

**MULTI-ROBOT SOCIALLY AWARE NAVIGATION WITH
DEEP REINFORCEMENT LEARNING**

BY

IBRAHIM KHALIL KABIR

A Thesis Presented to the
COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SYSTEMS AND CONTROL ENGINEERING

MAY 2025

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
COLLEGE OF GRADUATE AND INTERDISCIPLINARY STUDIES

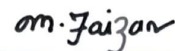
This thesis, written by **IBRAHIM KHALIL KABIR** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfilment of the requirements for the degree of **MASTER OF SCIENCE IN SYSTEMS AND CONTROL ENGINEERING**.



Dr. Mujahed Mohammad Al-Dhaifallah
Department Chairman



Prof. Suliman Saleh Al-Homidan
Dean of Graduate Studies



Dr. Muhammad Faizan Mysorewala
(Advisor)



Dr. Ali Nasir
(Member)



Dr. Yahya Ismail Osais
(Member)

Date



© Ibrahim Khalil Kabir

2025

This work is dedicated to my late dad, Dr. Adnan Kabir.

ACKNOWLEDGEMENTS

First of all I acknowledge Allah's bounties upon me. All praise due to Him, Lord of all creation. Alhamdulillah! I express my heartfelt gratitude to my mum, brothers, sister, and extended family at large. I am also indebted in gratitude and appreciation to King Fahd University of Petroleum & Minerals for making my academic environment conducive for excellence.

I thank my advisor, Dr. Muhammad Faizan Mysorewala, as well as the other members of my thesis committee: Dr Yahya Ismail Osais and Dr. Ali Nasir, whose constructive feedback and expertise significantly contributed to the success of this research. Lastly, I extend the utmost appreciation to the Chairman and faculty members of the Department of Control and Instrumentation Engineering.

I am truly and most sincerely grateful also to my friends, colleagues, and fellow graduate students, of notable mention among them are: Mr. Ghali Ahmad, Mr. Abdulbasit Hassan, Mr. Aminu Doguwa, Mr. Shehu Mohammed, and many others. My academic and personal journey has no doubt been positively been impacted by these individuals.

Ibrahim Khalil Kabir

May 2025

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	V
TABLE OF CONTENTS	VI
LIST OF TABLES	XI
LIST OF FIGURES	XII
LIST OF ABBREVIATIONS	XIV
ABSTRACT.....	XVI
المخلص الرسالة	XVIII
CHAPTER 1 : INTRODUCTION.....	1
1.1 Background	1
1.2 Problem Statement.....	5
1.3 Objectives.....	7
1.4 Contributions.....	7
1.5 Thesis Organization	8
CHAPTER 2 : LITERATURE REVIEW	10
2.1 Introduction.....	10
2.2 Traditional Navigation Approaches.....	10
2.3 Socially Aware Navigation Approaches.....	11
2.3.1 Reaction-based methods.....	12

2.3.2 Proactive methods.....	14
2.3.3 Learning-based methods	16
2.4 Multirobot Socially aware DRL methods.....	20
2.5 Summary of Related Work	22
2.6 Conclusion	24
CHAPTER 3 PRELIMINARIES	25
3.1 Introduction.....	25
3.2 Reinforcement Learning	25
3.2.1 Markov Decision Processes	26
3.2.2 Value Function	28
3.2.3 Q-Learning	29
3.2.4 Temporal Difference (TD) Learning.....	30
3.2.5 Deep Learning.....	31
3.2.6 Neural Networks	32
3.2.7 Feedforward Neural Network.....	36
3.2.8 Graph Neural Network.....	37
3.2.9 Attention Mechanism.....	38
3.3 Deep Reinforcement Learning.....	40
3.3.1 Value-based RL	40
3.3.2 Policy-based RL	42

3.3.3 Actor Critic RL	43
3.4 Imitation Learning.....	44
3.5 Summary.....	45
CHAPTER 4 : METHODOLOGY.....	46
4.1 Introduction.....	46
4.2 Assumptions.....	46
4.3 POMDP Formulation	47
4.3.1 State Space.....	48
4.3.2 Action Space	51
4.3.3 Reward Function.....	52
4.3.4 State Transition Model.....	54
4.3.5 Optimal Policy and Value Function	55
4.4 Interaction Modelling.....	56
4.4.1 Human-Human Interaction	57
4.4.2 Human-Ego Robot Interaction	57
4.4.3 Ego Robot-Other Robot Interaction	57
4.4.4 Human-Other Robot Interaction.....	57
4.4.5 Other Robot-Other Robot Interaction.....	58
4.5 Policy Formulation and Learning Algorithm.....	58
4.6 Imitation Learning-Based Pretraining	60

4.6.1 Initialization via Imitation Learning.....	61
4.6.2 The Optimal Reciprocal Collision Avoidance (ORCA) Policy	62
4.7 System Architecture.....	66
4.7.1 Interaction Module	67
4.7.2 Pooling Module.....	69
4.7.3 Planning Module	70
4.7.4 Implementation Details	71
4.8 Simulation Environment	72
4.9 Evaluation Metrics.....	74
4.10 Summary.....	75
CHAPTER 5 : RESULTS	77
5.1 Introduction.....	77
5.2 Simulation Configuration and Scenario Design.....	77
5.3 Multiple Scenarios Simulation.....	78
5.3.1 Open Space Scenario	78
5.3.2 Corridor Scenario	85
5.3.3 Static obstacles Scenario.....	87
5.4 Quantitative Results/ Experimental Setup	89
5.5 Impact of Imitation Learning on policy performance.....	93
5.6 Summary.....	95

CHAPTER 6 : CONCLUSION AND RECOMMENDATION	96
6.1 Summary of the Work	96
6.2 Recommendations for Future Work	97
REFERENCES.....	99
VITAE.....	106

LIST OF TABLES

Table 1 Comparison of Multi-robot socially aware schemes	23
Table 2. Agent Interaction Behaviour Matrix.....	58
Table 3 Training and Configuration Parameters.....	72
Table 4. Comparison of Similar work based on selected metrics.....	90

LIST OF FIGURES

Figure 1 A robot navigating while respecting social norms[7].....	3
Figure 2 Reinforcement Learning Loop	26
Figure 3 Illustration of biological neuron and artificial neuron[61]	32
Figure 4. A feedforward Neural Network [63]	37
Figure 5. Structure of a simple Graph Neural Network [66]	38
Figure 6. Scaled Dot-Product Attention (a) and Multi-Head Attention (b) [67]	39
Figure 7. Categories of agents in this work and their interaction relationship	56
Figure 8. Algorithm of learning framework	60
Figure 9. Robots A and B shown as disks (left) with their velocity obstacle (right) [22]	64
Figure 10 ORCA half-plane constraint in velocity space [22].....	65
Figure 11. Permissible velocity region formed by ORCA half-planes of 8 agents [22]...	66
Figure 12. Overall learning architecture of the ego robot in the navigation framework ..	67
Figure 13. Local Map creation.....	69
Figure 14. Circle and square crossings in Gym environment (a), and their setup (b) [79]	74
Figure 15. Scenario with 5 humans and 2 other robots at time: 0s, 2s, 4s, 5s, 6s, and 7s	80
Figure 16. 10 humans and 3 other robots scenario at: 0s, 2.5s, 3.75s, 4.5s, 5.75s, 8.5s ...	82
Figure 17. 10 Humans and 3 Other Robots trajectory plot	83
Figure 18. 7 humans and 5 other robots scenario at 0.25s, 1.75s, 3.5s, 4.75s, 6s, and 9s	84
Figure 19. 7 Humans and 5 Other Robots trajectory plot	85
Figure 20. Simple corridor scenario with 7 Humans and 2 other robots	86
Figure 21. 5 humans and 2 other robots scenario at: 0.5s, 3.75s, 4.75s, 6.25s, 7s, 8.5s ...	88
Figure 22. Performance metrics for the Navigation framework over Training episodes .	91

Figure 23. Comparison of cumulative reward and validation reward	92
Figure 24. Success rate and Cumulative discount reward	93
Figure 25. Ego Robot's time to reach goal without IL	94
Figure 26. Reward for DRL-only and IL with DRL models, with ± 1 standard deviation.	94

LIST OF ABBREVIATIONS

ANN	:	Artificial Neural Network
APF	:	Artificial Potential Field
CNN	:	Convolutional Neural Network
DL	:	Deep Learning
DNN	:	Deep Neural Network
DP	:	Dynamic Programming
DQN	:	Deep Q-Network
DRL	:	Deep Reinforcement Learning
FFNN	:	FeedForward Neural Network
GAT	:	Graph Attention Network
GCN	:	Graph Convolutional Network
GD	:	Gradient Descent
GNN	:	Graph Neural Network
GRU	:	Gated Recurrent Unit
LSTM	:	Long Short-Term Memory
MDP	:	Markov Decision Process

ML	:	Machine Learning
MLP	:	Multi Layer Perceptron
MSE	:	Mean Square Error
ORCA	:	Optimal Reciprocal Collision Avoidance
ReLU	:	Rectified Linear Unit
RL	:	Reinforcement Learning
RNN	:	Recurrent Neural Network
RVO	:	Reciprocal Velocity Obstacle
SAN	:	Socially Aware Navigation
SFM	:	Social Force Model
SGD	:	Stochastic Gradient Descent
TD	:	Temporal Difference
VFH	:	Vector Field Histogram
VO	:	Velocity Obstacle

ABSTRACT

Full Name : Ibrahim Khalil Kabir

Thesis Title : Multi-Robot Socially Aware Navigation with Deep Reinforcement Learning

Major Field : Systems and Control Engineering

Date of Degree : May 2025

The fast growth of technology is introducing robots, which were previously primarily utilized in labs and industries, into people's daily lives to interact in shared places. This makes it necessary to develop navigation frameworks that lead robots to move in non-threatening, human-friendly ways by considering social aspects. Such socially-aware navigation frameworks, must also scale to situations with several humans and robots traveling simultaneously. Recent advances in Deep Reinforcement Learning (DRL) have enabled development of robot navigation strategies that successfully accomplish their goals while including these social norms. This work develops a multi-robot socially aware navigation framework for mobile robots navigating in environments shared with humans and other robots. This approach, based on single-agent Deep Reinforcement Learning (DRL), involves modelling of all the interaction types in the environment between the ego robot, humans, and other robots. The DRL approach is used to train the framework with a reward function that balances task completion, collision avoidance, and the maintenance of comfortable distances from humans, and an attention mechanism is used to identify the relative importance of each agent to the navigation process. The developed framework is tested across a set of simulation scenarios including open space, corridor, and static

obstacle scenarios. The results show that in challenging, dynamic multi-agent environments the robot can create efficient, socially acceptable motion. Comparative assessments of current state-of-the-art methods highlight the advantages of the developed approach, especially in enhancing safety by reducing collisions and preserving suitable comfort distances between humans and other agents. These results confirm the success of the learned policy and support the applicability of the framework in practical multi-robot systems where social compliance is a fundamental need.

الملخص الرسالة

الاسم الكامل: إبراهيم خليل كبير

عنوان الرسالة: التنقل الاجتماعي الواعي متعدد الروبوتات باستخدام التعلم العميق المعزز

مجال التخصص: هندسة الأنظمة والتحكم

تاريخ الحصول على الدرجة: مايو 2025

النمو السريع للتكنولوجيا يؤدي إلى إدخال الروبوتات، التي كانت تُستخدم سابقًا بشكل رئيسي في المختبرات والصناعات، إلى حياة الناس اليومية للتفاعل في الأماكن المشتركة. وهذا يجعل من الضروري تطوير أطر تنقل تُوجّه الروبوتات للتحرك بطرق غير مُهدّدة وصديقة للبشر من خلال مراعاة الجوانب الاجتماعية. يجب على مثل هذه الأطر للتنقل المدرك اجتماعيًا أن تكون قابلة للتوسّع لتلائم المواقف التي تتواجد فيها عدة روبوتات وبشر يسافرون في نفس الوقت. مكّنت التطورات الأخيرة في مجال التعلم العميق المعزز (DRL) من تطوير استراتيجيات تنقل للروبوتات تتجح في تحقيق أهدافها مع تضمين هذه المعايير الاجتماعية. يطرّ هذا العمل إطارًا متعدد الروبوتات للتنقل المدرك اجتماعيًا للروبوتات المتنقلة في بيئات مشتركة مع البشر وروبوتات أخرى. تعتمد هذه المقاربة على التعلم العميق المعزز أحادي الوكيل (DRL)، حيث يتم نمذجة جميع أنواع التفاعلات في البيئة بين الروبوت نفسه، والبشر، والروبوتات الأخرى. تُستخدم طريقة التعلم العميق المعزز لتدريب الإطار من خلال وظيفة مكافأة توازن بين إكمال المهمة، وتجنب الاصطدامات، والحفاظ على مسافات مريحة من البشر، كما يُستخدم آلية الانتباه لتحديد الأهمية النسبية لكل وكيل في عملية التنقل. تم اختبار الإطار المطوّر عبر مجموعة من سيناريوهات المحاكاة بما في ذلك الأماكن المفتوحة، والممرات، والسيناريوهات ذات العوائق الثابتة. أظهرت النتائج أنه في البيئات الديناميكية متعددة الوكلاء الصعبة، يمكن للروبوت إنشاء حركات فعّالة ومقبولة اجتماعيًا. تُبرز التقييمات المقارنة مع الطرق الحديثة الرائدة في المجال مزايا النهج المطوّر، خاصةً في تعزيز السلامة من خلال تقليل الاصطدامات والحفاظ على مسافات راحة مناسبة بين البشر والوكلاء الآخرين. تؤكد هذه النتائج نجاح السياسة المستخلصة وتدعم قابلية تطبيق هذا الإطار في أنظمة الروبوتات المتعددة العملية حيث يُعد الامتثال الاجتماعي ضرورة أساسية.

CHAPTER 1 : INTRODUCTION

1.1 Background

The field of robotics has experienced rapid development in recent decades, moving from simple mechanical devices to complex machines with decision-making capacity. The sight of robots navigating in the presence of humans is now a common scene and one that will become much more common in the years to come. The future is tending towards seamless integration of robots with humans in daily activities. Robot navigation is a key capability of autonomous robotics. The issue is often characterized as the secure and autonomous attainment of a designated destination among diverse intricate circumstances [1]. Robots are increasingly being used in human environments and are progressively integrating into our everyday existence. Robots now navigate in human environments like factory floors, restaurants, hospitals and airports. Sharing the same navigation environment with humans definitely leads to unique challenges such as avoiding collisions with humans and going a step further to move in ways that do not cause discomfort or harm to humans. These robots must explicitly account for human presence in addition to achieving their navigation goals [2]. Hence it is crucial to study how humans navigate within the presence of each other and to use these navigation rules to guide robot navigation within the presence of humans.

Human navigation in shared settings is governed by social conventions, which are implicit norms that assist people in reading intentions and moving in ways that ensure mutual comfort [3]. A key concept that guides how humans move in presence of each other is proxemics [4], which describes how humans regulate interpersonal space. Humans

maintain a personal space buffer, and invading it without social cues or following acceptable conventions can cause discomfort. These spatial conventions vary by culture, age, and environment. Robots can be made to interact and move among humans in an acceptable way only if these conventions are understood and modelled in the navigation behaviour of the robots.

This research area is referred to by various terms within the robotics field, commonly including 'Human-aware navigation,' 'Social Navigation,' 'Socially Aware Navigation,' and 'Crowd Aware Navigation'. 'Human-Aware' denotes that the design and algorithm must specifically account for the presence of humans in the area of the robot, their activities, and preferences. The wording does not explicitly mean that the robot acts in a social manner. 'Social navigation' refers to navigation that incorporates social norms, rules, and roles typically employed by individuals throughout their interactions with others. 'Socially aware navigation' emphasizes the social dimension of navigation near humans but fails to explicitly convey that robot social navigation has to resemble human social navigation. In this Thesis, we decide to use the term Socially Aware Navigation (SAN), because it is comprehensive enough in meaning to convey the actual nature of the robot. The robots, and humans in the environment are called agents.

According to [5], a robot can be considered to be socially aware if:

1. It is designed to identify human agents and prioritize their safety by treating them as unique entities.
2. It is designed to reduce the human discomfort.
3. Whether overtly or covertly, it displays its navigation goals.

4. In the event of a potential collision, it considers the condition and implements actions it anticipates would resolve the conflict in a social manner, possibly at the expense of its own objectives.

While in [6] the delineated the prerequisites for socially conscious navigation as the physical safety of people, the perceived safety of individuals, the naturalness of robotic movement, and the adherence of robots to social standards. While Kruse et. Al in [2] define the fundamental requirements for navigation to be socially aware to must have taken comfort, naturalness, and sociability into consideration. Figure 1 illustrates a mobile robot planning its path based on social norms and conventions.

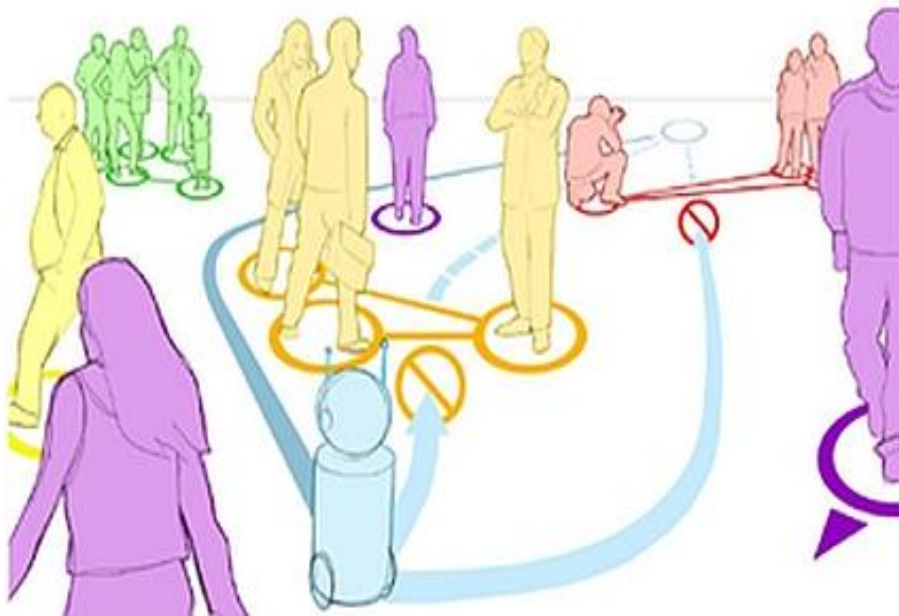


Figure 1 A robot navigating while respecting social norms[7]

Socially Aware Navigation is crucial in integrating robots into daily human activities and environments. It ensures that robots can cooperate and interact with humans in a way that will build trust and acceptance of them through respecting social norms and proxemics [8].

In environments like hospitals and healthcare settings, also robots need to assist humans and guide or navigate around them without causing discomfort or stress. Therefore, the need for mobile robots in social and human settings cannot be overstated. In a similar way that humans navigate in a given environment for the same purpose or different purposes, mobile robots may be navigating in that environment cooperatively with the same goal, or with different purposes and goals. These robots may take into consideration social norms to also be socially aware or not be encoded with these abilities, so it is central and important to develop navigation frame works for mobile robots to navigate in densely populated environments of humans and other mobile robots. These frameworks must also scale to high density and complex environments with high concentration of humans, and other various types of robots, and be able to navigate in different environmental settings as well, such as in indoor environments like corridors, elevators, doorways, and open space environments with many obstacles around. The interaction types are also of high importance. The framework also needs to accurately model all the different types of interactions in the navigation environment based on the characteristics and properties of the agents.

Indeed, when multiple agents or entities are moving and interacting in an environment, then it must clearly be encoded which type of agent is being considered, and which agent is interacting with which. For the case of multiple robots in the same environment, similarly, the robots must be able to differentiate between themselves and humans. Some methods also consider cooperation between robots, however such methods can be prone to communication or sensing delays. Generally, for multirobot schemes each robot makes navigation decisions based on its local observations and in some cases based on hard-coded

or predefined rules. Classical multi-robot co-planning methods can be divided into centralized methods and decentralized methods [9]. Centralized methods learn using information that is centrally shared between the agents. On the other hand, these procedures are dependent on the use of precise global information gathering or aggregation strategies. Furthermore, they are characterised by a variety of problems, including a lack of scalability and a discrete dimensional explosion, which specifically manifests itself as the number of agents rises. Hence it may be advantageous to use Decentralized training and executing paradigms due to ease of scalability and lack of need for communication or cooperation. Reinforcement learning (RL) is the theoretical foundation of this thesis. RL is a framework where agents learn efficient methods for decision-making by attempting different actions and obtaining rewards. The ultimate aim is for the agents to maximise their overall rewards over time. The entity that takes in data about its surroundings and then uses that data to make decisions about how to change that environment is called an agent. In single-agent RL, the decision-making entity is one, and all other entities form part of the environment. In Multi-Agent RL, more than one agent is involved and they each choose individual actions which jointly change the state of the environment, and obtain individual rewards as a result of this change as well as individual observations about the new environment state [10].

1.2 Problem Statement

Traditional robot navigation uses sensor-based perception to plan and execute collision-free pathways in real time. Socially aware navigation (SAN) requires robots to respect human social conventions, avoid discomfort, and move naturally. This includes avoiding obstacles and predicting and responding to human behavior. Multiple robots frequently

interact with people in real life, necessitating modelling socially acceptable interactions between them and humans in the environment.

Existing centralized multi-robot systems, however, rely on real-time global coordination, which is often challenging given sensor lag, communication delay, and processing limitations. Particularly in crowded, dynamic contexts where delays or interrupted communication can lead to dangerous or inappropriate behavior, these systems struggle to scale. Moreover, centralized approaches can ignore the subtle variations between types and navigation behaviors of the robots and the robot-robot and human-robot interactions. Although robots may collaborate collaboratively, pedestrian behavior is socially driven and less predictable, hence alternative modeling techniques are needed. Particularly in dense scenarios where human-human dynamics influence robot behavior, effective social aware navigation in such circumstances requires a unified model that can recognize and react to various interaction kinds.

To address these difficulties, this thesis incorporates social rules guiding navigation among humans in a decentralized multi-robot framework which allows the robot being controlled to learn and use its local observations to make real time navigation decisions without communicating with other robots or humans, and the other robots in the environment to follow a predefined navigation rule.

The problem is treated as a sequential decision-making process and is modeled as a Partially Observable Markov Decision Process (POMDP). It is considered as a decentralized framework because local information (state observations) available to each robot is enough for it to achieve its goal without communicating with other robots. This

thesis addresses specifically how to model the interactions between the humans and robots in a decentralized navigation framework employing deep reinforcement learning.

1.3 Objectives

This thesis aims to develop a multirobot SAN system that develops a framework based on Deep Reinforcement Learning that guides the navigation of multiple robots within the presence of humans.

I aim to achieve this, through the following objectives:

1. Develop a multi-robot socially aware navigation framework using Single-Agent Deep Reinforcement Learning, enabling the ego agent to navigate among other agents who use a predefined collision avoidance strategy, thereby considering human-human, human-robot, and robot-robot interactions.
2. Evaluate the developed framework in multiple scenarios, involving diverse configurations of humans, robots, and environmental layouts.
3. Benchmark the proposed framework against state-of-the-art methods in terms of navigation time, success rate, collision rate, and discomfort rate.

1.4 Contributions

The contributions I have made in my work can be summarized below:

1. Developed a navigation framework based on single-agent deep reinforcement learning, which allows the ego robot to move according to social norms among humans and other robots that follow a predefined collision avoidance behaviour.

2. Evaluated the framework in a variety of simulation scenarios including crowded spaces, narrow corridors, and environments with obstacles, demonstrating effective and socially aware path planning.
3. Showed improved safety performance over existing methods, especially in reducing collisions and maintaining comfortable distances from other human agents

1.5 Thesis Organization

The first chapter of this thesis introduces the concept of social awareness and its significance in navigation of mobile robots in human environments.

The second chapter presents a comprehensive background and a review of relevant literature in this field. This chapter discusses Traditional approaches, Multi-Robot approaches, and gives a tabular summary of the related work on the Multi-Robot approaches and how they compare to this work.

The third chapter comprehensively discusses preliminary information and theoretical background of this work. It gives a succinct introduction to the key concepts and theories that are fundamental to coming up with the methodology of this work and achieving it. In this chapter, the concepts of Reinforcement Learning, Deep Learning, Deep Reinforcement Learning, and Imitation Learning are discussed.

The fourth chapter presents the methodology of this work and discusses the approaches and methods that have been followed to achieve the objectives of this thesis. It gives the problem formulation, discussing Markov Decision Processes. It also discusses how the different interactions between humans and robots are modelled. The chapter then provides

the mechanism of the formulation or the navigation policy, the system architecture, the implementation details, the simulation environment, and finally the metrics used to evaluate the performance of the navigation algorithm.

The fifth chapter presents the outcomes of the application of the methodology to achieve the laid down objectives of the thesis. The results obtained for the different scenarios and conditions are presented and discussed to highlight their relevance and interpretation. Finally, they are then compared with results from other state of the art and relevant previous work in terms of some important performance metrics.

Finally, the sixth chapter concludes the thesis by giving an overview of what has been achieved by this thesis and its relevance and implications. Then the limitations of the thesis are outlined and future recommendations for the potential improvement and real-world application and deployment are discussed.

CHAPTER 2 : LITERATURE REVIEW

2.1 Introduction

The development of robot navigation technology enables robots to independently identify and avoid obstacles, plan suitable routes, and reach their destinations in human populated environments. This chapter discusses the relevant related literature on socially aware navigation, starting from the traditional and foundational methods up to more recent methods. Likewise, a discussion of the related work on multirobot socially-aware navigation is done, along with a summary and comparison of the related works to this work.

2.2 Traditional Navigation Approaches

Navigation is the basic requirement for any autonomous mobile robots. Traditional navigation systems purely rely on sensory inputs and use image processing or computer vision [11]. Robots equipped with this technology employ cameras and laser sensors (LiDAR) to detect their environments and determine their position and direction using algorithms. With an understanding of the environment, pre-built maps or maps created in real-time are used to plan paths that guide the robots to specific targets. Additionally, some algorithms have been developed to both construct maps of the environment plan paths simultaneously. Generally, there are two types of maps used for path planning: grid maps [12], and probabilistic maps [13].

To further improve traditional approaches for navigation and make them more reliable especially when there are specific navigation requirements, machine learning approaches

have been applied. Advanced techniques like neural networks and deep learning, have improved robot navigation in complex environments.

With the increasing overlap of robots with humans in spaces, it becomes more necessary to establish rules for social interactions between the two. 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, naturalness, and sociability during the navigation process, namely social awareness [14].

2.3 Socially Aware Navigation Approaches

With the advancement of social robotics, the significance of incorporating social behaviour in navigation of robots has become more evident. 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 [15].

One of the earliest efforts to incorporate social awareness into robot navigation systems was tour guide robots deployed in museums around the late 90s, such as the RHINO [16] and MINERVA [17] robots in Germany and the US, respectively. These robots used localization and mapping to engage users in natural human settings.

Existing works on social-aware robot navigation can be roughly grouped into three categories as characterized by [18], and these are: (1) reaction-based methods; (2) proactive methods; (3) learning-based methods.

2.3.1 Reaction-based methods

Robots navigating in dynamic or unknown environments, must respond quickly to avoid dynamic obstacles while planning their paths. This kind of reactive local navigation scheme is fundamental in maintaining a safe environment, and methods like these serve as the basic approach in social-aware robot navigation. In reaction-based methods, robots react with other mobile agents using one-step interaction approaches [19]. These methods can be considered to be purely rule-based, without any prediction or learning happening. Such methods include the Velocity Obstacles (VOs) [20], [21], [22], [23], [24], Social Force Models (SFMs) [17], [25], [26], Vector Field Histograms (VFHs) [27], [28] or Artificial Potential Fields (APFs) [29], [30].

Velocity Obstacles was developed in [20] and has been frequently implemented in traditional robot navigation, such as in [21], [22], [23] as well as in socially aware navigation works in general. The Velocity Obstacle of a moving obstacle with respect to an agent is the set of all velocities that would eventually lead to a collision. Therefore, during each planning step, the agent must select a velocity that does not fall within any of the VOs created by the moving obstacles. Reciprocal Velocity Obstacle (RVO) [23] prevents possible oscillations that might occur when approaching or crossing paths with one another in the case of multiple agents operating independently but in the same environment and where all agents are assumed to use the same navigation technique. The Optimal Reciprocal Collision Avoidance (ORCA) [22] improved RVO by introducing a cost function and an optimization method to select an optimal velocity. The Hybrid Reciprocal Velocity Obstacle (HRVO) [24] extends RVO by setting priorities in the interaction between robots.

The APF, derived from virtual potential fields, is a virtual force field methodology originally introduced by Khatib. [29]. It considers the environment around the robot as an energy field, where the motion of the robot is determined by the forces of attraction generated by the goal point and repulsion between the robot and obstacles. Yao et al. in [30] incorporated reinforcement learning to the APF to improve performance in dynamic scenes. VFHs [27] construct a two-dimensional histogram to model environmental information. The authors in [28] propose the VFH* to handle dynamic obstacles. In addition, SFM is proposed in [17], and is a physics-based method that infers the direction and speed of mobile agents' for collision-avoidance by simulating the interaction forces between individuals, such as attraction and repulsion forces. It was extended in [31] and applied to a robot companion that uses interactive learning to adjust the parameters of the model. By taking advantage of both the extended SFM and the HRVO, the authors in [26], propose Proactive SFM (PSFM) to achieve efficient and proactive collision-avoidance navigation. It considers both the human states in relation to the robot (position, mobility, and hand posture) and the social interaction data concerning human obstacles and group dynamics. However, the above approaches heavily depend on the precision 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 subsequent step based on one-step rule and the current state, the planning path is short sighted and unnatural. This may lead to a failure to adhere to social norms, especially in complex or crowded environments.

2.3.2 Proactive methods

Proactive methods assume that the humans cooperate towards the navigation and do not behave in adversarial manners. As a requirement, these approaches must predict the behaviour of humans through their trajectories and plan a path to follow. The trajectories or behaviours are obtained from large-scale datasets to form a Prediction Motion Model, and then a Path Planning method is incorporated. By proactively considering human actions, these methods achieve smoother interactions which lead to robot navigation within human social norms. 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 [32] 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. In [33], the authors combined Rapidly-exploring Random Trees (RRT) with Gaussian Processes to improve Gaussian predictions and identify probabilistically feasible paths. In scenarios where a robot shares the environment with multiple moving agents, it is absolutely necessary 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 accurately predict behavioral change of each agent over time, understanding and adapting to human social norms. Common methods for predicting trajectories include Kalman filters or Particle filters, but goal-based policy methods have been shown to provide more suitable and effective solutions for social navigation, achieving more natural and socially acceptable 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 human goal-oriented trajectories.

Furthermore, considering interactions with static and dynamic obstacles, data-driven methods have also been proposed to capture agent motion. In [34], a data-driven architecture called Social-LSTM was proposed to predict future human trajectories. It introduces a social pooling layer to capture the dependencies between multiple related sequences. Considering implicit cooperation between different agents in collision-free navigation, [35] proposed Social Attention to obtain the relative importance of all mobile agents while navigating, without considering the distance between them. After establishing the human motion model, trajectory-based methods define a planner that can find the optimal navigation policy. Svenstrup et al. in [36] propose a path planner based on RRT to avoid unforeseen pedestrians. Moreover, in order to avoiding crossing into personal space of people while avoiding collisions, [37] designed the Risk-RRT method, which employs 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) [38], where the planner predicts that all feasible routes are deemed unsafe, resulting in the robot being stuck and unable to move.

2.3.3 Learning-based methods

Researchers have been examining and using ML and AI-based techniques for robot navigation more and more in recent years. The number of works incorporating DL methods to Reinforcement Learning (RL) has been steadily increasing. These learning-based methods have shown improved navigation performance in advanced and dynamic environments. Deep Reinforcement Learning (DRL) is widely used in mobile robot navigation to learn optimal navigation strategies, by introducing deep neural networks to solve the bottleneck of RL in dealing with high dimensional state spaces. 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 accurately performs trajectory planning and collision avoidance simultaneously and effectively. In recent years, researchers have used DRL for collision avoidance in robot navigation, and some of these methods incorporate elements of social awareness. CADRL (Collision Avoidance with DRL) [39] was the earliest work replacing handcrafted 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 [40]. The algorithm 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. In [41], LSTM-RL was developed by using the Long-Short Term Memory (LSTM) to encode other agents and model the

collective impact of the crowd. It sequentially processes the current states of each neighbor in reverse order of proximity to the robot. To enhance the comfort of individuals coexisting with a robot in a crowded environment, [42] introduced the social stress index within a DRL, which captures local features and calculates social-attention scores through a multilayer perceptron. The authors in [43] introduced a method that combines robot velocities learned from an RL model (AutoRL [44]) with robot velocities calculated using an SFM to determine robot actions.

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 social environments as a one-way human-robot interaction problem, which could pose challenges as the size of the crowd increases. To overcome these limitations, [45] proposed a Social Attention Reinforcement Learning (SARL) approach, explicitly modeling the interactions between the robot and the crowd. The self-attention technique is applied in order to find the aggregate impact of the crowd. This is accomplished by taking into account both human-human and human-robot interactions. Furthermore, SOADRL [46] enhanced SARL by processing static and dynamic obstacle information independently, solving the difficulty of navigating with sensors giving only a restricted field of view within crowds. In [47], the most important individuals with respect to navigation are highlighted within the crowd. The work in [48] suggested simulating danger zones for the robot in order to react in real time to human actions. The definition of these zones is based on tracking human behavior in real-time and then storing all the potential actions that individuals may do at any given moment. The robot learns to navigate clear of these danger zones so as to achieve dependable and safe

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 social attention scores of humans when moving toward the goals. In order to accurately predict human attention for various agents within the crowd during navigation, the work in [47] employs a method based on human gaze data to train a Graph Convolutional Network (GCN), to accurately predict human attention to different agents within the crowd during navigation. Subsequently, they incorporated the acquired attention into a graph-based RL framework. Due to the dynamic interactions within a crowd generating non-Euclidean data, Graph Neural Networks (GNNs) can effectively extract representations for socially aware navigation. Graph Neural Networks (GNNs) combine the robust representational capabilities of graphs with the comprehensive learning potential of neural networks. 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 two most common forms of GNNs are Graph Attention Networks (GATs) and Graph Convolutional Networks (GCNs). The work in [49] introduces two-layer GCNs to model a Relation Graph Learning (RGL) structure to learn the agents' interactions. Unlike GCNs, which use preset weights in convolution operations, GATs rely on learnable attention mechanisms to dynamically assess the significance of each neighbor during input aggregation. The work in [50] improves two-layer GATs to extract graph representation, and the resulting model

is termed 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.

Imitation Learning (IL), which is a sample-efficient method that is commonly used to obtain a navigation model based on the training data, is one example of a method that employs supervised learning schemes to emulate expert demonstrations for the purpose of improving learning efficiency. The work in [51] proposes a real-time, socially aware method for movement of mobile robots among humans using visual inputs and Generative Adversarial Imitation Learning (GAIL). This approach improves safety and efficiency in real-world deployments, also offering a framework to model and simulate pedestrian behaviors. While in [52], the authors combined IL and RL for single-goal navigation in mapless scenarios. Inverse Reinforcement Learning (IRL) has also found good use due its ability to learn the reward function directly from data which is why it is termed ‘Inverse’. In [53], the authors used a probabilistic technique to enhance the parameters of the reward function using optimization based on maximum entropy, and this was used to learn human-aware navigation in densely populated human environments. Pérez-Higueras et al. in [54] also suggested IRL for social navigation. As an alternative to utilizing the costs for the purpose of path planning, they are utilized to acquire knowledge about local execution policies and to provide the robot with navigation directions.

2.4 Multirobot Socially aware DRL methods

Challenges arise when multiple robots are navigating within the same environment, in the presence of many humans. Policies must be developed that model both human-robot and robot-robot interactions. Recently, researchers have started to extend single-robot methods to multi-robot approaches because, often, it is required for multiple robots to navigate simultaneously in the same environment in the presence of humans to achieve their navigation goals collectively or independently. However, multi-robot navigation has its own challenges. Due to a greater number of agents in the environment collisions are more likely to happen, a significantly larger amount of computation is required, and coordination or cooperation between the robots in the environment must be ensured. Multirobot approaches in SAN include the ability to leverage collective information and distributed decision-making to improve navigation outcomes. The work in [55] developed a Multi-agent RL approach. It addresses robot dynamics through non-uniform linear motion constraints and models the environment using a Decentralized Partially Observable Semi-Markov Decision Process (Dec-POSMDP). A hybrid spatio-temporal transformer is also used to model human-robot interactions over time, enhancing the robots ability to make socially-aware decisions. The model uses separate spatial-temporal transformer blocks for modelling both human-robot interactions (HRI) and robot-robot interactions (RRI). An Attention Mechanism is also used to determine the importance of each agent (human or robot) relative to others. After processing spatial and temporal features separately, the model uses a multi-modal transformer to align and fuse these features, creating a comprehensive state representation that includes both human-robot and robot-robot interactions. Additionally, it uses a Centralized Training and Decentralized Execution

(CTDE) paradigm, where the state information is centrally shared and used for training of all agents, then the agent learn their policies using this information in a decentralized manner and perform individual actions.

The multi-robot technique presented by [9], which uses an Actor-Critic RL approach, where multiple robots cooperatively plan their navigation in pedestrian environments, using a Temporal-Spatial graph-based encoder to obtain the significance of social interactions. It also incorporates a K-step look ahead reward mechanism to allow robots to consider future consequences hence avoiding aggressive short short-sighted decisions. The problem is similarly modelled as a Dec-POSMDP. Robot-robot interactions are cooperative and are captured using a centralized value-network that shares necessary information about each robot needed to facilitate accurate decision-making. This method similarly uses the CTDE Multi-Agent RL paradigm in training and execution.

The method proposed in [56] takes a different approach. It uses a Graph Neural Network (GNN) to model interactions between robots and pedestrians combining their positions and field of view to get an accurate representation of the environment. An edge selector mechanism is also used to identify the most important interactions between agents by creating a sparse graph and applying attention on the nodes. A Gated Recurrent Unit is used to synchronize with previous actions by keeping track of the hidden state, which helps to make decisions based on prior experience. It also employs the CTDE paradigm. The framework is simulated and compared with other multirobot frameworks.

Another method, [57] has modelled navigation among humans and multiple robots as a POMDP. It develops a Heterogeneous Relational Single-agent DRL method for

Decentralized Multi-Robot Crowd Navigation. A two-layer heterogeneous GNN is employed to transfer and aggregate information among different types of agents (robots and humans). All types of interactions, namely human-human, robot-robot, and human-robot have their specific subgraph to model the relationship then aggregate it with others to obtain necessary navigation information.

2.5 Summary of Related Work

The section presents a summary of relevant existing multirobot studies which are related to this work. The previously mentioned methods in the previous subsection are compared in Table 1 to show the differences and similarities with the proposed work. This work, unlike previous ones that only do single-agent RL for single robot SAN or multi-agent RL for multi-robot SAN, develops a socially acceptable navigation framework for multi-robot, multi-human environments using a single-agent deep reinforcement learning (DRL) policy. By relying on single agent RL, we avoid training complexity, stability issues, and poor generalization which are common in MARL. A single trained ego robot is trained through imitation learning and reinforcement learning to navigate with social awareness, with the other robot and humans following a predefined policy, that enables other robots to navigate with more social distance in presence of humans. This way of modelling more realistically reflects shared spaces where robots may not be jointly trained or have communication abilities.

Table 1 Comparison of Multi-robot socially aware schemes

Work	RL Framework	Architecture	Human-Robot vs Robot-Robot Modelling	Training strategy	Social awareness level
[56]	Multi-Agent Proximal Policy Optimization (MAPPO)	Graph Neural Network with Attention Mechanisms	Human-robot and robot-robot interactions are modelled the same	CTDE without Imitation Learning	All robots are fully socially aware
[55]	Multi-Agent Proximal Policy Optimization (MAPPO)	Hybrid spatio-temporal transformers with Attention Mechanisms	Interactions are modelled differently	CTDE without Imitation Learning	All robots are fully socially aware
[9]	Multi-Agent Actor-Critic RL	Temporal-spatial graphs with Attention Mechanisms	explicitly differentiates humans and robots in the architecture	CTDE without Imitation Learning	All robots are fully socially aware
[57]	Single-Agent Q-Learning	Graph Neural Networks	robot-robot interaction is modelled same as human-robot interaction	Single-Agent Decentralized without Imitation Learning	Full social awareness for Controlled robot and no social awareness for other robots
Proposed Work	Single-Agent Q-Learning	Feedforward Neural Network with Attention Mechanism	robot-robot interaction modelled same as human-robot interaction	Single-Agent Decentralized with Imitation Learning	Full social awareness for ego robot and partial social awareness for other robots

2.6 Conclusion

This chapter has presented the relevant related literature on socially aware navigation. The different categories of approaches were presented, starting with traditional up to the latest learning-based approaches. Additionally, the relevant related works on Multirobot SAN were highlighted and then compared to this work.

CHAPTER 3 PRELIMINARIES

3.1 Introduction

The fundamental concepts and theoretical background relevant to this thesis is discussed in this chapter. It introduces reinforcement learning, Markov Decision Processes, policy and value functions, Q-learning, and Temporal difference learning. It also details some relevant concepts and types of Deep Neural Networks, such as feedforward neural networks, and graph neural networks. Then it discusses the types of deep reinforcement learning methods, and finally it introduces Imitation learning which is also used in this work.

3.2 Reinforcement Learning

Reinforcement Learning [58] is a subset of Machine Learning where an algorithm aims to acquire solutions for sequential decision processes by iterative engagement with the environment. The decision maker or controller is called the agent and it receives state information from the environment and takes specific actions based on its current state which then influences the environment and subsequently gets a reward value (positive or negative) based on the desired goal or outcome. The goal of the agent is to learn an action policy by continuously adjusting strategies through trial and error when interacting with the environment, with the aiming of maximizing the long-term cumulative reward. In the case of our work the agent is the robot and the environment comprises of everything else beyond the robot which includes other robots in the environment, humans, obstacles, and walls. The robot needs to select appropriate actions, such as turn left, right, go straight, or

stop based on the current state information like velocity and position. The environment then provides corresponding punishments or rewards based on the robot's actions. These may include reward for reaching goal, and punishments for collision, coming into the discomfort zone of a human, or having jerky unnatural motion. Figure 2 depicts the agent-environment interaction in a basic RL loop, where the agent makes use of environmental state information to take an action in the environment which changes the state and gives a reward (or penalty) to the agent. The outcome of this repeated process is an optimal policy which selects actions in each state to maximize the expected return. However, this learning process is characterized by the exploration-exploitation problem, which aims to balance exploring the outcomes of different actions against staying with actions that have currently been the best.

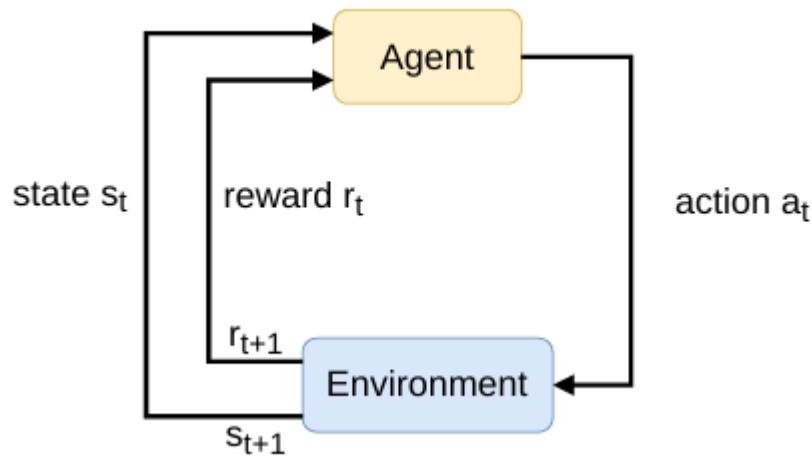


Figure 2 Reinforcement Learning Loop

3.2.1 Markov Decision Processes

An important concept to introduce is the Markov Decision Process (MDP) which is utilized to formalize Sequential decision processes. It serves as a conceptual framework for representing decision situations in which outcomes are affected by a series of decisions and

their random events. The MDP describes a decision maker (agent) in a state, s , taking an action a , which based on probability changes the state and gets a reward R . In this new state, it must make a choice on which action to subsequently execute, and this continues until a stopping criteria is met. The MDP can be described by a tuple $\langle S, A, P, R \rangle$ where— s belongs to a set of states, S , called State Space, a also belongs to a set of actions, A , called Action Space, P is the probability that an action a_t in a state s_t , will cause a transition to state s_{t+1} . This is called the transition probability, and is given by $P = \mathbb{P}(s_{t+1}|s_t, a_t)$. R is the scalar value of the reward received after the change from s_t to s_{t+1} while taking the action a_t .

A discount factor, γ is introduced to assess the impact 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 policy, π , defines how the agent behaves and can be considered as a link from states to actions that specifies the action to be taken at each state. At state s_t , the agent takes an action a_t , based on policy π and then moves to the next state s_{t+1} based on P and gets feedback in form of a reward signal R_t from the environment. The generated sequence is thus: $S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3 \dots$

The goal in Reinforcement Learning is to choose the best possible mapping from state to action, called policy, that will lead to the maximum possible expected reward G_t , defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^T \gamma^k R_{t+k+1} \quad (3.1)$$

Where γ is a value between 0 and 1 and T is ∞ or a finite value for infinite horizon or finite problems, respectively. And the optimal policy itself is defined as

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \mathbb{E}[G_t] \quad (3.2)$$

3.2.2 Value Function

Two measures frequently used to obtain expected cumulative reward for each state or state-action pair are the value $V_\pi(s)$ and action value $Q_\pi(s, a)$ respectively. These values can evaluate the strategy and guide the agent's decision-making under different states.

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] \quad (3.3)$$

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

These equations can be obtained recursively through the action-value function of the successor state. These equations are called Bellman Equations, and they may be estimated using Dynamic Programming. (DP) [59]. Using incremental updates, the states' optimal value function can be obtained. 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:

$$\pi^* = \underset{a}{\operatorname{argmax}} V^*(s) \quad (3.5)$$

$$V^*(s) = \max_a R_s^a + \gamma \sum_{s'} P_{ss'}^a V^*(s') \quad (3.6)$$

3.2.3 Q-Learning

The term $Q_\pi(s, a)$ can also be called the Q-function. It is the value of an optimal policy when the first selected action is imposed. Upon application of the Bellman Equation we obtain the equation for the Q-function as:

$$Q_\pi(s_t, a_t) = \mathbb{E}[r(s_t, a_t) + \gamma \mathbb{E}[Q_\pi(s_{t+1}, a_{t+1})]] \quad (3.7)$$

The optimal Q-function can be obtained by using a model-free reinforcement learning algorithm called the Q-learning algorithm. 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. In Algorithm 1, we present its pseudo code sequentially. The exploration rate, ε is used in 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. The process function is highlighted in Algorithm 1.

Algorithm 1 Q-learning

Input: the policy π

Initialization: $Q(s, a)$ Using arbitrary values (e.g. zeros) initialize $Q(s, a)$

for each episode **do**

 initialize s

for each non-terminal state s in each episode **do**

$a \leftarrow$ action for s derived by Q

 select 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

Output: action-value function Q

3.2.4 Temporal Difference (TD) Learning

Temporal Difference (TD) [60] is critical to RL. This is what is used to estimate the value function and is capable of evaluating the value function after performing an action. Temporal Difference (TD) error, which is the variation between the present state's estimated value and the future state's predicted value, is the foundation of TD learning. This error is utilized to update the state value function with the purpose of obtaining better returns. Through continuous iterations, the agent is able to make the best possible predictions as TD learning converges to the optimal value function over time. Its update rule is:

$$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)] \quad (3.8)$$

where $r + \gamma V(s') - V(s)$ is the TD error, and α is a learning rate. Its pseudo-code is presented in Algorithm 2 to introduce the implementation method of the TD learning algorithm.

Algorithm 2 TD learning

Input: the policy π

Initialization: Using arbitrary values (e.g. zeros) initialize $V(s)$

for each episode **do**

 initialize s

for each non-terminal state s in each episode **do**

$a \leftarrow$ action for s obtained from π

 select action a , observe r, s'

$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$

$s \leftarrow s'$

end for

end for

Output: value function $V(s)$

3.2.5 Deep Learning

In the domain of reinforcement learning there is typically a large search space which makes it necessary to use parameterized representations of the value or policy functions. The objective is to approximate the true value or policy function as closely as possible using these parameterized models. Accordingly, the common problem of the curse of dimensionality in which the cost of computation increases with increasing number of dimensions of the state space needs to be addressed. This can be done through Deep

learning by using neural networks. The objective is to find the parameter of a function that minimizes the loss between the estimated Q function and the real Q function. So, the parameter can be updated by taking the gradient of the loss function, where the loss function can be the mean square error. DL approximates non-linear functions by training a Neural Network with vast amounts of layers to learn the intrinsic representations and essential features of the input data.

3.2.6 Neural Networks

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) usually consist of layers upon layers, but basically comprise of input, hidden, and output layers in that order. The input and output layers take in and give out data, respectively, and there is at least one hidden layer in between. Each layer uses the output of the previous layer as its input. NNs are a powerful tool for approximating functions due to their highly flexible structure and parameter tuning capabilities. Deep NNs (DNNs) (with multiple hidden layers) are particularly good at approximating complex nonlinear functional relationships.

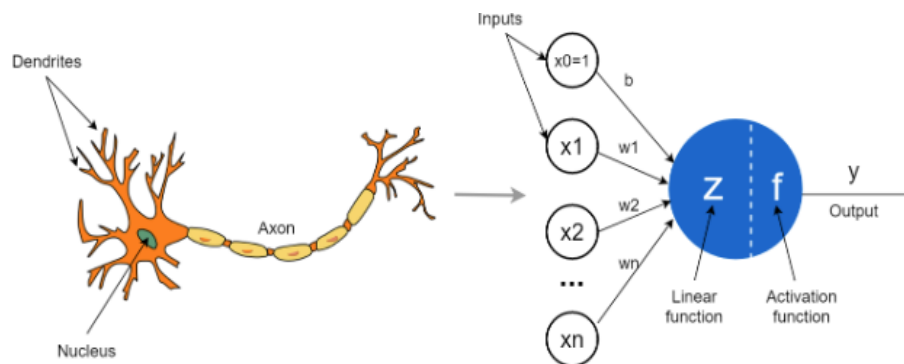


Figure 3 Illustration of biological neuron and artificial neuron[61]

3.2.6.1 Neuron

Artificial neurons are the basic processing units of neural networks. Inspired by biological neurons. 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 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:

$$y = f(b + \sum w_i x_i) \quad (3.8)$$

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 the layer.

3.2.6.2 Output Layer

The output layer gives the final representation in the network, and is 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, and their selection is task specific. When dealing with linear regression

problems, linear activation functions are the always choice because they directly allow for straightforward prediction results. A linear activation function is given by:

$$f_{linear}(x) = x \quad (3.9)$$

However, their input-output relationship is limited to simple linear transformations which can limit the ability of deep neural networks to model complex patterns.

For the majority of tasks, and in this thesis nonlinear activation functions are favored, such as Sigmoid, ReLU, Softmax. 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:

$$f_{sigmoid}(x) = \frac{1}{1 + e^{-x}} \quad (3.10)$$

which 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:

$$f_{softmax}(s_i) = \frac{e^{s_i}}{\sum_j e^{s_j}} \quad (3.10)$$

where s_i is the output of the i -th neuron. The softmax function transforms the input values into probability distributions for various classes, ensuring that the sum of all its output values is equal to 1. In addition, to mitigate the problem of vanishing gradient, ReLU is used to keep the gradient from decaying when $x > 0$:

$$f_{ReLU}(x) = \begin{cases} x, & x \geq 0 \\ 0, & x < 0 \end{cases} \quad (3.11)$$

3.2.6.3 Objective Function

The objective function, in this context also called the loss function or cost function, is used to measure the divergence or error between the output and the desired value of a neural network. It guides the network's parameter learning and updates by backpropagating the measurement error, aiming to minimize the objective function. The neural network's ability to accurately match the training data improves as the parameters are fine-tuned, this improves model performance and generalization ability. A widely used objective function is the mean squared error (MSE):

$$L_{MSE}(\theta) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}(x_i, \theta))^2 \quad (3.12)$$

where the loss function 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. 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.

3.2.6.4 Network Training

A neural network is trained in a process of parameter tuning to make it learn from the input data and gradually fit the desired output. The main and most common neural network training methods are Gradient Descent (GD) and Stochastic Gradient Descent (SGD). GD

performs optimization by computing the objective function's gradient with respect to the parameters and updating the parameter values in the gradient's descent direction, it progressively reduces the objective function.

$$\theta_{t+1} = \theta_t - \alpha \nabla L \theta_t \quad (3.13)$$

where θ_t is the parameter at the t -th iteration. L is the objective function and $\nabla L \theta_t$ is its derivative with respect to θ_t . The learning rate α regulates the step size at each update in negative gradient direction. The value of the learning rate significantly affects the effectiveness and speed of training. If a very small value of α is chosen, the step size of each parameter update becomes small, and the training process will become very slow leading to a delayed rate of convergence. But, if α is set too large, the training may become unstable, and may even lead to the objective function constantly jumping or even diverging from an optimal solution. In machine learning applications, a common optimization algorithm, Adam [62], is often preferred because it automatically adjusts the learning rate for each parameter, helping to stabilize training and accelerate convergence without the need for extensive manual tuning of the learning rate.

3.2.7 Feedforward Neural Network

If the information in a neural network flows in a forward manner without forming cyclic connections, then this neural network model is called a FeedForward neural network or Multilayer Perceptron (MLP). This model of NNs is widely used due to its simplicity and ability of other models to build upon it. A typical feedforward neural network with three inputs neurons, three hidden layers, and two output neurons is shown in Figure 4.

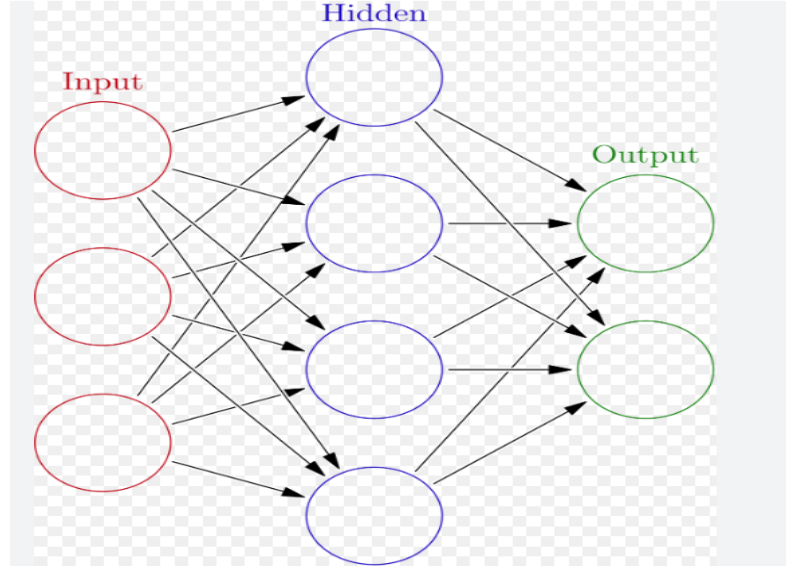


Figure 4. A feedforward Neural Network [63]

3.2.8 Graph Neural Network

The Graph Neural Network (GNN) [64], [65] is another widely used type of network and is shown in Figure 5. It is another neural network architecture specifically designed for handling data structured in the form of graphs. They can handle data with irregular spatial structures commonly found in real-life scenarios, i.e., data in non-Euclidean spaces. The GNN is made up of graphs which are built upon interconnected nodes and edges. It can be represented as $G = (V, E, A)$ with V being the nodes, E the edges, and A representing the adjacency matrix of the graph. $v_i \in V$ is the i th node, and $e_{ij} \in E$ is the edge between i th node and j th node. If there exists $e_{ij} \in E$, then $A_{ij} = w_{ij} > 0$, otherwise $A_{ij} = 0$ if $e_{ij} \notin E$.

The goal of GNNs is to learn feature representations between nodes and graphs. GNNs 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 most common models of GNNs are Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs).

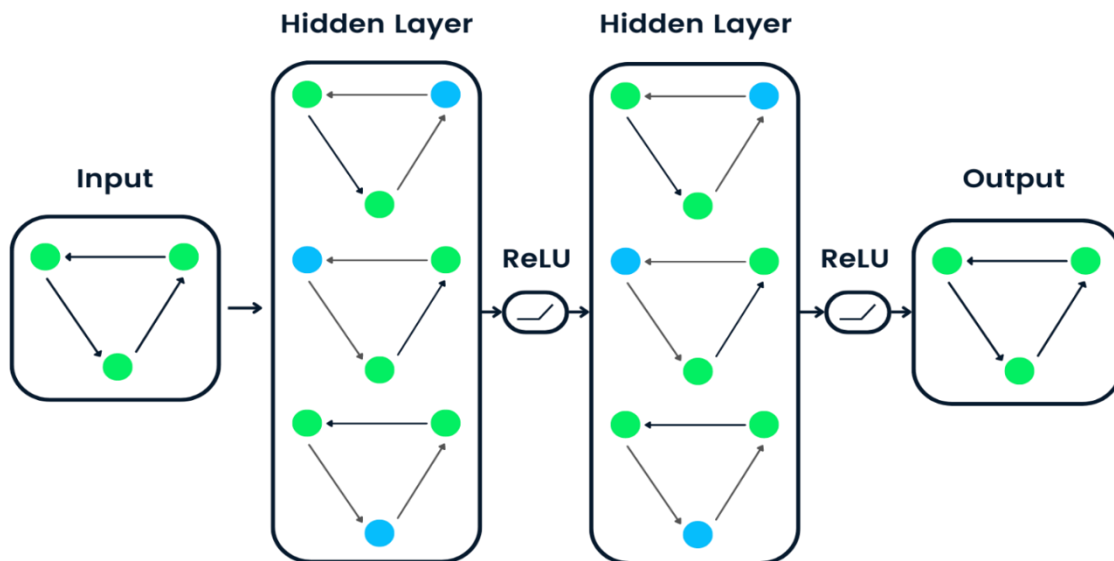


Figure 5. Structure of a simple Graph Neural Network [66]

3.2.9 Attention Mechanism

The attention mechanism [67] was initially introduced in the context of sequence modeling to enable models to dynamically weigh input elements based on their relevance to a given task. At its core, attention operates by computing a compatibility score between a query vector and a set of key-value pairs. In formal terms, given a query q , a set of keys $K = \{k_1, k_2, \dots, k_n\}$ and corresponding values $V = \{v_1, v_2, \dots, v_n\}$, the attention output is a weighted sum of the values

$$Attention(q, K, V) = \sum_{i=1}^n \alpha_i v_i \quad (3.14)$$

where

$$\alpha = \frac{\exp(q^T k_i)}{\sum_{j=1}^n \exp(q^T k_j)} \quad (3.15)$$

This softmax-weighted aggregation allows the model to dynamically focus on more relevant parts of the input, a property that has found substantial utility in navigation tasks with multiple agents and dynamic obstacles. Figure 6 shows the most common types of Attention architectures.

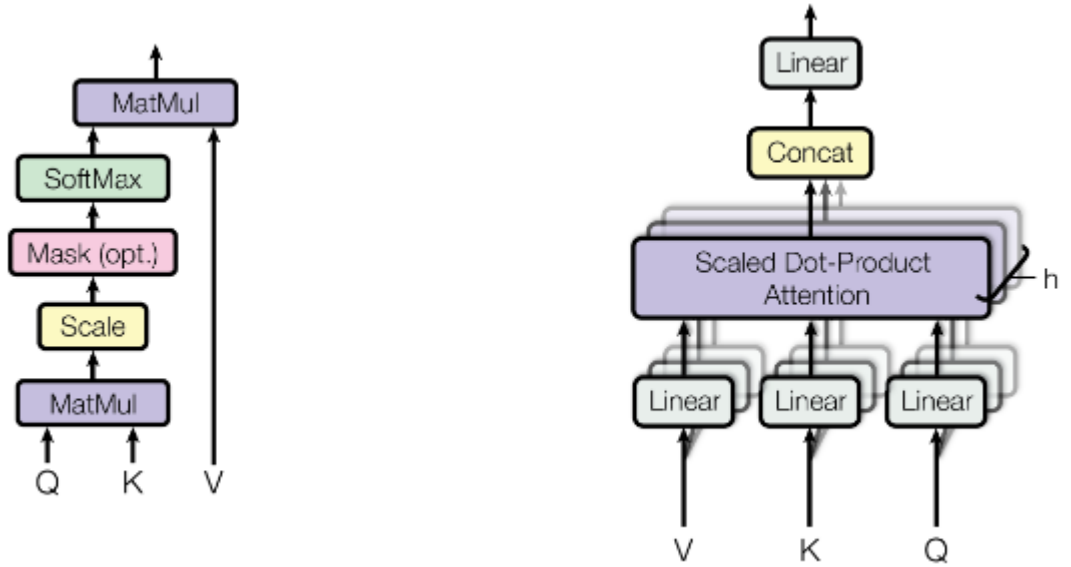


Figure 6. Scaled Dot-Product Attention (a) and Multi-Head Attention (b) [67]

In multirobot SAN, attention mechanisms facilitate selective interaction modeling, allowing each robot to prioritize which neighboring agents influence its motion planning. Instead of treating all surrounding entities uniformly, attention enables the robot to allocate computational focus toward those agents whose trajectories are most relevant to the robot's current goal and state. This selective encoding is critical in densely populated environments where not all agents contribute equally to a robot's decision-making process. Similarly, in

this work we apply it to focus on most relevant interactions in the environment when navigating.

3.3 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is an offshoot of Machine Learning that integrates Deep Neural Networks with RL to solve specific types of problems. In DRL, the ultimate objective is to learn a policy that maximises cumulative rewards over time using a process of trial and error coupled with feedback from the environment [68]. It uses deep learning to estimate the Q-function and deal with high-dimensional state and action spaces, which may be difficult for traditional RL algorithms. Another benefit of using DRL is its ability to learn from raw data and make decisions without the need for manual feature engineering. This makes explains its widespread use in game development and control of robots in complex environments. Classical RL relies on hand-designed features to represent the states and the actions that can be taken in the environment. DRL is classified into value-based and policy-based types, but some researchers add a third type; actor-critic based technique.

3.3.1 Value-based RL

Value based RL involves a function that takes in the state and one possible action and outputs the value of taking that action. The value is the total expected reward from a state up to future states. The policy of the algorithm is to check the value of every possible action and choose the one with the highest value. This can be represented using a lookup table called Q-table. The algorithm uses the Bellman equation through Dynamic programming to update the value of actions. In Value-based DRL, neural networks are introduced to

represent the value functions. The value-based DRL obtains the agent's policy by continuously updating the value function [69]

The Deep Q-Network [70] is a combination of using Neural Network with Q-Learning to give Deep Q-Learning. It extends the principles of Q-learning to handle high-dimensional data, 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". The agent's previous experience data is stored in the Experience replay buffer which is sampled at every time step and used for training, updates and optimization. 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 $Q(s, a; \theta_i)$. 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:

$$L_i = \mathbb{E}[(r + \gamma \max_a Q(s', a'; \theta'_i) - Q(s, a; \theta_i))^2] \quad (3.16)$$

If the V-value is optimized rather than the more 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.

3.3.2 Policy-based RL

In this method, the policy is learnt directly without explicitly learning the value function. It is represented by any function approximation technique or, in the case of DRL, by neural networks. This provides more flexibility in selecting actions compared to Value-based RL algorithms and makes it possible to represent policies for continuous action spaces [10].

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

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:

$$\theta_* = \arg \max_{\theta} J(\theta) \quad (3.17)$$

Policy gradient methods perform parameter updates to update the parameter:

$$\theta \leftarrow \theta + \alpha \nabla_{\theta} J(\theta) \quad (3.18)$$

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

3.3.3 Actor Critic RL

Actor-critic methods, which are originally a family of policy-gradient algorithms, have been recently categorised by researchers and practitioners as a separate DRL approach because they combine properties of both value and policy-based RL types [71]. Actor-Critic (AC) Architecture [70], [72] combines the Actor obtained from the policy network of Policy Gradient algorithms and the Critic obtained from the value function network of Value-based RL algorithms. The Actor learns the policy in the environment and finds the optimal actions for given states. The Critic sends feedback to the Actor after estimating the value of states or state-action pairs, thus helping it update the policy for better exploration and exploitation of the environment. The actor network chooses the most suitable action given the current state, while the critic network evaluates the quality or value of the state-action pair chosen by the actor. Actor-critic networks inherently thrive in balancing exploration and exploitation when training the DRL policy. They can easily explore new behaviours towards the navigation goal while using the critic network to refine the policy based on feedback or reward signals. Additionally, they are well suited to handle continuous high-dimensional state and action spaces, which are required in the most advanced navigation algorithms to learn advanced techniques like context-awareness, intention prediction, and trajectory prediction [73], [74].

3.4 Imitation Learning

Imitation Learning (IL), traditionally known in reinforcement learning literature as behavioral cloning, is a technique wherein an agent learns to replicate expert behavior through demonstration. Unlike reinforcement learning, which relies on interaction with the environment and exploration of trial-and-error policies, IL leverages labeled expert trajectories to directly train a policy via supervised learning. This makes IL particularly attractive in navigation scenarios that demand sample efficiency and safety during training.

A common practice in recent work is to use imitation learning as a preparatory phase in deep reinforcement learning pipelines. For instance, a socially compliant but rigid expert policy like ORCA or SFM may be used to generate training data. The agent learns to imitate this behavior, thereby acquiring a stable starting policy that ensures basic safety and social compliance. This pretraining significantly reduces the burden on subsequent reinforcement learning stages, which then focus on refining the policy and handling novel interactions

Despite its sample efficiency and ease of training, IL is not without limitations. It is fundamentally constrained by the quality and diversity of the expert demonstrations. If key situations are missing from the data, the agent may perform poorly when encountering them during deployment. Moreover, since IL lacks a mechanism for learning from its own mistakes, it can be less robust than reinforcement learning in long-term autonomous settings.

To deal with some of the limitations of IL, particularly for SAN, the work in [52] presented a data-driven end-to-end motion planner, wherein the robot learns to navigate according to the preferences of the human user. Expert examples on navigating a specific training

environment are made available for accomplishing this. The objective is to not only duplicate the given demos in a singular situation but also to acquire collision avoidance rules and apply them to novel contexts. In [75], a combined approach known as Reinforced Imitation Learning (R-IL) merges imitation learning and RL into a single framework for target-oriented navigation which doesn't use a map. Initially, the navigation strategy is trained using expert demonstration data. Then, during the reinforcement learning phase, constraint handling is introduced through Constrained Policy Optimization (CPO) [76]. The study demonstrates that this hybrid method can reduce the time needed to reach comparable performance levels with traditional reinforcement learning by up to five times.

3.5 Summary

Chapter 3 presents the preliminaries and necessary theoretical background to develop a methodology for achieving the objectives of this thesis. It begins with an introduction to reinforcement learning, including key components such as Markov Decision Processes (MDPs), value functions, Q-learning, and temporal difference learning. It then discusses deep learning fundamentals, covering the different type of neural network architectures. These foundations are integrated in the discussion of deep reinforcement learning (DRL), which explores value-based, policy-based, and actor-critic methods for sequential decision-making. The chapter concludes with an outline of imitation learning, highlighting its relevance in leveraging expert demonstrations to bootstrap the learning process in complex environments.

CHAPTER 4 : METHODOLOGY

4.1 Introduction

This chapter discusses the application of the preliminaries discussed in Chapter 3 to develop the research methodology for this thesis. We start with formulating the problem as a Partially Observable Markov Decision Process (POMDP) and define its components. The chapter discusses how Deep Reinforcement Learning is used to develop the navigation policy that enables robots to move towards their goals in a socially aware manner. The chapter then discusses modelling of the human-human, human-robot, and robot-robot interactions. It then highlights the learning algorithm, policy formulation, and system architecture. Finally, the implementation details, simulation environment, and metrics used for evaluation are presented.

4.2 Assumptions

The following assumptions are made in this work:

- This work is based on Single agent reinforcement learning framework where an agent is trained and all other entities form part of the environment.
- The robot that the learned policy runs on is called the ego robot, and the rest of the robots in the environment are called the other robots
- The robots are modeled as holonomic, i.e. they can move in any direction instantly without rotation constraints.
- Humans and other robots in the environment follow the ORCA policy

- Information from the environment is obtained in form states and not physical observations from sensors
- There is no explicit communication of navigation intent between ego robot, other robots and humans, they can only observe the states of each other.
- The navigation and scenarios are modelled in Two-Dimensional (2D) plane
- The developed policy is obtained from training and implementation in a simulation environment without deployment on hardware to real world environment

4.3 POMDP Formulation

The following section covers formulation of the problem multiple robots navigating in environments shared with humans while respecting social norms, such as maintaining comfort distances. Many real-world applications only require robot navigation in only two dimensions. Their navigation objective is to determine the most efficient and safe route from the initial position to the target location. 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. Determining the goal point's relative location to the initial point is essential for robot navigation, which may be viewed as a point-to-point movement that includes obstacle avoidance. This may be accomplished by having the goal point's absolute coordinates a priori and then finding the relative position information by subtracting the goal point's current position from its absolute coordinates.

In this work, the problem of multi-robot SAN in 2D Euclidian space is formulated as a Partially Observable Markov Decision Process (POMDP) within an RL framework. The objective is to empower robots to move based on social norms of navigation in a heterogeneous environment of humans, robots and obstacles, taking into account partial observability, environmental uncertainty, and heterogeneous interactions. The navigation issue is fundamentally modelled as a POMDP because of limited access to the objectives and intents of other agents. The proposed DRL framework does not explicitly compute belief states, but rather the neural network policy implicitly learns representations that approximate belief through end-to-end training on observation histories. This facilitates efficient and scalable decision-making without formal belief tracking as used in classical POMDP methods.

The POMDP is defined by the tuple $\langle S, O, A, P, R, \gamma \rangle$ where S is the state space, A the action space, O the observation space, P is the transition probability function and γ is the discount factor. These elements together form the foundation for the ability of the agent to make decisions and learn within the environment and 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.

4.3.1 State Space

In RL literature state space and observation space are sometimes used interchangeably. In this work we have a key distinction, we refer to the state space to include both the observable and unobservable state of all agents. The observation space O refers to the

information observed by the ego robot, including the self-state of the ego robot and the observable states of other agents. Fundamentally, for all agents in the environment, the position, speed, and radius, can be observed by others, however, the ego robots goal position, preferred speed, and orientation are unobservable and only available to the ego robot itself.

The state space represents the environment and the agents (other robots and humans) within it. Each agent has observable and unobservable states. The ego robot is fully observable, while the other robots and humans are partially observable.

The state space S includes the states of the ego robot, humans, and other robots.

Likewise, the observable state comprises of the full state of the ego robot and the observable states of the other robots and humans. The state of the ego robot at time t is denoted as s_t^{er}

$$s_t^{er} = [d_g, v_{pref}, \theta, r, v_x, v_y] \quad (4.1)$$

Where, d_g is the distance to the goal, v_{pref} is the preferred velocity of the robot, θ is the heading angle of the robot, r is the radius of the robot, v_x, v_y are the velocity components of the robot.

The observable state of the i -th human at time t is denoted as $o_t^{h_i}$:

$$o_t^{h_i} = [p_x^{h_i}, p_y^{h_i}, v_x^{h_i}, v_y^{h_i}, r^{h_i}, d_a^{h_i}, r^{h_i} + r] \quad (4.2)$$

Where $p_x^{h_i}, p_y^{h_i}$ are the components of the position of the human, $v_x^{h_i}, v_y^{h_i}$ are velocity components of the human, r^{h_i} is the radius of the human, and $d_a^{h_i}$: Distance between the ego robot and the human.

The observable state of the j -th other robot at time t is denoted as o_t^{orj} :

$$o_t^{orj} = [p_x^{orj}, p_y^{orj}, v_x^{orj}, v_y^{orj}, r^{orj}, d_a^{orj}, r^{orj} + r] \quad (4.3)$$

Where p_x^{orj}, p_y^{orj} are the components of the position of the other robot, v_x^{orj}, v_y^{orj} are velocity components of the other robot, r^{orj} is the radius of the other robot, and d_a^{orj} is the distance between the other robot and the ego robot. Only the agent itself has full knowledge of its unobservable states. In terms of observability, the humans are partially observable, ego robot is fully observable, and the other robots are partially observable. Additionally, the unobservable states are the target position g_x, g_y , preferred velocity v_{pref} , and direction angle θ . For the purpose of constructing the full state of the agent, both the observable and unobservable states are combined. Then, this full state is utilized to construct the joint state, s_t^{jn} which is a combination of the full state of the robot and the observable state of the other agents (human and ego robot).

For the sake of simplicity, researchers introduced a new coordinate system that sets the ego robot at the origin, and they transformed the coordinate system from a global x, y Cartesian to a local polar one. In the original global system, the representation of the ego robot's current and goal positions is absolute, which might lead to coordinate transformations or rotations that do not affect the optimal decisions. For example, the ego 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. With the transformation of the ego robot's current position to the origin, the x -axis becomes directed toward the target in this new agent-centric local system. Next, we transform the target point's representation into relative coordinates relative to where the ego robot is. As a result, the

ego robot's decision-making process becomes streamlined and improved, leading to the removal of the associated ambiguity and redundancy from the global coordinate system. Furthermore, agent-centric parameterization streamlines the representation of the ego robot's state and action selection, reducing computational complexity and enhancing efficiency.

4.3.2 Action Space

The action space encompasses the full range of actions an agent can perform, determined by its motion capabilities and the specific demands of the navigation task. Robot motion is generally categorized as either holonomic or non-holonomic, based on its movement constraints. 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 nonholonomic 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. We assume that the robot's velocity can be immediately reached after receiving the action command. Action space size or range has a significant impact on the flexibility and efficiency of the robot's 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. In this work we use a discrete action space that is two-dimensional $[v_t, \phi_t]$, where v_t represents the velocity at time t , taking 5 discrete values uniformly distributed between 0 and v_{pref} , and ϕ_t denotes the heading angle at time t , taking 16 discrete values uniformly distributed between 0 and 2π . The final combination yields a total of 80 discrete actions.

- Speed command: $v_t \in [0, v_{pref}]$
- Heading direction: $\phi_t \in [0, 2\pi]$

4.3.3 Reward Function

In DRL, a reward function assigns a numerical score (reward or penalty) to the agent's actions based on how desirable those actions are in a given state. The success of every RL framework depends significantly on the formulation of the reward function, as it incentivises the agent to engage in advantageous behaviours through positive incentives and penalises detrimental behaviours with negative rewards. Thus, the agent seeks to optimise its methods to maximise long-term accumulated rewards, resulting in enhanced navigation behaviour.

In social-aware robot navigation, the reward function is often responsible for enabling the robot to move in the direction of the target to accomplish the objective that has been set..

For instance, significant positive rewards may be given upon reaching the goal, motivating the robot to successfully complete the navigation task. Furthermore, collision avoidance can be achieved by offering positive rewards for actions that avoid collisions, and negative rewards for actions that result in collisions or getting close to obstacles. We have considered many factors in the design of the reward function.

Arrival rewards provide a reward value when the robot has achieved its navigation goal, and uncomfortable distance penalties penalize a robot coming too close to a human, i.e., having a distance less than a minimum predefined comfortable distance. Another vital reward mechanism is Time-based penalties, which penalize the robot for exceeding a pre-set limit without arriving at the target. This thesis research did not assign a penalty to it, but rather was given a value of 0. Designing the reward function presents 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.

The reward function below was used. The total reward R_T at time t includes rewards R_g for navigating the robot to its goal, penalties R_c for collision, and discomfort rewards R_d . The expression for the cumulative reward is as below

$$R_T = R_g + R_c + R_d \quad (4.4)$$

$$R_g = \begin{cases} 1, & \text{if } d_g = 0 \\ 0, & \text{Otherwise} \end{cases} \quad (4.5)$$

$$R_c = \begin{cases} -0.25, & \text{if } d_a^i < r + r^i \\ 0, & \text{Otherwise} \end{cases} \quad (4.6)$$

$$R_d = \sum_{j=1}^N f(d_{min}, d_{th}) \quad (4.7)$$

$$f(d_{min}, d_{th}) = \begin{cases} d_{min} - d_{th}, & \text{if } d_{min} < 0.2 \\ 0, & \text{Otherwise} \end{cases} \quad (4.8)$$

where d_g is the ego robot distance to the goal, and d_a^i is the ego robot distance from another agent i at timestep t , then $d_{th} = 0.2$ is the threshold distance, below which pedestrians will feel discomfort, and d_{min} is the assigned value of minimum distance of separation between the ego robot and the j th human at timestep t .

4.3.4 State Transition Model

The state transition function $P = \mathbb{P}(s_{t+1}|s_t, a_t)$ denotes the probability of moving to a subsequent state after the robot performs an action in its current state. It outlines all potential future states along with their associated probabilities of transitioning between them based on the states and the actions selected. In our Multirobot social-aware robot navigation, the state transition model is determined by the environment's dynamics and motion rules, which depend on factors such as the robot's motion capabilities, how obstacles are co-located in the navigation environment, and the positions of other robots. In practice, the state transition model is typically unknown due to the complexity of the environment and its ever-changing nature which makes coming up with an accurate and precise model challenging. However, in simulated environments, while the transition model might initially be unknown, it often becomes clearer as training continues. Furthermore, interactions with the other agents (humans and other robots) increases the uncertainty of state transitions.

Consequently, model-free reinforcement learning approaches like Deep Q-Networks (DQN) are widely adopted. These methods operate without requiring a priori knowledge of the model that governs state transitions. They instead gather experience by interacting with the environment and use this data to optimize policies and estimate value functions. This allows the robot to develop and refine navigation strategies that can adapt to unknown dynamics and complex environments.

4.3.5 Optimal Policy and Value Function

The value function V_s also called action-value function $Q(s, a)$ represents the expected total reward an agent can accumulate by being in a particular state (or taking a specific action in that state) and following a certain policy afterwards. It is used to help the agent in making optimal decisions by having information about how good each action is. Therefore, in this work the goal will be to find the optimal value function $V^*(a_t^{jn})$ which in our case will be:

$$V^*(s_t^{jn}) = \sum_{t'=t}^T \gamma^{t'-t} v_{pref} R(s_{t'}^{jn}, \pi^*(s_{t'}^{jn})) \quad (4.9)$$

where γ is a value between 0 and 1 (but closer to 1) termed the discount factor. In this formulation, v_{pref} which is nominally the agent's target speed, is employed purely to normalize the value function, because without this scaling, slowly moving agents would produce values that are increasingly become small. Once the value function has been optimally normalized, the corresponding optimal policy $\pi^*(s_t^{jn})$ can then be directly extracted from it.

$$\pi^*(s_t^{jn}) = \arg\max_{a_t} R(s_t^{jn}, a_t) + \gamma^{\Delta t \cdot v_{pref}} \int_{s_{t+\Delta t}^{jn}} P(s_t^{jn}, a_t, s_{t+\Delta t}^{jn}) V^*(s_{t+\Delta t}^{jn}) ds_{t+\Delta t}^{jn} \quad (4.10)$$

4.4 Interaction Modelling

This subsection discusses the main part of the objectives which describes the modelling of the interactions that happen between all agents in the environment (ego robot, humans, other robots). The five distinct type of interactions are: Human-Human, Human-Ego Robot, Ego Robot-Other Robot, Human-Other Robot, and Other Robot-Other Robot interactions, and are shown in Figure 7, and their distinct relationships are shown in an agent interaction matrix in Table 2.

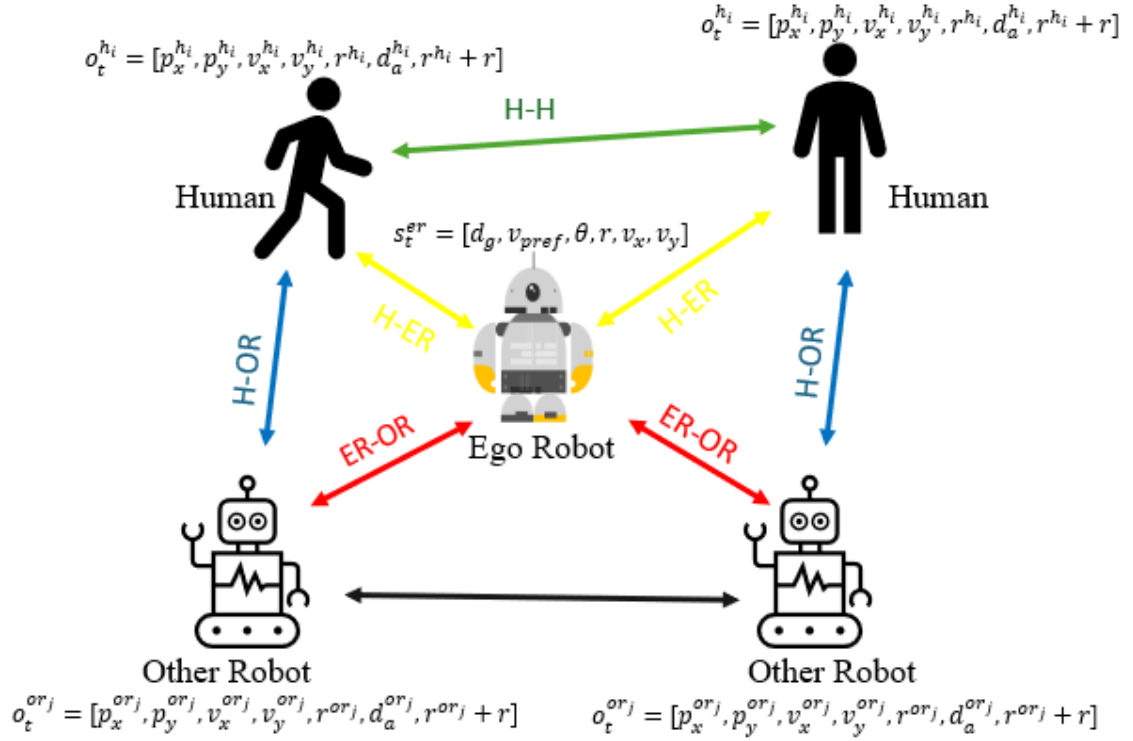


Figure 7. Categories of agents in this work and their interaction relationship

4.4.1 Human-Human Interaction

Human agents are modeled using the ORCA policy, which enables reciprocal collision avoidance based on local sensing and velocity adjustments. Each human responds reactively to the motion of nearby humans, maintaining safe distances and navigating toward individual goals while avoiding collisions. This interaction captures the decentralized and socially compliant nature of human navigation in shared spaces.

4.4.2 Human-Ego Robot Interaction

The ego robot employs a reinforcement learning policy trained to exhibit socially aware behavior. It perceives and interprets the state of surrounding humans, including position, velocity, and relative orientation, to make navigation decisions that align with human social norms. The policy accounts for proxemic constraints and collision risk, allowing the robot to adjust its trajectory appropriately in the presence of humans.

4.4.3 Ego Robot-Other Robot Interaction

While the ego robot is the only agent trained via reinforcement learning, other robots in the environment are modeled using the ORCA policy. These other robots are treated as dynamic obstacles by the ego robot, so it does not enforce avoiding entering the personal space or discomfort zone of the other robots, rather it only does collision avoidance.

4.4.4 Human-Other Robot Interaction

Interactions between humans and other robots are governed by the same reciprocal collision avoidance principles as human–human interactions. There is no social awareness in this interaction, only purely collision avoidance. However, other robots are designed to

maintain slightly larger safety margins when navigating around humans. This ensures that their presence in the environment reflects behaviour of robots.

4.4.5 Other Robot-Other Robot Interaction

Interactions among the other robots follows standard ORCA dynamics. Each robot independently computes its velocity based on the positions and velocities of its neighbors, including other robots. These interactions are symmetric and reciprocal, enabling decentralized collision avoidance without explicit communication. Although these robots are not trained, their behavior is sufficient to support socially aware motion of other agents in the environment.

Table 2. Agent Interaction Behaviour Matrix

Agent	In presence of Human	In presence of Ego Robot	In presence of Other Robot
Human	Reciprocal collision avoidance	Reciprocal collision avoidance	Reciprocal collision avoidance
Ego Robot	Implements learned DRL policy with social awareness	-	Reciprocal collision avoidance (no social awareness)
Other Robot	Reciprocal collision avoidance with larger safety margin	Reciprocal collision avoidance (no social awareness)	Reciprocal collision avoidance

4.5 Policy Formulation and Learning Algorithm

In this single-agent DRL framework for SAN, the proposed learning architecture integrates both demonstration-based imitation learning and value-based reinforcement learning. The ego robot is the only agent trained via reinforcement learning, while surrounding agents (humans and other robots) navigate based on the ORCA policy. The learning process starts with initializing a value network and an experience replay buffer using demonstration data

generated through Imitation Learning. It uses ORCA-based trajectories to jump start the training and provides a stable foundation for subsequent reinforcement learning.

In every time step, the ego robot gets state observations of all nearby agents as well as its own self-state. These inputs, represented in an agent-centric coordinate system aligned with the ego robot's goal, include relative positions, velocities, radii, and orientations. For each surrounding agent, a local occupancy map is constructed to capture the spatiotemporal dynamics of its immediate neighbors. This map is concatenated with both the agent's state and the ego robot's state and processed by a neural network to generate an interaction embedding. These embeddings, which encode the individual influence of each agent on the ego robot, are then aggregated using an attention-based pooling mechanism. The mean of all interaction embeddings is computed to represent the global crowd context, which is then combined with each individual embedding and passed through a Feedforward neural network to produce attention scores. These values reflect the relative importance of each agent, and are used to calculate a weighted aggregation of the interaction features, resulting in a final crowd representation. This is highlighted in Figure 4.1 as the system algorithm.

The joint state, comprising the ego robot's state and the attention-weighted representation of all agents, is fed into another multilayer perceptron known as the value network. This network computes an expected return for every action option, then picks and carries out the one with the largest value estimate. The system then advances to the subsequent state, where a reward is generated according to the specified reward function. The replay buffer is where the experience tuple that was generated as a result is saved. The tuple comprises the current state, the action selected, the reward obtained, and the resulting next state. Through the process of temporal-difference learning, batches of experiences are extracted

from this buffer and used to update the value network while training is going on. This process continues iteratively, allowing the ego robot to progressively refine its navigation policy to exhibit the desired socially-aware behavior while navigating efficiently in the presence of both humans and other robots. This process is shown in Figure 8.

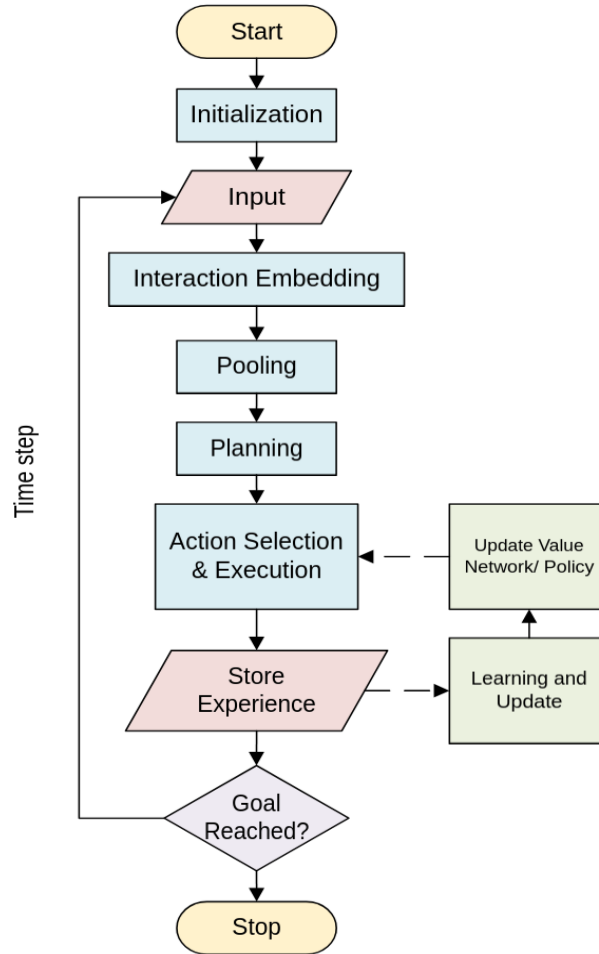


Figure 8. Algorithm of learning framework

4.6 Imitation Learning-Based Pretraining

This subsection discusses how the developed framework uses Imitation Learning to initialize training and increase sample efficiency for the Reinforcement Learning stage. It

is a learning framework in which policies are obtained directly from demonstrations by ‘experts’ which are provided in form of sample trajectories. Its main approaches are Inverse Reinforcement Learning (IRL) and Behaviour Cloning. We utilize the approach of Behaviour Cloning which performs the process of imitation learning in a supervised manner using the data collected. It does this through direct learning the policy of the expert, which may be a human or an agent. The policy is a rule that maps the states in the recorded trajectories to their corresponding labels.

4.6.1 Initialization via Imitation Learning

This work uses an Imitation Learning (IL) phase as a gateway to Deep Reinforcement Learning (DRL) to enable stable policy learning right from the beginning of the training of the framework. Using expert demonstrations produced by the Optimal Reciprocal Collision Avoidance (ORCA) policy, the IL phase thus initializes the network. Imitation Learning initializes the learning process and reduces the burden of exploration in subsequent reinforcement learning (RL) phases. It also improves sample efficiency in the DRL phase.

For IL, Let $\mathcal{D} = \{(s_i^{jn}, a_i^*)\}_{i=1}^N$ define the dataset of state-action pairs obtained from ORCA-generated trajectories, where $s_i \in \mathbb{R}^d$ represents the ego robot’s joint state with the other environment agents, and $a_i^* \in A$ denotes the action of the expert at state s_i . Thus the objective of IL is to learn a policy $\pi_\theta : \mathbb{R}^d \rightarrow A$ parameterized by θ , that minimizes the supervised loss:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_{CE}(\pi_\theta(s_i), a_i^*) \quad (4.11)$$

Where $\mathcal{L}_{CE}$ is the cross-entropy loss. In practice, we use a neural network to approximate π_{θ} , sharing the architecture with the value-based DRL model that follows.

We simulated 2000 navigation episodes for 50 epochs with a learning rate of 0.01 to generate the state-action pairs representing the behavior of the ego robot navigating within the environment. The generated trajectories served as data for the supervised learning, with the training objective being reduction of cross-entropy loss between predicted actions obtained through training and expert actions across all states. By use of this method, the ego robot policy learnt to mimic the conservative and reciprocal actions displayed by the ORCA expert.

4.6.2 The Optimal Reciprocal Collision Avoidance (ORCA) Policy

The Optimal Reciprocal Collision Avoidance (ORCA) policy is a velocity-based, decentralized algorithm that is specifically designed for real-time multi-agent collision avoidance. By modeling the behaviors of neighboring agents under the assumption of reciprocal cooperation, it allows each agent to independently select collision-free velocities. The policy assumes that all agents are rational and will comply with the same collision avoidance rules.

ORCA determines, for each agent, the allowable velocity options by taking into account its current velocity, the forecasted paths of surrounding agents, and the spatial constraints formed by their relative positions and speeds. The agent picks, from the collision-free velocities, the one that most closely matches its preferred speed and direction toward the goal.

In ORCA, each agent is modelled as an open disc with radius r_A , at a defined position $p_A \in \mathbb{R}^2$, velocity v_A and a preferred velocity v_A^{pref} which is directed toward its goal. The fundamental concept of ORCA is to create, for every agent B , a half-plane in velocity space that represents the set of velocities agent A can select to avoid collisions with agent B over a predetermined time horizon τ . These definitions and constraints are adapted from the concept of Velocity Obstacles (VO).

The Velocity Obstacle $VO_A^\tau(B)$, as shown in Figure 9, defines the set of relative velocities between agents A and B that would result in a collision within time τ . It is formally given by:

$$VO_A^\tau(B) = \{v | \exists t \in [0, \tau]: t(v - v_B) \in D(p_B - p_A + r_B)\} \quad (4.12)$$

Where $D(p, r) = \{q | \|q - p\| < r\}$ is an open disc centered at p with radius r . As shown in Fig f $VO_A^\tau(B)$ is diagrammatically represented in the form of its geometry as a cone that is truncated at its peak which is at the origin and the sides tangent to the circle defined by the sum of the radii of A and B with origin at $p_B - p_A$.

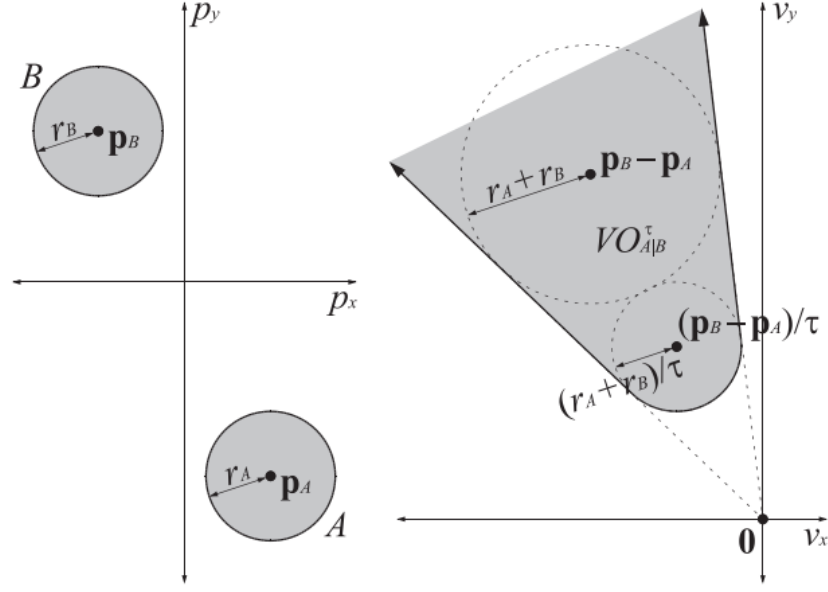


Figure 9. Robots A and B shown as disks (left) with their velocity obstacle (right) [22]

To avoid collisions, agent A must choose a velocity $v \notin VO_A^i(B)$. In ORCA, this requirement is softened to a half-plane constraint which gives each agent half the responsibility of avoiding the collision. Let u be the minimum change in relative velocity required to avoid a collision:

$$u = \left(\arg \min_{v \in \partial VO_A^i(B)} \|v - (v_A^{opt} - v_B^{opt})\| \right) - (v_A^{opt} - v_B^{opt}) \quad (4.13)$$

As shown in Figure 9, the set of permitted velocities for agent A with respect to agent B is then given by the half-plane:

$$ORCA_A^i(B) = \left\{ v \left| \left(v - \left(v_A^{opt} + \frac{1}{2}u \right) \right) \cdot n \geq 0 \right. \right\} \quad (4.14)$$

Where n is the outward-facing normal vector to the boundary of the velocity obstacle at the point $(v_A^{opt} - v_B^{opt}) + u$. The half-plane, shown in Fig, and bounded by the line

perpendicular to u via the point $v_A^{opt} + \frac{1}{2}u$ defines the range of allowed velocities for A to prevent collisions with B in an optimal and reciprocal manner.

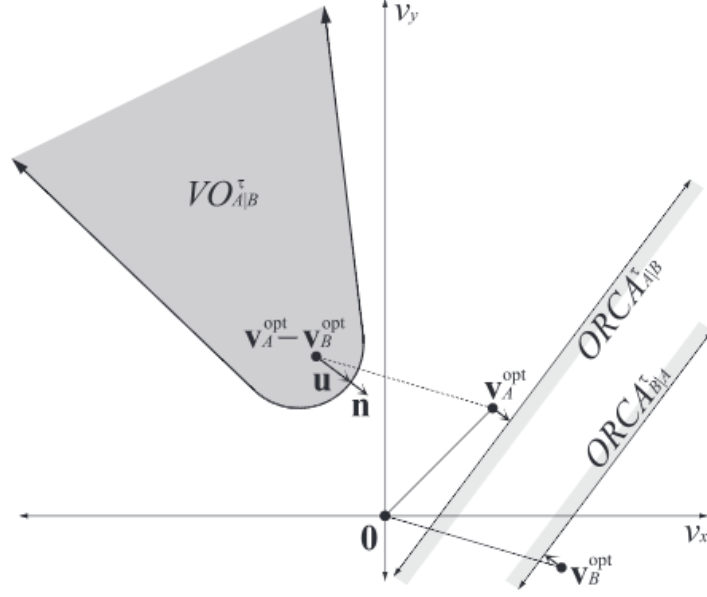


Figure 10 ORCA half-plane constraint in velocity space [22]

To remain collision-free with respect to all neighbouring agents, a feasible velocity space for agent A is computed as the intersection of all permitted half-planes:

$$ORCA_A^\tau = D(0, v_A^{max}) \cap \bigcap_{B \neq A} ORCA_A^\tau(B) \quad (4.15)$$

The optimal new velocity v_A^{new} is then selected as the one closest to the preferred velocity v_A^{pref} within the feasible region:

$$v_A^{new} = \arg \min_{v \in ORCA_A^\tau} \|v - v_A^{pref}\| \quad (4.16)$$

This optimization problem is solved as a linear program over convex constraints enabling efficient execution even in dense scenarios. We illustrate the application of ORCA for eight

robots in Figure 11. Their current velocities are shown using arrows on the left. And in the right we can see the half-planes of allowed velocities for agent A which is induced by each of the other agents. The shaded region represents the $ORCA_A^\tau$ and comprises of the permissible velocities for A in relation to all other agents.

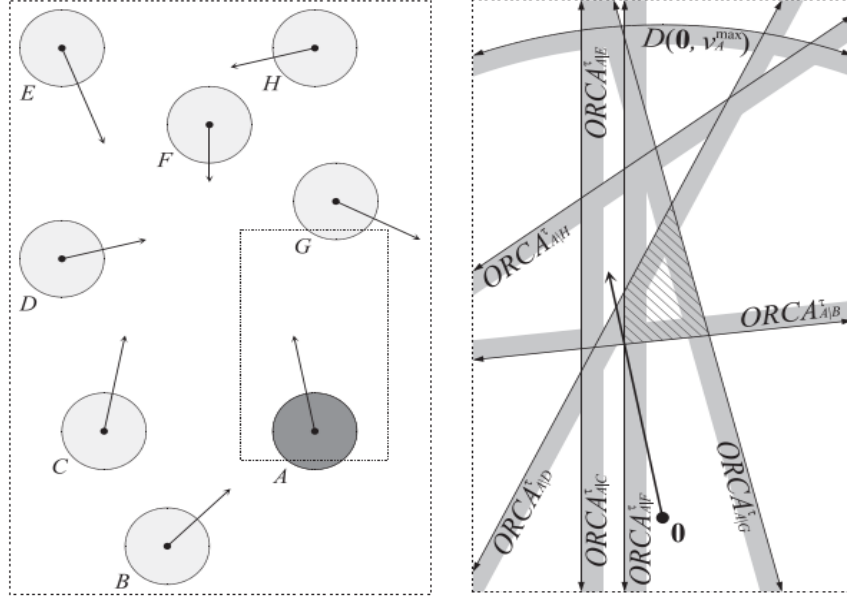


Figure 11. Permissible velocity region formed by ORCA half-planes of 8 agents [22]

4.7 System Architecture

The proposed framework contains three modules as adapted from [45]. They are:

Interaction, Pooling, and Planning Modules. The overall architecture of the system with all three modules combined is shown in Figure 12.

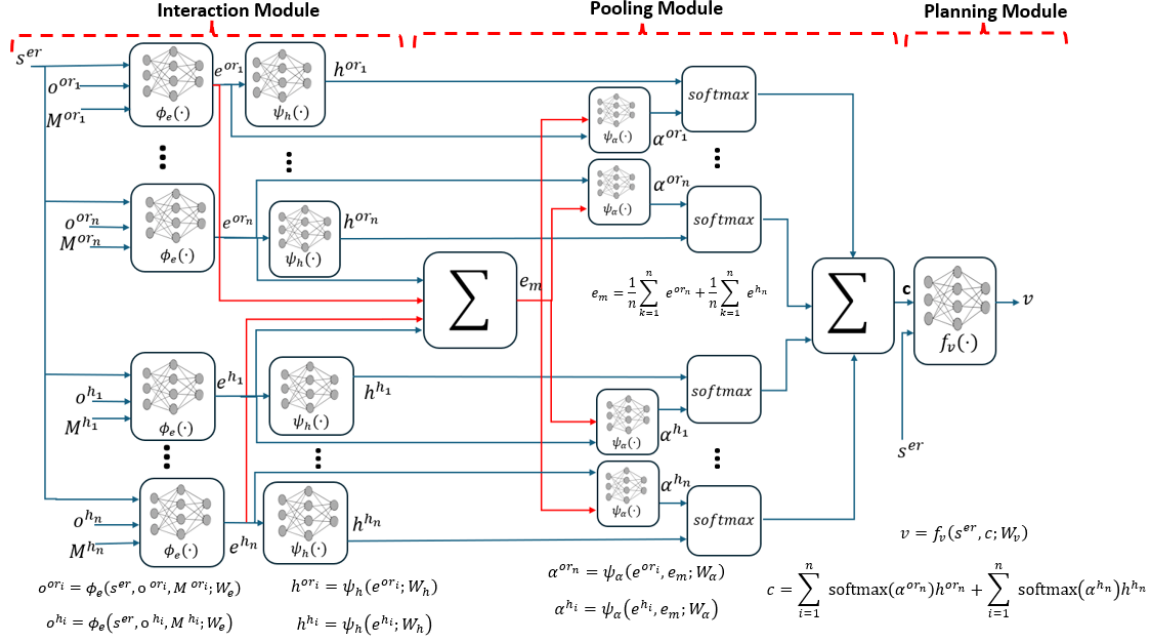


Figure 12. Overall learning architecture of the ego robot in the navigation framework

4.7.1 Interaction Module

It is assumed that all agents (human, ego-robot, other robot) have an impact on each other in the environment. It would be too computationally expensive and almost impossible to model all these interactions as it would lead to an $O(N^2)$ complexity. Local maps will be employed as a representation for the Human-Human, Human-Other Robot, and Other robot-Other Robot interactions, and a pairwise interaction module will be used to simulate both Human-Ego Robot and Ego Robot-Other Robot interactions. To capture the presence and velocities of neighbours in a neighbourhood of size L , Figure 13 shows a $L \times L \times 3$ map tensor M^{*i} centred at each agent (with $*$ represents or for other robot and h for human) i . This map is referred to as the local map.

$$M^{*i}(a, b, :) = \sum_{j \in \mathcal{N}_i} \delta_{ab}[x_j - x_i, y_j - y_i] \xi'_j \quad (4.17)$$

where $\xi'_j = (v_{xj}, v_{yj}, 1)$ is a local state vector for agent j , $\delta_{mn}[x_j - x_i, y_j - y_i]$ is an indicator function.

Using an MLP, we encode the state of the ego robot, humans, other robots, and the map tensor M^{ori} or M^{hi} into a fixed-length vector e^{ori} for the interaction of an other robot or e^{hi} for interaction of a human respectively.

$$e^{ori} = \phi_e(s^{er}, o^{ori}, M^{ori}; W_e) \quad (4.18)$$

$$e^{hi} = \phi_e(s^{er}, o^{hi}, M^{hi}; W_e) \quad (4.19)$$

Where W_e denotes the embedding weights and $\phi_e(\cdot)$ is a function with ReLU activations for embedding the states, weights, and map tensor. The interaction feature between the ego robot and a human or other robot is obtained by feeding the embedding vectors e^{ori} and e^{hi} to subsequent MLPs:

$$h^{ori} = \psi_h(e^{ori}; W_h) \quad (4.20)$$

$$h^{hi} = \psi_h(e^{hi}; W_h) \quad (4.21)$$

where $\psi_h(\cdot)$ is another layer with RectiLinear Unit non-linearity and W_h denotes the weight of this MLP.

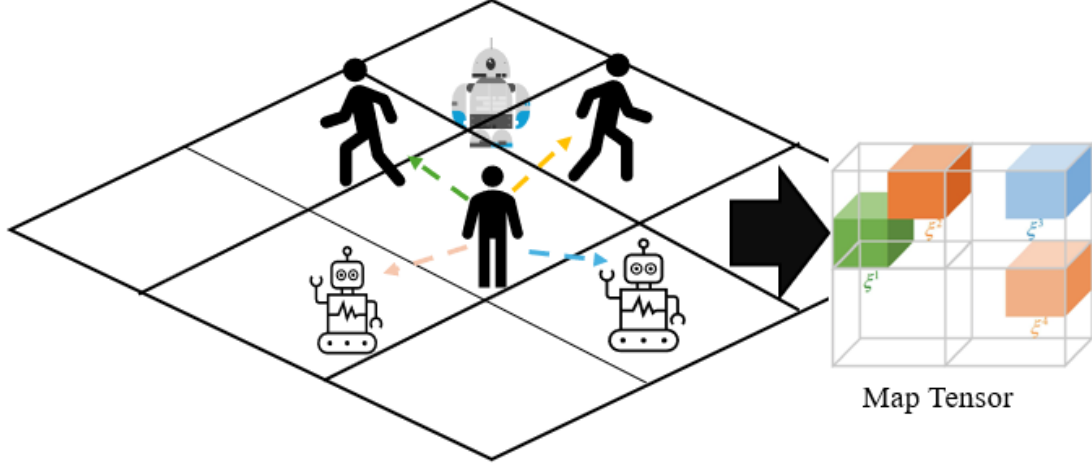


Figure 13. Local Map creation

4.7.2 Pooling Module

A self-attention mechanism is proposed for this module based on [45]. This is because the nearest other robot or human is not necessarily the most important in terms of navigation, especially if the direction and speed of the agent is not in conflict with that of the robot. Therefore, an attentive module used for pooling while incorporating social aspects is proposed to model the importance of each agent in relation to other agents in the environment, and the impact of each agent collectively on all other agents.

The interaction vector is then operated on to obtain an attention score α^{hi} and α^{ori} for a human and robot respectively follows:

$$e_m = \frac{1}{n} \sum_{k=1}^n e^{or_n} + \frac{1}{n} \sum_{k=1}^n e^{h_n} \quad (4.22)$$

$$\alpha^{or_n} = \psi_\alpha(e^{or_i}, e_m; W_\alpha) \quad (4.23)$$

$$\alpha^{h_i} = \psi_\alpha(e^{h_i}, e_m; W_\alpha) \quad (4.24)$$

Where W_α represents the weights, $\psi_\alpha(\cdot)$ is a Multi-Layer Perceptron with Rectified Linear Unit activations, and e_m is a vector with a bounded length that results from taking the mean after pooling all vectors of interactions.

In the final stage of crowd representation, we employ the previously obtained pairwise interaction vectors and their accompanying attention scores to obtain a linear weighted-sum of all the pairs which makes up the crowd representation:

$$c = \sum_{i=1}^n \text{softmax}(\alpha^{or_n}) h^{or_n} + \sum_{i=1}^n \text{softmax}(\alpha^{h_n}) h^{h_n} \quad (4.25)$$

4.7.3 Planning Module

This is the module that determines the value of the network through obtaining the value of the planning state v . The basis for this is from the previously established model of the setting and crowd dynamics. Hence v is:

$$v = f_v(s^{er}, c; W_v) \quad (4.26)$$

where $f_v(\cdot)$ is a Multilayer Perceptron with ReLU activations, the weights are denoted by W_v .

We compute the value for the pairwise attention for each individual based on the mean of the vectors and the individual embedding vector using a multi-layer perceptron. A weighted total of the pairwise interactions constitutes the final joint representation.

4.7.4 Implementation Details

The Details for the implementation of this work including the training parameters, configuration parameters, and environment parameters are discussed in this section as well as in Table 3.1. The policy was implemented using PyTorch library [77] of Python in OpenAI Gym[78] and trained using Adam optimizer [62]. During the training stage, DRL is utilized to train the robot’s navigation policy. the robot interacts with the simulation environment, collecting sample data, and continuously updating value functions, policy networks, or other learning parameters. The robot gradually learns the optimal strategies for different actions in various states to maximize long-term cumulative rewards. While in 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. 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

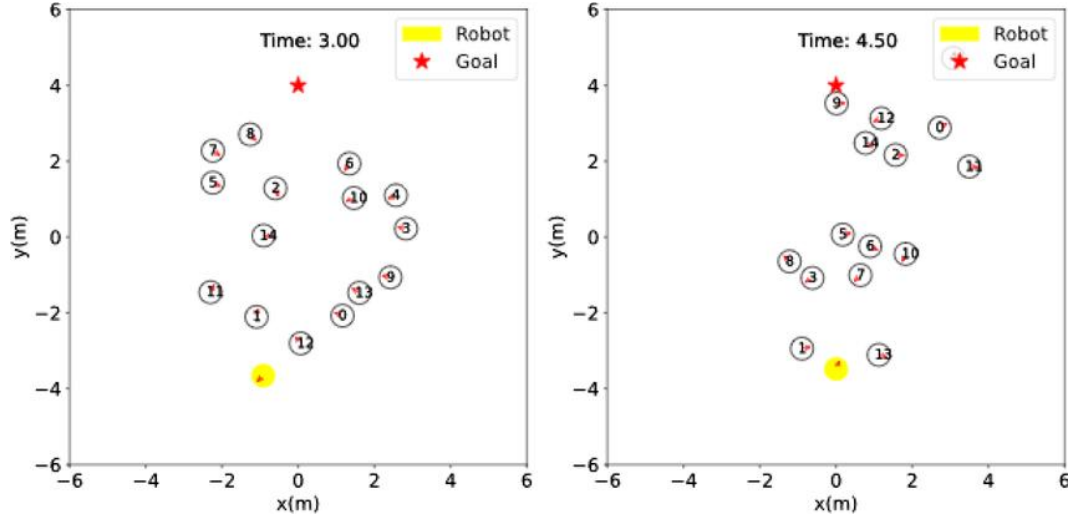
Table 3 Training and Configuration Parameters

Parameter	Value
Preferred velocity	1.0 m/s
Radius of agent	0.3m
Discomfort distance	0.2m
Hidden units of $\phi_e(.)$	150,100
Hidden units of $\psi_h(.)$	100,50
Hidden units of $\psi_\alpha(.)$	100,100
Hidden units of $f_v(.)$	150,100,100
IL training episodes	2000
IL policy	ORCA
IL epochs	50
IL learning rate	0.01
RL learning rate	0.001
Discount factor	0.9
Training batch size	100
RL training episodes	6000
Exploration rate in first 4000 episodes	0.5 to 0.1
Training time	17 hours on AMD Ryzen CPU

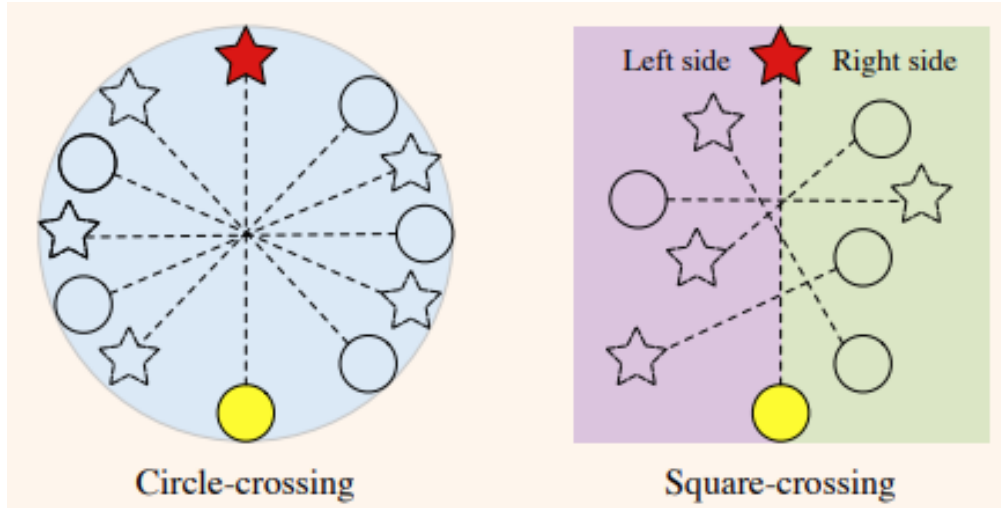
4.8 Simulation Environment

Using a simulation platform provides a safe virtual environment, allowing researchers to conduct experiments without real risks. 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.

The simulation is done in OpenAI Gym [78]. OpenAI Gym is an open-source software tool for reinforcement learning research. It is a single-agent reinforcement learning environment. The humans' behaviors in the environment usually follow the ORCA policy, with parameters randomly drawn from a Gaussian distribution to ensure an appropriate variety of behaviors. The circle-crossing scenarios and square-crossing scenarios are two common simulation scenarios for robot crowd navigation, shown in Figure 4.6. 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 setting, all the agents in the environment are initialized from points along the circumference of the circle, and their goal positions are assigned to be also on the circumference of the circle, but at opposite ends of their starting positions. This type of setting is more common for situations where agents interacting in an environment need to converge to a certain point to carry out a task before heading to their destinations, or situations that resemble a roundabout. In the square crossing setting, all the agents, apart from the ego robot are randomly initialized at either side of the environment, and then their goal position is similarly randomly selected at the opposite side of the start position. This scenario is more reflective of real-world scenarios. Small random offsets are applied to the x and y coordinates of both the start and target locations to enhance the RL algorithm's robustness and capability of generalization. Additionally, obstacles are placed in the simulation environment in some scenarios in our work, and the robots and humans have to plan their paths around them.



(a)



(b)

Figure 14. Circle and square crossings in Gym environment (a), and their setup (b) [79]

4.9 Evaluation Metrics

We quantitatively evaluate the Multirobot SAN scheme using some commonly used evaluation metrics from literature [15] and [80].

The selected metrics are:

1. Success rate: This represents the percentage of successful cases in which the robot arrives at the designated target location within the allotted time frame without collisions.
2. Collision rate: This represents the frequency at which the robot experiences collisions with the other agents in the environment.
3. Timeout rate: This is the frequency of cases in which the ego robot does not reach the target and does not collide with other agents within the maximum permitted time.
4. Navigation time: the average time for the successful navigation cases.
5. Discomfort numbers/frequency: refers to as the frequency in time percentage where robot is within the discomfort region of humans i.e. The total number of times discomfort cases occur.

4.10 Summary

This chapter presented the proposed methodology for developing a multirobot socially aware navigation framework for a dynamic multi-agent environment. The chapter begins by stating the key assumptions and formally defines the problem as a Partially Observable Markov Decision Process (POMDP), with explicit descriptions of the state, observation, and action spaces, as well as the reward function and state transition dynamics. The formulation of the optimal policy and value function is followed by a detailed modeling of the various agent interactions. The chapter then presents the learning algorithm and policy architecture adopted, including the use of attention-based interaction modeling, pooling mechanisms, and planning modules. The system architecture is described in terms of its modular components, followed by implementation details and the design of the simulation

environment. Finally, the chapter concludes with the evaluation metrics used to assess the performance and social compliance of the trained navigation policy.

CHAPTER 5 : RESULTS

5.1 Introduction

This section presents a systematic analysis of the results obtained after implementing the methodology of this work. We obtain and verify multirobot socially aware navigation, then present a discussion and analysis highlighting the qualitative and quantitative results with the aim of verifying the success of this approach.

The evaluation is structured to reflect real-world scenarios and considerations where social norms, environment awareness, and inter-agent interactions are important and considered. The trained policy is tested in settings involving humans, static agents, and other mobile robots, reflecting the diversity and complexity of crowd configurations likely to be encountered in practical deployments. The results are presented in form of snapshots of the simulation video, and trajectory plots from start to finish. Results are also compared quantitatively with other approaches from the surveyed literature. Finally, we highlight the successes and limitations across representative navigation episodes.

5.2 Simulation Configuration and Scenario Design

The evaluation framework for the proposed multi-robot system was designed to replicate a broad range of possible real-world crowd configurations. These scenarios were devised to test the system’s ability to generalize under complex social interactions while maintaining navigational safety and goal-oriented behavior. All experiments were carried out in a discrete 2D simulation space. The workspace consists of a 6×6 -meter bounded square grid, which is typical of an open-plan pedestrian environment. To challenge the policy in tighter conditions, and test real world

conditions, a second configuration resembling a corridor scenario was introduced, defined by a rectangular area measuring 12 meters in length and 4 meters in width. Additionally, a static obstacle scenario was defined to assess the navigation policy's performance in the presence of fixed obstacles. These obstacles required both human and robot agents to react to them to go to their goals.

5.3 Multiple Scenarios Simulation

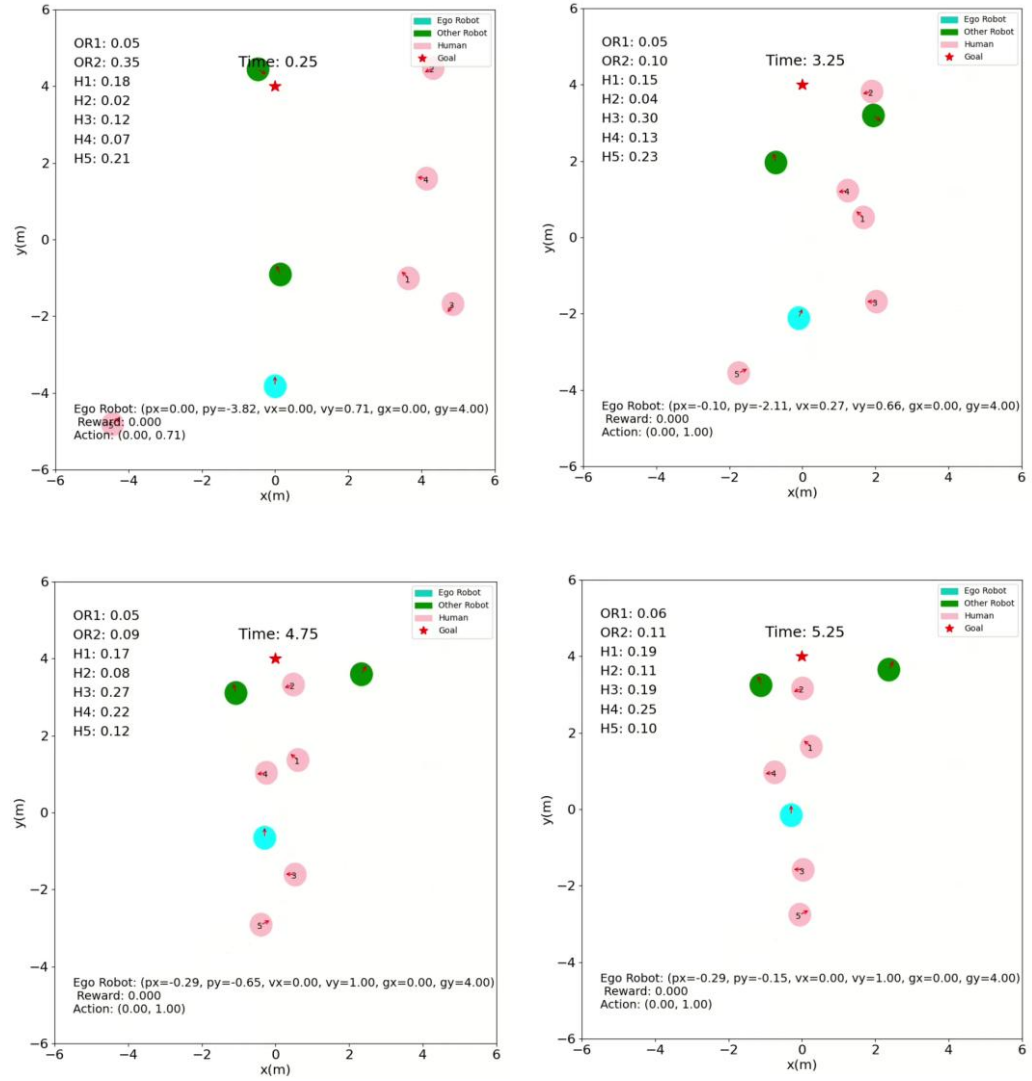
To test the effectiveness of the developed algorithm and its generalization capabilities we have tested it in multiple simulation scenarios which include a varying number of humans and other robots, an environment with static obstacles, and a corridor passage scenario.

5.3.1 Open Space Scenario

The open space scenario involves an open environment with no obstacles with the only limit being the environment is bounded to a 6m by 6m square space. This scenario involves the ego robot and a combination of humans and other robots each with their goals. The ego robot is initialized from position (0, -4), and the target is set at (0,4) which the ego robot must navigate to in a socially aware manner in the presence of the humans and other robots. For the cases in this scenario, we implement a square crossing scenario, where all the agents, apart from the ego robot are randomly initialized either in the left or right halves of the environment, and their goal positions similarly randomly selected at the opposite side of the start position. This scenario is more reflective of real-world scenarios where a robot has to move with humans attempting to cross its path.

We test this scenario across a diverse range of human and other robots setups to show the generalization ability of the developed framework. These setups include:

5.3.1.1 5 Humans and 2 Other Robots



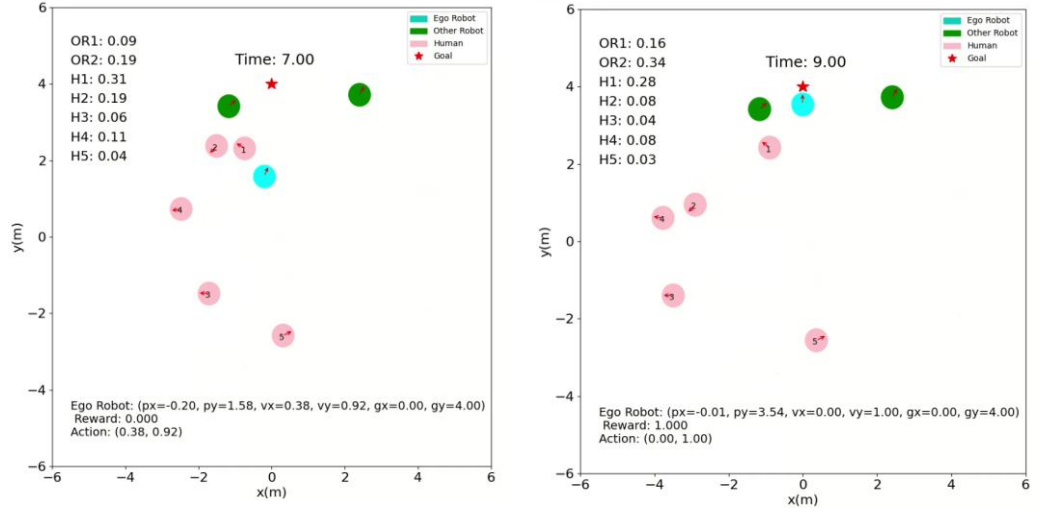


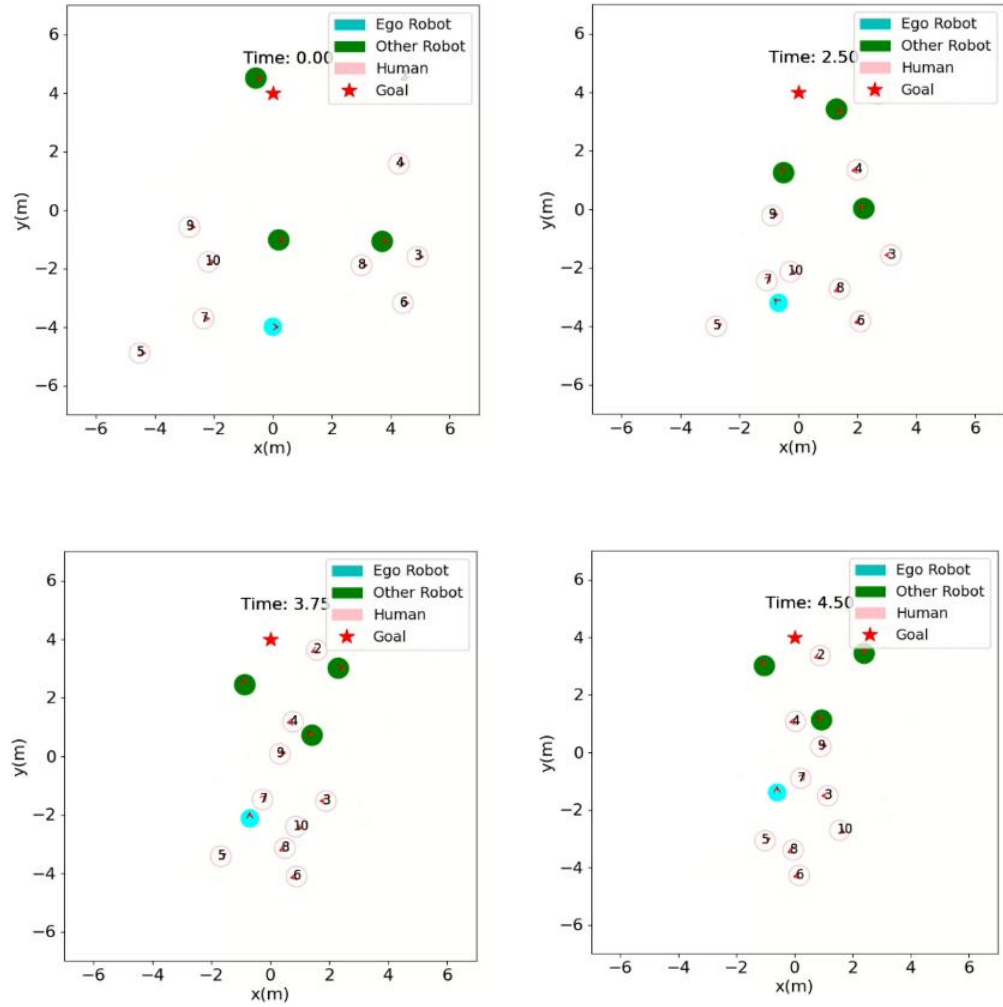
Figure 15. Scenario with 5 humans and 2 other robots at time: 0s, 2s, 4s, 5s, 6s, and 7s

In the scenario above (Figure 15) involving 5 humans and 2 other robots, the ego robot navigates in a relatively sparse environment. The ego robot is initialized with its own self state vector which encodes its position, preferred velocity, and heading angle. It perceives the 7 surrounding agents, each with its relative position and velocity inputs to the state vector. This forms part of the information that is passed to the interaction module to start the process. The action that is obtained and selected as the outcome of the algorithm is a velocity and heading command (e.g. $[0.8\text{m/s}, \pi]$) executed in the environment. A scalar reward value is obtained based on the reward function and the next state is observed after the action has been taken, and the loop continues.

From the onset ($T= 0.00\text{s}$), the ego robot identifies a feasible trajectory that avoids early collisions with humans or other robots. As the simulation progresses ($T= 2\text{s}$ to $T=4\text{s}$) the ego robot venture more into the crowded area, and slows down and allows human 4 and human 1 to pass at $T=5\text{s}$, and $T=6\text{s}$ respectively. The robot maintains a safe space and utilizes the attention of the agents to weigh the priority of each agent. In the final frame (T

= 7s), the robot completes its path to the goal by efficiently passing between two other robots while preserving smooth and collision-free navigation.

5.3.1.2 10 Humans and 3 Other Robots



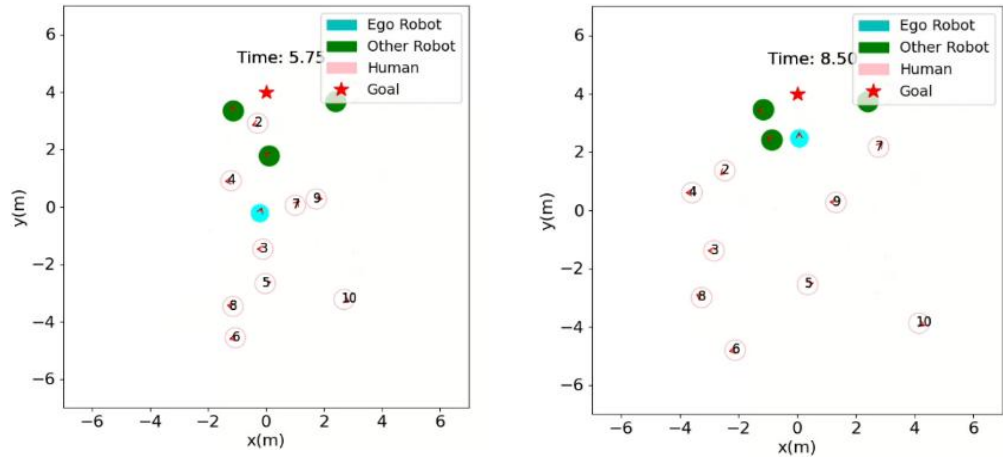


Figure 16. 10 humans and 3 other robots scenario at: 0s, 2.5s, 3.75s, 4.5s, 5.75s, 8.5s

Figure 16 shows the scenario of 10 humans with 3 other robots at the timesteps 0s, 2.5s, 3.75s, 4.5s, 5.75s, 8.5s of the simulation. This scenario is a more densely populated and complex environment. The density of both humans and other robots imposes tighter constraints on the available free space. As the ego robot begins to advance ($T = 2.50s$), it dynamically adjusts its trajectory, and by $T = 3.75s$ to $T = 5.75s$, the robot can be seen socially avoiding humans 3, 4, and 8, as well as responding to the trajectory of a robot approaching it head-on. Between $T=3.75s$ to $T=8.50s$, the robot has moved in between agents to reach toward its goal. Figure 5.3 shows

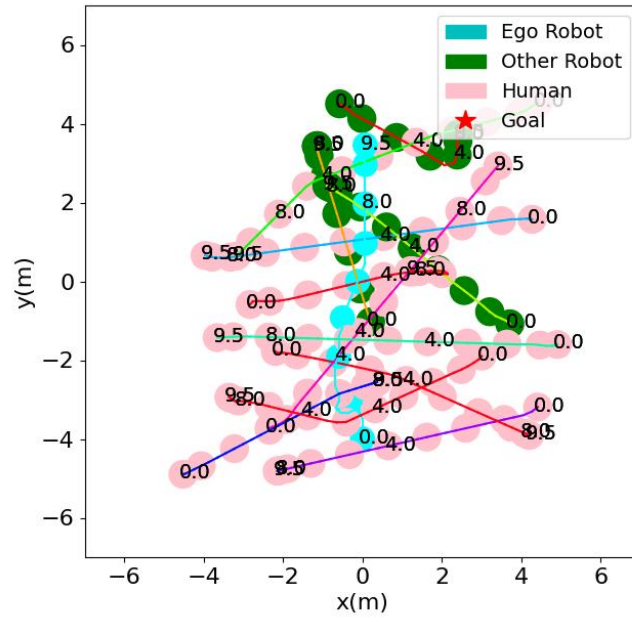
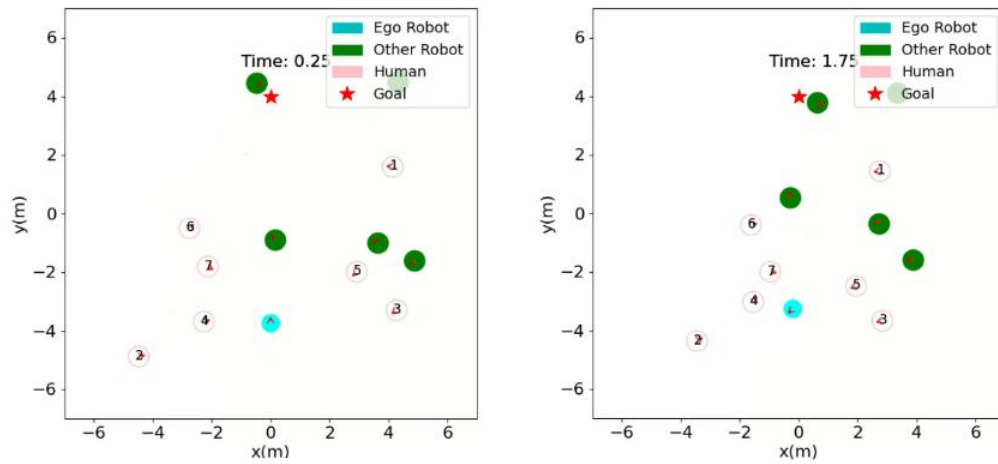


Figure 17. 10 Humans and 3 Other Robots trajectory plot

Figure 17 shows the spatial dynamics and temporal evolution of all agents throughout this navigation scenario of 10 humans and 3 other robots.

5.3.1.3 7 Humans and 5 Other Robots



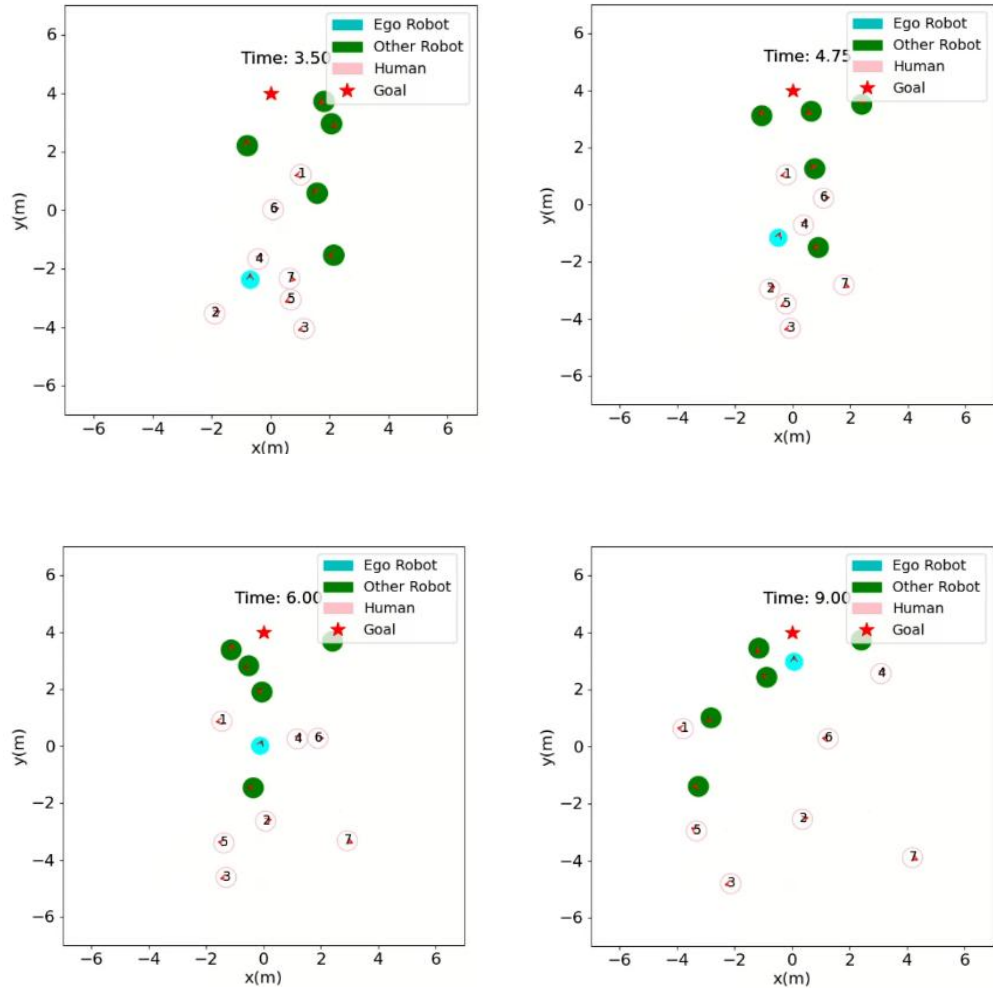


Figure 18. 7 humans and 5 other robots scenario at 0.25s, 1.75s, 3.5s, 4.75s, 6s, and 9s

As can be seen in Figure 18 the ego robot demonstrates effective planning around the crowd. Rather than taking a direct but congested route to the goal, it initiates a curved trajectory (as seen between 0.25s and 3.5s), thereby minimizing collision risks and avoiding socially inappropriate proximity to surrounding agents. A notable aspect of the navigation strategy is the robot's ability to identify and exploit low-density pathways as they emerge. For instance, at 4.75s, the robot advances through a channel formed between two other robots and three human agents without abrupt movements or hesitation. This

shows the importance of the attention-based mechanism which prioritizes most important interactions, while filtering out less relevant influences.

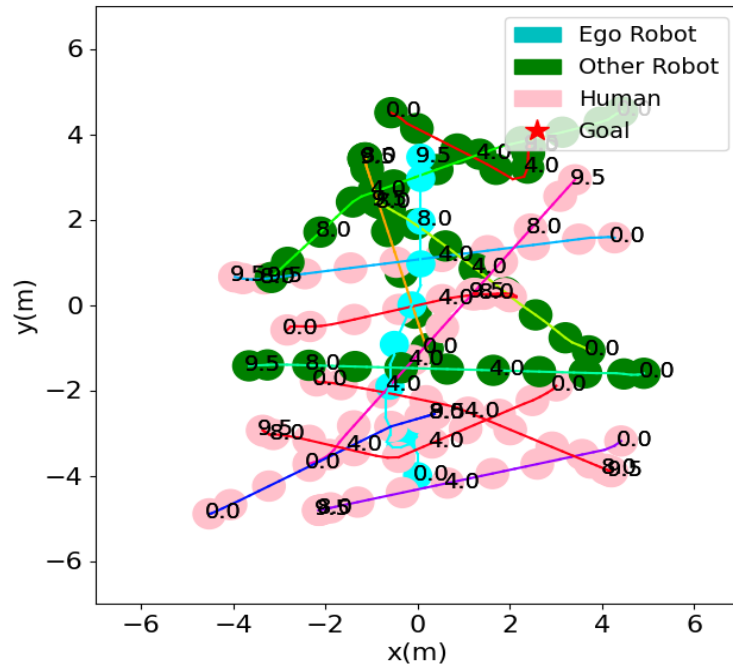


Figure 19.7 Humans and 5 Other Robots trajectory plot

Figure 19 shows the trajectory plot of the navigation scenario of 5 Humans and 5 Other Robots with their spatial dynamics and temporal evolution over time as shown by the timestamps.

5.3.2 Corridor Scenario

Here we have a simple corridor scenario with multiple humans and other robots as shown in Figure 20. It is a narrow corridor passage populated with seven humans and two other robots, designed to test the ego robot's ability to navigate safely and efficiently in a spatially constrained and socially complex setting. The corridor configuration restricts free movement sideways and forces all agents to interact within a confined vertical passage.

Walls are defined on both sides. At $T= 0.75s$ the ego robot is centrally positioned in the corridor among humans and other robots, with its goal situated further ahead along the y -axis. By $T=3.25s$, the ego robot has reached mid-corridor and it can be noticed it yielded way to human 1 despite the presence of human agents 1 and prioritized it above the other robot closest to it. This behavior indicates the attention mechanism's effectiveness in assigning higher weights to nearby obstructive agents.

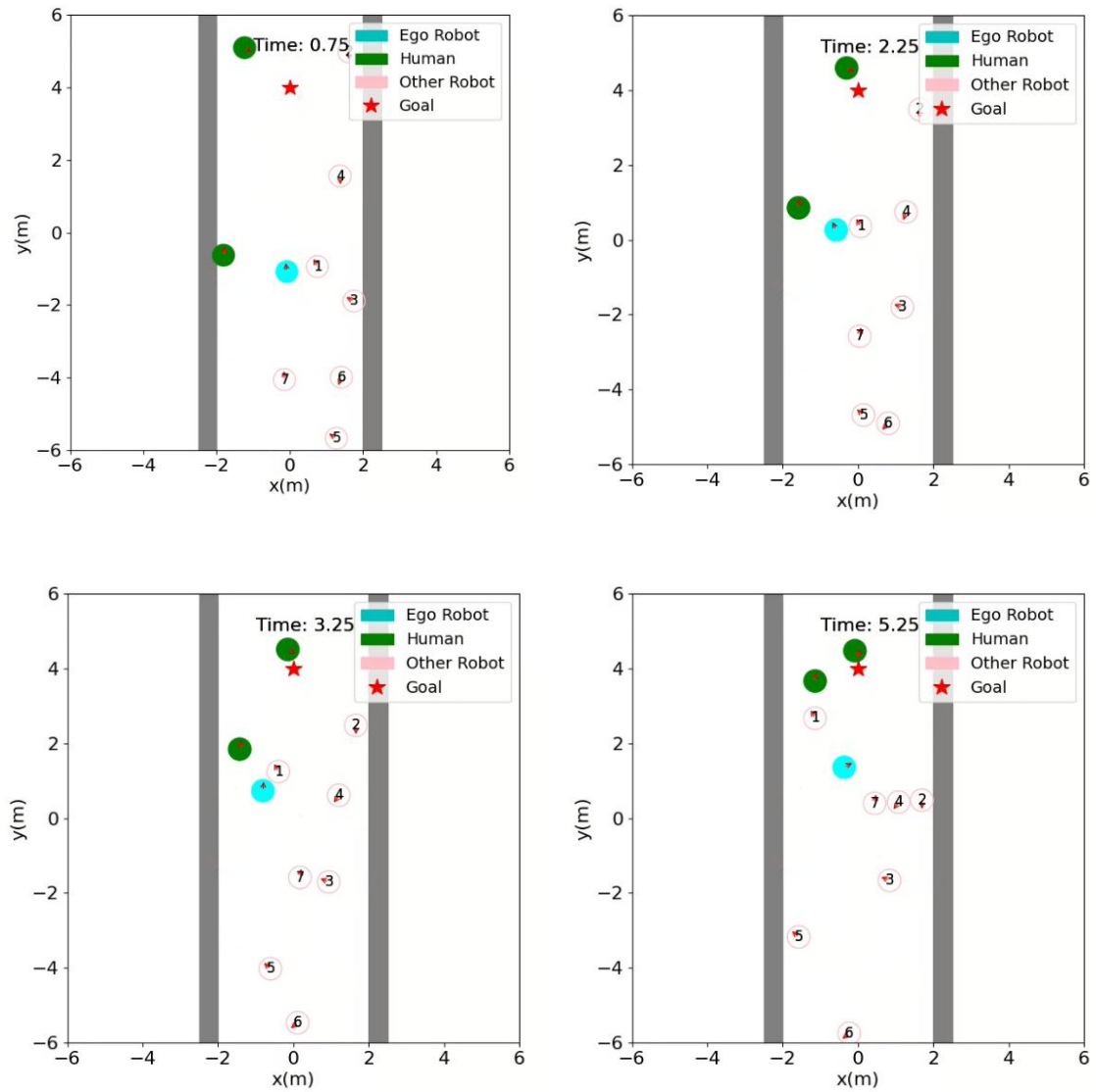
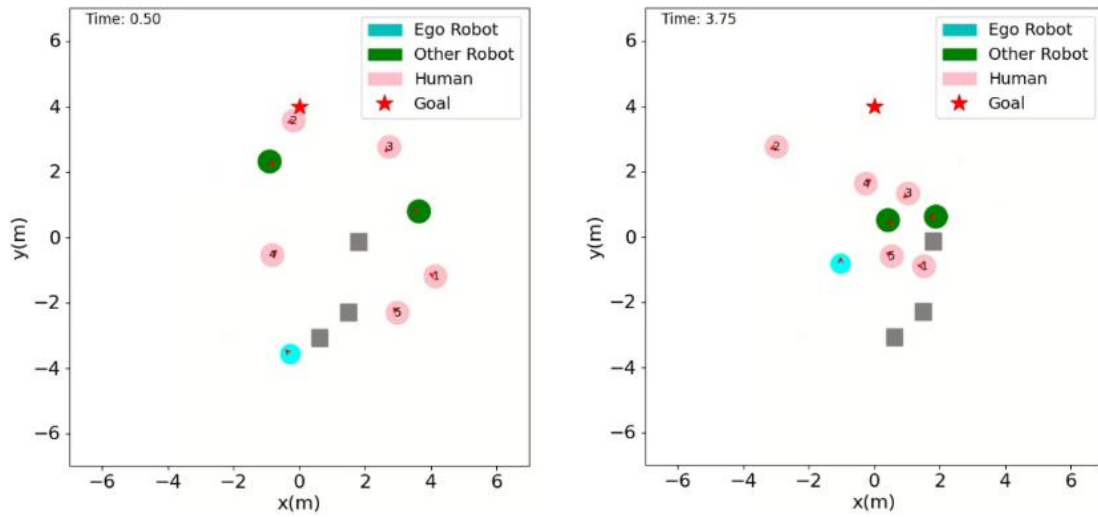


Figure 20. Simple corridor scenario with 7 Humans and 2 other robots

5.3.3 Static obstacles Scenario

Here we have scenarios of the robots navigating in an environment with static obstacles.

As is common, human environments are filled with obstacles that are static and not moving, humans easily recognize these obstacles when navigating, likewise it is essential to equip the ego robot and other robots with the capability to detect static obstacles and plan their paths around them. We test the effectiveness of the developed framework in an environment with static obstacles which more realistically depicts natural human environments.



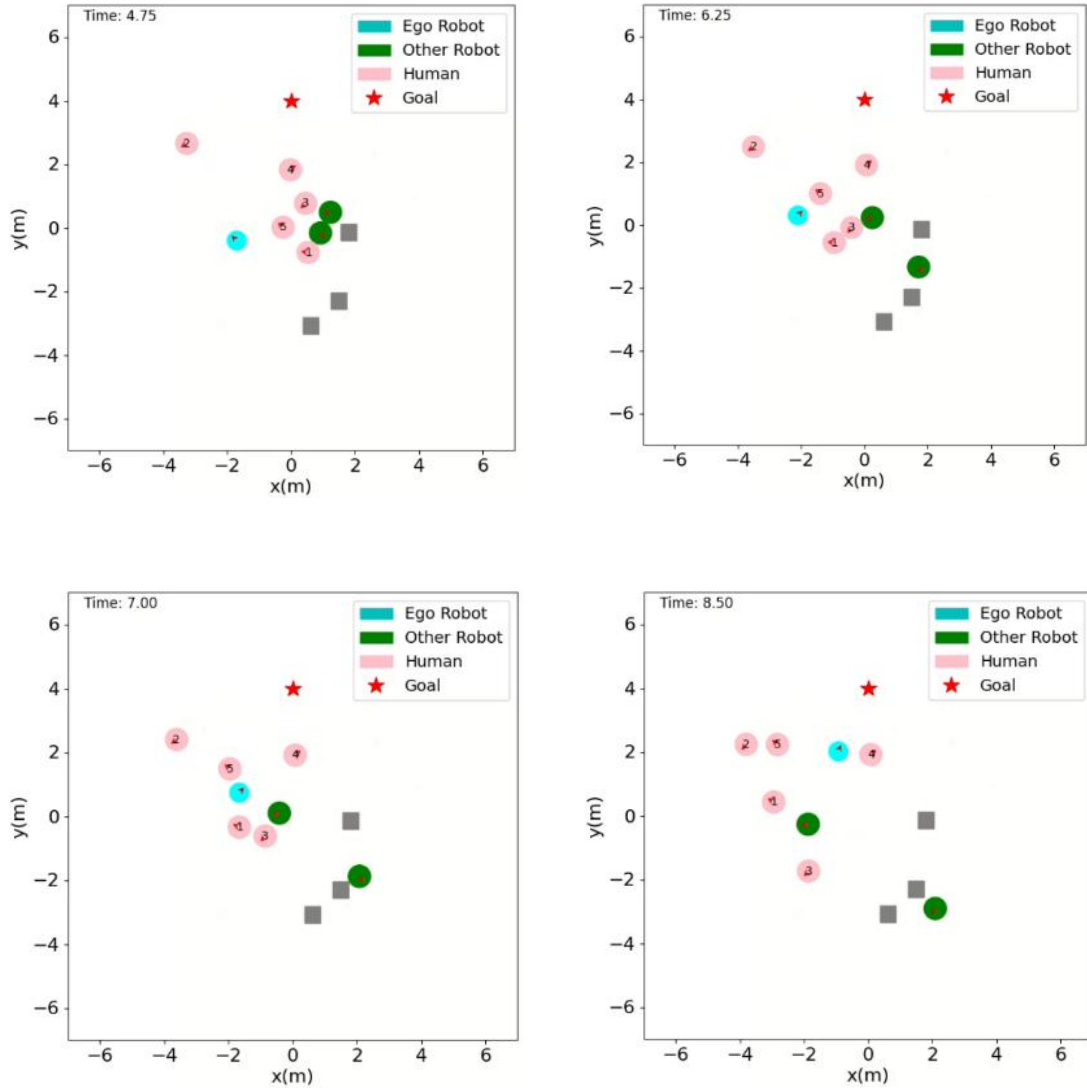


Figure 21. 5 humans and 2 other robots scenario at: 0.5s, 3.75s, 4.75s, 6.25s, 7s, 8.5s

In this static obstacle scenario as shown in Figure 21, the effectiveness of the ego robot's socially aware navigation strategy is evaluated in a setting with 5 humans, 2 other robots, and several static obstacles (grey squares). This navigation case illustrates the ego robot's decision-making process as it traverses a complex environment while maintaining safe and socially acceptable distances from both moving agents and static objects. At the onset, the ego robot begins reasoning about obstacles to avoid and starts anticipating the future

positions of nearby humans and other robots. As the scenario develops ($T = 3.75s$ to $T = 4.75s$), the ego robot actively moves towards the denser part of the environment while taking precaution not to venture inwards only at the right time. The successful social navigation of the ego robot can be seen as respects comfort zones while it moves through narrow passages between moving humans and other robots ($T = 6.25s$ to $T = 7.0s$). By the final shown time step ($T = 8.50s$), the ego robot has approached the goal region successfully, with a trajectory that reflects a balance between comfort and efficiency.

5.4 Quantitative Results/ Experimental Setup

Table 5.1 presents a comparative evaluation of the developed navigation framework against two previous approaches [57]. The proposed method achieves a success rate of 89%, which, even though is slightly lower than that of HeR-DRL (96.54%) and HoR-DRL (96.05%), is of an acceptable value considering our approach is more conservative to increase generalization ability. A key difference is that the proposed approach achieves a zero collision rate, highlighting its superior safety behavior. In contrast, both HeR-DRL and HoR-DRL experience non-negligible collision rates of 3.14% and 3.06%, respectively.

Regarding navigation time, this work achieves an average of 12.13 seconds, slightly longer than the baseline methods. This increase is expected due to the more complex interaction dynamics and the conservative decisions made by the ego robot to avoid both humans and other robots. However, the trade-off results in a higher minimum separation distance of 0.16 meters, surpassing the values of 0.154 m and 0.15 m for HeR-DRL and HoR-DRL, respectively. This indicates enhanced social compliance and a lower likelihood of discomfort among surrounding agents. The timeout rate for this work is 11%, which is higher than that of the compared models. This again can be attributed to the cautious

planning required in dense, heterogeneous environments. However, this can be justified since the method needs to prioritize avoiding collisions even while maintaining safe interpersonal distances.

Overall, while the proposed method has slight trade-offs in success rate and navigation time, it significantly improves safety and social compliance. This makes it particularly suitable for real-world scenarios where collisions are unacceptable and multi-robot socially aware behavior is required.

Table 4. Comparison of Similar work based on selected metrics

Works	Success Rate (%)	Collision Rate (%)	Navigation time (s)	Time out Rate (%)	Minimum separation distance (m)
This work	89	0	12.13	11	0.16
HeR-DRL[57]	96.54	3.14	10.88	0.32	0.154
HoR-DRL[57]	96.05	3.06	10.91	0.89	0.15

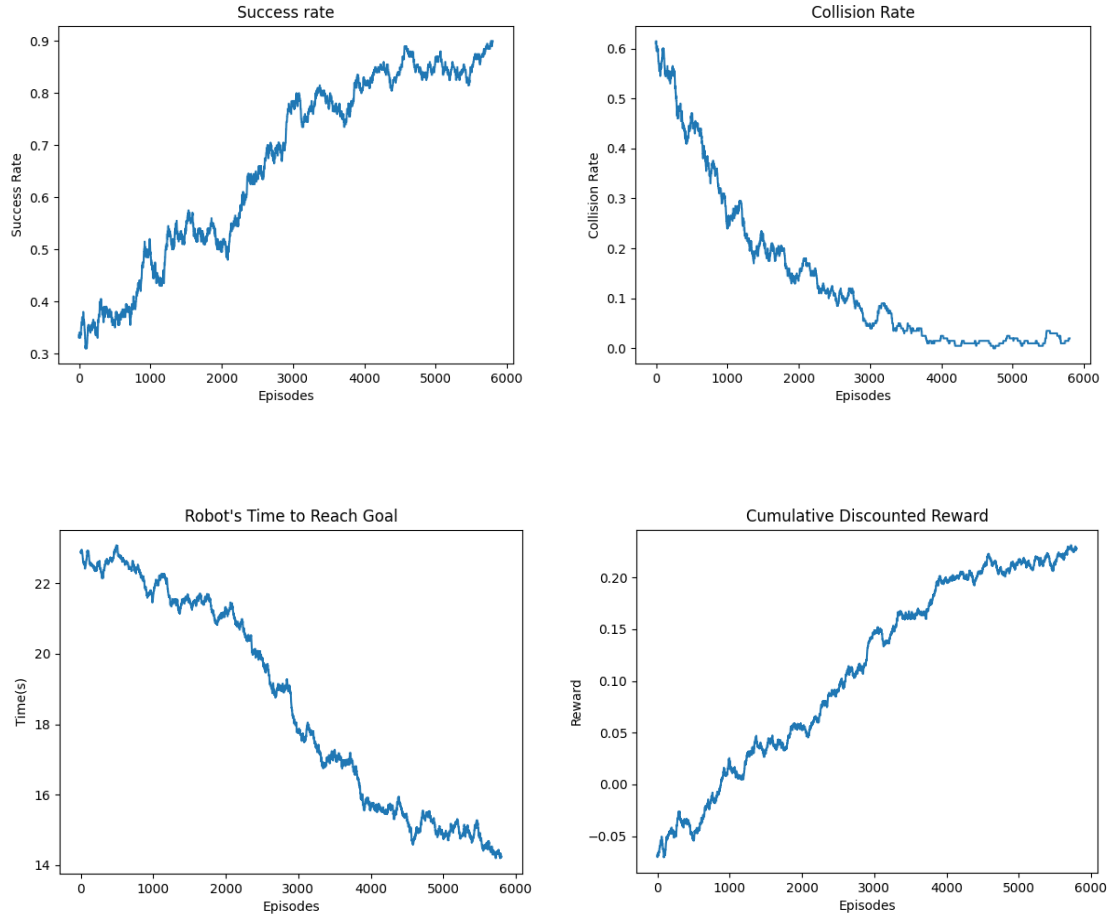


Figure 22. Performance metrics for the Navigation framework over Training episodes

From Figure 22 we can see that the success rate improves from around 30% at the beginning to approximately 90% by the end of training, which shows that the agent is learning. The occasional dips reflect exploration or unstable policy updates, which are common during RL training. Correspondingly, the collision rate decreases sharply from an initial value of over 60% to nearly 0%. This directly indicates the policy learning to avoid humans and other robots in its path. The steep decline before episode 3000 implies early gains in safety behavior, likely guided by imitation learning and reward shaping.

A downward trend in the time taken to reach the goal shows that the ego robot is learning to avoid collisions and doing so with increasing efficiency. The time drops from around 23 seconds to under 15 seconds, which suggests improved path planning and better interaction modeling. The steadily increasing cumulative reward confirms that the learning objective is being met. It is initially negative due to frequent collisions and penalties, then it transitions to positive values as the agent learns to maximize returns by reaching the goal quickly and safely. The reward curve's saturation near the end suggests policy convergence.

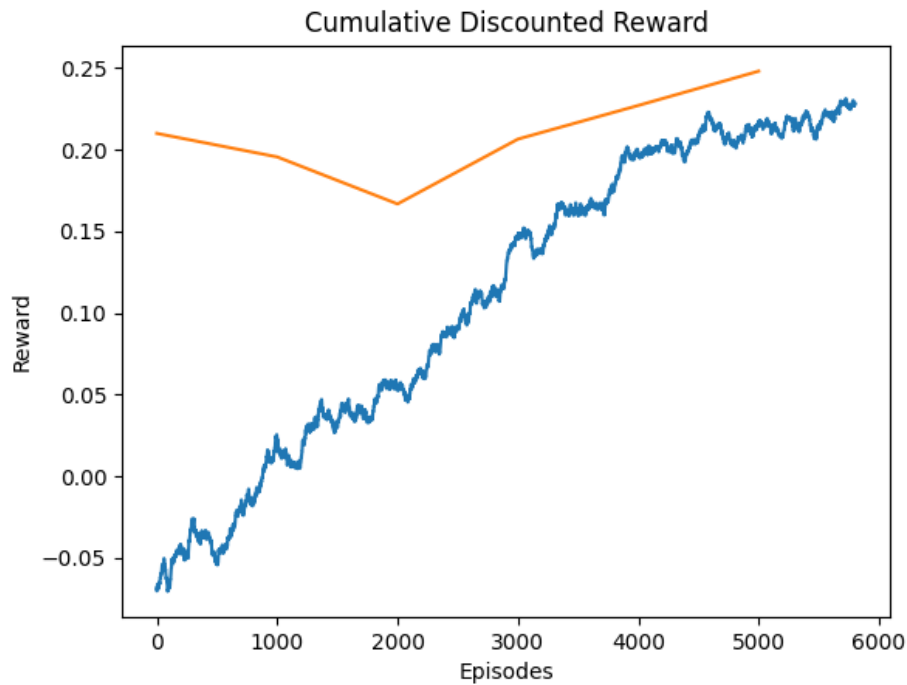


Figure 23. Comparison of cumulative reward and validation reward

Figure 23 shows a graph comparing the cumulative reward and validation reward. The evaluation or validation reward is the reward obtained when the agent's policy is tested without exploration (i.e., greedy policy). The evaluation rewards are consistent and

improve in later stages, showing the policy is generalizing well. Validation rewards trend upward and don't diverge from training rewards. It was plotted at regular intervals of 1000 episodes.

5.5 Impact of Imitation Learning on policy performance

We performed a comparative experiment between two models: one trained exclusively through Deep Reinforcement Learning (DRL), and another initiated using imitation learning with ORCA-generated demonstrations prior to DRL thus quantitatively evaluating the contribution of imitation learning (IL) to the training process. This was done under same environments, reward systems, and training duration (6,000 episodes).

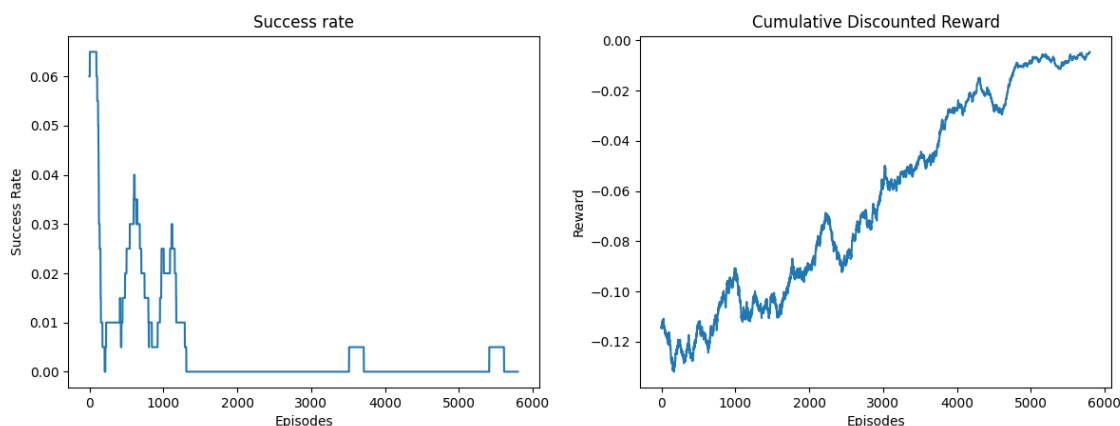


Figure 24. Success rate and Cumulative discount reward

Figure 24 shows for both settings, the training reward over episodes. With a steeper reward gradient and higher final reward, the policy initialized with IL routinely surpasses the DRL-only policy throughout all training epochs. Likewise the graphs are shown with ± 1 standard deviation in Figure 25 and Figure 26. As can be seen the success rate and time to reach goal graphs demonstrate the poor performance of the policy without initializing with

reinforcement learning leading to an overwhelming majority of episodes ending up as timeouts.

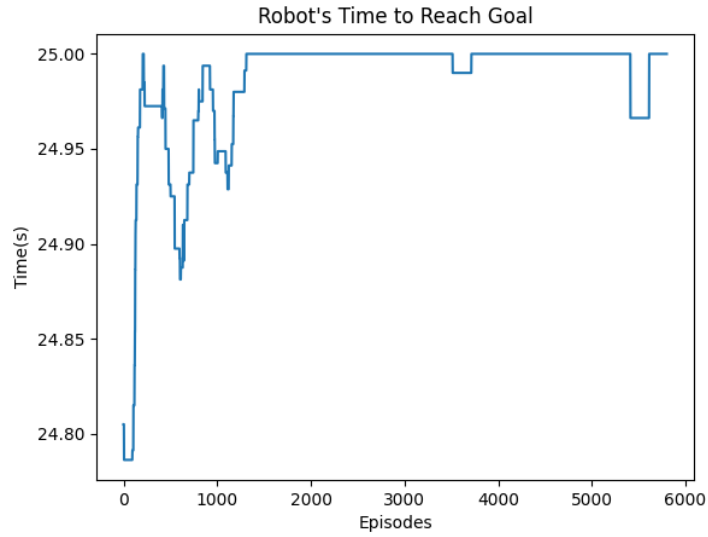


Figure 25. Ego Robot's time to reach goal without IL

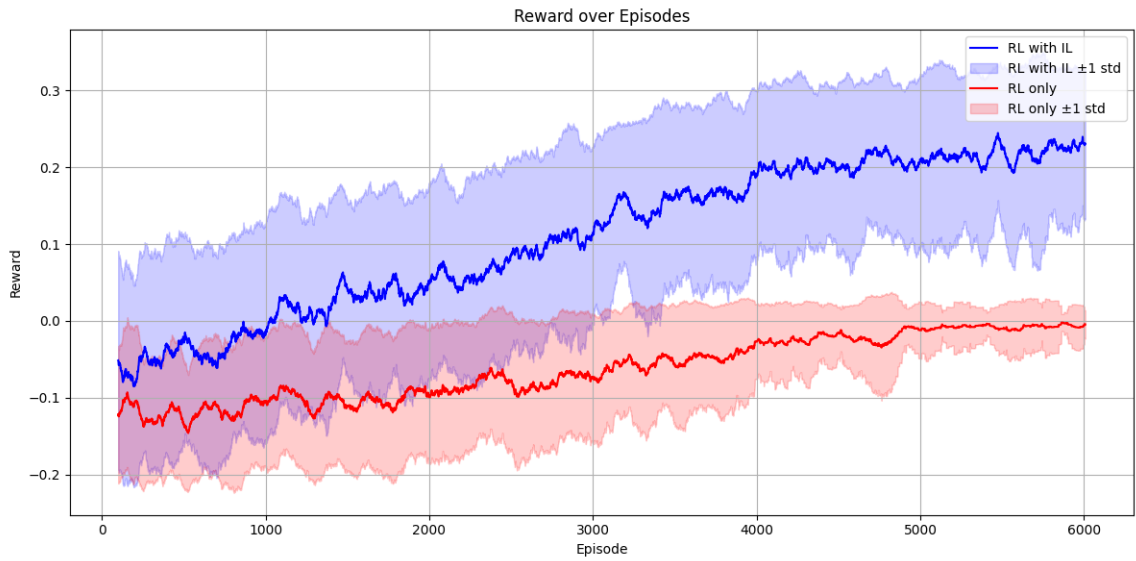


Figure 26. Reward for DRL-only and IL with DRL models, with ± 1 standard deviation.

These findings taken together verify that imitation learning serves as a powerful prior, leading early in training the policy towards safer and more optimal behavior. The IL-based approach reduces the trial-and-error burden of RL, hence improving sample efficiency, stability, and performance. In multi-robot socially-aware situations, where unsafe behavior could cause collisions or even harm, this is very important.

5.6 Summary

This chapter has presented the results of the developed multirobot SAN framework and provided a comprehensive evaluation. Through a series of simulation scenarios ranging from crowded scenes with up to 10 humans and multiple robots to constrained environments with static obstacles and a corridor environment, the developed framework has been shown to demonstrate effective path planning and social awareness for the ego robot. Visual trajectory analyses confirmed the robot’s ability to move through dynamic agents while preserving comfort distances. The training performance, reflected by steadily increasing success rates and cumulative rewards, and decreasing collision and timeout rates, has shown that indeed the RL policy converged and achieved learning stability.

Quantitative comparisons with existing baselines (HeR-DRL and HoR-DRL) revealed that while the proposed method incurred slightly higher navigation times and timeout rates, it outperformed both baselines in collision avoidance and maintained higher minimum separation distances. These findings validate the safety-first design of the policy and its potential for real-world deployment in diverse environments of multiple humans and multiple robots. The results collectively affirm the viability of single-agent deep reinforcement learning for multi-robot socially aware navigation with heterogeneous agent modeling.

CHAPTER 6 : CONCLUSION AND RECOMMENDATION

This section presents the summary of the research undertaken in the thesis. It highlights the significant findings from the study and draws relevant conclusions. In addition, the limitations encountered during this research are outlined, and the potential directions for future research in these areas are suggested.

6.1 Summary of the Work

This thesis addresses the challenge of enabling multiple robots to navigate human-populated environments in a manner that is socially acceptable through the use of deep reinforcement learning (DRL). The primary objective is to develop a navigation framework that allows multiple robots to interact safely and naturally with humans and each other, while achieving their navigation tasks efficiently. Social awareness is embedded into the navigation policy by modeling all the important interactions within the DRL framework. The proposed approach uses attention mechanisms and imitation learning to improve navigation and decision-making in dense, dynamic environments.

Chapter 1 introduces the background and motivation behind socially aware navigation. It defines the scope and objectives of the study and outlines the need for scalable multirobot navigation frameworks capable of operating in complex human environments.

Chapter 2 provides a detailed review of existing approaches in this research field. It discusses traditional methods including reactive and trajectory-based navigation and moves on to learning-based methods such as DRL. It also presents a comparison of related work in multirobot SAN and highlights the gaps addressed by this thesis.

Chapter 3 presents the theoretical foundations of the research. It covers reinforcement learning, deep learning models such as MLP and GNN, and introduces the attention mechanism and imitation learning as essential tools for policy learning. This chapter lays the groundwork for understanding the proposed navigation framework.

Chapter 4 explains the methodology used to implement the proposed system. The problem is formulated as a Partially Observable Markov Decision Process (POMDP). This chapter details the system architecture, modeling of agent interactions, and the design of the reward function, simulation environment, and evaluation metrics.

Chapter 5 presents the results of the experiments conducted using the proposed framework. The system is evaluated in various scenarios involving multiple humans and robots. Results are compared with state-of-the-art methods using metrics such as navigation success rate, collision rate, and discomfort level to demonstrate the advantages of the proposed approach.

Chapter 6 concludes the thesis by summarizing the major findings and contributions. It discusses limitations of the current framework and outlines recommendations for future work, including real-world deployment and extension to more complex interaction settings.

6.2 Recommendations for Future Work

- **Real-World Validation:** Extend the current framework to physical robots operating in real environments such as hospitals, offices, or crowded public spaces to evaluate robustness under perceptual uncertainty and control noise.

- **Spatio-Temporal Encoding:** Incorporate spatio-temporal graph neural networks or attention-based temporal encoders to better model how human and robot behaviors evolve over time.
- **Efficient and Scalable Architectures:** Design lightweight interaction encoders or sparse attention mechanisms that reduce computational load while preserving performance in large-scale multi-agent systems.
- **Multi-Agent Reinforcement Learning (MARL):** Explore decentralized or centralized MARL approaches to enable cooperative task completion (e.g., formation navigation) and handle adversarial interactions (e.g., competitive access to shared space).
- **Deployment on Humanoid Robots:** Adapt and deploy the learned navigation policies on humanoid platforms to address scenarios requiring expressive motion, better human-robot interaction, or operation in structurally complex 3D environments.
- **Policy Interpretability Tools:** Integrate attention visualization, saliency maps, or rule extraction methods to enhance transparency and facilitate debugging, safety verification, and user trust.

REFERENCES

- [1] S. Nahavandi *et al.*, ‘A Comprehensive Review on Autonomous Navigation’, *ACM Comput Surv*, Mar. 2025, doi: 10.1145/3727642.
- [2] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, ‘Human-aware robot navigation: A survey’, *Robot. Auton. Syst.*, vol. 61, no. 12, pp. 1726–1743, Dec. 2013, doi: 10.1016/j.robot.2013.05.007.
- [3] ‘Personal space: An evaluative and orienting overview.’ Accessed: May 09, 2025. [Online]. Available: <https://psycnet.apa.org/record/1979-23561-001>
- [4] ‘Proxemics [and Comments and Replies] | Current Anthropology: Vol 9, No 2/3’. Accessed: May 09, 2025. [Online]. Available: <https://www.journals.uchicago.edu/doi/abs/10.1086/200975>
- [5] P. T. Singamaneni *et al.*, ‘A survey on socially aware robot navigation: Taxonomy and future challenges’, *Int. J. Robot. Res.*, p. 02783649241230562, Feb. 2024, doi: 10.1177/02783649241230562.
- [6] J. Karwowski, W. Szynekiewicz, and E. Niewiadomska-Szynekiewicz, ‘Bridging Requirements, Planning, and Evaluation: A Review of Social Robot Navigation’, *Sensors*, vol. 24, no. 9, p. 2794, Apr. 2024, doi: 10.3390/s24092794.
- [7] ‘SPENCER - Social Situation-aware Perception and Action for Cognitive Robots - Örebro University’. Accessed: Nov. 20, 2024. [Online]. Available: <https://www.oru.se/english/research/research-projects/rp/?rdb=p966>
- [8] ‘Autonomous human–robot proxemics: socially aware navigation based on interaction potential | Autonomous Robots’. Accessed: Apr. 27, 2025. [Online]. Available: <https://link.springer.com/article/10.1007/s10514-016-9572-2>
- [9] L. Dong, Z. He, C. Song, X. Yuan, and H. Zhang, ‘Multi-robot social-aware cooperative planning in pedestrian environments using attention-based actor-critic’, *Artif. Intell. Rev.*, vol. 57, no. 4, p. 108, Apr. 2024, doi: 10.1007/s10462-024-10739-w.
- [10] S. V. Albrecht, F. Christianos, and L. Schäfer, ‘Multi-Agent Reinforcement Learning: Foundations and Modern Approaches’.
- [11] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, ‘ORB-SLAM: A Versatile and Accurate Monocular SLAM System’, *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.
- [12] S. Thrun, ‘Integrating Grid-Based and Topological Maps for Mobile Robot Navigation’.
- [13] J. Levinson and S. Thrun, ‘Robust vehicle localization in urban environments using probabilistic maps’, in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 4372–4378. doi: 10.1109/ROBOT.2010.5509700.
- [14] J. Rios-Martinez, A. Spalanzani, and C. Laugier, ‘From Proxemics Theory to Socially-Aware Navigation: A Survey’, *Int. J. Soc. Robot.*, vol. 7, no. 2, pp. 137–153, Apr. 2015, doi: 10.1007/s12369-014-0251-1.
- [15] Y. Gao and C.-M. Huang, ‘Evaluation of Socially-Aware Robot Navigation’, *Front. Robot. AI*, vol. 8, Jan. 2022, doi: 10.3389/frobt.2021.721317.

- [16] C. Mavrogiannis *et al.*, ‘Core Challenges of Social Robot Navigation: A Survey’, *ACM Trans. Hum.-Robot Interact.*, vol. 12, no. 3, pp. 1–39, Sep. 2023, doi: 10.1145/3583741.
- [17] D. Helbing and P. Molnár, ‘Social force model for pedestrian dynamics’, *Phys. Rev. E*, vol. 51, no. 5, pp. 4282–4286, May 1995, doi: 10.1103/PhysRevE.51.4282.
- [18] S. Guillén-Ruiz, J. P. Bandera, A. Hidalgo-Paniagua, and A. Bandera, ‘Evolution of Socially-Aware Robot Navigation’, *Electronics*, vol. 12, no. 7, p. 1570, Mar. 2023, doi: 10.3390/electronics12071570.
- [19] Y. Chen, F. Zhao, and Y. Lou, ‘Interactive Model Predictive Control for Robot Navigation in Dense Crowds’, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 52, no. 4, pp. 2289–2301, Apr. 2022, doi: 10.1109/TSMC.2020.3048964.
- [20] P. Fiorini and Z. Shiller, ‘Motion planning in dynamic environments using the relative velocity paradigm’, in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, May 1993, pp. 560–565 vol.1. doi: 10.1109/ROBOT.1993.292038.
- [21] Z. Shiller, F. Large, and S. Sekhavat, ‘Motion planning in dynamic environments: obstacles moving along arbitrary trajectories’, in *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, May 2001, pp. 3716–3721 vol.4. doi: 10.1109/ROBOT.2001.933196.
- [22] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, ‘Reciprocal n-body collision avoidance’, in *Robotics Research: The 14th International Symposium ISRR*, Springer, 2011, pp. 3–19. Accessed: May 31, 2024. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-642-19457-3_1
- [23] J. Van Den Berg, Ming Lin, and D. Manocha, ‘Reciprocal Velocity Obstacles for real-time multi-agent navigation’, in *2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA: IEEE, May 2008, pp. 1928–1935. doi: 10.1109/ROBOT.2008.4543489.
- [24] J. Snape, J. van den Berg, S. J. Guy, and D. Manocha, ‘The Hybrid Reciprocal Velocity Obstacle’, *IEEE Trans. Robot.*, vol. 27, no. 4, pp. 696–706, Aug. 2011, doi: 10.1109/TRO.2011.2120810.
- [25] ‘Social Cues in the Autonomous Navigation of Indoor Mobile Robots | International Journal of Social Robotics’. Accessed: Apr. 27, 2025. [Online]. Available: <https://link.springer.com/article/10.1007/s12369-020-00721-1>
- [26] X.-T. Truong and T. D. Ngo, ‘Toward Socially Aware Robot Navigation in Dynamic and Crowded Environments: A Proactive Social Motion Model’, *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 4, pp. 1743–1760, Oct. 2017, doi: 10.1109/TASE.2017.2731371.
- [27] J. Borenstein and Y. Koren, ‘The vector field histogram-fast obstacle avoidance for mobile robots’, *IEEE Trans. Robot. Autom.*, vol. 7, no. 3, pp. 278–288, Jun. 1991, doi: 10.1109/70.88137.
- [28] A. Babinec, F. Duchoň, M. Dekan, Z. Mikulová, and L. Jurišica, ‘Vector Field Histogram* with look-ahead tree extension dependent on time variable environment’, *Trans. Inst. Meas. Control*, vol. 40, no. 4, pp. 1250–1264, Feb. 2018, doi: 10.1177/0142331216678062.

- [29] O. Khatib, ‘Real-time obstacle avoidance for manipulators and mobile robots’, in *1985 IEEE International Conference on Robotics and Automation Proceedings*, Mar. 1985, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.
- [30] Q. Yao *et al.*, ‘Path Planning Method With Improved Artificial Potential Field—A Reinforcement Learning Perspective’, *IEEE Access*, vol. 8, pp. 135513–135523, 2020, doi: 10.1109/ACCESS.2020.3011211.
- [31] G. Ferrer, A. G. Zulueta, F. H. Cotarelo, and A. Sanfeliu, ‘Robot social-aware navigation framework to accompany people walking side-by-side’, *Auton. Robots*, vol. 41, no. 4, pp. 775–793, Apr. 2017, doi: 10.1007/s10514-016-9584-y.
- [32] P. Trautman, J. Ma, R. M. Murray, and A. Krause, ‘Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation’, *Int. J. Robot. Res.*, vol. 34, no. 3, pp. 335–356, Mar. 2015, doi: 10.1177/0278364914557874.
- [33] G. S. Aoude, B. D. Luders, J. M. Joseph, N. Roy, and J. P. How, ‘Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns’, *Auton. Robots*, vol. 35, no. 1, pp. 51–76, Jul. 2013, doi: 10.1007/s10514-013-9334-3.
- [34] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, ‘Social LSTM: Human Trajectory Prediction in Crowded Spaces’, presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 961–971. Accessed: Apr. 27, 2025. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Alahi_Social_LSTM_Human_CVPR_2016_paper.html
- [35] A. Vemula, K. Muelling, and J. Oh, ‘Social Attention: Modeling Attention in Human Crowds’, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4601–4607. doi: 10.1109/ICRA.2018.8460504.
- [36] M. Svenstrup, T. Bak, and H. J. Andersen, ‘Trajectory planning for robots in dynamic human environments’, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 4293–4298. doi: 10.1109/IROS.2010.5651531.
- [37] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, ‘Probabilistic navigation in dynamic environment using Rapidly-exploring Random Trees and Gaussian processes’, in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sep. 2008, pp. 1056–1062. doi: 10.1109/IROS.2008.4650959.
- [38] P. Trautman and A. Krause, ‘Unfreezing the robot: Navigation in dense, interacting crowds’, in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 797–803. doi: 10.1109/IROS.2010.5654369.
- [39] Y. F. Chen, M. Liu, M. Everett, and J. P. How, ‘Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning’, in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, Singapore: IEEE, May 2017, pp. 285–292. doi: 10.1109/ICRA.2017.7989037.
- [40] Y. F. Chen, M. Everett, M. Liu, and J. P. How, ‘Socially aware motion planning with deep reinforcement learning’, in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 1343–1350. doi: 10.1109/IROS.2017.8202312.
- [41] M. Everett, Y. F. Chen, and J. P. How, ‘Motion Planning Among Dynamic, Decision-Making Agents with Deep Reinforcement Learning’, in *2018 IEEE/RSJ*

- International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 3052–3059. doi: 10.1109/IROS.2018.8593871.
- [42] ‘Crowd-Comfort Robot Navigation Among Dynamic Environment Based on Social-Stressed Deep Reinforcement Learning | International Journal of Social Robotics’. Accessed: Apr. 27, 2025. [Online]. Available: <https://link.springer.com/article/10.1007/s12369-021-00838-x>
 - [43] ‘Social Robot Navigation Tasks: Combining Machine Learning Techniques and Social Force Model’. Accessed: Apr. 27, 2025. [Online]. Available: <https://www.mdpi.com/1424-8220/21/21/7087>
 - [44] A. Francis *et al.*, ‘Long-Range Indoor Navigation With PRM-RL’, *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1115–1134, Aug. 2020, doi: 10.1109/TRO.2020.2975428.
 - [45] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, ‘Crowd-Robot Interaction: Crowd-Aware Robot Navigation With Attention-Based Deep Reinforcement Learning’, in *2019 International Conference on Robotics and Automation (ICRA)*, Montreal, QC, Canada: IEEE, May 2019, pp. 6015–6022. doi: 10.1109/ICRA.2019.8794134.
 - [46] L. Liu, D. Dugas, G. Cesari, R. Siegwart, and R. Dube, ‘Robot Navigation in Crowded Environments Using Deep Reinforcement Learning’, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 5671–5677. doi: 10.1109/IROS45743.2020.9341540.
 - [47] Y. Chen, C. Liu, B. E. Shi, and M. Liu, ‘Robot Navigation in Crowds by Graph Convolutional Networks With Attention Learned From Human Gaze’, *IEEE Robot. Autom. Lett.*, vol. 5, no. 2, pp. 2754–2761, Apr. 2020, doi: 10.1109/LRA.2020.2972868.
 - [48] S. S. Samsani and M. S. Muhammad, ‘Socially Compliant Robot Navigation in Crowded Environment by Human Behavior Resemblance Using Deep Reinforcement Learning’, *IEEE Robot. Autom. Lett.*, vol. 6, no. 3, pp. 5223–5230, Jul. 2021, doi: 10.1109/LRA.2021.3071954.
 - [49] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, ‘Relational Graph Learning for Crowd Navigation’, in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2020, pp. 10007–10013. doi: 10.1109/IROS45743.2020.9340705.
 - [50] Z. Zhou, P. Zhu, Z. Zeng, J. Xiao, H. Lu, and Z. Zhou, ‘Robot Navigation in a Crowd by Integrating Deep Reinforcement Learning and Online Planning’, Feb. 25, 2021, *arXiv*: arXiv:2102.13265. Accessed: May 25, 2024. [Online]. Available: <http://arxiv.org/abs/2102.13265>
 - [51] L. Tai, J. Zhang, M. Liu, and W. Burgard, ‘Socially Compliant Navigation Through Raw Depth Inputs with Generative Adversarial Imitation Learning’, in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 1111–1117. doi: 10.1109/ICRA.2018.8460968.
 - [52] M. Pfeiffer, M. Schaeuble, J. Nieto, R. Siegwart, and C. Cadena, ‘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)*, May 2017, pp. 1527–1533. doi: 10.1109/ICRA.2017.7989182.
 - [53] P. Henry, C. Vollmer, B. Ferris, and D. Fox, ‘Learning to navigate through crowded environments’, in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 981–986. doi: 10.1109/ROBOT.2010.5509772.

- [54] C. Perez-D'Arpino, C. Liu, P. Goebel, R. Martin-Martin, and S. Savarese, 'Robot Navigation in Constrained Pedestrian Environments using Reinforcement Learning', in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China: IEEE, May 2021, pp. 1140–1146. doi: 10.1109/ICRA48506.2021.9560893.
- [55] W. Wang, L. Mao, R. Wang, and B.-C. Min, 'Multi-Robot Cooperative Socially-Aware Navigation Using Multi-Agent Reinforcement Learning', May 15, 2024, *arXiv: arXiv:2309.15234*. Accessed: May 25, 2024. [Online]. Available: <http://arxiv.org/abs/2309.15234>
- [56] E. Escudie, L. Matignon, and J. Saraydaryan, 'Attention Graph for Multi-Robot Social Navigation with Deep Reinforcement Learning', Jan. 31, 2024, *arXiv: arXiv:2401.17914*. Accessed: May 28, 2024. [Online]. Available: <http://arxiv.org/abs/2401.17914>
- [57] X. Zhou, S. Piao, W. Chi, L. Chen, and W. Li, 'HeR-DRL:Heterogeneous Relational Deep Reinforcement Learning for Decentralized Multi-Robot Crowd Navigation', Mar. 15, 2024, *arXiv: arXiv:2403.10083*. doi: 10.48550/arXiv.2403.10083.
- [58] R. S. Sutton and A. Barto, *Reinforcement learning: an introduction*, Nachdruck. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2014.
- [59] 'Dynamic Programming | Science'. Accessed: Apr. 27, 2025. [Online]. Available: <https://www.science.org/doi/abs/10.1126/science.153.3731.34>
- [60] R. S. Sutton, 'Learning to predict by the methods of temporal differences', *Mach. Learn.*, vol. 3, no. 1, pp. 9–44, Aug. 1988, doi: 10.1007/BF00115009.
- [61] R. Pramoditha, 'The Concept of Artificial Neurons (Perceptrons) in Neural Networks', Towards Data Science. Accessed: May 24, 2025. [Online]. Available: <https://towardsdatascience.com/the-concept-of-artificial-neurons-perceptrons-in-neural-networks-fab22249cbfc/>
- [62] D. P. Kingma and J. Ba, 'Adam: A Method for Stochastic Optimization', Jan. 30, 2017, *arXiv: arXiv:1412.6980*. doi: 10.48550/arXiv.1412.6980.
- [63] automaticaddison, 'Artificial Feedforward Neural Network With Backpropagation From Scratch'. Accessed: May 24, 2025. [Online]. Available: <https://automaticaddison.com/artificial-feedforward-neural-network-with-backpropagation-from-scratch/>
- [64] T. N. Kipf and M. Welling, 'Semi-Supervised Classification with Graph Convolutional Networks', Feb. 22, 2017, *arXiv: arXiv:1609.02907*. doi: 10.48550/arXiv.1609.02907.
- [65] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio, 'Graph Attention Networks', Feb. 04, 2018, *arXiv: arXiv:1710.10903*. doi: 10.48550/arXiv.1710.10903.
- [66] 'How to Use Graph Neural Networks (GNNs) for Big Data - Datatas'. Accessed: Apr. 28, 2025. [Online]. Available: <https://datatas.com/how-to-use-graph-neural-networks-gnns-for-big-data/>
- [67] A. Vaswani *et al.*, 'Attention is All you Need', in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2017. Accessed: May 02, 2025. [Online]. Available:

- <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [68] Y. Wu, S. Liao, X. Liu, Z. Li, and R. Lu, ‘Deep Reinforcement Learning on Autonomous Driving Policy With Auxiliary Critic Network’, *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 7, pp. 3680–3690, Jul. 2023, doi: 10.1109/TNNLS.2021.3116063.
 - [69] K. Zhu and T. Zhang, ‘Deep reinforcement learning based mobile robot navigation: A review’, *Tsinghua Sci. Technol.*, vol. 26, no. 5, pp. 674–691, Oct. 2021, doi: 10.26599/TST.2021.9010012.
 - [70] V. Mnih *et al.*, ‘Asynchronous Methods for Deep Reinforcement Learning’, in *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, Jun. 2016, pp. 1928–1937. Accessed: Apr. 28, 2025. [Online]. Available: <https://proceedings.mlr.press/v48/mniha16.html>
 - [71] D. Han, B. Mulyana, V. Stankovic, and S. Cheng, ‘A Survey on Deep Reinforcement Learning Algorithms for Robotic Manipulation’, *Sensors*, vol. 23, no. 7, Art. no. 7, Jan. 2023, doi: 10.3390/s23073762.
 - [72] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour, ‘Policy Gradient Methods for Reinforcement Learning with Function Approximation’, in *Advances in Neural Information Processing Systems*, MIT Press, 1999. Accessed: Apr. 28, 2025. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1999/hash/464d828b85b0bed98e80ade0a5c43b0f-Abstract.html
 - [73] E. Escudie, L. Matignon, and J. Saraydaryan, ‘Attention Graph for Multi-Robot Social Navigation with Deep Reinforcement Learning’, Jan. 31, 2024, *arXiv*: arXiv:2401.17914. Accessed: Sep. 25, 2024. [Online]. Available: <http://arxiv.org/abs/2401.17914>
 - [74] Y. Zhou and J. Garcke, ‘Learning Crowd Behaviors in Navigation with Attention-based Spatial-Temporal Graphs’, Jan. 11, 2024, *arXiv*: arXiv:2401.06226. Accessed: Sep. 25, 2024. [Online]. Available: <http://arxiv.org/abs/2401.06226>
 - [75] M. Pfeiffer *et al.*, ‘Reinforced Imitation: Sample Efficient Deep Reinforcement Learning for Mapless Navigation by Leveraging Prior Demonstrations’, *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 4423–4430, Oct. 2018, doi: 10.1109/LRA.2018.2869644.
 - [76] J. Achiam, D. Held, A. Tamar, and P. Abbeel, ‘Constrained Policy Optimization’, in *Proceedings of the 34th International Conference on Machine Learning*, PMLR, Jul. 2017, pp. 22–31. Accessed: May 02, 2025. [Online]. Available: <https://proceedings.mlr.press/v70/achiam17a.html>
 - [77] A. Paszke *et al.*, ‘Automatic differentiation in PyTorch’, Oct. 2017, Accessed: May 02, 2025. [Online]. Available: <https://openreview.net/forum?id=BJJsrnfCZ>
 - [78] G. Brockman *et al.*, ‘OpenAI Gym’, Jun. 05, 2016, *arXiv*: arXiv:1606.01540. doi: 10.48550/arXiv.1606.01540.
 - [79] ‘Attention-based advantage actor-critic algorithm with prioritized experience replay for complex 2-D robotic motion planning’, ResearchGate. Accessed: May 25, 2025. [Online]. Available: https://www.researchgate.net/publication/362540404_Attention-based_advantage_actor-

critic_algorithm_with_prioritized_experience_replay_for_complex_2-D_robotic_motion_planning

- [80] J. Wang, W. P. Chan, P. Carreno-Medrano, A. Cosgun, and E. Croft, ‘Metrics for Evaluating Social Conformity of Crowd Navigation Algorithms’, in *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*, May 2022, pp. 1–6. doi: 10.1109/ARSO54254.2022.9802981.

VITAE

Name	Ibrahim Khalil Kabir
Nationality	Nigerian
Date of Birth	15 th June, 1998
Email	ibrahimkkabir@gmail.com
Address	No. 5A Zobe Road, Unguwan Sarki GRA, Kaduna State, Nigeria. Postal code: 800221
Academic Background	M.Sc. in Systems and Control Engineering (2025)
	• <i>King Fahd University of Petroleum & Minerals, KSA.</i>
	B.Eng. Mechatronics Engineering (2021)
	• <i>Bayero University Kano, Kano, Nigeria.</i>
Publication	Kabir, Ibrahim Khalil, Muhammad Majid Gulzar, Adnan Shakoor, Salman Habib, and Hasnain Ahmad. "Trajectory Tracking of Unmanned Aerial Vehicles using Backstepping Approach-based Sliding Mode Control." In <i>2024 International Conference on Control, Automation and Diagnosis (ICCAD)</i> , pp. 1-6. IEEE, 2024.
	Surur, Khaled, Ibrahim Kabir, Ghali Ahmad, and Mohammad A. Abido. "Optimal Gain Scheduling for Fault-Tolerant Control of Quadrotor UAV Using Genetic Algorithm-Based Neural Network." <i>Arabian Journal for Science and Engineering</i> (2025): 1-15.

	Mosally, Amer, Ibrahim Kabir, and Mohammad Hammoudeh. "Post-Quantum Cryptography in Embedded Systems, Future Trends, and Research Directions." Quantum Computing (2024): 159-173.
	RESEARCH AND TEACHING ASSISTANT (Jan 2023 to Date) <i>King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia</i>