zhou et al.*, 2022) lacks the ability to capture time-varying motion features, thus reducing predictive accuracy. In comparison, our ASTG exhibits significantly slower increases in collision rates and navigation times in pure dynamic environments and environments containing both dynamic and static humans, demonstrating excellent generalization performance.

From Table 6.1 and Table 6.2, they show that the baselines enjoy a better performance in $t_{succ.nav}$, which is defined as the average navigation time of the successful cases in seconds. However, overall, they have either a higher discomfort frequency or higher collision rate, which means, to achieve a shorter navigation time, the robot takes a lot of aggressive actions, badly affecting human comfort or even having collisions. To better measure and analyze the behavior, I propose a weighted navigation time metric $t_{weighted.nav}$ that penalizes aggressive policies:

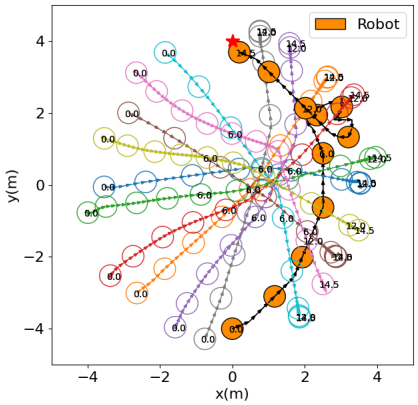
$$t_{weighted.nav} = \frac{t_{dnav} + N_{coll} * t_{limit}}{N_{succ} + N_{coll}}, \tag{6.15}$$

where N_{succ} and N_{coll} are the number of success and collision cases, respectively. t_{limit} is the maximum navigation time, which set as 25s in our simulations. t_{dnav} is extended from the original navigation time t_{nav} to penalize the impact of discomfort. Specifically, if a step is causing discomfort, I add half of a time step to $t_{succ.nav}$, which equals $t_{dnav} = N_{succ} * t_{succ.nav} + N_{disc} * 0.5 * \Delta t$ and N_{disc} is the number of discomfort steps in success cases. Thus, when considering collision cases and the discomfort frequency, our method shows a shorter weighted navigation time $t_{weighted.nav}$.

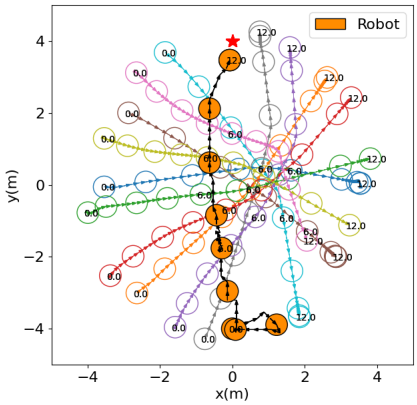
Overall, our method outperforms the baseline because it more effectively captures the evolving relationships in the environment, including changes in positional information and inferences about crowd intentions. Combined with spatial and temporal GATs, our method is able to better adapt to dense and partially observable scenarios with dynamic varying numbers of humans.

6.3.4 Ablation Study

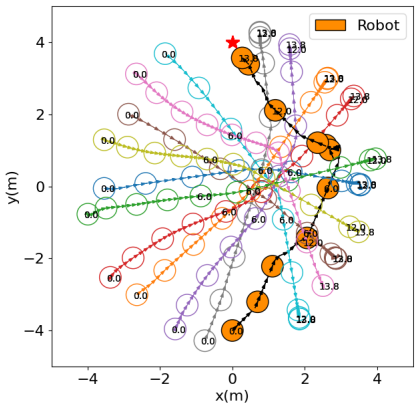
To assess the individual contributions of the different components in our model, I conduct an ablation study where I independently train and test the spatial and temporal branches with the same conditions as before. Fig. 6.6 shows that the spatial graph and ASTG learn to navigate a safer side away, while the spatial graph has a longer navigation time. Furthermore, the temporal graph and ASTG are good at reasoning about the dynamics of the crowd, while ASTG demonstrates better social norm compliance. These findings suggest that incorporating the spatial and temporal graph can significantly improve the performance and better adapt to dense scenarios with dynamic varying numbers of humans.



(a) Only spatial graph



(b) Only temporal graph



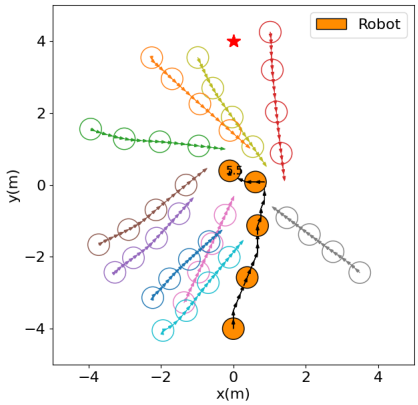
(c) Full ASTG

Figure 6.6: Simulation trajectories on the same testing case.

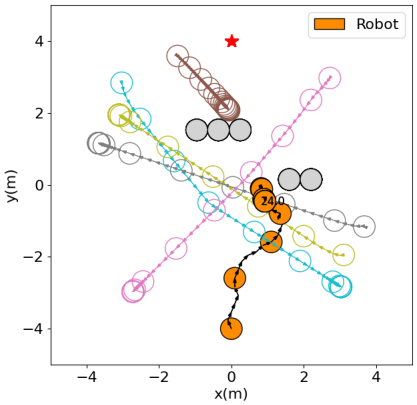
6.3.5 Qualitative Evaluation

To qualitatively evaluate the effectiveness of our method, I show in Figure 6.7 the robot's trajectories for different methods under simple and complex scenarios. In the simple scenario with ten dynamic humans, Figure 6.7(left), SARL (*Chen et al.*, 2019) collides with humans, while SG-DQN (*Zhou et al.*, 2022) detours largely to achieve the goal. In contrast, our method slightly detours at first and then directly reaches the goal, resulting in a shorter path.

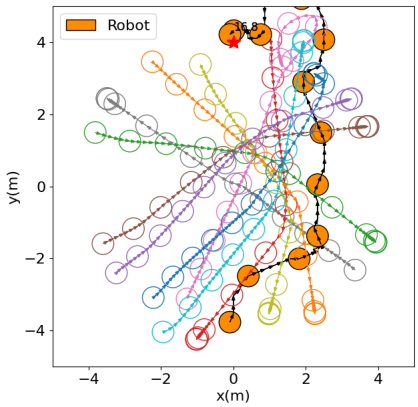
When the scenarios are extended from simple to complex, Figure 6.7(right) shows SARL (*Chen et al.*, 2019) and RGL (*Chen et al.*, 2020b) are having timeouts in the static group scenario. Combined with Figure 6.5, I can find that SARL (*Chen et al.*, 2019) and RGL (*Chen et al.*, 2020b) have high timeout rates for all static group scenarios, where the rates increase significantly as the group becomes more complex. This indicates that the freezing robot problem is prevalent for these two methods. As shown in Figure 6.7(f), our ASTG outperforms these two methods by slowing down at first and then going through the crowd. This is because our method is far-sighted when exploring the



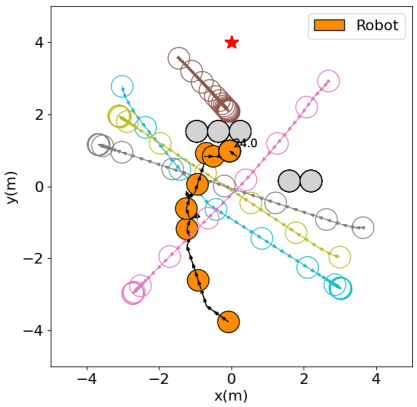
(a) SARL



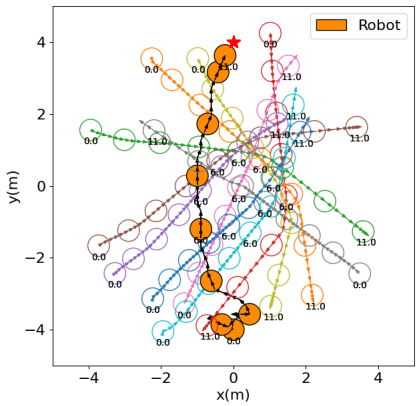
(d) SARL



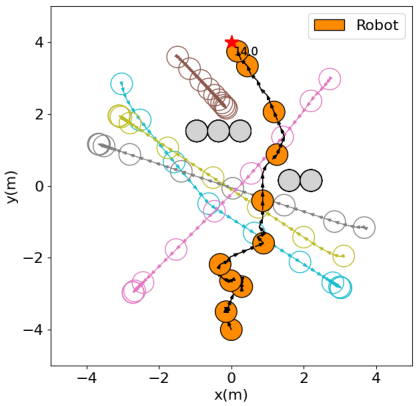
(b) SG-DQN



(e) RGL

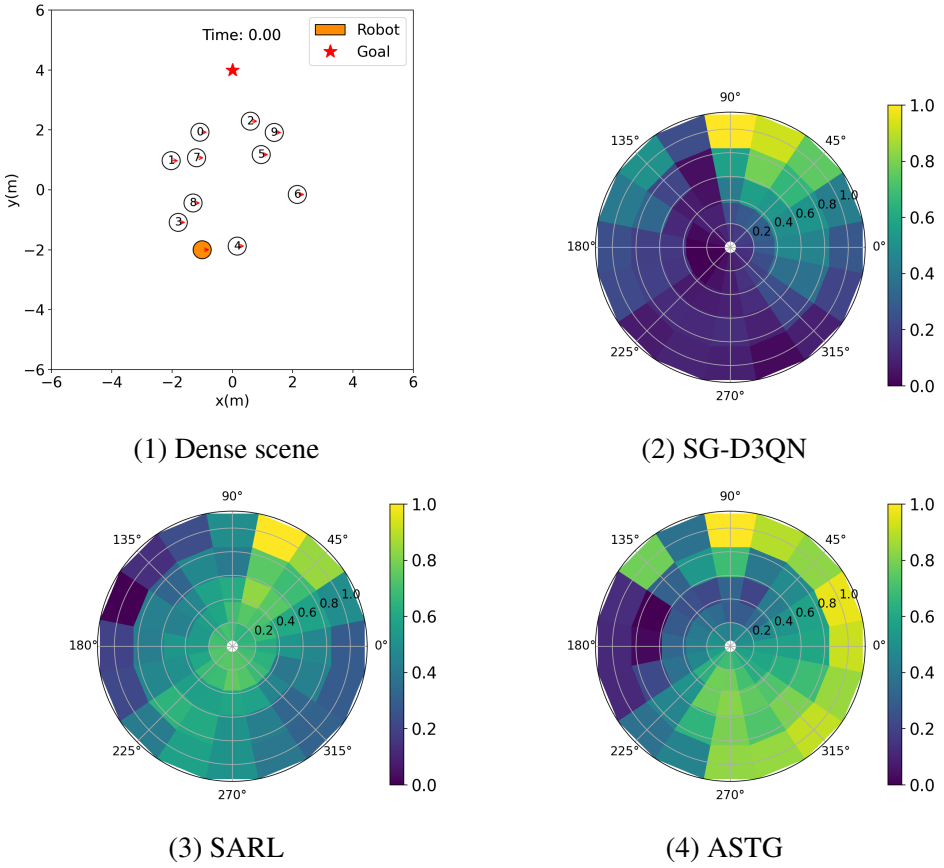


(c) Ours



(f) Ours

Figure 6.7: Trajectory comparisons of different methods under simple and complex scenarios. (a)(b)(c) The left pictures show the trajectories of SARL, SG-DQN and our ASTG in the scenarios with 10 dynamic humans. (d)(e)(f) The right pictures show the trajectories of SARL, RGL and our ASTG in the scenario with group (row3and2) static humans. The red star denotes the robot's goal. The filled orange circle denotes the robot. The hollow circles are dynamic humans while the filled gray circles are static humans.



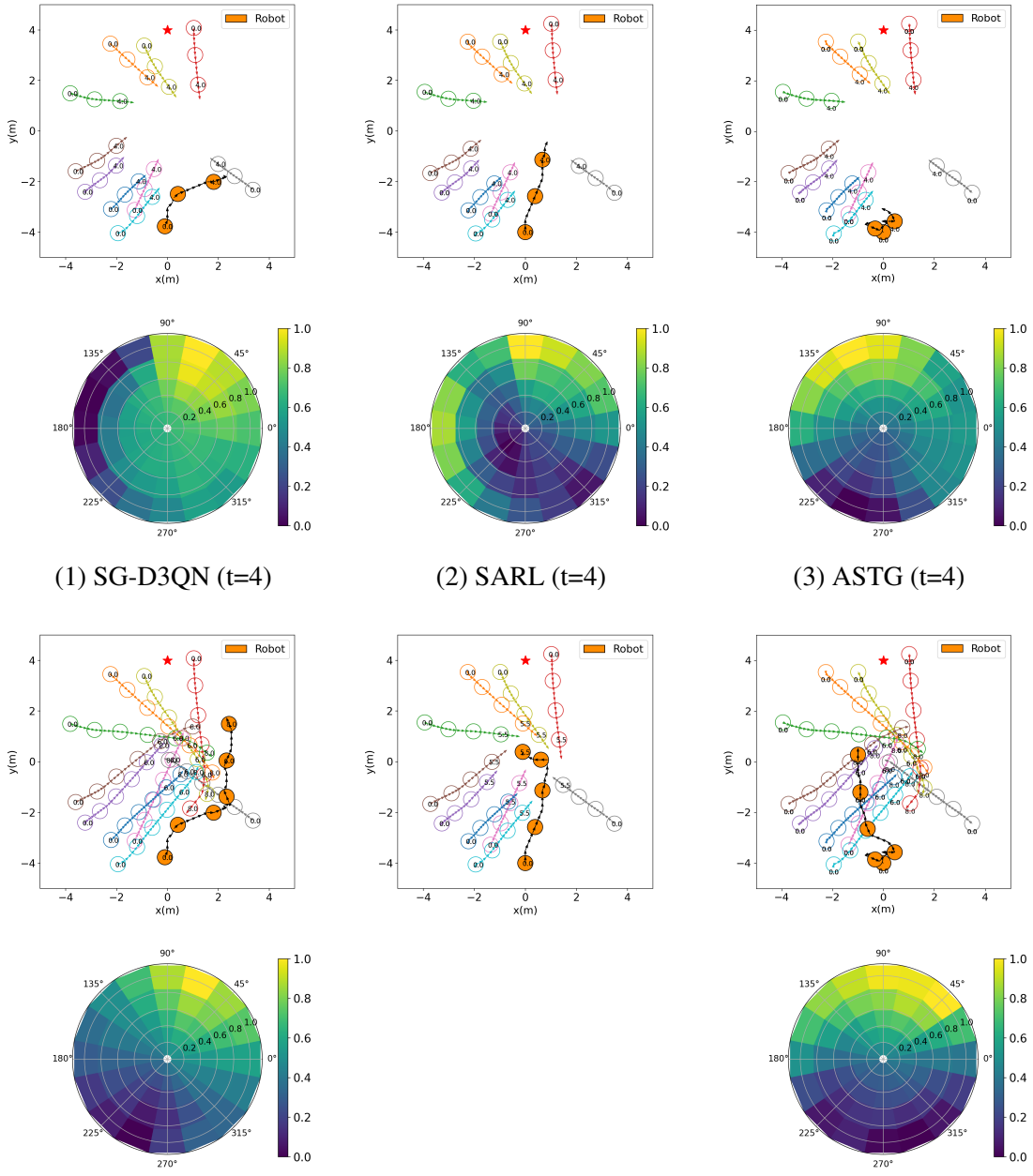
(1) Dense scene (2) SG-D3QN

(3) SARL (4) ASTG

Figure 6.8: Value estimations by different methods for the dense scene (1). The dense scene (1) is formed by 10 dynamic humans and one robot. SARL model prefers to take full speed at 67°, while SG-D3QN model and SARL model achieve the high value for high speeds on the rotation of 90°, preparing to pass behind dynamic humans #3 and #8.

environment. Also in the example of qualitative trajectories, I see the advantages of spatial-temporal graph reasoning, which helps to understand relationships between agents over time and space and use that to make more informed decisions. Our ASTG method thus is able to better adapt to dense and complex situations in uncertain environments.

In the following, I also compare the state values estimated by different methods under the same environment in Figure 6.8. Considering that dynamic humans in dense crowds tend to gather on the robot's linear path to its target, it's crucial to note that these aggregations spread out over time. Therefore, the robot should opt for paths with lower crowd density to minimize the risk of collisions. SARL model, taking into account the current spatial position, predicts danger from 135° to 180° (humans #3 and #8), thus choosing to move at full speed in the 67° direction to avoid collisions. It is also observed that the SARL model's likelihood of choosing other actions is almost zero. Similarly, both the SG-D3QN model and the ASTG model tend to turn towards 90° taking full speed, moving behind dynamic humans #3 and #8, and they effectively bypass the congregated crowd. However, unlike the ASTG model, the SG-D3QN model predicts taking full-speed values from 45° to 90°, with almost



(1) SG-D3QN (t=4) (2) SARL (t=4) (3) ASTG (t=4)

(4) SG-D3QN (t=8) (5) SARL (collision) (6) ASTG (t=8)

Figure 6.9: Value estimations and environments for SG-D3QN, SARL and our ASTG with the same episode at different timestep ($t = 4$ and $t = 8$). The first row and the third rows show the trajectories of different methods at $t = 4$ and $t = 8$, separately. The second and fourth rows correspond the value estimations at $t = 4$ and $t = 8$, separately. The full trajectories of each method are depicted in Figure 6.7 (a), (b), (c). (1) and (4) show that SG-D3QN is overly conservative, choosing to turn right at a large angle to avoid the dynamic dense crowd aggregation, creating a timeout. (2) and (5) show that SARL was too risky in rushing to its destination at full speed, resulting in a collision. In contrast, (3) and (6) show that our ASTG would initially slow down from left to right to avoid the approaching dynamic humans and then navigate past the gathered crowd at a small angle, balancing efficiency and safety.

zero likelihood of choosing actions in other directions. From this, I can infer that with increasing crowd density, both SG-D3QN and SARL models are likely to struggle to take wise actions, leading to the "freezing robot problem".

In addition, I compare the value estimation and environment's states of different methods at different time steps ($t = 4$ and $t = 8$) in the same episodes in Figure 6.9. By combining the complete trajectories in the Figure 6.7 (a), (b) and (c), I can compare the different methods more clearly. The SARL model based on attention reasoning is very short-sighted, it is too risky to move towards the destination at full speed, resulting in a collision with the dynamic crowd. SG-D3QN model and ASTG model are both based on spatial reasoning and choose to deviate from rightward/leftward to avoid dense crowd gatherings. However, compared to the small-angle arc trajectory of ASTG, the trajectory generated by SG-D3QN, which lacks the intention reasoning of the dynamic crowd, is too conservative, and thus a timeout occurs. Therefore, combined with spatial graph and temporal graph, ASTG could reason about both spatial information and the intention of dynamic humans.

6.3.6 Comparison with FSRL

Further, I conducted comparative experiments involving the standalone ASTG method, the FSRL method, and their combination - the ASTG+FSRL method. In the ASTG+FSRL method, I adopted a dense reward function Equation (5.16) to replace the original sparse reward function (Equation 5.10). The experiments were conducted separately for two different types of robots: non-holonomic robots and holonomic robots (Figure 4.2).

According to the Figure 6.10and Figure 6.11, in almost all test environments, the standalone ASTG method demonstrated significant advantages over both the FSRL method and the ASTG+FSRL method, regardless of whether the robots were non-holonomic or holonomic. The ASTG method not only showed a higher success rate but also significantly fewer timeout cases, resulting in a shorter overall weighted navigation time. This outcome highlights the strong advantage of the ASTG method in integrating spatial and temporal graphs, particularly in understanding and adapting to dynamic crowd behaviors in various environments.

However, for non-holonomic robots, the ASTG+FSRL method exhibited the lowest discomfort rates. This could be due to the more complex movement and control constraints faced by non-holonomic robots, such as the inability to move or turn freely in all directions. In these situations, the ASTG+FSRL method adapts better to these specific movement and control constraints of the robots, effectively reducing discomfort instances. On the other hand, for holonomic robots, which can move freely in any direction, showing greater spatial flexibility, the standalone ASTG method alone suffices for effective navigation, especially in dealing with complex crowd dynamics and social norms. This advantage of the ASTG method is particularly important for freely moving holonomic robots, making it perform best in terms of discomfort rates.

Figure 6.10: Quantative results comparison for a nonholonomic robot under FSRL method, ASTG method, and the combination method ASTG+FSRL.

Metrics	$Disc.(\%)$			$t_{succ.nav}$			$t_{weighted.nav}$		
N dynamic humans without static humans									
N(dynamic)	5	10	15	5	10	15	5	10	15
FSRL	0.03	0.13	0.25	**10.68**	**12.02**	12.98	**11.02**	13.97	18.29
ASTG	**0.01**	0.07	0.18	11.11	12.15	**12.78**	11.15	**13.03**	**15.20**
ASTG + FSRL	**0.01**	**0.05**	**0.14**	11.37	12.51	13.38	11.47	13.86	15.40
N static humans with 5 dynamic humans									
N(static)	2	4	8	2	4	8	2	4	8
FSRL	0.07	0.10	0.22	**10.53**	**10.49**	**10.80**	11.13	11.87	13.88
ASTG	**0.03**	0.06	0.19	10.95	11.16	11.95	**11.11**	**11.69**	**13.37**
ASTG + FSRL	**0.03**	**0.05**	**0.10**	11.18	11.68	12.79	11.40	12.08	13.75
5 dynamic humans with different static groups									
N(static)	DS	RO	CO	DS	RO	CO	DS	RO	CO
FSRL	0.13	0.11	0.15	**10.56**	**10.94**	**10.66**	12.66	12.50	12.65
ASTG	0.09	0.08	**0.07**	11.19	11.30	11.23	**12.01**	**11.99**	**12.57**
ASTG + FSRL	**0.07**	**0.06**	**0.07**	11.99	11.59	11.69	12.56	12.19	12.90

Table 6.4: Evaluation performance comparison on different scenarios.. "Disc. (%)" is the average frequency of duration that the human invades the comfort area of the robot. "$t_{succ.nav}$" is the average navigation time of success cases. "$t_{weighted.nav}$" (Equation 6.15 is a novel weighted navigation time metric considering the impact of the discomfort steps and collision cases.

Figure 6.11: Quantative results comparison for a holonomic robot under FSRL method, ASTG method, and the combination method ASTG+FSRL.

Metrics	$Disc.(\%)$			$t_{succ.nav}$			$t_{weighted.nav}$		
N dynamic humans without static humans									
N(dynamic)	5	10	15	5	10	15	5	10	15
FSRL	0.01	0.08	0.17	**11.34**	**12.32**	**13.08**	11.46	14.58	17.58
ASTG	**0.01**	**0.04**	**0.08**	11.40	12.65	13.28	**11.45**	**12.97**	**14.13**
ASTG + FSRL	**0.01**	0.06	0.11	11.66	13.25	14.23	11.73	13.72	15.18
N static humans with 5 dynamic humans									
N(static)	2	4	8	2	4	8	2	4	8
FSRL	0.04	0.09	0.16	**10.78**	**10.88**	**11.36**	11.25	11.56	12.83
ASTG	**0.03**	**0.06**	**0.10**	11.09	11.22	11.97	**11.06**	**11.56**	**12.61**
ASTG + FSRL	**0.03**	0.07	0.13	11.54	12.16	13.83	11.74	12.98	14.80
5 dynamic humans with different static groups									
N(static)	DS	RO	CO	DS	RO	CO	DS	RO	CO
FSRL	0.10	0.08	0.04	**10.88**	**11.09**	**10.77**	11.75	11.76	12.21
ASTG	**0.06**	**0.06**	**0.03**	11.35	11.30	11.05	**11.65**	**11.75**	**12.07**
ASTG + FSRL	0.07	0.05	**0.03**	12.88	12.33	12.49	12.98	12.68	13.66

Table 6.5: Evaluation performance comparison in the simple scenarios. "Disc. (%)" is the average frequency of duration that the human invades the comfort area of the robot. "$t_{succ.nav}$" is the average navigation time of success cases. "$t_{weighted.nav}$" (Equation 6.15 is a novel weighted navigation time metric considering the impact of the discomfort steps and collision cases.

6.4 Summary

In this chapter, I proposed a novel attention-based spatial-temporal graph learning model (ASTG) for crowd robot navigation. By formulating spatial and temporal graphs with GATs, I implicitly compute both direct and indirect spatial interactions and temporal crowd features so that our model can reason for decision-making under changing scenarios. Integrating an RNN, our model implicitly incorporates the past trajectories into the temporal graph to also reason about the future intentions of each agent. In the experiments, I showed our model outperforms baseline methods and can handle both uncertain simple, and complex environments. Based on the promising results, I intend in future work to further analyse and extend the GAT-module, for example using multiple graph layers, which can help to extract deeper information from the crowd

CHAPTER 7

Social-Aware Robot Navigation in Partially Observable Environments

In the previous chapter, I introduced a novel attention-based spatial-temporal graph learning model (ASTG). While this method demonstrated excellent performance in social-aware robot navigation, it is not entirely applicable in scenarios with limited sensor perception range. This is because the limited range of sensors can lead to incomplete and even highly uncertain environmental information. Therefore, navigating robots in partially observable environments remains a significant challenge. To achieve collision avoidance in congested, partially observable environments, I propose a novel deep reinforcement learning architecture in this chapter. This approach combines spatial graphs and attentional reasoning to address this issue. Human-observed relative positions and velocities are treated as nodes in the spatial graph. At the same time, interactions between the robot and humans are modeled as nodes in the attention graph, capturing the spatial relationships between the robot and humans. In this manner, the proposed approach enhances the modeling of relationships between the mobile robot, static obstacles, and surrounding individuals. Consequently, in congested scenarios where the robot has limited sensor range, the proposed navigation framework significantly outperforms state-of-the-art methods in collision reduction. Additionally, the application of parallel Double Deep Q-learning substantially reduces training time.

Contents

7.1 Introduction

In reinforcement learning, observed information holds primary importance as the foundation for effective decision-making. The information an agent acquires from the environment directly influences

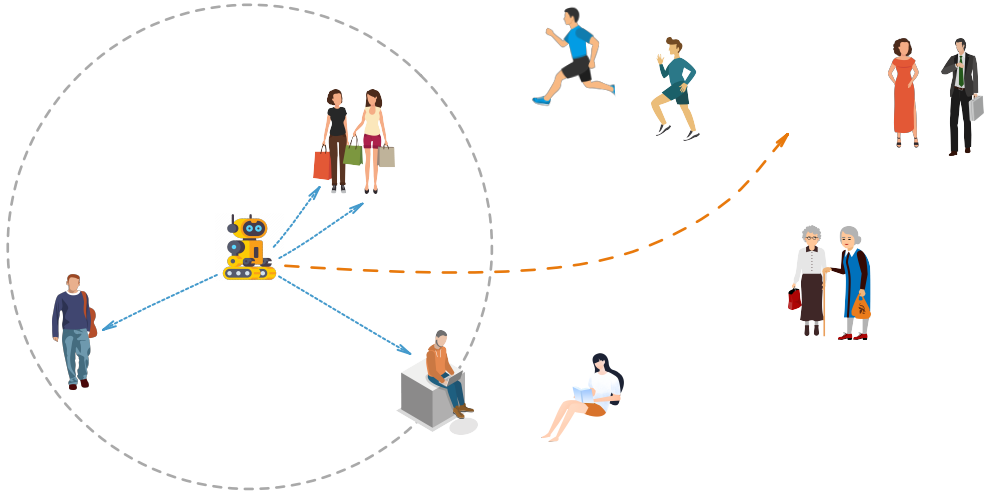


Figure 7.1: Motivation of our work. In the navigation process, the robot's observation is limited to humans within the specified sensor range. I represent the relationship between the robot and the observed humans using two distinct graphs. Initially, I consider the positions and velocities of humans relative to the robot as nodes in a spatial graph. Subsequently, I treat each robot-human pair as a node within the attention graph. By integrating these two graphs within a deep reinforcement learning framework, the robot's navigation behavior is notably enhanced compared to the current state of the art.

its actions taken to maximize cumulative rewards over time. Accurate and relevant observations enable the agent to establish a robust understanding of its surrounding environment, leading to more effective strategy learning and action selection.

In the previous chapters, it was assumed that all human states could be observed throughout the navigation process. While the proposed methods have achieved success, they rely entirely on fully observable environmental information. Specifically, this assumption implies that the robot can acquire complete, accurate, and real-time information, enabling it to perceive and understand all environmental elements relevant to the navigation task. By leveraging these comprehensive observational inputs, socially aware robots can create a comprehensive representation of the environment, identify potential collision risks, and optimize their paths while respecting social norms and minimizing interference.

However, this assumption is not feasible under real-world conditions. Typically, depth-based observations are obtained from sensors such as LiDAR or depth cameras. Due to limitations in the perceptual range of sensors, whether visual or laser-based, they cannot cover the vast environment entirely, preventing the robot from obtaining information about all human states. This leads to the realization that learning methods based on this idealized assumption are not universally applicable across different sensor setups. Furthermore, since the environment often comprises both static and dynamic agents, and an agent's dynamics may change at any moment due to individual awareness and implicit social interactions, the uncertainty within the crowd increases. This heightened uncer-

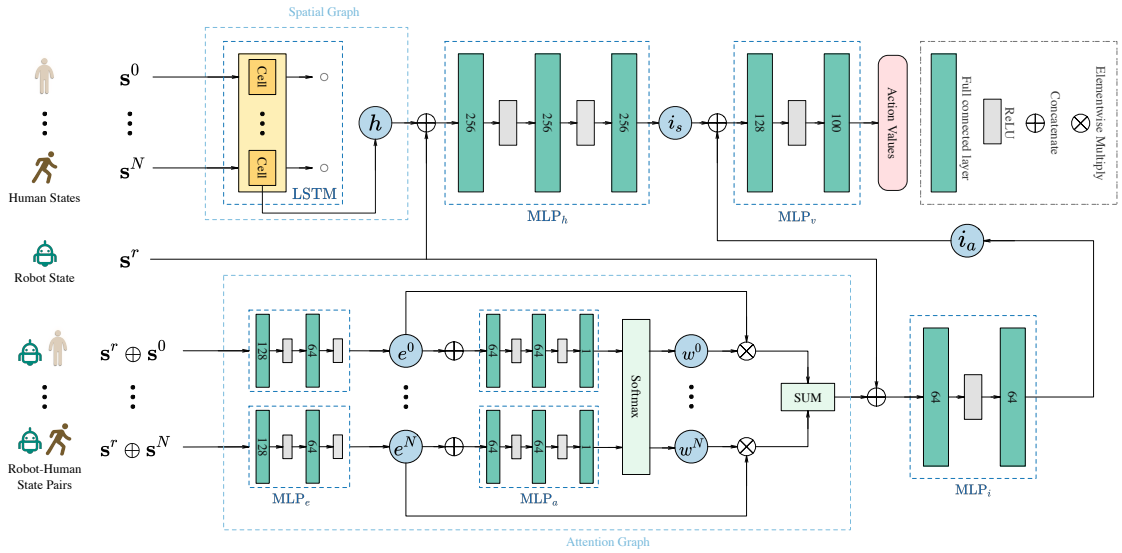


Figure 7.2: Overview of our ESA graph architecture. It illustrates three key components: (a) The upper spatial graph shows our utilization of an LSTM (*Hochreiter and Schmidhuber*, 1997) for encoding spatial relationships between the robot and other agents. Here, I first arrange each observed human in descending order according to the weighted sum of their present distance and potential future distance. And then, I utilize their states (comprising positions and velocities) as inputs for the LSTM. (b) The below attention graph involves combining the robot's state and the state of the i-th human, channeling this through the embedding network MLP$_e$ and the attention network MLP$_a$ to formulate human-robot interactions. (c) At last, the final network combines the outputs from both modules to generate action values for guiding decision-making.

tainty in modeling the environment under partial observability contributes to increased navigation complexity.

To address these challenges, in this chapter, I propose a novel approach named Enhanced Spatial Attention (ESA) graph for robot navigation in crowded, partially observable scenes. I model high-level relationships using Spatial Long Short-Term Memory (LSTM) and attention mechanisms Given that agents cannot access complete information about the entire environment state in partially observable settings, this limits their comprehensive understanding of the current state, leading to state value estimations in Deep V Learning (Section 3.3.1.1) based on incomplete or biased information. Therefore, in such special cases, I prefer to choose Deep Q Learning (Section 3.1.4) over Deep V Learning (Section 3.3.1.1) for training the network. Additionally, to mitigate the issue of overestimation, I opt for training the network using Double Deep Q Learning (DDQN) (*Van Hasselt et al.*, 2016; *Fujita et al.*, 2021) (Section 3.3.1.2). I first model the crowd navigation scene as a distributed spatial graph to encode spatial relationships between the robot and other agents. Moreover, I capture the significance of each robot's interaction with humans to form an attention graph. Lastly, I combine these two graphs to generate the robot's next action. Moreover, due to the high parallelizability of DQN, which allows training on high-performance hardware like GPUs, DDQN significantly shortens the time required for value estimation.

7.2 Social-Aware Navigation with Partial Observation

7.2.1 Problem Formulation

Based on the general problem formulation of DRL-based social-aware navigation presented in Section 4.2, $\mathbf{s}_t^r$ represents the robot's state and $\mathbf{s}_t^i$ represents the state of the i-th human at timestep t. Different from previous chapters, I define the partially observed state $\mathbf{s}_t^o = [\mathbf{s}_t^r, \mathbf{s}_t^0, \mathbf{s}_t^1, ..., \mathbf{s}_t^N] \in \mathcal{S}$ as a joint state that is composed of the robot's state $\mathbf{s}_t^r$ and the states of N humans who are scanned by the robot's limited sensors at time step t, where $N(\geqslant 0)$ may vary at each time step.

When navigating in a new episode, the robot starts from the initialized observed state $\mathbf{s}_0^o \in \mathcal{S}$. At each time step t, based on the unknown state transition function $\mathcal{P}(\mathbf{s}_{t+1}^o|\mathbf{s}_t^o, \mathbf{a}_t)$, the robot samples an action from the learning policy $\pi(\mathbf{a}_t|\mathbf{s}_t^o)$, transitioning to its next state $\mathbf{s}_{t+1}^r$. Meanwhile, all other agents take their actions and move to their next states $\mathbf{s}_t^i$ according to their respective policies. At the end of an episode, I employ Monte Carlo value estimation (*Kroese et al.*, 2013) (Section 3.1.5) to estimate the state-action values for each time step within the episode. Finally, the optimal policy π^* can be expressed as follows:

$$\pi^*(\mathbf{s}_t^o, \mathbf{a}_t) = \underset{\mathbf{a}_t}{\operatorname{argmax}} \left[R(\mathbf{s}_t^o, \mathbf{a}_t) + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}_{t+\Delta t}^o, \mathbf{a}') \right] \tag{7.1}$$

where Q^* refers to the optimal action-state value function from DDQN (*Van Hasselt et al.*, 2016), and Δt represents the duration of each time step.

I adopt the reward function formulation as defined in *Chen et al.* (2019) outlined in Section 4.2.3, which provides rewards for task accomplishment while simultaneously imposing penalties for collisions or uncomfortable distances.

7.2.2 Enhanced Spatial Attention (ESA) Graph Struture

As illustrated in Figure 7.2, a novel framework ESA is presented for social-aware robot navigation under a partially observed environment. It comprises two primary pipelines, processing three distinct types of inputs:

(i) the states of static and dynamic humans observed within the robot's sensor range,

(ii) the own state of the robot

(iii) the concatenation of the robot's state and the states of observed humans.

In the following subsections, I introduce the architecture and formulations of each pipeline, namely the spatial graph and the attention graph, both focused on the observed humans. These graphs are in the form of fully connected graphs, where the number of nodes corresponds to the number of observed people. In these graphs, each node represents an individual, and the edges between nodes are weighted based on spatial or attention relationships. This structure allows us to precisely capture and analyze the interactions and relationships between agents, providing a comprehensive perspective for understanding and addressing human behavior. For simplicity, I omit the time index t in the subsequent discussions.

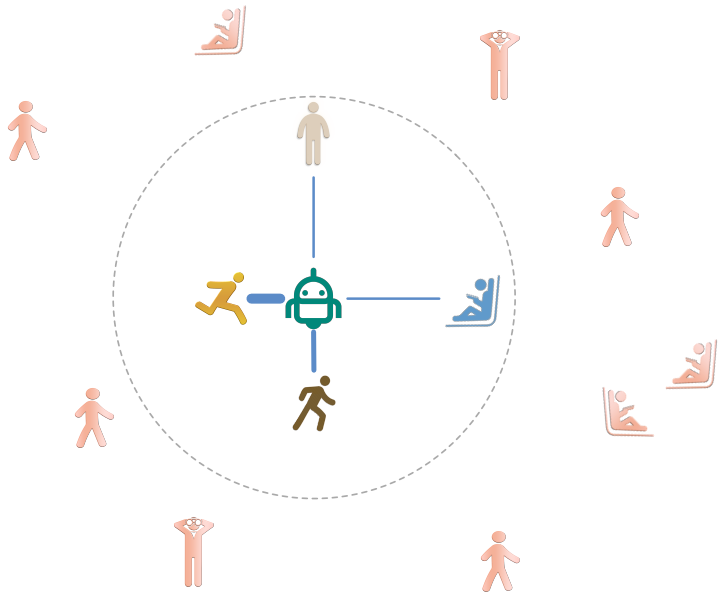


Figure 7.3: Structure of the spatial graph. The states of the robot and humans denote the nodes and the spatial relationship between the robot and its neighbor denote the edges of the graph. The light-grey dashed circle signifies the sensor range of the robot, within which the robot can perceive the positions and velocities of the humans and establish spatial relationships with those within the sensor range. When evaluating the significance of each human to the robot, I consider not only current positions but also future positions estimated by observed velocities. The importance level is indicated by the width of the line, where closer distances correspond to greater importance, resulting in wider lines. The states are then fed into the LSTM, ordered in descending distance.

7.2.2.1 Spatial Graph on Observed Humans

In order to handle the spatial features of humans, I treat each sensored agent (the robot or humans) as a node of the graph, while the edges signify the spatial relationships, as illustrated in Figure 7.3. Most of the learning-based methods for local obstacle avoidance have traditionally employed feed-forward neural networks to encode these spatial relationships (*Chen et al.*, 2017b,a). However, such approaches often necessitate fixing the input size or converting spatial relations into occupancy grid maps, followed by the utilization of CNNs (*LeCun et al.*, 2010) and pooling techniques. Grid maps can prove to be either coarse or computationally demanding when aiming for high precision. In our case, as shown in Figure 7.4, I leverage an LSTM to manage the various quantities of observed humans and designate each human state $\mathbf{s}_t^h$ at the current time t as the input for LSTM cells, ordered inversely according to their balanced distance from the robot:

$$\mathbf{h}^i = \text{LSTM}(\mathbf{h}^{i-1}, \mathbf{s}_t^i), \tag{7.2}$$

where $\mathbf{h}_i$ refers to the i-th hidden state of the LSTM. For simplicity, from now on $s_t^{r/i}$ will be dentoed as $s^{r/i}$.

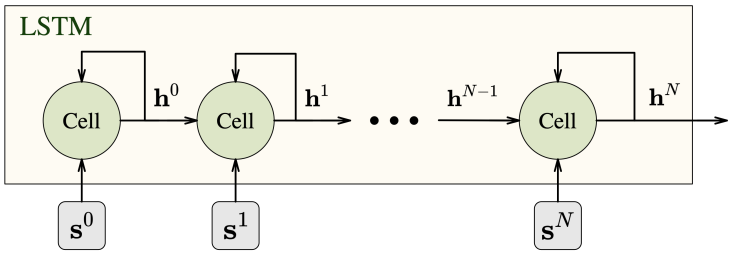


Figure 7.4: Structure of the LSTM unfolded to show each input. At each step, the detected human feeds its states into an LSTM cell sequentially, which stores the related information in the hidden states $\mathbf{h}^i$. The final hidden state $\mathbf{h}^N$ encodes the entire state of all detected humans into a fixed-length vector. The order of humans is sorted by decreasing the balanced distance (Equation 7.4) so that the closest human has the most latest effect on $\mathbf{h}^N$.

As shown in Chapter 4.2.1, the transformed states of the robot and humans are:

$$\begin{aligned} s^r &= [d_g, v_x^r, v_y^r, v_{pref}, r], \\ s^i &= [p_x^i, p_y^i, v_x^i, v_y^i, r^i, d^i, r^i + r], \end{aligned} \tag{7.3}$$

where d_g and d^i represent the robot's distance to the goal and to its i-th neighbor respectively.

For humans, I balance their current distance (the distance between the human's current position and the robot) and the estimated next distance (the distance between the position where the human is expected to move and the robot) to establish a new distance as the criterion for sorting the input data for the LSTM. The predicted future position is determined based on their current moving speed. With the above notions, for each human i, it balanced distance is computed as:

$$\mathbf{dist}' = w_c({p'_x}^2 + {p'_y}^2)^{1/2} + (1 - w_c)\left((p'_x + v'_x\Delta t)^2 + (p'_y + v'_y\Delta t)^2\right)^{1/2}, \tag{7.4}$$

where $p'_{x/y} = p^i_{x/y} - p^r_{x/y}$ and $v'_{x/y} = v^i_{x/y} - v^r_{x/y}$. The hyper-parameter $w_c \in [0, 1]$ is used to determine the significance of the current distance to the robot and the potential future distance. In our experimental setup, I assign a value of 0.8 to w_c.

Furthermore, I enable the LSTM network to retain information about all human states by maintaining a high-dimensional output hidden state, without significantly increasing the time overhead. As depicted in the upper module of Figure 7.2, I initially feed the states of humans and the robot's own state into a multi-layer perception (MLP):

$$\mathbf{i}_s = \mathrm{MLP}_h(\mathbf{h}^i, \mathbf{s}^r), \tag{7.5}$$

which ultimately yields encoded spatial relationships $\mathbf{i}_s$ between the robot and the observed humans.

7.2.2.2 Attention Graph on Observed Humans

In addition, drawing inspiration from *Chen et al.* (2019), I establish an attention graph that encodes the interactions between the robot and observable humans in pairs.

In order to deal with the variable amounts of scanned humans during each observation, I first employ a non-linear transformation, such as ReLU activation function, to each robot-human interaction pair to obtain preliminary embedding features:

$$e^i = \text{MLP}_e(\mathbf{s}^r, \mathbf{s}^i), \tag{7.6}$$

Each embedding vector e_i is subsequently fed into another MLP and transformed into its corresponding attention weight:

$$w^i = \text{MLP}_a(e^i), \tag{7.7}$$

Here, please be aware that the final layer of MLP_a lacks a layer of ReLU activation. Additionally, both MLP_e and MLP_a utilize the same parameters during the calculation of each interaction. In comparison to SARL (*Chen et al.*, 2019), our ESA network architecture is simplified by dropping the mean pooling module in Figure 5.3 and deeper embedding. This adjustment allows us to achieve comparable performance in less time.

Subsequently, I obtain the interaction representation $\mathbf{a}$ of all pairs through a weighted sum of pairwise interaction vector e^i and the corresponding attention weight w^i:

$$\mathbf{a} = \sum_{i}^{N} w^i e^i, \tag{7.8}$$

Finally, I feed both the interaction representation $\mathbf{a}$ and the robot's state into an additional MLP to get the encoded attention feature:

$$\mathbf{i}_a = \text{MLP}_i(\mathbf{a}, \mathbf{s}^r) \tag{7.9}$$

7.2.2.3 Final Output

Upon independently processing information through the attention branch and the spatial graph branch, I construct an extra non-linear transformation. This transformation incorporates ReLU activations and utilizes the spatial feature $\mathbf{i}_s$ and attention feature $\mathbf{i}_a$ as inputs. The resulting output is estimated action value $\mathbf{q}$, serving as the learned indicator for cooperative path planning:

$$\mathbf{q} = \text{MLP}_v(\mathbf{i}_s, \mathbf{i}_a) \tag{7.10}$$

It can be observed that when the number of people in the environment changes, ESA algorithm's spatial graph processes a variable number of observed humans using LSTM, outputting fixed-length spatial vectors. Meanwhile, the attention graph aggregates interaction features between pairs through a pooling mechanism. Ultimately, ESA combines features from both the spatial and attention graphs as inputs for value function estimation. This integration of spatial and attention information enhances the flexibility and adaptability of the ESA algorithm in partially observable environments, allowing it to effectively manage diverse crowd sizes.

7.3 Experiments

7.3.1 Environment Setup

In line with the preceding chapters, I establish the simulation environment with Gym (*Brockman et al.*, 2016) and RVO (*Van den Berg et al.*, 2008) as detailed in *Chen et al.* (2019). In partially

observable environments, my primary focus is on testing and validating the effectiveness and feasibility of new navigation algorithms. Robots with non-holonomic dynamics face more movement constraints, which can significantly increase the complexity of training. Therefore, my experiments have only considered the robot with holonomic dynamics. Navigation of robots with non-holonomic dynamics in crowded settings is one of our future research directions.

The behavior of all agents, except the robot, adheres to the ORCA policy (*Van Den Berg et al.*, 2011). Throughout all experiments, the robot remains in an invisible setting to prevent excessive human reactions. This approach ensures that the robot learns a policy primarily focused on direct goal-reaching, rather than adopting an overly aggressive strategy. For the conducted experiments, I configure scenarios involving circle-crossing, with the robot's initial position set at $(0, -4)$ and its goal position set at $(0, 4)$. These circle-crossing scenarios have a radius of 5m for both training and testing. To maintain a realistic representation of real-world conditions, I restrict the robot's sensor range to 2.5m. This means that the robot's observations are limited to humans within this radius, rather than encompassing all humans in the environment at all times.

7.3.2 Training and Testing

7.3.2.1 Training

I employed the parallel Double DQN algorithm (Section 3.3.1.2) to train our ESA, utilizing PyTorch (*Paszke et al.*, 2017) and a deep reinforcement learning library, PFRL (*Fujita et al.*, 2021) for implementation. The batch size was set to 32, and I utilized the Adam optimizer (*Kingma and Ba*, 2014). To assess performance, I conducted a comparative analysis between our ESA method and GA3C-CADRL (*Everett et al.*, 2018) as well as SARL (*Chen et al.*, 2019). All three methods were trained using 10k episodes in the environment featuring 5 dynamic humans and 2 static humans. To ensure a fair comparison, I standardize the reward function, optimizer, training batch size, and learning rate across all methods. Unlike the approach presented in *Chen et al.* (2019), I abandon the utilization of imitation learning. This decision was based on the poor performance of ORCA (*Van Den Berg et al.*, 2011) in partially observable environments. ORCA relies heavily on the accurate perception of surrounding agents to compute collision-free trajectories. In partially observable environments, where a robot or agent's sensors cannot detect the entire environment, this could lead to inadequate or incorrect collision avoidance decisions. This renders it unsuitable as a qualified expert for demonstration in partially observable environments. Importantly, our modifications led to a significant reduction in training time. On a Mac Mini equipped with the M1 chip, I was able to achieve this reduction from approximately 10 hours to around 2 hours, as compared to the findings reported in *Chen et al.* (2019)

7.3.2.2 Testing

For testing phases, I configured scenarios with varying numbers of static humans and dynamic humans that different scenarios are the mostly same as Chapter 6. During training, I established a timeout limit of 60 seconds, ensuring adequate exploration for the agent. However, this limit was reduced to 24 seconds for testing. Across both the training and testing phases, distinct random seeds were assigned to different episodes. Consequently, within a given test episode, all approaches encountered scenarios with identical robot start and goal positions, maximum velocity, and humans' start and goal positions. However, the humans' start and goal positions differ in the episodes by

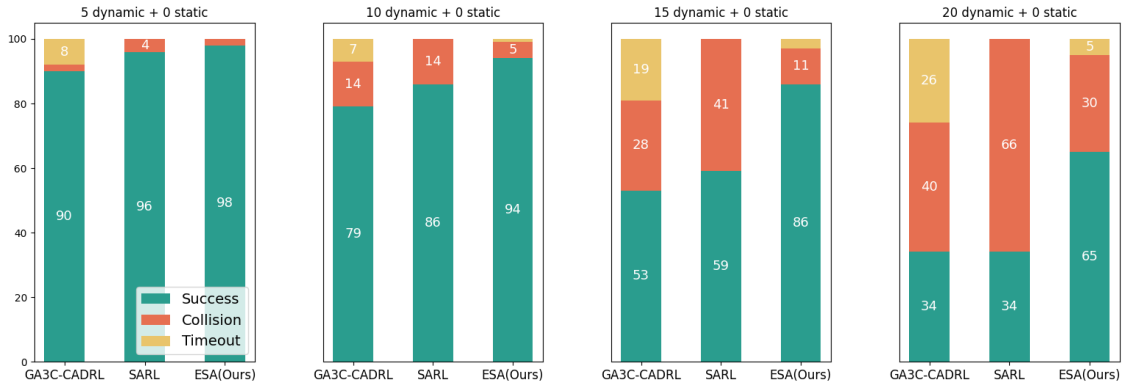


Figure 7.5: Quantitative evaluation of three methods in scenarios with varying numbers of dynamic humans. It shows that our ESA consistently exhibited the highest success rate and the lowest collision rate among these three methods. Compared to the other two approaches, GA3C-CADRL (*Everett et al.*, 2018) had the highest timeout rate and experienced more collisions than our ESA. In contrast, SARL (*Chen et al.*, 2019) never reached a timeout, but as the number of humans increased, its collision rate sharply rose.

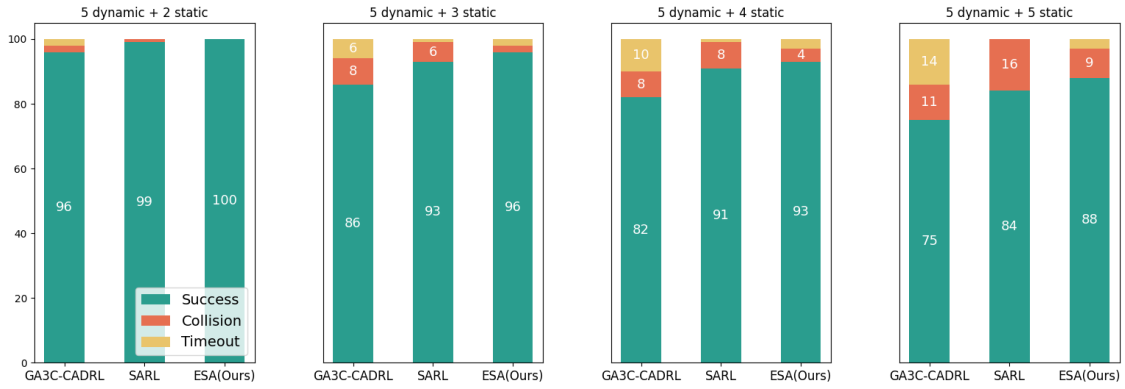


Figure 7.6: Quantitative evaluation on scenarios with 5 dynamic humans and varying numbers of static humans.

re-randomization. The metrics used in the testing phases are presented in Section 4.3.3.

7.3.3 Quantitative Evaluation

I conducted a comprehensive comparison across various metrics, including success rate, collision rate, timeout rate, average navigation and the average reward across all tested episodes. The results for the three methods in scenarios featuring different combinations of dynamic and static humans are illustrated in Figure 7.5, Figure 7.6, and Figure 7.7. Comparatively, SARL (*Chen et al.*, 2019) demonstrates a more aggressive behavior, resulting in shorter navigation time. However, it becomes

Methods	Numbers of dynamic humans			
	5	10	15	20
SARL	0.949	0.795	0.425	0.1
GA3C-CADRL	0.891	0.737	0.438	0.204
ESA	**0.972**	**0.909**	**0.800**	**0.531**

Table 7.1: Average rewards across 1,000 test cases in simple scenarios only with varying numbers of dynamic humans. Our ESA achieves the highest average rewards. SARL outperforms GA3C-CADRL in the first two scenarios but performs less effectively in scenarios involving 15 and 20 humans due to the much higher collision rate.

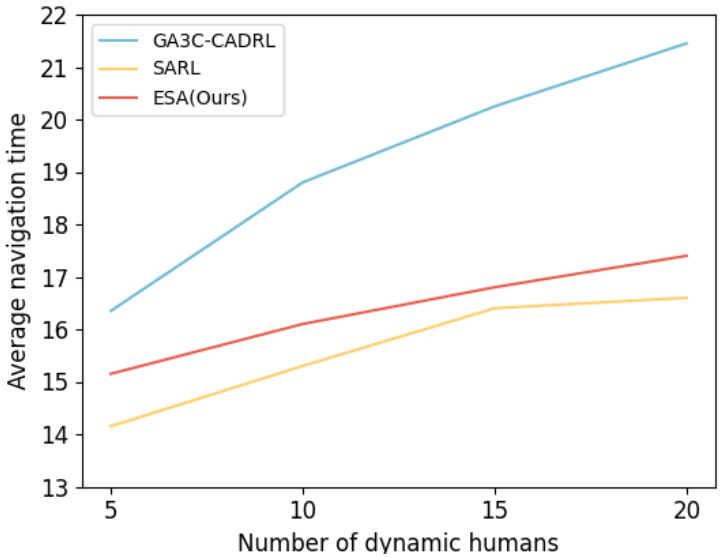
Methods	5 dynamic + additional number of static humans				
	1	2	3	4	5
SARL	0.953	0.944	0.902	0.876	0.770
GA3C-CADRL	0.930	0.889	0.828	0.781	0.696
ESA	**0.981**	**0.956**	**0.945**	**0.912**	**0.837**

Table 7.2: Average rewards across 1,000 test cases in complex scenarios with 5 dynamic humans and varying numbers of static humans. Similarly, our ESA outperforms the other two

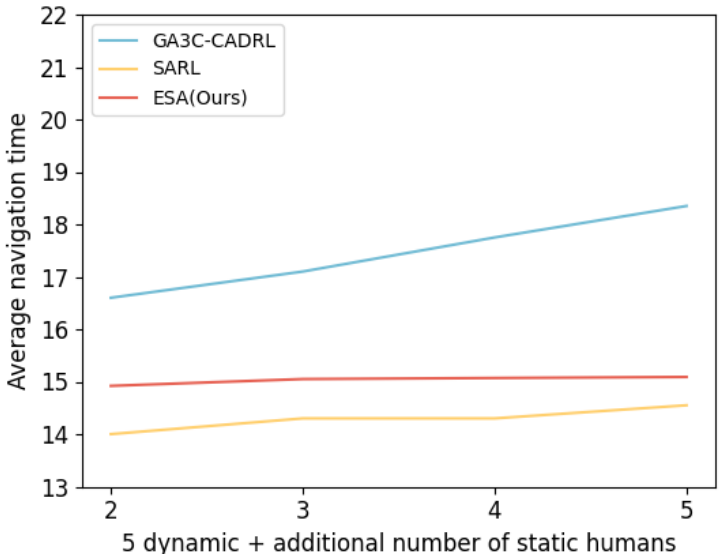
more prone to collisions as the number of obstacles increases. On the other hand, GA3C-CADRL (*Everett et al.*, 2018) adopts a more conservative approach, leading to a higher frequency of timeouts and longer navigation time, even in successful episodes. In contrast, our approach shows effective obstacle avoidance while maintaining navigation times comparable to SARL (*Chen et al.*, 2019) as the number of humans increases. The collision rate and navigation time of our method exhibit slower increments in both scenarios: pure dynamic environments and those featuring both dynamic and static humans. I also collected data on the average navigation time. SARL (*Chen et al.*, 2019) maintains a steady navigation time across various numbers of humans, whereas GA3C-CADRL (*Everett et al.*, 2018) and our ESA show slight increases as the human count grows. Despite a slight increase in our inference time, the maximum difference remains only 0.0003ms.

Furthermore, the average episode rewards for the various test scenarios are presented in Table 7.1 and Table 7.2. It is evident that our ESA outperforms the other two methods, particularly as the number of humans increases. Statistical analysis through paired t-tests confirms that the differences in success rate, collision rate, and average rewards are statistically significant. This reaffirms the robustness of our approach compared to others, especially considering that collision rate is a crucial metric in crowd navigation.

Although both Chapter 6 and this Chapter focus on testing the proposed method in scenarios different from the training environment, their core emphases are distinct. The former concentrates on verifying the method's robustness and generality in fully observable environments, while the latter focuses on its effectiveness in unknown environments with limited sensor perception. This difference highlights the various aspects of adaptability and effectiveness of the proposed methods



(a) Simple scenarios only with different numbers of dynamic humans.



(b) Scenarios with 5 dynamic humans plus different numbers of static humans as indicated.

Figure 7.7: Average navigation time for the episodes in different scenarios.

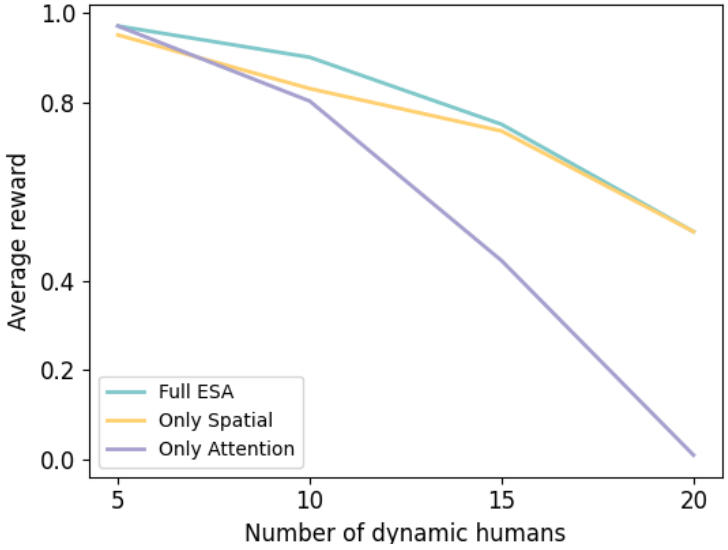


Figure 7.8: Average rewards on 1,000 test episodes with varying human numbers. I can clearly see that the attention graph outperforms the spatial graph when the number of humans is low, but its performance significantly deteriorates as the human count increases. However, when both graphs are combined for reasoning, our model excels across nearly all test scenarios.

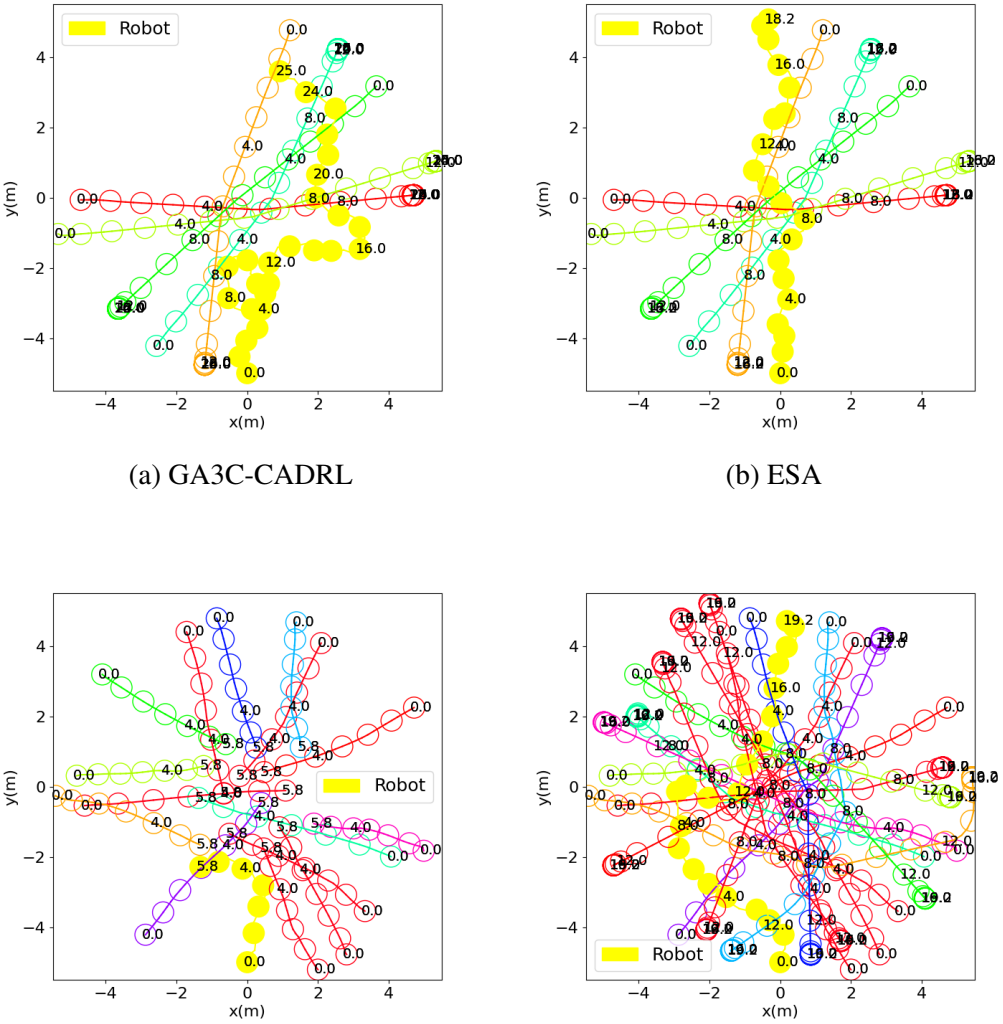
under different conditions.

7.3.4 Ablation Study

In order to assess the relative contributions of different components to the overall performance, I divided the spatial and attention branches into two distinct parts. I conducted training and testing using identical conditions as before. The results of the ablation study, presented in Figure 7.8, demonstrate that our ESA leverages the strengths of both branches, resulting in the most optimal performance.

7.3.5 Qualitative Evaluation

I further investigated the enhanced performance of our model through qualitative outcomes. All three RL-based methods can successfully navigate to the goal in scenarios containing just a few humans. However, GA3C-CADRL (*Everett et al.*, 2018) tries to keep conservative and waits for humans to move away. Figure 7.9 illustrates two representative scenarios involving obstacle avoidance while navigating through human crowds. When confronted with an environment containing five dynamic huamns, the robot controlled by GA3C-CADRL (*Everett et al.*, 2018) initially moves, but then drastically slows down and alternates between left and right movements from 5.0s to 16.0s, resulting in a timeout case. In contrast, our ESA robot initially hesitates, but subsequently identifies a more favorable path through the center, resulting in a shorter navigation time. As the number of dynamic humans increases to 15, SARL (*Chen et al.*, 2019) becomes more risk of moving directly into crowds, resulting in collisions. In contrast, our ESA robot first reduces its speed and then selects a shorter path toward the goal.



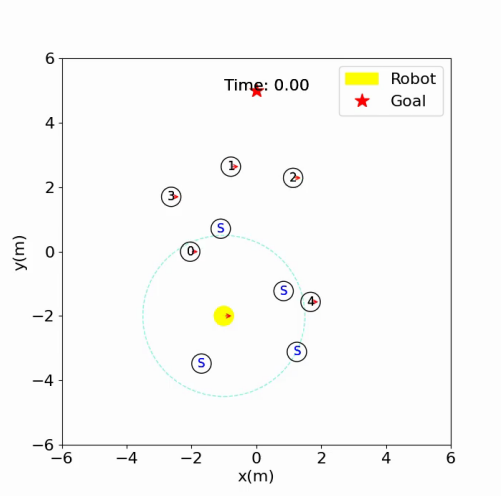
(a) GA3C-CADRL

(b) ESA

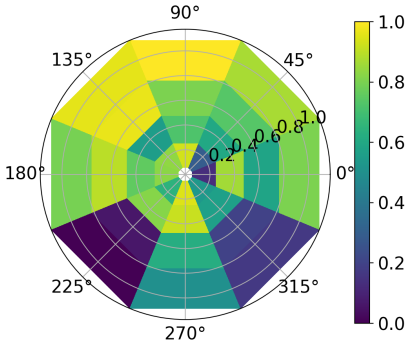
(c) SARL

(d) ESA

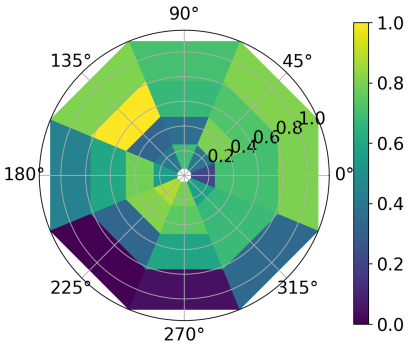
Figure 7.9: Illustration of the resulting trajectories. Fig. (a) and (b) show the navigation behavior over time in the scenario with five dynamic humans. ESA takes the shorter time to reach the goal, while GA3C-CADRL meets the timeout case. Fig. (c) and (d) show how the robot navigates in the environment with 15 dynamic humans. Although the robot's observed states are limited in crowded environments, the ESA is still able to find a safe navigation path, while SARL collides with the crowd on its way.



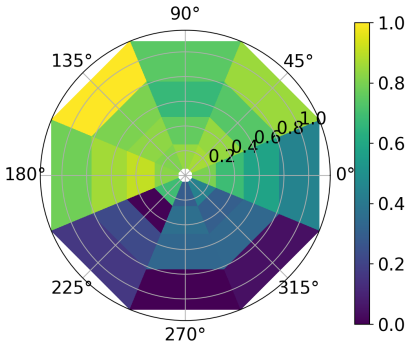
(a) Dense scene



(b) SARL



(c) GA3C-CADRL



(d) ESA

Figure 7.10: Value estimations by different methods for the dense scene (a). The dense scene (a) is formed by 5 dynamic humans, 4 static humans and one robot. The light dashed blue circle is the robot's sensor range with a radius of 2.5m, where the humans could be observed by the robot. SARL model prefers to take full speed at 90°, while GA3C-CADRL model prefers to take less speed at 135°. ESA model achieves the high value for high speeds on the rotation of 135°, preparing to pass behind dynamic humans #0.

In the following, I also compared the value estimations of several different methods in the same simulation environment Figure 7.10. It's important to note that the robot can only observe the state of humans within a circle centered on itself, defined by the radius of its sensor range, and not the state of all humans in the simulation environment. This requires the robot to utilize incomplete environment observation information to understand the environment more accurately and comprehensively for safe and effective navigation. Among these methods, the SARL model's attention mechanism significantly focuses on stationary humans at 225° and 0°. This focus might lead the robot to choose more risky behaviors, such as moving full speed at a 90-degree angle, increasing the risk of conflicts with static humans outside the sensor detection range. In contrast, both GA3C-CADRL and ESA consider spatial position information, thus tending to turn towards 135° to bypass dynamic human #0. However, GA3C-CADRL does not move at full speed, adopting a more conservative behavior to prevent potential collisions. Conversely, ESA can understand the environment more accurately based on the current partial observations, strengthening its focus on the stationary human at 0° and dynamic human #0. It chooses to avoid the 90° direction, which is the riskiest area (potential human gathering spot) and infers that the 135° direction poses less risk, thereby choosing to move full speed in that direction to avoid potential obstacles.

7.4 Summary

In this chapter, I introduce an efficient method called Enhanced Spatial and Attention Graph (ESA) for crowded and partially observable environments. While robots can only perceive human states within a short range of sensors, ESA integrates spatial and attention reasoning for effective navigation. Spatial graphs capture the spatial relationships between the robot and its environment, including dynamic and static objects, aiding in understanding and predicting environmental dynamics. The attention mechanism enhances the focus on key environmental information, enabling more accurate identification and response within limited perception. Parallel Double DQN is used for network training. Experiment results show that ESA outperforms state-of-the-art baselines in challenging simulated environments with various dynamic and static human numbers, maintaining low collision rates even in high-density scenarios.

CHAPTER 8

Conclusion

In this chapter, I summarize the contributions of the thesis for social-aware robot navigation. Furthermore, I discuss future directions for research to advance state-of-the-art approaches for social-aware robot navigation.

8.1 Summary

In this thesis, I introduced new contributions to the field of socially aware robot navigation. I propose several methods based on deep reinforcement learning to overcome the limitations of current learning-based navigation approaches, such as poor generalizability, short-sightedness, and reduced performance due to incomplete observational information in partially observable environments, particularly in crowded and complex human environments. Our developed methods prioritize optimizing human-robot social interactions. I aim to devise navigation algorithms that facilitate intent-aware and socially compliant robot motions, enhancing human comfort and safety while reducing instability in human social dynamics. The proposed methods augment exploration capabilities by extending sparse reward functions. They leverage graph attention networks to encode the temporal and spatial dimensions of crowd interactions and employ attention mechanisms for planning and enhancing robot navigation in partially observable environments. Consequently, robot navigation is improved, enabling faster target reach and minimizing collisions, all while maintaining a high level of robustness. In the following, I will discuss these contributions in detail.

Previous robot navigation approaches only rely on the current state to make decisions, resulting in shortsighted actions that may exhibit inflexibility or even become trapped when encountering sudden events or increasing environmental complexity. For instance, traditional methods might struggle to respond promptly when other individuals suddenly change direction, or they might freeze in place when faced with group obstacles. In Chapter 5, I address this limitation through the development of an extended reward function and the introduction of a novel foresight social-aware reinforcement learning (FSRL) framework. On one hand, the new reward function incorporates the ability to foresee potential future collisions, enhancing its responsiveness to sudden events. The FSRL operates by predicting potential future collisions and taking proactive measures to avoid them, thereby increasing navigational safety. On the other hand, by introducing efficiency constraints into the new reward function, I have significantly reduced navigation time. Moreover, the synergistic incorporation of foresight penalties and efficiency constraints enables the robot to reduce collisions while effectively balancing efficiency and user comfort (minimizing disturbances). Our model has demonstrated superior performance across three progressively challenging simulated environments. It not only delivers smoother and safer navigation trajectories but also accomplishes navigation in a relatively shorter time frame, substantially improving the efficacy and efficiency of the navigation methods.

To enhance the generalizability of robot navigation strategies, I introduce an Attention-based

Spatial-Temporal Graph (ASTG) learning model in Chapter 6. This model leverages graph attention networks (GAT) to capture and understand complex interactions and relationships between agents, utilizing a self-attention mechanism to propagate and aggregate information from neighbors. With the incorporation of GAT, I can implicitly calculate both direct and indirect spatial interactions and temporal group features, empowering the model to make informed decisions in dynamically changing scenarios. ASTG further integrates recurrent neural networks (RNN) to incorporate past trajectory information, enabling more precise predictions of each agent's future intentions. This approach goes beyond making decisions based solely on the current environmental state, offering a more foresighted decision-making process by deeply analyzing spatial-temporal graphs to anticipate future changes and potential challenges. This significantly boosts the robot's navigation abilities in a variety of complex environments, enhancing its adaptability and universality. To better measure and analyze behaviors, I introduce a new performance metric called "weighted navigation time," which effectively penalizes situations involving collision risks or inappropriate strategies, thereby offering a more comprehensive and practical assessment standard for robot navigation. In our experiments, ASTG demonstrated superior performance, not only significantly reducing collision rates compared to baseline methods but also maintaining efficient navigation speeds. This showcases remarkable robustness and reliability, ensuring that robots can effectively navigate in a range of environments, from simple to complex.

Finally, the limitations imposed by sensors present a significant challenge, as the restricted perception range can lead to incomplete and uncertain environment data. In response to this, I devised an advanced deep reinforcement learning architecture called the "Enhanced Spatial Attention Graph (ESA)" based on graph learning methodologies, detailed in Chapter 7, aiming to optimize robot navigation strategies. This system unites spatial and attention reasoning, facilitating improved navigation in crowded and partially observable environments. On one hand, it focuses on analyzing the spatial relationships and layouts of both static and dynamic elements present in the environment; on the other hand, it lays emphasis on determining action strategies for the robot by analyzing the significance of different entities that interact with the robot. To mitigate issues of overfitting and learning biases, enhancing both efficiency and performance of the learning process, I employed a parallel Double Deep Q-Network (DQN) strategy for training the network. Our experiments indicate that, compared to other advanced methods, the ESA method not only significantly reduces training time but also exhibits higher success rates and lower collision rates in a variety of test scenarios. This method can effectively avoid obstacles while maintaining a navigation time similar to that of SARL, especially in cases of increased human presence, its growth rate of collision rate and navigation time is slower than other methods.

8.2 Future Work

While a variety of methods addressing social-aware crowd navigation have achieved success, it is imperative to acknowledge and address their respective limitations for practical, real-world deployment. In this section, I intend to briefly discuss some open challenges and possible extensions of the method proposed in this paper.

Adaptive Reward Function. In the field of robot crowd navigation using deep reinforcement learning, a key future research direction is the development of adaptive reward functions. These func-

tions could adjust according to the dynamic complexities of the environment, thereby enhancing the algorithm's generalizability and adaptability in varied social settings. Traditionally, reinforcement learning depends on effective reward signals to guide agent learning. Typically, an agent receives a positive reward for achieving goals and a negative one for colliding with obstacles. However, these rewards are often generated only at the end of each training epoch, leading to sparsity in reward signals. Sparse rewards can exacerbate data inefficiency, resulting in insufficient training convergence and prolonged training times. To address this, reward shaping techniques were introduced in Chapter 5, where a comprehensive reward function was constructed by setting multiple objectives such as safety, efficiency, and comfort. This approach allows robots to consider other social factors while achieving their primary navigation goals. Additionally, developing reward functions that adapt to complex environmental changes is another effective method for mitigating the issue of sparse rewards. This could potentially be achieved by integrating meta-learning techniques, enabling robots to dynamically adjust their reward functions based on the current environment and context. Such a strategy is expected to reduce overfitting and enhance adaptability across diverse social environments. However, introducing such adaptive reward functions adds complexity to reinforcement learning, which could impact the algorithm's stability and convergence. Thus, realizing a self-adaptive reward function capable of navigating through varied and unknown situations without compromising the stability and efficiency of the learning process is a highly challenging task. In the context of robot crowd navigation, such a reward system must not only consider the effectiveness of navigation but also how to adapt to complex social dynamics while maintaining algorithmic stability and efficiency.

Enhancing Generalization Ability. In Chapter 6, I introduced an attention-based spatial-temporal graph learning model to address social-aware robot navigation tasks in crowded scenarios. This model can effectively capture and analyze complex crowd interactions, thereby aiding robots in comprehending the underlying behaviors and intentions of each agent. The attention mechanism plays a pivotal role, encouraging learning across diverse simulation environments by weighing the interactive influences amongst various agents. However, despite the method achieving good generalization performance in simulated environments, it encounters some challenges when transitioning to physical environments. In real environments, the system needs to handle more uncertainties and dynamic changes, making the direct transition from simulated to real environments very difficult. To overcome this, researchers have begun exploring various solutions. *Zhang et al.* (2017) initiated the agent from a random position and utilized four various maze environments in training phases. *Long et al.* (2018) devised a multi-scenario training framework to learn optimal strategies, employing numerous robots for concurrent training in rich, complex scenarios. Only a small amount of fine-tuning is needed to transfer the model from the simulated environment to the real environment. This demonstrates that adding randomness to the simulated environment to expand the sample space is an effective way to solve the generalization problem. Incorporating elements of randomness - be it through sensor noise, random target positions, or unpredictable obstacles - expands the sample space, enhancing data diversity across various scenarios and facilitating a smoother transition between simulated environments. I believe that a promising approach to enhancing the model's generalization performance further is to broaden the sample space through introducing randomness into the simulated environments. This can be achieved by incorporating various elements such as random sensor noise, random target positions, and random obstacles, building a richer and more diverse training dataset. In this way, the model can

be trained in an environment closer to the real-world dynamics, enhancing its generalization ability and effectiveness in practical applications.

Prediction on Partially Observable Environments I delved into navigation models in partially observable environments in Chapter 7. Leveraging LSTM to distinguish and encode the spatial dynamics between robots and other agents has fundamentally enhanced robots' comprehension of their environments, establishing a solid foundation for navigation adapted to social cues. An interesting future direction is to further explore the potential of recurrent neural networks (RNN), especially their powerful memory functions, to enhance agents' autonomous memory and ability to process previous observable information. For instance, adopting Bidirectional LSTM (Bi-LSTM) can simultaneously capture past and future information to better understand and predict the surrounding environment and pedestrian behavior. Additionally, by integrating features from different sources, such as visual information, LiDAR data, or other sensor data, a more comprehensive understanding of the environment can be provided, thereby assisting LSTM in making more accurate predictions. Such advancements promise not just heightened adaptability to diverse environments but also smooth and natural navigation in complex social scenarios. Moreover, although I have explored the application of RNNs in processing temporal information in navigation tasks based on deep reinforcement learning (DRL) in Chapter 6, there are still many unexplored possibilities in this field. I recognize that RNNs and its variants (such as LSTM and GRU) may have long-term dependency issues, making the choice and design of network architecture particularly important.

Learning the Interactions in Crowds Human behavior is diverse at the individual level, but it becomes more complex in terms of group structure and dynamics. An interesting research direction in future is delve deeper into how to design a social interaction model that not only considers pairwise interactions between agents but also accounts for group-level interactions. By deeply understanding group dynamics, robots can more effectively predict the movement trends and potential changes of crowds, which is crucial for improving obstacle avoidance and path planning accuracy. In certain scenarios, such as emergency situations, robots may also need to coordinate or cooperate with crowds to guide evacuation. Such a model can provide a comprehensive and multifaceted perspective, helping robots better understand and interpret social environments, thereby offering more precise and flexible navigation solutions. Previous studies have explored methods to capture inter-group differences (for instance, *Shao et al.* (2014)) in dynamic groups and intra-group coherence (for instance, *Taylor et al.* (2020); *Luber and Arras* (2013)), but group properties may differ between static and dynamic settings. Future research could explore how to construct models capable of handling multi-level social interactions, including interactions among individuals, small groups, and large crowds. Moreover, combining deep learning techniques with social science theories can provide robots with a deeper understanding of social environments. This combination can help robots navigate more effectively and naturally in complex crowd navigation tasks. Understanding aspects such as cultural differences, psychological factors, and group psychology will significantly impact the robots' social adaptability and decision-making abilities. Overall, this multi-level, multi-angle research approach will propel future studies on different structural levels of interaction within crowd environments, bringing new breakthroughs in the field of robot crowd navigation based on deep reinforcement learning.

Bibliography

Achiam, Joshua; Held, David; Tamar, Aviv, and Abbeel, Pieter. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017. (Cited on page 15.)

Alahi, Alexandre; Goel, Kratarth; Ramanathan, Vignesh; Robicquet, Alexandre; Fei-Fei, Li, and Savarese, Silvio. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. (Cited on page 13.)

Aoude, Georges S; Luders, Brandon D; Joseph, Joshua M; Roy, Nicholas, and How, Jonathan P. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35:51–76, 2013. (Cited on page 12.)

Babinec, Andrej; Duchon, Frantisek; Dekan, Martin; Mikulova, Zuzana, and Jurisica, Ladislav. Vector field histogram* with look-ahead tree extension dependent on time variable environment. *Transactions of the Institute of Measurement and Control*, 40(4):1250–1264, 2018. (Cited on pages 11 and 12.)

Batalin, Maxim A; Sukhatme, Gaurav S, and Hattig, Myron. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, volume 1, pages 636–641. IEEE, 2004. (Cited on page 10.)

Bellman, Richard. Dynamic programming. *Science*, 153(3731):34–37, 1966. (Cited on pages 18 and 19.)

Bengio, Yoshua; Simard, Patrice, and Frasconi, Paolo. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994. (Cited on page 29.)

Bennewitz, Maren; Burgard, Wolfram, and Thrun, Sebastian. Learning motion patterns of persons for mobile service robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3601–3606. IEEE, 2002. (Cited on page 13.)

Borenstein, Johann and Koren, Yoram. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989. (Cited on page 52.)

Borenstein, Johann and Koren, Yoram. Real-time obstacle avoidance for fast mobile robots in cluttered environments. In *Proceedings., IEEE International Conference on Robotics and Automation*, pages 572–577. IEEE, 1990. (Cited on page 52.)

Borenstein, Johann; Koren, Yoram, and others, . The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE transactions on robotics and automation*, 7(3):278–288, 1991. (Cited on pages 11, 12 and 52.)

BostonDynamics, . URL `https://bostondynamics.com/`. (Cited on page 9.)

Breazeal, Cynthia. *Designing sociable robots*. MIT press, 2004. (Cited on page 11.)

Brockman, Greg; Cheung, Vicki; Pettersson, Ludwig; Schneider, Jonas; Schulman, John; Tang, Jie, and Zaremba, Wojciech. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. (Cited on pages 46, 60 and 101.)

Chen, Changan; Liu, Yuejiang; Kreiss, Sven, and Alahi, Alexandre. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6015–6022. IEEE, 2019. (Cited on pages vii, 14, 41, 46, 51, 52, 53, 54, 55, 56, 60, 61, 62, 65, 76, 77, 79, 80, 81, 84, 85, 98, 100, 101, 102, 103, 104 and 106.)

Chen, Changan; Hu, Sha; Nikdel, Payam; Mori, Greg, and Savva, Manolis. Relational graph learning for crowd navigation. In *IROS*, pages 10007–10013, 2020a. (Cited on pages 15, 80, 81 and 84.)

Chen, Yiyang; Cheng, Chuanxin; Zhang, Yueyuan; Li, Xinlin, and Sun, Lining. A neural network-based navigation approach for autonomous mobile robot systems. *Applied Sciences*, 12(15):7796, 2022. (Cited on page 10.)

Chen, Yu Fan; Everett, Michael; Liu, Miao, and How, Jonathan P. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1343–1350. IEEE, 2017a. (Cited on pages 13 and 99.)

Chen, Yu Fan; Liu, Miao; Everett, Michael, and How, Jonathan P. Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 285–292. IEEE, 2017b. (Cited on pages 13, 41, 43, 44, 56 and 99.)

Chen, Yujing; Zhao, Fenghua, and Lou, Yunjiang. Interactive model predictive control for robot navigation in dense crowds. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52 (4):2289–2301, 2021. (Cited on page 11.)

Chen, Yuying; Liu, Congcong; Shi, Bertram E, and Liu, Ming. Robot navigation in crowds by graph convolutional networks with attention learned from human gaze. *IEEE Robotics and Automation Letters*, 5(2):2754–2761, 2020b. (Cited on pages 14, 15 and 85.)

Cho, Kyunghyun; van Merrienboer, Bart; Gulcehre, Caglar; Bahdanau, Dzmitry; Bougares, Fethi; Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1724. Association for Computational Linguistics, 2014. (Cited on page 29.)

Choset, Howie; Lynch, Kevin M; Hutchinson, Seth; Kantor, George A, and Burgard, Wolfram. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005. (Cited on page 12.)

Doucet, Arnaud; De Freitas, Nando; Gordon, Neil James, and others, . *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001. (Cited on page 12.)

Engel, Jakob; Schöps, Thomas, and Cremers, Daniel. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014. (Cited on page 10.)

Everett, Michael; Chen, Yu Fan, and How, Jonathan P. Motion planning among dynamic, decision-making agents with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3052–3059. IEEE, 2018. (Cited on pages 14, 41, 52, 60, 61, 62, 102, 103, 104 and 106.)

Ferrer, Gonzalo; Zulueta, Anaís Garrell; Cotarelo, Fernando Herrero, and Sanfeliu, Alberto. Robot social-aware navigation framework to accompany people walking side-by-side. *Autonomous robots*, 41(4):775–793, 2017. (Cited on pages 11 and 12.)

Fiorini, Paolo and Shiller, Zvi. Motion planning in dynamic environments using the relative velocity paradigm. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 560–565. IEEE, 1993. (Cited on page 11.)

Fong, Terrence; Nourbakhsh, Illah, and Dautenhahn, Kerstin. A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3-4):143–166, 2003. (Cited on page 52.)

Francis, Anthony; Faust, Aleksandra; Chiang, Hao-Tien Lewis; Hsu, Jasmine; Kew, J Chase; Fiser, Marek, and Lee, Tsang-Wei Edward. Long-range indoor navigation with prm-rl. *IEEE Transactions on Robotics*, 36(4):1115–1134, 2020. (Cited on page 14.)

Fujita, Yasuhiro; Nagarajan, Prabhat; Kataoka, Toshiki, and Ishikawa, Takahiro. Chainerrl: A deep reinforcement learning library. *The Journal of Machine Learning Research*, 22(1):3557–3570, 2021. (Cited on pages 97 and 102.)

Gao, Xingyuan; Sun, Shiying; Zhao, Xiaoguang, and Tan, Min. Learning to navigate in human environments via deep reinforcement learning. In *International Conference on Neural Information Processing*, pages 418–429. Springer, 2019. (Cited on page 14.)

Gao, Xueshan; Gao, Rui; Liang, Peng; Zhang, Qingfang; Deng, Rui, and Zhu, Wei. A hybrid tracking control strategy for nonholonomic wheeled mobile robot incorporating deep reinforcement learning approach. *IEEE Access*, 9:15592–15602, 2021. (Cited on page 53.)

Gao, Yuxiang and Huang, Chien-Ming. Evaluation of socially-aware robot navigation. *Frontiers in Robotics and AI*, 8:721317, 2022. (Cited on page 11.)

Ge, Shuzhi Sam. Social robotics: Integrating advances in engineering and computer science. In *The 4th annual international conference organized by Electrical Engineering/Electronics, Computer, Telecommunication and Information Technology*, page 2007, 2007. (Cited on page 10.)

Gil, Óscar; Garrell, Anaís, and Sanfeliu, Alberto. Social robot navigation tasks: Combining machine learning techniques and social force model. *Sensors*, 21(21):7087, 2021. (Cited on page 14.)

Goodfellow, Ian; Bengio, Yoshua, and Courville, Aaron. *Deep learning*. MIT press, 2016. (Cited on page 22.)

Graves, Alex and Graves, Alex. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pages 37–45, 2012. (Cited on page 74.)

Guillén-Ruiz, Silvia; Bandera, Juan Pedro; Hidalgo-Paniagua, Alejandro, and Bandera, Antonio. Evolution of socially-aware robot navigation. *Electronics*, 12(7):1570, 2023. (Cited on page 11.)

Hastings, W Keith. Monte carlo sampling methods using markov chains and their applications. 1970. (Cited on page 20.)

He, Kaiming; Zhang, Xiangyu; Ren, Shaoqing, and Sun, Jian. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. (Cited on pages 27 and 78.)

Helbing, Dirk and Molnar, Peter. Social force model for pedestrian dynamics. *Physical review E*, 51 (5):4282, 1995. (Cited on pages 11 and 12.)

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997. (Cited on pages 13, 29 and 97.)

Howard, Ronald A. Dynamic programming and markov processes. 1960. (Cited on page 18.)

Hu, Zhengxi; Zhao, Yingli; Zhang, Sen; Zhou, Lei, and Liu, Jingtai. Crowd-comfort robot navigation among dynamic environment based on social-stressed deep reinforcement learning. *International Journal of Social Robotics*, 14(4):913–929, 2022. (Cited on page 14.)

Hussein, Ahmed; Gaber, Mohamed Medhat; Elyan, Eyad, and Jayne, Chrisina. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017. (Cited on page 15.)

Joseph, Joshua; Doshi-Velez, Finale; Huang, Albert S, and Roy, Nicholas. A bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31:383–400, 2011. (Cited on page 12.)

Kalman, Rudolph Emil. A new approach to linear filtering and prediction problems. 1960. (Cited on page 12.)

Khatib, Oussama. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE international conference on robotics and automation*, volume 2, pages 500–505. IEEE, 1985. (Cited on page 11.)

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on pages 26, 62 and 102.)

Kipf, Thomas N and Welling, Max. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. (Cited on pages 31 and 32.)

Kretzschmar, Henrik; Spies, Markus; Sprunk, Christoph, and Burgard, Wolfram. Socially compliant mobile robot navigation via inverse reinforcement learning. *The International Journal of Robotics Research*, 35(11):1289–1307, 2016. (Cited on page 52.)

Kroese, Dirk P; Taimre, Thomas, and Botev, Zdravko I. *Handbook of monte carlo methods*. John Wiley & Sons, 2013. (Cited on page 98.)

Kruse, Thibault; Pandey, Amit Kumar; Alami, Rachid, and Kirsch, Alexandra. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013. (Cited on page 52.)

Kuderer, Markus; Kretzschmar, Henrik; Sprunk, Christoph, and Burgard, Wolfram. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: science and systems*, 2012. (Cited on page 10.)

Laboratories, Hiroshi Ishiguro. URL `http://www.geminoid.jp/en/robots.html`. (Cited on page 9.)

LaValle, Steven M; Kuffner, James J; Donald, BR, and others, . Rapidly-exploring random trees: Progress and prospects. *Algorithmic and computational robotics: new directions*, 5:293–308, 2001. (Cited on page 12.)

LeCun, Yann; Bottou, Léon; Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. (Cited on pages vii, 26 and 27.)

LeCun, Yann; Kavukcuoglu, Koray, and Farabet, Clément. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010. (Cited on page 99.)

LeCun, Yann; Bengio, Yoshua, and Hinton, Geoffrey. Deep learning. *nature*, 521(7553):436–444, 2015. (Cited on page 22.)

Levinson, Jesse and Thrun, Sebastian. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE international conference on robotics and automation*, pages 4372–4378. IEEE, 2010. (Cited on page 10.)

Lillicrap, Timothy P; Hunt, Jonathan J; Pritzel, Alexander; Heess, Nicolas; Erez, Tom; Tassa, Yuval; Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. (Cited on page 37.)

Lindner, Felix and Eschenbach, Carola. Towards a formalization of social spaces for socially aware robots. In *Spatial Information Theory: 10th International Conference, COSIT 2011, Belfast, ME, USA, September 12-16, 2011. Proceedings 10*, pages 283–303. Springer, 2011. (Cited on page 10.)

Liu, Jun S and Chen, Rong. Sequential monte carlo methods for dynamic systems. *Journal of the American statistical association*, 93(443):1032–1044, 1998. (Cited on page 12.)

Liu, Lucia; Dugas, Daniel; Cesari, Gianluca; Siegwart, Roland, and Dubé, Renaud. Robot navigation in crowded environments using deep reinforcement learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5671–5677. IEEE, 2020. (Cited on page 14.)

Liu, Shuijing; Chang, Peixin; Liang, Weihang; Chakraborty, Neeloy, and Driggs-Campbell, Katherine. Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3517–3524. IEEE, 2021. (Cited on page 52.)

Liu, Yuejiang; Xu, An, and Chen, Zichong. Map-based deep imitation learning for obstacle avoidance. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8644–8649. IEEE, 2018. (Cited on page 15.)

Long, Pinxin; Liu, Wenxi, and Pan, Jia. Deep-learned collision avoidance policy for distributed multiagent navigation. *IEEE Robotics and Automation Letters*, 2(2):656–663, 2017. (Cited on page 15.)

Long, Pinxin; Fan, Tingxiang; Liao, Xinyi; Liu, Wenxi; Zhang, Hao, and Pan, Jia. Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 6252–6259. IEEE, 2018. (Cited on page 113.)

Luber, Matthias and Arras, Kai Oliver. Multi-hypothesis social grouping and tracking for mobile robots. In *Robotics: Science and Systems*, 2013. (Cited on page 114.)

Maxwell, Bruce A. Building robot systems to interact with people in real environments. *Autonomous Robots*, 22:353–367, 2007. (Cited on page 11.)

McCulloch, Warren S and Pitts, Walter. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. (Cited on page 23.)

Medsker, Larry R and Jain, LC. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001. (Cited on page 75.)

Michaud, François; Côté, Carle; Létourneau, Dominic; Brosseau, Yannick; Valin, J-M; Beaudry, Éric; Raïevsky, Clément; Ponchon, Arnaud; Moisan, Pierre; Lepage, Pierre, and others, . Spartacus attending the 2005 aaai conference. *Autonomous Robots*, 22:369–383, 2007. (Cited on page 11.)

Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Graves, Alex; Antonoglou, Ioannis; Wierstra, Daan, and Riedmiller, Martin. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. (Cited on page 34.)

Mnih, Volodymyr; Kavukcuoglu, Koray; Silver, David; Rusu, Andrei A; Veness, Joel; Bellemare, Marc G; Graves, Alex; Riedmiller, Martin; Fidjeland, Andreas K; Ostrovski, Georg, and others, . Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015. (Cited on page 34.)

Mnih, Volodymyr; Badia, Adria Puigdomenech; Mirza, Mehdi; Graves, Alex; Lillicrap, Timothy; Harley, Tim; Silver, David, and Kavukcuoglu, Koray. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016. (Cited on page 37.)

Moon, Todd K. The expectation-maximization algorithm. *IEEE Signal processing magazine*, 13(6): 47–60, 1996. (Cited on page 13.)

Mur-Artal, Raul; Montiel, Jose Maria Martinez, and Tardos, Juan D. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015. (Cited on page 10.)

Osa, Takayuki; Pajarinen, Joni; Neumann, Gerhard; Bagnell, J Andrew; Abbeel, Pieter; Peters, Jan, and others, . An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018. (Cited on page 15.)

Pacchierotti, Elena; Jensfelt, Patric, and Christensen, Henrik I. Tasking everyday interaction. *Autonomous Navigation in Dynamic Environments*, pages 151–168, 2007. (Cited on page 10.)

Paszke, Adam; Gross, Sam; Chintala, Soumith; Chanan, Gregory; Yang, Edward; DeVito, Zachary; Lin, Zeming; Desmaison, Alban; Antiga, Luca, and Lerer, Adam. Automatic differentiation in pytorch. 2017. (Cited on page 102.)

Pfeiffer, Mark; Schaeuble, Michael; Nieto, Juan; Siegwart, Roland, and Cadena, Cesar. From perception to decision: A data-driven approach to end-to-end motion planning for autonomous ground robots. In *2017 ieee international conference on robotics and automation (icra)*, pages 1527–1533. IEEE, 2017. (Cited on pages 10 and 15.)

Pfeiffer, Mark; Shukla, Samarth; Turchetta, Matteo; Cadena, Cesar; Krause, Andreas; Siegwart, Roland, and Nieto, Juan. Reinforced imitation: Sample efficient deep reinforcement learning for mapless navigation by leveraging prior demonstrations. *IEEE Robotics and Automation Letters*, 3 (4):4423–4430, 2018. (Cited on page 15.)

Pitt, Michael K and Shephard, Neil. Filtering via simulation: Auxiliary particle filters. *Journal of the American statistical association*, 94(446):590–599, 1999. (Cited on page 12.)

Reddy, Arun Kumar; Malviya, Vaibhav, and Kala, Rahul. Social cues in the autonomous navigation of indoor mobile robots. *International Journal of Social Robotics*, 13:1335–1358, 2021. (Cited on page 11.)

Rios-Martinez, Jorge; Spalanzani, Anne, and Laugier, Christian. Understanding human interaction for probabilistic autonomous navigation using risk-rrt approach. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2014–2019. IEEE, 2011. (Cited on page 13.)

Rios-Martinez, Jorge; Spalanzani, Anne, and Laugier, Christian. From proxemics theory to socially-aware navigation: A survey. *International Journal of Social Robotics*, 7:137–153, 2015. (Cited on page 10.)

Rosenblatt, Frank. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958. (Cited on page 10.)

Rummery, Gavin A and Niranjan, Mahesan. *On-line Q-learning using connectionist systems*, volume 37. University of Cambridge, Department of Engineering Cambridge, UK, 1994. (Cited on page 20.)

Samsani, Sunil Srivatsav and Muhammad, Mannan Saeed. Socially compliant robot navigation in crowded environment by human behavior resemblance using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 6(3):5223–5230, 2021. (Cited on page 14.)

Schmidhuber, Jürgen. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. (Cited on page 22.)

Shao, Jing; Change Loy, Chen, and Wang, Xiaogang. Scene-independent group profiling in crowd. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2219–2226, 2014. (Cited on page 114.)

Shi, Chao; Shimada, Michihiro; Kanda, Takayuki; Ishiguro, Hiroshi, and Hagita, Norihiro. Spatial formation model for initiating conversation. *Proceedings of robotics: Science and systems VII*, pages 305–313, 2011. (Cited on page 10.)

Shi, Weixian; Zhou, Yanying; Zeng, Xiangyu; Li, Shijie, and Bennewitz, Maren. Enhanced spatial attention graph for motion planning in crowded, partially observable environments. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4750–4756. IEEE, 2022. (Cited on page 8.)

Shiller, Zvi; Large, Frederic, and Sekhavat, Sepanta. Motion planning in dynamic environments: Obstacles moving along arbitrary trajectories. In *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No. 01CH37164)*, volume 4, pages 3716–3721. IEEE, 2001. (Cited on page 11.)

Sisbot, Emrah Akin; Marin-Urias, Luis F; Broquere, Xavier; Sidobre, Daniel, and Alami, Rachid. Synthesizing robot motions adapted to human presence: A planning and control framework for safe and socially acceptable robot motions. *International Journal of Social Robotics*, 2(3):329–343, 2010. (Cited on page 10.)

Smith, Trevor; Chen, Yuhao; Hewitt, Nathan; Hu, Boyi, and Gu, Yu. Socially aware robot obstacle avoidance considering human intention and preferences. *International journal of social robotics*, pages 1–18, 2021. (Cited on page 13.)

Snape, Jamie; Van Den Berg, Jur; Guy, Stephen J, and Manocha, Dinesh. The hybrid reciprocal velocity obstacle. *IEEE Transactions on Robotics*, 27(4):696–706, 2011. (Cited on pages 11 and 52.)

Sutton, Richard S. Learning to predict by the methods of temporal differences. *Machine learning*, 3: 9–44, 1988. (Cited on pages 19 and 20.)

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*. MIT press, 2018. (Cited on pages vii, 17, 18, 20 and 37.)

Sutton, Richard S; McAllester, David; Singh, Satinder, and Mansour, Yishay. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999. (Cited on page 37.)

Svenstrup, Mikael; Bak, Thomas, and Andersen, Hans Jørgen. Trajectory planning for robots in dynamic human environments. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4293–4298. IEEE, 2010. (Cited on page 13.)

Tai, Lei; Zhang, Jingwei; Liu, Ming, and Burgard, Wolfram. Socially compliant navigation through raw depth inputs with generative adversarial imitation learning. In *2018 IEEE ICRA*, pages 1111–1117, 2018. (Cited on page 15.)

Taylor, Angelique; Chan, Darren M, and Riek, Laurel D. Robot-centric perception of human groups. *ACM Transactions on Human-Robot Interaction (THRI)*, 9(3):1–21, 2020. (Cited on page 114.)

Teh, Yee Whye and others, . Dirichlet process. *Encyclopedia of machine learning*, 1063:280–287, 2010. (Cited on page 12.)

Thomaz, Andrea; Hoffman, Guy; Cakmak, Maya, and others, . Computational human-robot interaction. *Foundations and Trends® in Robotics*, 4(2-3):105–223, 2016. (Cited on page 11.)

Thrun, Sebastian. An approach to learning mobile robot navigation. *Robotics and Autonomous systems*, 15(4):301–319, 1995. (Cited on page 10.)

Thrun, Sebastian and Bücken, Arno. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the national conference on artificial intelligence*, pages 944–951, 1996. (Cited on page 10.)

Trautman, Pete; Ma, Jeremy; Murray, Richard M, and Krause, Andreas. Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation. *The International Journal of Robotics Research*, 34(3):335–356, 2015. (Cited on page 12.)

Trautman, Peter and Krause, Andreas. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803. IEEE, 2010. (Cited on page 13.)

Truong, Xuan-Tung and Ngo, Trung Dung. Toward socially aware robot navigation in dynamic and crowded environments: A proactive social motion model. *IEEE Transactions on Automation Science and Engineering*, 14(4):1743–1760, 2017. (Cited on pages 11 and 12.)

Van den Berg, Jur; Lin, Ming, and Manocha, Dinesh. Reciprocal velocity obstacles for real-time multi-agent navigation. In *2008 IEEE international conference on robotics and automation*, pages 1928–1935. Ieee, 2008. (Cited on pages 11, 46, 52, 60 and 101.)

Van Den Berg, Jur; Guy, Stephen J; Lin, Ming, and Manocha, Dinesh. Reciprocal n-body collision avoidance. In *Robotics Research: The 14th International Symposium ISRR*, pages 3–19. Springer, 2011. (Cited on pages 11, 46, 52, 60, 61, 62 and 102.)

Van Hasselt, Hado; Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. (Cited on pages 35, 97 and 98.)

Velickovic, Petar; Cucurull, Guillem; Casanova, Arantxa; Romero, Adriana; Lio, Pietro; Bengio, Yoshua, and others, . Graph attention networks. *stat*, 1050(20):10–48550, 2017. (Cited on pages 31 and 33.)

Vemula, Anirudh; Muelling, Katharina, and Oh, Jean. Social attention: Modeling attention in human crowds. In *2018 IEEE international Conference on Robotics and Automation (ICRA)*, pages 4601–4607. IEEE, 2018. (Cited on page 13.)

Wang, Jack; Hertzmann, Aaron, and Fleet, David J. Gaussian process dynamical models. *Advances in neural information processing systems*, 18, 2005. (Cited on page 12.)

Wang, Junxian; Chan, Wesley P; Carreno-Medrano, Pamela; Cosgun, Akansel, and Croft, Elizabeth. Metrics for evaluating social conformity of crowd navigation algorithms. In *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, pages 1–6. IEEE, 2022. (Cited on page 47.)

Watkins, Christopher JCH and Dayan, Peter. Q-learning. *Machine learning*, 8:279–292, 1992. (Cited on page 20.)

Welch, Greg; Bishop, Gary, and others, . An introduction to the kalman filter. 1995. (Cited on page 12.)

Wiering, Marco A and Van Otterlo, Martijn. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012. (Cited on page 17.)

Wurm, Kai M; Hornung, Armin; Bennewitz, Maren; Stachniss, Cyrill, and Burgard, Wolfram. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, page 3, 2010. (Cited on page 10.)

Xiao, Xuesu; Liu, Bo; Warnell, Garrett, and Stone, Peter. Motion planning and control for mobile robot navigation using machine learning: a survey. *Autonomous Robots*, 46(5):569–597, 2022. (Cited on page 10.)

Yao, Qingfeng; Zheng, Zeyu; Qi, Liang; Yuan, Haitao; Guo, Xiwang; Zhao, Ming; Liu, Zhi, and Yang, Tianji. Path planning method with improved artificial potential fieldâa reinforcement learning perspective. *IEEE access*, 8:135513–135523, 2020. (Cited on page 11.)

Zhang, Ji and Singh, Sanjiv. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and systems*, volume 2, pages 1–9. Berkeley, CA, 2014. (Cited on page 10.)

Zhang, Jingwei; Springenberg, Jost Tobias; Boedecker, Joschka, and Burgard, Wolfram. Deep reinforcement learning with successor features for navigation across similar environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2371–2378. IEEE, 2017. (Cited on page 113.)

Zhou, Yanying; Li, Shijie, and Garcke, Jochen. Foresight social-aware reinforcement learning for robot navigation. In *2023 35th Chinese Control and Decision Conference (CCDC)*, pages 3501–3507. IEEE, 2023. (Cited on page 7.)

Zhou, Zhiqian; Zhu, Pengming; Zeng, Zhiwen; Xiao, Junhao; Lu, Huimin, and Zhou, Zongtan. Robot navigation in a crowd by integrating deep reinforcement learning and online planning. *Applied Intelligence*, pages 1–17, 2022. (Cited on pages 15, 80, 81, 84 and 85.)

Ziebart, Brian D; Ratliff, Nathan; Gallagher, Garratt; Mertz, Christoph; Peterson, Kevin; Bagnell, J Andrew; Hebert, Martial; Dey, Anind K, and Srinivasa, Siddhartha. Planning-based prediction for pedestrians. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3931–3936. IEEE, 2009. (Cited on page 13.)