Université de Lille

IMT Nord Europe
École Mines-Télécom
IMT-Université de Lille

Université Catholique de Lille 1875

THÈSE

en vue de l'obtention du grade de

DOCTEUR,
SPÉCIALITÉ INFORMATIQUE

Présenté et soutenue par :

ADAM GOUGUET

le 28 novembre 2025

Human-Aware Robot Navigation: From Comprehensive and Realistic Benchmarks to Versatile and Efficient Solutions

Navigation Robotique Consciente de l'Humain : Des Benchmarks Complets et Réalistes aux Solutions Polyvalentes et Efficaces

Thesis supervised by	Luc FABRESSE	Supervisor
	Guillaume LOZENGUEZ	Co-Monitor
	Abir KARAMI	Co-Monitor

Committee members

Committee President	⟨*Prénom*⟩ ⟨*Nom*⟩	Professor
Referees	Anne SPALANZANI	Professor at INRIA, University Grenoble Alpes
	Damien PELLIER	Professor at INRIA, University Grenoble Alpes
Examiner	Abdel-illah MOUADDIB	Professor at University of Caen
Supervisors	Luc FABRESSE	Professor at IMT Nord Europe
	Guillaume LOZENGUEZ	Assistant Professor at IMT Nord Europe
	Abir KARAMI	Associate Professor at Institut Catholique de Lille

Préparée au sein des laboratoires :

Centre d'Enseignement de Recherche et d'Innovation Systèmes Numériques
764 Bd Lahure, 59500 Douai, France

Laboratoire Interdisciplinaire des Transitions de Lille
60 Bd Vauban, F-59000 Lille, France

Human-Aware Robot Navigation: From Comprehensive and Realistic Benchmarks to Versatile and Efficient Solutions

Abstract

A major challenge for mobile robotics is Human-Aware Robot Navigation (HAN); requiring robots to navigate safely and effectively in human-shared environments while adhering to social norms and adapting to dynamic contexts. Despite extensive research, key challenges remain due to the topic's inherent complexity. On the one hand, evaluation protocols frequently use limited sets of scenarios, usually concentrating on either dense crowds or sparse environments, rarely covering both, and ignoring mixed or intermediate scenarios. State-of-the-art approaches are still struggling to present solutions that are robust in a wide range of environments, socially acceptable, efficient in navigation, and real-time.

Three complementary contributions are made in this thesis to address these issues. First, we present Robot Social Navigation Assessment Platform (RobotSNAP), a benchmarking platform that allows for systematic comparison of current approaches by offering realistic simulation, a variety of scenarios, and extensive social and performance metrics. Second, we present model-based social navigation techniques: MDP-Based Social Navigation (MBSN), which is a Markov Decision Process (MDP) over a navigation graph, and its computationally efficient variant Heuristic-Based Social Navigation (HBSN), which employs a heuristic to provide real-time socially compliant navigation. Third, to manage humans and the environment, we create a robust Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC) method. It leverages a novel Curriculum Learning (CL) strategy and a multi-branch architecture integrating a new model of Human-Robot Interaction (HRI) to enable stable learning across a variety of environments and scenarios.

Extensive simulation using RobotSNAP and real-world experiments have confirmed that each of these solutions has distinctive trade-offs: MBSN offers interpretability and strong results but suffers from high computational cost. Meanwhile, MDHC offers real-time execution and adaptability but requires extensive training, is less interpretable, and struggles with robustness. Although it is simple, HBSN strikes the best balance by combining real-time performance, robustness, and interpretability.

Overall, this thesis prepares the way for future research on implementing such solutions in increasingly complex and realistic environments while building the foundation for scalable, interpretable, and reliable human-aware navigation.

Keywords: robotics, artificiel intelligente, social robotics navigation

Navigation Robotique Consciente de l'Humain : Des Benchmarks Complets et Réalistes aux Solutions Polyvalentes et Efficaces

Résumé

Un défi majeur pour la robotique mobile est la navigation robotique consciente de l'humain (HAN) ; nécessitant que les robots naviguent en toute sécurité et efficacement dans des environnements partagés avec des humains tout en respectant les normes sociales et en s'adaptant aux contextes dynamiques. Malgré des recherches approfondies, des défis majeurs subsistent en raison de la complexité inhérente du sujet. D'une part, les protocoles d'évaluation utilisent fréquemment des ensembles limités de scénarios, se concentrant généralement soit sur des foules denses, soit sur des environnements clairsemés, couvrant rarement les deux, et ignorant les scénarios mixtes ou intermédiaires. Les approches de pointe ont encore du mal à présenter des solutions robustes dans une large gamme d'environnements, socialement acceptables, efficaces en navigation et en temps réel.

Trois contributions complémentaires sont apportées dans cette thèse pour aborder ces problèmes. Tout d'abord, nous présentons la Plateforme d'Évaluation de la Navigation Sociale des Robots (RobotSNAP), une plateforme de référence qui permet une comparaison systématique des approches actuelles en offrant une simulation réaliste, une variété de scénarios et des métriques sociales et de performance étendues. Deuxièmement, nous présentons des techniques de navigation sociale basées sur des modèles : Navigation sociale basée sur les MDP (MBSN), qui est un processus de décision de Markov (MDP) sur un graphe de navigation, et sa variante computationnellement efficace Navigation Sociale Basée sur l'Heuristique (HBSN), qui utilise une heuristique pour fournir une navigation conforme aux normes sociales en temps réel. Troisièmement, pour gérer les humains et l'environnement, nous créons une méthode robuste d'apprentissage par renforcement profond multi-branches pour le contrôle conscient des humains (MDHC). Il exploite une nouvelle stratégie d'apprentissage par programme (CL) et une architecture multi-branches intégrant un nouveau modèle d'interaction homme-robot (HRI) pour permettre un apprentissage stable dans une variété d'environnements et de scénarios.

Une simulation extensive utilisant RobotSNAP et des expériences réelles ont confirmé que chacune de ces solutions présente des compromis distincts : MBSN offre une interprétabilité et des résultats solides, mais souffre d'un coût de calcul élevé. Pendant ce temps, MDHC offre une exécution en temps réel et une adaptabilité, mais nécessite une formation approfondie, est moins interprétable et a des difficultés avec la robustesse. Bien qu'il soit simple, HBSN trouve le meilleur équilibre en combinant performance en temps réel, robustesse et interprétabilité. Dans l'ensemble, cette thèse prépare le terrain pour des recherches futures sur la mise en œuvre de telles solutions dans des environnements de plus en plus complexes et réalistes, tout en jetant les bases d'une navigation consciente de l'humain évolutive, interprétable et fiable.

Mots clés : robotique, intelligence artificielle, navigation robotique sociale

Acronyms

MDHC Multi-branch Deep Reinforcement-learning for Human-aware Control. iii, iv, clxii, clxiii, 87, 90, 97, 99, 101, 103–106, 108–112, 117, 122, 123, 139, 146, 149, 155
MDP Markov Decision Process. iii, clxii, 2–4, 31, 54–56, 58, 60, 63, 65–68, 72, 82, 83, 87, 92, 107, 108, 120
MPC Model Predictive Control. 8, 9, 43, 51
MPC-EBM Model Predictive Control With One-Shot Energy-Based Multi modal Motion Prediction. 43, 45–49, 51, 139

R

RL Reinforcement Learning. 87
RobotSNAP Robot Social Navigation Assessment Platform. iii, iv, clxii, clxiii, 2–4, 33, 34, 37, 40, 41, 43, 50, 64, 86, 89, 91, 99, 105–107, 120, 122, 123, 139
ROS Robotic Operating System. 8, 9, 20, 27, 30, 34, 35, 42, 50, 99, 113

S

SAF Social Angular Field. 86, 91, 96–98, 100, 101, 104, 121, 149–151
SARL Socially Attentive Reinforcement Learning. 15, 42, 45–51, 139
SFM Social Force Model. 36
SLAM Simultaneous Localization And Mapping. 7
ST-Graph Spatio-Temporal Graph. 86, 91, 96–98, 100, 101, 104, 121, 151

T

TEB Timed Elastic Band. 8, 9, 42

V

VO Velocity Obstacles. 7, 42

Foreword

For the writing of this manuscript, we exclusively and punctually used generative artificial intelligence tools (ChatGPT - OpenAI) for tasks of reformulation and translation into English. These tools did not produce any scientific, methodological, or analytical content; all ideas, results, and conclusions are entirely the responsibility of the author.

Pour la rédaction de ce manuscrit, nous avons fait appel à des outils d'intelligence artificielle générative (ChatGPT - OpenAI) exclusivement et ponctuellement pour des tâches de reformulation et de traduction en anglais. Ces outils n'ont produit aucun contenu scientifique, méthodologique ou analytique ; toutes les idées, résultats et conclusions relèvent entièrement de la responsabilité de l'auteur.

Contents

Introduction

Recent advancements in robotics have expanded the presence of robots in our daily lives. From hospitals and restaurants to museums, homes, and industries, robots are increasingly being deployed across a wide range of environments. Many of these robots are mobile, requiring them to navigate in complex and dynamic spaces due to the numerous obstacles present in their environment, often shared with humans. This shift has brought about new challenges, especially those involving Human-Robot Interaction (HRI). Traditionally, mobile robots treated humans as dynamic obstacles to be avoided, focusing only on collision-free navigation. However, this behavior has started to appear inadequate as robots step closer to operating alongside humans.

Thus, the sub-discipline of Human-Aware Robot Navigation (HAN), also known as Social Navigation, evolved out of HRI with the intent to enable robots to avoid humans in ways that respect implicit social norms such as personal space [Hall 1969], direction of movement, and group formation. Because the subject is highly complex, HAN has been studied for several years, and even today, studies are still being conducted to define what social navigation is and its various aspects, such as the work proposed in [Mavrogiannis, Baldini, et al. 2023a]. This shows that relying solely on social norms becomes insufficient, as depending on the context and the robot's mission, these social norms become either changeable or anecdotal. For example, in a context where a robot needs to urgently deliver a medical product, respecting the personal space of humans along the trajectory is no longer the robot's priority. Studies like [Francis et al. 2025] have provided several principles defining what is social navigation, such as proactivity, agent understanding, contextual appropriateness, in addition to five more classic ones related to comfort and safety. Moreover, they have shown that the evaluation of HAN methods is clearly not an easy task. They introduced some guidelines for understanding how to properly evaluate robotic social navigation methods. This need arose due to the numerous works done in the field of HAN and the extent of the different methods used.

In recent years, a wide variety of HAN methods have been proposed, ranging from model-based approaches, which exploit explicit representations of the environment, robot dynamics, and human behavior [Singamaneni and Alami 2020], to learning-based methods, such as supervised learning, Deep Reinforcement Learning (DRL) [Li, Xu, et al. 2019], or more recently Large Language Model (LLM), and even hybrid strategies combining both [Xie et al. 2023]. Each family has distinct advantages—learning-based approaches adapt to complex situations, while model-based approaches offer interpretability and anticipation—but

they also have significant disadvantages, such as high computational costs and manually constructed assumptions for model-based approaches, limited robustness and generalization for learning-based approaches, and challenges striking a balance between efficiency and flexibility for hybrid solutions. Beyond these particular trade-offs, their assessment is a common limitation. Most methods are tested in a small number of highly similar and often simplified environments, frequently omitting realistic aspects such as static obstacles, occlusions in human perception, or realistic human behaviors. This lack of diversity and realism, particularly critical for learning-based methods, severely restricts the fair assessment of robustness and generalizability.

This thesis explores the HAN problem and initially provides a tool for evaluating and comparing HAN methods. As well as a range of new solutions, the aim is to address the issues encountered in the literature. Proposing solutions that can be generalized to any environment while presenting a social aspect that allows for no disruption to humans, with the robot anticipating and making correct decisions.

Contributions

This thesis presents three principal contributions to the field of Human-Aware Robot Navigation (HAN), detailed in the following subsections.

Human-Aware Robotic Navigation Assessment Method

Our first contribution is a systematic evaluation of state-of-the-art HAN approaches to gain insight into their strengths and weaknesses. To do this, we propose Robot Social Navigation Assessment Platform (RobotSNAP) that includes a simulation tool, several scenarios and a multitude of metrics to evaluate any method. Using RobotSNAP, we compared six state-of-the-art solutions over several hundred experiments. The results revealed that the proposed solutions were often good in one or two scenarios, but very poor in others: typically, they succeeded in either sparse or crowded settings, but not both. And they particularly struggle in scenarios requiring the most decision-making and proactivity, such as narrow passages or corners. This highlighted the need for more general and anticipatory navigation strategies

Markov Decision Process Based Social Navigation

Our second contribution is the formalization of the HAN problem as a Markov Decision Process (MDP) - a discrete stochastic control formalism especially suited to sequential decision making under uncertainty.

We first use a navigation graph to define key spatial positions and possible movements of agents in the environment. This helps the robot better understand the state of the environment it is navigating in, and more specifically, the social context in which it operates. Because the graph provides possible future routes for humans, the robot is able to better

predict interactions, thus improving its ability to anticipate and make informed decisions. Solving the MDP requires computing a policy: a plan that tells the robot to execute a selected action in any possible circumstance so as to maximize its long-term expected performance. This is generally to consider short-term reward and long-term impact in making socially aware and cost-effective navigation choices with the goal of having a good HAN strategy.

To address the limitations of this graph representation—namely its dependency on expert input navigation graph and its computational cost, we then proposed a second version of this approach. We have slightly changed the robot's representation of the world by transforming the navigation graph into a polygon map, avoiding excessive discretizations of the world, as it can be the case with a navigation graph, which requires one graph per scenario. We initially explored the use of the Monte Carlo Tree Search (MCTS) algorithm for online decision-making, but found it too slow for real-time navigation. As a result, we designed a heuristic solution based on the polygonal grid, making MDP-based navigation more practical for human-aware contexts.

Deep Reinforcement Learning (DRL) Robust Method

Finally, our third contribution focuses on enhance the adaptability of DRL-based social navigation solutions. Although learning-based approaches can adapt to different situations, they require a new learning phase to achieve good performance in environments not seen during learning. We created a new multi-branch architecture to address this, with each branch concentrating on a distinct facet of social navigation: interactions between humans and robots, the robot itself, and the environment in which it operates. Training was carried out across a large variety of maps and scenarios, using Curriculum Learning (CL) to progressively increase task difficulty and promote more stable, robust learning. This combination promotes more stable learning and produces policies more resilient to environmental variability, and to our knowledge this has never been explored before for HAN.

Thesis Outline

The thesis consists of five chapters. The first two chapters provide an overview of the literature and the performance of existing solutions, while the last three chapters address the modeling, implementation and evaluation of new methods.

Chapter 1 provides context for the thesis, including the basics of robotic navigation and to the HAN problem, providing an overview of the work carried out in the state of the art. It also highlights the limitations regarding the evaluation of HAN methods, motivating the need for more systematic approaches.

Chapter 2 presents our first contribution: the development of Robot Social Navigation Assessment Platform (RobotSNAP), a benchmarking framework for HAN. We describe the choices made regarding its implementation, architecture, and the evaluation tools used. This

chapter also includes an evaluation of off-the-shelf methods available in the literature and reveals the strengths and weaknesses of these methods.

Chapter 3 introduces our second contribution, our own HAN solution. We first present a formalization of the robot environment under a Markov Decision Process (MDP) using a navigation graph, enabling predictive and anticipatory decision-making. Secondly, we extend it to a more general polygon grid representation, thus removing the need for expert-provided graph and supports efficient online resolution.

Chapter 4 presents our third contribution: a versatile DRL-based approach for social navigation. We describe a novel multi-branch architecture that separates different aspects of navigation and details the use of curriculum learning in diverse environments. Preliminary experiments demonstrate interest in this architecture and the use of CL.

Chapter 5 provides a comprehensive evaluation of our contributions. We first evaluate the performance of our methods in simulation, comparing them against state-of-the-art baselines in diverse scenarios using RobotSNAP. We then extend the evaluation to real-world experiments on a robotic platform, highlighting the challenges of transferring solutions from simulation to reality.

Publications

A. Gouguet, A. Karami, G. Lozenguez and L. Fabresse, *Benchmarking Off-the-shelf Human-Aware Robot Navigation Solutions*, in Intelligent Systems Conference (IntelliSys), 2023.

A. Gouguet, A. Karami, G. Lozenguez and L. Fabresse, *A Versatile Heuristic-based Social Robot Navigation Solution Efficient in Sparse and Crowded Environments*, in IEEE International Conference on Tools with Artificial Intelligence (ICTAI), 2025.

L. Naury, A. Gouguet, G. Lozenguez and L. Fabresse, *Communication Isolation For Multi-Robot Systems Using ROS2*, in 40th ACM/SIGAPP Symposium on Applied Computing (SAC), 2025.

Chapter 1

State of the Art: Human-Aware Robot Navigation

Outline of the current chapter

This chapter begins by detailing the fundamental of mobile robot navigation, setting the scene for Human-Aware Robot Navigation (HAN) systems. It subsequently provides a comprehensive state of the art of the leading classes of methods developed for HAN. In this thesis, the term robot refers exclusively to an indoor ground mobile platform, namely an Automated Guided Vehicle (AGV). The chapter ends with an investigation into modern approaches to testing and comparing these methods. This includes the use of quantitative metrics, common scenarios, standard benchmarking tools and user studies. Finally, it provides a summary and introduction to our approach.

1.1 Mobile Robot Navigation

The main goal of mobile robot navigation is to find a feasible path from the present robot position to a desired target while ensuring that no collision occurs. In addition to the feasibility, the navigation process may seek to optimize some performance criteria such as shortest path length, traversal time, or energy consumption. To achieve these objectives, the robot leverages its internal knowledge base and the available computational tools to determine an optimal trajectory. In the following section, we first outline the general navigation pipeline and subsequently review traditional methodologies employed in mobile robot navigation.

1.1.1 Mobile Robot Navigation Pipeline

To calculate an optimal path, the robot must have accurate knowledge of its current position, its surroundings, and at least a basic understanding of environmental topology. As illustrated in Figure 1.1, the robot uses a multitude of sensors on board to capture and analyze the environment. The acquired data is processed and interpreted to update the internal representation of its state, which in turn is used in the navigation module. After calculating the optimal path, the system generates the relevant motor control command required to activate the mobile base and execute the planned path.

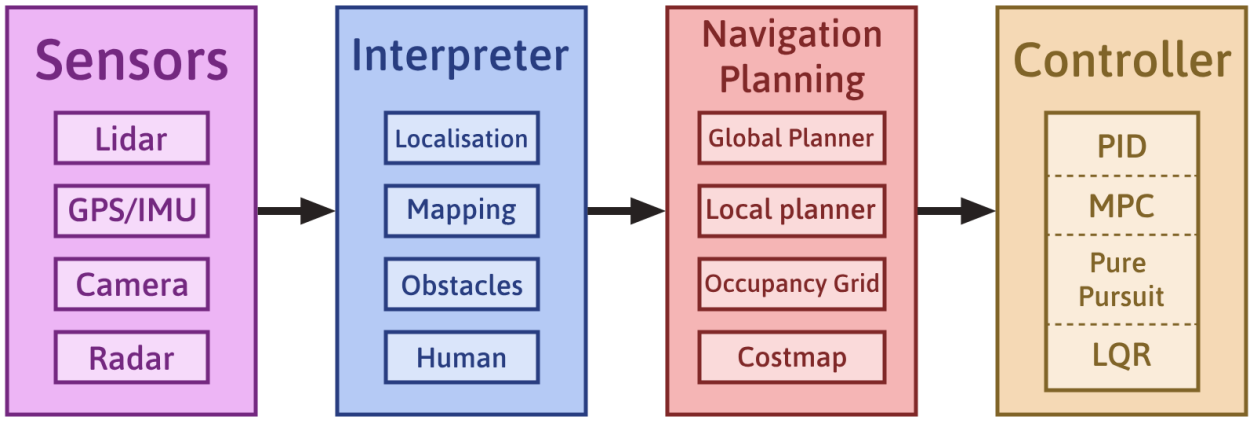


Figure 1.1 – A overview of the four primary modules that make up the robotic navigation pipeline: sensors, interpreter, navigation planning, and controller.

Sensors

This is the first stage of the navigation pipeline and represents the robot's connection to the physical world. The sensors collect raw data from the world to give the robot information about its internal state and external scene (the environment and the elements around the robot). Common sensors are Light Detection And Ranging (LiDAR) for 2D/3D spatial understanding, cameras for vision and object identification, IMUs (Inertial Measurement Unit) for

acceleration and orientation, GPS (Global Positioning System) for outdoor positioning, and wheel encoders for odometry. Additionally, internal sensors such as battery monitors provide crucial information about the robot's power status. These sensors form the foundation of all perception, mapping, and subsequent decision-making.

Interpreter

The interpreter is the intelligence that converts raw sensor data into valuable information. It contains modules for mapping, localization, and perception of the environment. For example, it can help estimate the robot's position in an environment accurately, build or contribute to a map, or both with Simultaneous Localization And Mapping (SLAM) algorithms, and interpret LiDAR or camera data to detect and track obstacles. The interpreter can also recognize and categorize the obstacles into different classes as furniture, human, object, wall... This enables the robot to localize itself, and sense the scene around it, allowing a safe and clever planning within complex real-world surroundings.

Navigation Planning

This is the factor that decides how the robot will navigate to a final destination while avoiding obstacles. It has both global planning, for the computation of a long-term path, and local planning, which gives real-time, dynamically feasible paths that consider near obstacles and the robot's mobility constraints. The planner ensures that the robot not only moves towards its target but also does so safely and effectively, always adjusting as it gets new information from the interpreter.

- **Global Planning** Global planning maps the long-term path from the robot's current location to its destination via an occupancy map, i.e., a map of the accessible environment with static obstacles, as shown in Figure 1.2(A). It returns a collision-free path in the form of a way-point sequence and, thus, tends to be commonly used with A* [Hart et al. 1968], Dijkstra [Dijkstra 1959], or sampling-based approaches such as Rapidly-Exploring Random Tree (RRT) [LaValle 1998] or Probabilistic Roadmap (PRM) [Kavraki et al. 1996]. These differ in operating paradigms, A* and Dijkstra are grid-based, deterministic algorithms, while RRT and PRM are probabilistic algorithms better suited to high-dimensional or complex spaces.

- **Local Planning** Local planning tracks the global path and adaptively adjusts it to suit the robot's present environment. Local planners ensure real-time collision avoidance through sensor feedback (e.g., LiDAR, cameras), and enable the robot to follow the path safely despite evolving environments. A key element of this process is the cost map (Figure 1.2(B)), a grid-like representation of the robot's environment, based on the occupancy map, where each cell is assigned a cost based on the presence and proximity of obstacles. Short-term, feasible trajectories are developed by local planners and generate velocity commands. Methods like Dynamic Window Approach (DWA) [Fox et al. 1997], Velocity Obstacles (VO) [Fiorini et al.

1998], Timed Elastic Band (TEB) [Roesmann et al. 2012; Rösmann et al. 2013], and Model Predictive Control (MPC) [Rosmann et al. 2021] are some of the most popular ones.

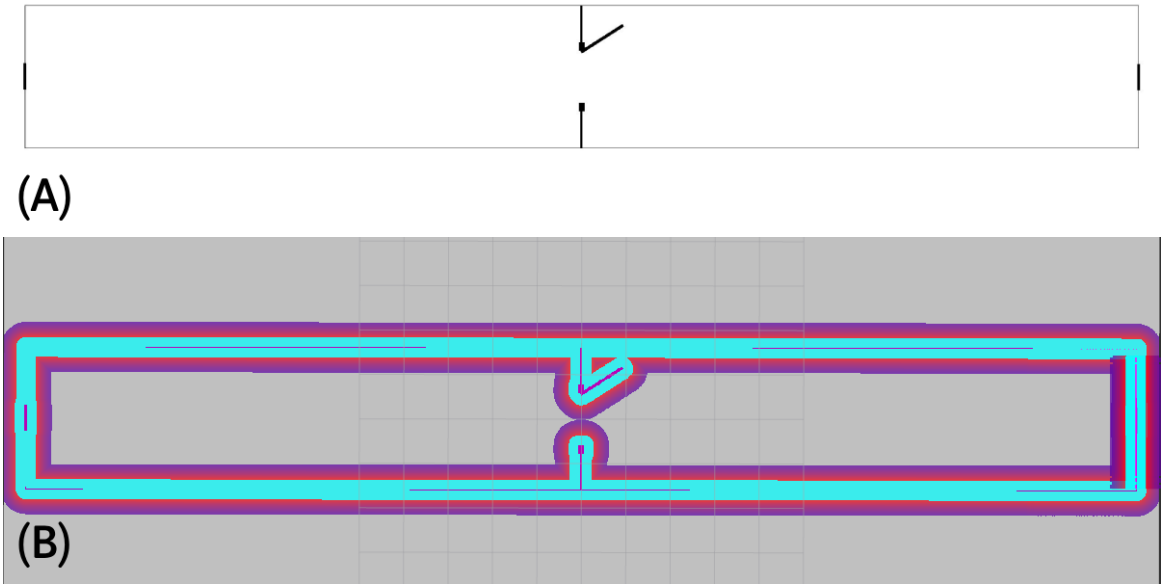


Figure 1.2 – (A) Occupancy grid map of a corridor with a doorway. Black cells represent obstacles and white denotes free space. (B) Corresponding cost map visualized in rviz, where the color gradient represents cost values: blue indicates high cost areas (near obstacles), while purple indicates low cost.

Controller

The controller translates planned trajectory for motor commands. This ensures that the robot follows the right way while compensating for sensor noise, wheel slippage or the faults caused by external forces. Control algorithms such as proportional–integral–derivative (PID), Model Predictive Control (MPC) or pure pursuit with steady and stable speed, reaction loops, allowing the robot to always change the movement. This phase completes the loop between planning and physical execution, which leads the robot to behave in the real world according to the plan.

1.1.2 Mobile Robot Application with ROS2 Middleware

Robotic Operating System (ROS) 2 is a modern, open-source software framework that can construct robust and scalable robot software systems. It is founded on the principles of the original ROS (Robot Operating System) but geared towards addressing performance, real-time, and multi-robot problems. ROS 2 is a collection of tools, libraries, and patterns for easy distributed component communication, which makes it suitable for numerous robotic applications, from research to industry.

Navigation Stack in ROS2

The ROS2 Navigation Stack (Nav2) is a modular and extensible system for supporting autonomous navigation of mobile robots. Navigation is decomposed into independent modules, each dealing with specific activities like path planning, motion control, and behavior management (Figure 1.3). The stack includes a Planner Server to calculate a global path, a Controller Server to execute that path, and a Behavior Tree Navigator to manage the whole navigation process, including fallback actions upon failure. Nav2 is built on top of ROS2 lifecycle management and plugin framework to be scalable to different types of robots and situations. Its robustness is compatible with indoor and outdoor uses and offers a solution to real-time, intelligent robot navigation. In this stack, there are several global planners (several derived from A*), and several local planners already implemented (DWA, TEB, MPC, ...).

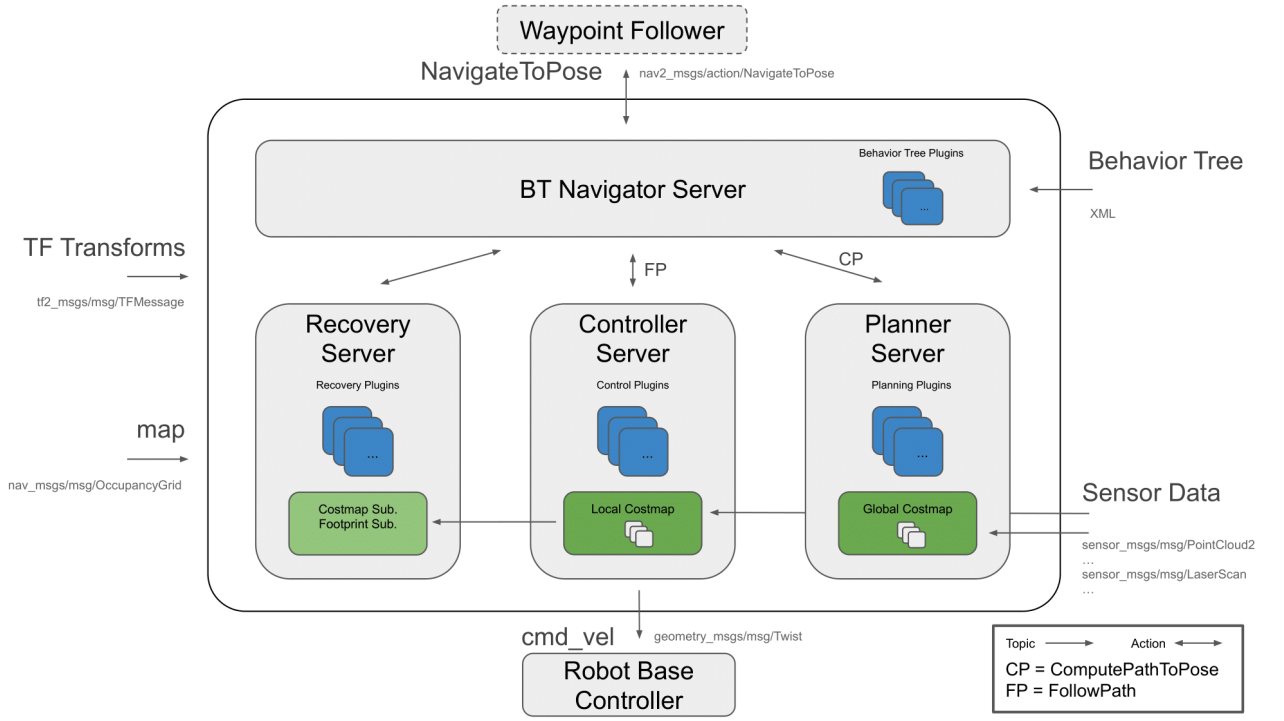


Figure 1.3 – The official architecture of the Nav2 stack[1] composed of 4 functional software modules (BT Navigator Server, Recovery Server, Controller Server and Planner Server).

1.2 Social Mobile Robot Navigation

Human-Aware Robot Navigation (HAN), also called social navigation, seeks to enhance traditional navigation strategies while minimizing disturbances to nearby humans. Consequently, the planning process is formulated as a multi-objective optimization problem, often involving conflicting objectives. As a first abstraction, the humans can be considered as dynamic obstacles by the robots. However, human interactions are guided by established social norms, and there is an implicit expectation that both humans and robots will adhere to these

[1] https://docs.nav2.org/

conventions. Numerous survey papers have been dedicated to the field of social navigation. [Möller et al. 2021] presents an interdisciplinary review encompassing areas such as active vision, robotic navigation, human-robot interaction, and human activity recognition relevant to socially compliant navigation. Additionally, [Cheng et al. 2018] discusses the advantages and disadvantages of different social navigation strategies, including reactive, predictive, model-based, and learning-based approaches. Also, [Xiao et al. 2022] explores machine learning techniques for mobile robot navigation, with a part devoted to approaches for social navigation. Finally, even very recently, new taxonomies have been proposed [Singamaneni, Bachiller-Burgos, et al. 2024] that address HAN's problem from new perspectives. These foundational works inform both the taxonomy we propose and the subsequent comparative analysis.

In this section, we first present a state of the art of HAN, divided into different categories of solutions as presented in Figure 1.4. Presenting each category and subcategory in detail, we provide an overview of existing solutions. Thereafter, we will analyze this same state of the art by presenting the advantages and disadvantages of each solution category from several perspective.

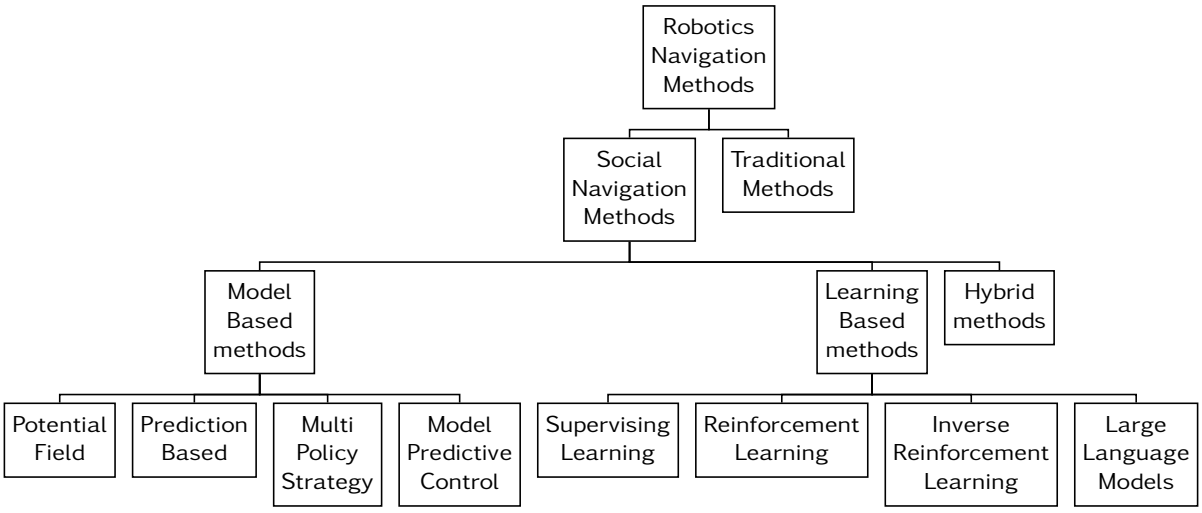


Figure 1.4 – Hierarchical taxonomy of traditional and social robotics navigation methods, including model-based, learning-based, and hybrid approaches.

1.2.1 Model-Based

Model-based methods refer to strategies in which explicit models of the environment, robot dynamics, and human behavior serve as a guide for decision-making. These models give the robot structured knowledge that allows it to make socially acceptable motion plans, reason about its interactions, and predict future states. Particularly in dynamic and populated environments, model-based navigation seeks to improve safety, interpretability, and human acceptance by utilizing formal representations of physical constraints and social norms. This subsection will examine a number of significant model-based approach categories: methods that use potential fields to simulate navigation; Multi-Policy Strategy, which change strategies

depending on the situation and context in which the robot operates; Model Predictive Control (MPC), which optimizes control actions over a finite horizon based on dynamic models; prediction-based techniques, which use predicted information such as human movement, activity detection or emotion detection to inform planning; and, lastly, other model-based techniques that do not fit into the previously mentioned categories but still depend on explicit modeling of the environment, dynamics, or human behaviors.

1.2.1.1 Potential Field

The potential field method guides a robot toward an objective while avoiding obstacles. Using the idea of an artificial force field in the surroundings, this technique applies an attractive force in the direction of the objective while obstacles create repulsive forces. To ensure that the robot's path respects the human's comfort zone, potential field navigation methods model the human as a source of repulsive force.

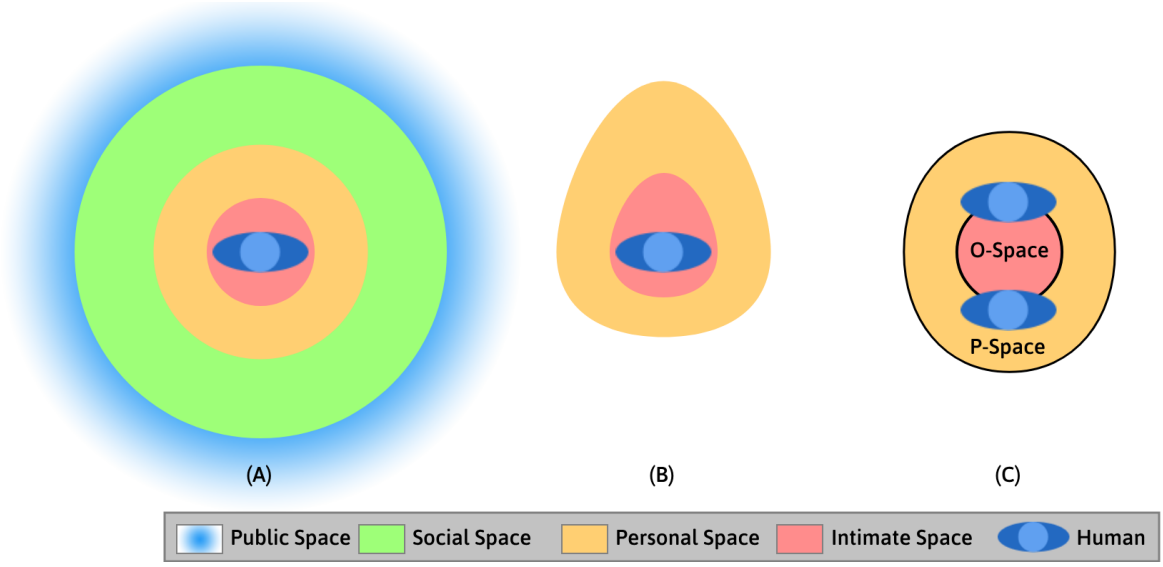


Figure 1.5 – Social zones modeling

One of the earliest approaches to modeling human presence in a shared environment focused on representing human social spaces [Hall 1969; Rios-Martinez, Spalanzani, et al. 2015]. For a stationary individual, this model defines four concentric zones centered around the person, each characterized by specific radii, as illustrated in Fig. 1.5 (a): public space ($> 3.5m$), social space ($> 1.2m$), personal space ($> 0.45m$), and intimate space ($\leq 0.45m$). Since interpersonal distances are highly context-dependent, the boundaries of social spaces, particularly the intimate zone, vary according to the environment. For example, the acceptable intimate distance is typically smaller in confined spaces such as elevators compared to open environments. These spatial boundaries can be dynamically adapted based on the individual's activity; for instance, during locomotion, the social spaces may deform, as shown in Figure Fig. 1.5 (b). Furthermore, in group settings, formations such as F-formations concept [Kendon 1990] including O-spaces are used to model the collective social space. Their configuration depending on the relative positions of individuals within

the group [Melo et al. 2022; Yang et al. 2019], as depicted in Fig. 1.5 (c), or influenced by the nature of the group's activity [Gines Clavero et al. 2021; Abir et al. 2022]. [Rios-Martinez, Spalanzani, et al. 2011] expanded the Risk-RRT by incorporating pedestrians personal space knowledge and potential human interactions. A common way to model these personal space is to use an asymmetric Gaussian function to represent either a personal space of a human or group space [Sousa et al. 2022; Bacchin et al. 2024; Shao et al. 2025]. [Sebastian et al. 2017] proposed a Gaussian Mixture Model to differentiate people's behavior and calculate a social-appropriateness score for each possible trajectories. Then, it chooses the one which maximizes this score, the most relevant and respectful trajectory towards people.

1.2.1.2 Prediction-Based

One of the key challenges in HAN is enabling robots to perceive and interpret human behavior in order to predict future actions or intentions. A fundamental component of this process is the prediction of human trajectories, which allows the robot to anticipate and adapt its motion accordingly. For instance, in addition to employing asymmetric Gaussian functions, [Kollmitz, Hsiao, et al. 2015] incorporates trajectory prediction to avoid intruding into personal spaces and to adhere to social conventions, such as yielding the way or avoiding obstruction of a human's path. [Rudenko et al. 2018] introduces a novel approach for predicting human movement by integrating Markov Decision Process (MDP)-based planning with stochastic policies and a social forces model to account for interactions between individuals and groups. Along same way, [Rios-Martinez, Renzaglia, et al. 2012] proposes an adaptive stochastic optimization (CAO) method based on data from sensor's robot, making it possible to both anticipate human and taking into account of complex dynamic. Activity detection can also be an asset to increase the efficiency of robotic social navigation methods [Charalampous et al. 2023]. [Bilen et al. 2024] uses information about human emotions (happy, neutral, angry) to adjust the personal zones of humans, thus changing the robot's trajectory. [Mavrogiannis and Knepper 2019] presents a navigation planning for dynamic environments composed of a probabilistic inference mechanism predicting the collective avoidance strategy, based on the observation of past behaviors. Even if the prediction models provide a lot of social information to help the navigation of the robot, it requires a significant computational cost, especially when it comes to dense crowded environments. With solutions that add too many cost maps, the robot often cannot access its target without encroaching on the cost space representing the personal space of humans (present or future). In this case, the robot will stop until access is freed, this situation is called "Freezing Robot Problem" [Trautman et al. 2010]. This situation often arises from each agent focusing on its own individual consideration, with little regard for interactions with nor consideration of other agents in the environment.

1.2.1.3 Multi-Policy Strategy

One possible approach is to adapt navigation strategies based on the robot's context and surrounding environment. A method that selects the policy with the highest expected utility

from a predefined set of behaviors (go, follow, stop for example) has been explored [Mehta et al. 2016; Mehta et al. 2017]. In a similar spirit, Human-Aware Timed-Elastic Band (HATEB) [Khambhaita et al. 2019; Singamaneni and Alami 2020] introduces a socially aware navigation framework that uses an asymmetric Gaussian cost function and a modular architecture with three switchable planning modes, selected according to environmental conditions. One of these modes leverages the elastic band concept, incorporating human trajectory prediction and cooperative behavior between the robot and nearby humans.

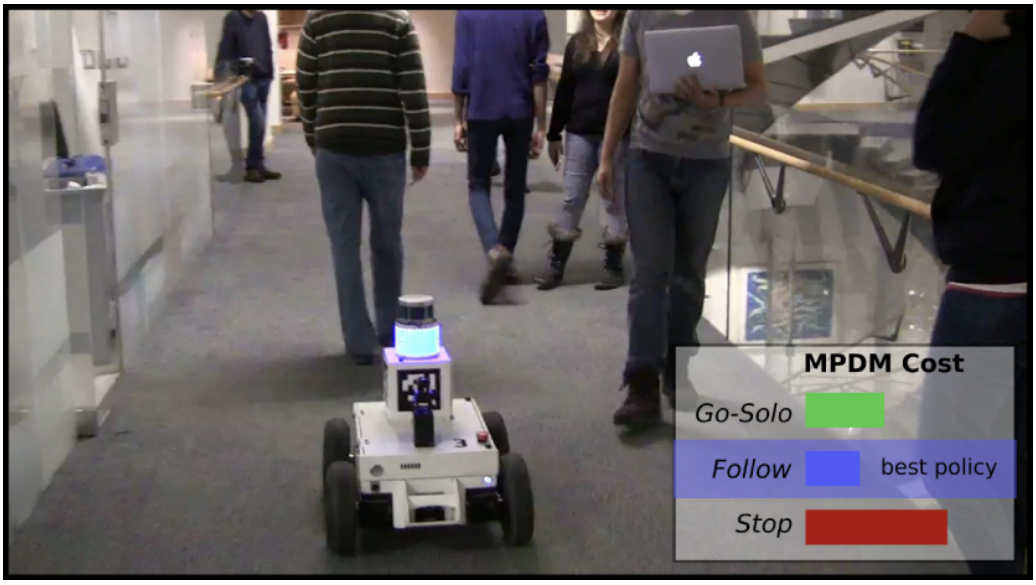


Figure 1.6 – Multi-Policy Decision Making example from [Mehta et al. 2016]

1.2.1.4 Model Predictive Control (MPC)

Model Predictive Control (MPC) computes optimal control commands by directly incorporating predicted intentions of dynamic obstacles, allowing the robot to proactively and smoothly avoid collisions. A nonlinear Model Predictive Contouring Control (MPCC) framework [Brito et al. 2019] has been extended to incorporate static map information by computing convex free-space regions online, while modeling dynamic obstacles as ellipsoids with provable collision bounds. To account for human motion, a bilevel nonlinear MPC formulation [Samavi et al. 2025] jointly solves for robot and crowd trajectories in closed-loop, where each human is modeled using an Optimal Reciprocal Collision Avoidance (ORCA) scheme embedded as a constraint in the robot's local planner. For path tracking, MPC has also been adapted using a curvature-based online reference update strategy, combined with an enhanced Dynamic Window Approach (DWA) for improved local planning performance [Wang, Wang, Xie, et al. 2025].

1.2.1.5 Miscellaneous Model-Based Approaches

Other forms of model-based methods exists but less represented in the literature such as game theory-based methods or static braids. In game theory methods, multi-robot navigation

and human-robot scenarios are designed as a possible cooperative game of two players [Le et al. 2023]. To improve human acceptance, game-theoretic path planning methods have been proposed [Galati et al. 2022], where non-cooperating game theory are used to design navigation behavior of multiple humans in populated environment. The Bayesian Recursive Nash Equilibrium (BRNE) [Muchen Sun et al. 2024] expands this line of work by enabling stochastic planning that estimates the pedestrian path, which improves navigation in a crowded environment. Interactions between agents, such as maneuvering passage, are mathematically referred to as rotational behaviors grounded in topological invariance [Mavrogiannis, Balasubramanian, et al. 2022], which is preferred by a socially appropriate robot track with cost functions. In addition, the static braid theory has been used to develop movement planners who are able to clearly argue about the topological structures that develop multi-agent interactions in a dense human environment [Mavrogiannis, Alves-Oliveira, et al. 2022].

1.2.2 Learning-Based

Traditional approaches have largely relied on rule-based systems or optimization techniques that encode predefined social behaviors, often falling short in capturing the complexity and variability of human interactions. In recent years, learning-based methods have emerged as a promising alternative, enabling robots to infer navigation strategies directly from data and adapt to complex, dynamic social contexts.

1.2.2.1 Supervised Learning

Regarding supervised learning methods, researchers use expert demonstrations for the training of social navigation methods. This is because humans are naturally accustomed to crowd movements and therefore have good performance in navigation taking others into account. [Pfeiffer et al. 2017] provided a target-oriented end-to-end robot navigation using 2D laser data as input and steering commands as output. They used an existing motion planner to generate expert demonstrations and showed the possibility to transfer the learned motion planner to unseen environments whether in simulation or in real-world. Similarly to [Pfeiffer et al. 2017], [Pérez-Higueras, Caballero, et al. 2018a] used expert demonstration as learning data, but the intended purpose is different. Indeed, [Pérez-Higueras, Caballero, et al. 2018a] used a Fully Convolutional Neural Network to learn a cost map that works as a prediction. [Groshev et al. 2018] used a deep neural network to perform a reactive policy, i.e., mapping a state of the environment to an action. Learning social cost functions is also possible, as shown by [Eirale et al. 2024] by teaching the robot to queue up. The work in [Martinez-Baselga et al. 2024] presents an innovative motion planner that uses topological paths and a deep neural network trained on real-world human motion data to select navigation strategies that align with human behavior, ensuring socially intelligent and context-aware navigation.

1.2.2.2 Reinforcement Learning

Reinforcement learning solutions are useful to generate human-like robot navigation. As mentioned, the prediction task often required expensive computation, [Chen, Liu, Everett, et al. 2017] proposed to offload the online computation (prediction) to an offline training process. [Chen, Everett, et al. 2017] pointed out that it is easier to specify what the robot should not do than what it should do. They used deep reinforcement learning to achieve robotic social navigation that doesn't violate social rules. Contrary to previous works, [Everett et al. 2018] assumed that agents do not follow particular behavioral rules. The proposed method (CADRL) allows collecting observations of an arbitrary number of other agents for learning. By rethinking pairwise interactions with a self-attention mechanism and jointly modeling Human-Robot as well as Human-Human interactions, [Chen, Liu, Kreiss, et al. 2019] propose a social navigation method Socially Attentive Reinforcement Learning (SARL) with the ability to better anticipate humans mouvements using the deep reinforcement learning. [Li, Xu, et al. 2019] has improved the SARL algorithm (SARL*) by introducing a dynamic local goal and a map-based safe action space to avoid real-world distance limitation or neglect obstacles other than humans. The method proposed by [Chen, Hu, et al. 2020] models inter-agent relationships by leveraging latent feature representations of all agents and employs a Graph Convolutional Network (GCN) to capture higher-order interactions within each agent's state encoding. This capability to anticipate human motion dynamics facilitates multi-step lookahead planning, thereby enabling the consideration of the temporal progression of human crowd behavior. [Xie et al. 2023] proposes a learning-based control policy that generalizes to novel environments, enabling autonomous robot navigation in cluttered and crowded spaces. The policy combines recent LiDAR history, pedestrian kinematics, and a sub-goal to compute steering and velocity commands.

1.2.2.3 Inverse Reinforcement Learning

There exist some limitations for reinforcement learning. Indeed, with reinforcement learning, a handcrafted cost function is needed, and in other words, such solutions requires expertise in robotics, sensing and motion planning [Wulfmeier et al. 2017]. [Kim and Pineau 2016] uses an inverse reinforcement learning module to learn an expert's behavior through a set of demonstration trajectories and represent it as a cost function that respects social rules. [Pérez-Higueras, Caballero, et al. 2018b] combined inverse reinforcement learning with Rapidly-exploring Random Trees (RRT*) to learn the RRT*'s cost function from demonstrations. [Kollmitz, Koller, et al. 2020] employed inverse reinforcement learning to learn from physical human-robot interaction, so the robot learns social rules and can adjust his behavior according to present user's preferences. [Kathuria et al. 2025] introduces Smooth Maximum Entropy Deep Inverse Reinforcement Learning (S-MEDIRL), an algorithm that generalizes beyond expert demonstrations to infer scene navigability from few-shot data. The agent learns cost maps by reasoning over trajectory information and scene geometry.

1.2.2.4 Large Language Model

Recent progress in robotics and Large Language Model (LLM) has intensified interest in human-robot collaboration and embodied intelligence. While classical deep reinforcement learning methods combining HRI and path planning show strong performance, they often lack adaptability to novel scenarios. LLM offer potential for robust navigation via commonsense reasoning, though existing frameworks predominantly rely on centralized control. Some efforts focus on bridging high-level intentions with accurate low-level execution. [Elnoor et al. 2025] presents an attention-based distillation approach that transfers socially compliant navigation knowledge from a large Vision-Language Model (VLM) and a pretrained vision-action model into a compact transformer-based policy. [Wang, Obi, et al. 2025] proposes a decentralized multi-agent LLM actor-critic framework for social robot navigation, where parallel LLM actors—each modeling different robot configurations—independently generate control commands.

1.2.3 Hybrid Solutions

Hybrid approaches merge the advantage of model-based and learning-based approaches. Hybrid approaches attempt more robust, adaptive, and social navigation in dynamic environments by merging pre-specified human behavior models with machine learning-based methods. [Pokle et al. 2019] proposes a hybrid navigation system that integrates hierarchical planning with learning-based components, combining a global planner for optimal path computation with a deep local planner and velocity controller for real-time motion commands. [Gil et al. 2021] introduces two human-aware navigation tasks—social robot navigation and robot accompaniment—by integrating machine learning with the Social Force Model (SFM) to enable socially compliant behaviors. [Han et al. 2025] presents Deep Residual Model Predictive Control (DR-MPC), which fuses MPC with model-free deep reinforcement learning to enable efficient and safe policy learning from real-world crowd navigation data, addressing challenges related to data efficiency and initial safety.

1.2.4 Analysis

Having seen the different categories of solutions for HAN, this section presents a general analysis of these categories. The table 1.1 summarizes the advantages and disadvantages of each category of navigation solutions based on the following properties: Safety, Extensibility of Social Information, Computation Cost, Data Requirements and Adaptability. These five criteria were chosen as they capture the core challenges of real-world socially-aware navigation: ensuring safe and appropriate behavior, handling diverse and evolving social norms, operating under resource constraints, functioning with limited data, and generalizing to new environments.

In Table 1.1, we assign to each subcategory a qualitative score for each criterion, using a scale from -- to ++ to denote very poor to excellent performance respectively. We provide

here, a detailed criterion-by-criterion analysis. This analysis is based on a wide review of the literature, including survey works [Kruse et al. 2013; Cheng et al. 2018; Möller et al. 2021; Xiao et al. 2022; Mavrogiannis, Baldini, et al. 2023b], detailed examination of representative navigation methods, and our own comparisons across categories.

Category / Sub-category	Safety	Extensibility of Social Info	Computation Cost	Data Hungry	Adaptability
Model-Based					
Potential Field	+	--	++	++	-
Prediction-Based	++	+	-	+	+
Model Predictive Control	++	-	--	+	+
Multi Policy Strategy	+	-	-	+	+
Learning-Based					
Supervised Learning	-	+	+	-	+
Reinforcement Learning	-	+	-	-	+
Inverse Reinforcement Learning	-	++	-	-	+
Large Language Models	--	++	--	--	++
Hybrid	++	++	-	-	++

Table 1.1 – Comparison of human-aware robot navigation category and their sub-categories across key evaluation criteria.

Safety. *Is the generated robot behavior safe? (--: Absolutely not; ++: Absolutely)*

Navigation safety is the degree to which a method consistently avoids physical and social harm. Model-based approaches, most notably Model Predictive Control (MPC), work well by computing and optimizing explicitly future trajectories, though reactive methods like Potential Fields are inadequate in dense or intricate scenarios. Multi-Policy Strategies improve safety by context-dependent switching but still rely on model quality. Learning-based methods, and Reinforcement Learning in particular, have weak safety guarantees due to uncontrolled behavior and unsafe exploration. Supervised Learning is safer but limited by data quality, and Inverse Reinforcement Learning is safer provided norms are properly represented [Mirsky et al. 2024]. Large Language Models currently pose the greatest safety challenges due to opaque reasoning. Hybrid methods, blending model-based constraints with learning flexibility offer the best safety-performance tradeoff [Selim et al. 2022].

Extensibility of Social Information. *Is it easy to add social information to be taken into account in the solution? (--: Difficult to add or remove a social aspect; ++: Easy to add or remove a social aspect)*

Extensibility measures how well a method extends to complex or shifting social norms. Model-based methods do poorly, as they are based on simplified human models and hand-crafted rules. Potential Fields are particularly limited, with no awareness of subtle cues.

MPC and Prediction-Based methods can capture simple social behaviors but require major redesigns for more complicated interactions. Multi-Policy Strategies achieve limited benefits by switching between pre-defined social behaviors. On the other hand, learning-based methods are very extensible. Supervised Learning learns social patterns from demonstration, and RL and IRL learn to contextualize and infer implied norms. IRL is particularly good at picking up unwritten rules from behavior. LLMs lead the way here, interpreting and reasoning about social dynamics using large, subtle data, though real-time deployment is challenging. Hybrid methods shine by integrating learned social awareness—either via IRL or LLMs—into structured planners, combining flexibility and faithful execution for excellent extensibility.

Computation Cost. *Does this require significant computing costs? (--: Complex to have real-time; ++: Real-time ready)*

Computation cost reflects both the cost of training and online execution. Model-based methods differ in their intensity: Potential Fields are lightweight and fast, while Prediction-Based and especially MPC methods tend to be more intense owing to online optimization [Pintos Gómez de las Heras et al. 2023]. Multi-Policy Strategies can perhaps somewhat counter runtime cost by switching between precomputed actions, though complexity grows with the number of policies. Learning-based methods tend to be computation-intensive, especially during training. Supervised Learning is efficient for inference but RL and IRL are computationally intensive for iterative learning. LLMs are most computationally intensive, primarily for real-time use, typically employing high-performance hardware or cloud computing resources. Hybrid solutions employ both costs. Offline learning or model simplification are feasible, but uniting learning and planning incurs runtime expense. However, well-considered design makes these systems capable of supporting efficiency and performance for deployment in the real world.

Data Requirements. *Does this require a lot of data? (--: Need a lot of data; ++: Very little data required)*

Data Requirements is the size of data a method needs in order to function well. Model-based approaches are data efficient and rely on physical principles or very simple heuristics—Potential Fields, MPC, and even Prediction-Based methods only need very little data except for simple motion models. In contrast, learning-based methods often have high data requirements [Guillen-Ruiz et al. 2023]. Supervised Learning requires large labeled datasets, while RL and IRL require large-scale interactions or demonstrations, with IRL needing in particular rich behavioral information. LLMs are most data-demanding, relying on large pretraining corpora and fine-tuning. Hybrid methods take a middle road, requiring medium data for training learned components but relying on model-based planning to reduce overall dependency. This allows them to remain functional even with minimal or absent data sets.

Adaptability. *Does the solution adapt well to the new environment? (--: Does not work in an environment never encountered; ++: Can operate in an environment never encountered before)*

Adaptability assesses a method's robustness in unfamiliar settings, including unseen environments or new social interactions. Model-based approaches, while offering structured decision-making, often rely on fixed assumptions about dynamics or interactions, which can limit their performance in unfamiliar settings. While traditional model-based methods rely on fixed assumptions, variants incorporating online estimation or learned models (e.g., trajectory prediction) introduce adaptability, though they remain less flexible than end-to-end learning approaches. Potential Fields react to changes but cannot learn. Prediction-Based and MPC approaches adapt to environmental changes but not shifting social norms in the absence of periodic updating. Multi-Policy Strategies offer limited adaptability, bounded by pre-defined actions. In principle, learning-based methods offer a higher degree of flexibility, but there are trade-offs. While RL and IRL methods can adapt themselves to new environments through retraining/fine-tuning [Fahmy et al. 2024], pre-trained policies typically underperform. Adapting such a model requires time (to collect additional data) and computation (to train the new model), which restricts real-time adaptability. Supervised Learning models generalize well provided the training distribution was not too narrow, but remain sensitive to domain shift. LLMs demonstrate remarkable flexibility because they can adapt through prompting or continue to engage in training-fine-tuning allowing for a flexible form of reasoning in a new situation, though they have not yet demonstrated full feasibility in real-time embodied deployment. Hybrid methods exhibit adaptability to social rules and environmental change. Learning methods enable adaptation to new data, while model-based planning ensures consistency—allowing both to be applied effectively in diverse and dynamic social environments.

Summary

No socially aware robot navigation method optimally meets all of the most critical requirements—safety, social extensibility, computational cost, data dependency, and adaptability. Each class of methods—model-based, learning-based, and hybrid—has specific trade-offs well suited to particular goals. Model-based approaches are highly safe and reliable, with very good performance in well-understood environments but rigid to adapt to rich social conventions and, in their current form, hard to modify without redesign. Their computational cost varies, with some being light and others heavy. Learning-based methods excel at social nuance capture and generalizability to new situations but require big data, large computation, with weaker safety guarantees, limiting their use in safety-critical settings. Hybrid methods take the advantage of both, combining learned social representations with constrained planners to attain the best of safety, adaptability, and efficiency. A well-crafted hybrid solution attains the best overall performance for deployment in dynamic social settings.

1.3 Assessment of Human-Aware Robot Navigation

As discussed in the previous section, a wide range of approaches have been proposed for robotic navigation in dynamic environments. Despite their methodological differences, a systematic framework for evaluating and comparing these methods is essential. Consequently, significant research has focused on assessing HAN through diverse metrics, scenarios, benchmarks, and questionnaires [Kruse et al. 2013; Gao et al. 2022; Wang, Chan, et al. 2022; Karwowski et al. 2024; Francis et al. 2025]. This section presents an overview of the key tools and methodologies used to evaluate social robotic navigation approaches, beginning with a description of the relevant metrics. We then identify the scenarios that are generally applied to evaluate HAN methods, followed by an overview of current benchmarks uniting both metrics and scenarios. We then conclude with an overview of user studies in the literature, and finally address the main challenges of evaluating social robotic navigation.

1.3.1 Metrics

The primary tools for evaluating HAN are quantitative metrics, which can be broadly categorized into two groups: performance metrics and social metrics. These categories reflect the dual objectives of optimizing navigation efficiency and ensuring socially appropriate behavior. Table 1.2 summarizes the most commonly used metrics in the literature, organized according to these two evaluation dimensions and detailed in the following of this section. Each row corresponds to a specific metric, with the columns providing the metric name, a short label used in plots or code, the unit of measurement, and a brief description of its purpose.

Performance Metrics

Classic performance metrics were used to compare different solutions like DWA and TEB in ROS [Cybulski et al. 2019], [Naotunna et al. 2020]. Those metrics are: the length of the path, the time required, the average speed and acceleration, whether the robot has reached its goal, the final distance between the robot and the goal, and finally the number of collisions. In addition to these very common metrics, we can find metrics like path efficiency [Mavrogiannis, Hutchinson, et al. 2019] and path irregularity [Guzzi et al. 2013]. The efficiency of the path consists in comparing the length of the shortest path with the length of the path made by the robot. The irregularity corresponds to the number of unnecessary turns performed by the robot. Knowing the number of seconds the robot did not move is also interesting, because it can be due to several situations: a period of calculation time, a moment when the robot was blocked by an obstacle or a situation where the robot gave priority to a human. However, it is necessary to observe the path of the robot in more detail to detect the exact reason.

	Metric	Short Name	Unit	Description
Performance Metrics	Sucess	S	bool	A binary variable indicating if the robot achieves the goal.
	Path Length	PL	m	The trajectory's length.
	Success weighted by path length	SPL	-	Success weighted using normalized inverse path length
	Path Efficiency	PE	-	The ratio of the distance between a segment's endpoints to the length of the path which an agent actually followed.
	Path Irregularity	PI	rad/m	Path irregularity quantifies how much turning is unnecessary along an entire path traveled by a robot.
	Time to Reach Goal	T	s	The duration from task assignment to completion.
	Error Distance to Target	E	m	The distance between the robot's final position and the target position.
	Total Rotation	R	deg	The total rotation of the trajectory.
	Time Not Moving	TNM	s	The duration when the robot is stationary.
	Collision with Static Obstacle	C	-	The count of collisions with static obstacles.
	Velocity	V	m/s	Velocity during the navigation, linear and angular speed (average, min, max).
	Acceleration	A	m/s^2	Acceleration during the navigation, linear and angular acceleration (average, min, max).
	Movement Jerk	J	m/s^3	Movement jerk (the second-order derivative of the speed), linear and angular jerk. (average, min, max)
Social Metrics	Violation of Human Personal Space	VPS	-	The number of times the robot violates human personal space.
	Violation of Human Personal Space While the Robot is Stationary	VPSS	-	The number of times the robot violates human personal space while stationary.
	Time of Robot Violate Human Personal Space	TVPS	s	The duration when the robot is within human personal space.
	Minimum Distance to Human	DH	m	The closest distance to a human observed throughout the trajectory.
	Collision with Human	HC	-	The count of collisions involving humans.
	Minimum Time-to-Collision	TTC	m	The minimum time until potential collision with a human agent, assuming linear trajectories for both.
	Clearing Distance	CD	m	The closest distance to an obstacle observed throughout the trajectory.

Table 1.2 – List of metrics for evaluating human-aware robotic navigation methods. The first part corresponds to the so-called performance metrics and the second concerns the social quality of navigation.

Social Metrics

There are also metrics that assess the robot's human-awareness. A metric that reflects this well is the number of times the robot will strongly disturb a human. The easy way is to consider that a human is disturbed when a robot enters his personal space [Hall 1969] (distance less than 1.2 meters) like described in section 1.2.1.1. It is possible to differentiate 2 versions of the metric: number of times the robot will violate a human's personal space and the same number during which the robot remains stationary. If the robot is not moving and the distance drops below 1.2 meters, it can be assumed that this does not disturb the human, as it was their choice to approach the robot. In addition to the number of times the robot has disturbed a human, we can measure the degree of disturbance by measuring the time duration the robot has remained in the personal space of the human and also the minimum distance between the robot and human during the navigation. To capture more detail, we distinguish between collisions with static objects and those with humans. There are also metrics comparing robot behavior to human behavior; Average Displacement Error [Pellegrini et al. 2009] is the computation of a mean difference between a predicted trajectory and a human trajectory. In estimating the socially conscious navigation method, the measure is used to compare the path produced by the navigation method with respect to a human path [Gao et al. 2022]. And Final Displacement Error [Alahi et al. 2016] quantifies the distance between the human's final position and that of the estimated trajectory at each time step. These are usually utilized to quantify the performance of the human prediction algorithms within HAN, rather than the robot's performance. Researchers have also proposed innovative metrics from a human perspective [Singamaneni, Favier, et al. 2023], defining cost metrics that correspond to human feelings (fear, panic, react, shock, visibility). Unfortunately, new metrics like these are not widely used in method evaluations in recent literature.

Metrics Analysis

We now evaluate the previously introduced metrics to understand their impact on the two key criteria: performance and social interaction. The table 1.3 illustrates a classification of both social and performance measures commonly utilized in measuring socially-aware robot navigation, assessed across four dimensions: interpretability, efficiency, social awareness, and safety. We assess each metric by addressing the following questions:

- **Interpretable.** Is the metric easily interpretable?
- **Efficiency.** Does the metric enable the evaluation of the navigation method's effectiveness?
- **Socially Aware.** Does the metric assess the robot's level of social awareness?
- **Safety.** Does the metric give information about the safety of the navigation method?

	Metric	Interpretable	Efficiency	Socially Aware	Safety
Performance	Sucess	++	++	--	--
	Path Length	+	++	-	--
	Success weighted by path length	+	++	-	--
	Path Efficiency	+	++	-	-
	Path Irregularity	+	++	+	-
	Time to Reach Goal	++	++	-	--
	Error Distance to Target	++	+	--	--
	Total Rotation	-	+	--	--
	Time Not Moving	+	-	-	--
	Collision with Static Obstacle	++	-	-	++
	Velocity	+	++	--	+
	Acceleration	+	+	--	+
	Movement Jerk	+	-	-	+
Social	Violation of Human Personal Space	++	--	++	++
	Violation of Human Personal Space While the Robot is Stationary	-	--	+	++
	Time of Robot Violate Human Personal Space	+	--	++	+
	Minimum Distance to Human	++	--	++	++
	Collision with Human	++	-	++	++
	Minimum Time-to-Collision	-	--	++	++
	Clearing Distance	+	--	-	++

Table 1.3 – Classification of metrics presented in table 1.2 under four criteria

The table assigns values ranging from --, indicating that the criterion does not answer the corresponding question at all, to ++, indicating that the metric answers the question very efficiently.

Performance-oriented metrics such as Success, Path Length, Path Efficiency, Path Irregularity, and Time to Reach Goal are typically efficient and easy to interpret, but they fail to capture the subtleties of social interaction. In particular, Path Efficiency is a measure of how directly a robot is moving towards its goal, and Path Irregularity is a measure of the smoothness and naturalness of movement—both of which are critical in measuring the realism and effectiveness of navigation behavior.

Conversely, socially-relevant measures like Violation of Human Personal Space, Minimum Distance to Human, and Collision with Human provide critical information about the robot's

understanding of and respect for human-centred norms and therefore have a significant contribution to safety and acceptability. Intermediate measures like Velocity, Acceleration, and Movement Jerk are a compromise: they relate to efficiency and smooth motion, but carry limited interpretability toward social acceptability.

Generally, measures of performance are useful in assessing accomplishment of tasks and robust control, while social measures are crucial to ensure robot navigation satisfies human belief and comfort in shared spaces. It must be pointed out that in current literature, the most frequent reported measures are the success rate, the number of collisions, and the minimum human distance, revealing an interest in both task success and elementary safety factors.

1.3.2 Scenarios

Robot social navigation addresses all plausible scenarios in which a robot may be placed in environments including humans. As it is not possible to anticipate all the situations a robot may face when it is deployed, some common recurring cases have been outlined in the literature. Table 1.4 provides a non-exhaustive classification of the scenarios in four increasing levels of complexity: traditional, sparse, crowded, and interpersonal scenarios. Each scenario includes its name, description, the topology of the frequently used environment, and the behavior of the human within the scenario. A visual of these scenarios is also shown in Figure 1.7.

- Traditional scenarios : Traditional scenarios generally applied in assessing classical robotic navigation methodologies. In such scenarios, the robot navigates in human-free environments with objectives of maximizing performance measures like travel time and path length. These tend to involve navigation around static obstacles.

- Sparse scenarios : Sparse situations are typical situations of everyday life with sparse presence of humans and specific environmental geometries. Here, the robot must balance goal-oriented navigation with the need for human comfort awareness. Some prototypical examples are head-on situations in narrow passageways, navigating intersections, passing pedestrians on the outside, or passing doorways.

- Crowded scenario : Crowded scenarios include spaces with dense human presence, generally in open space environments. These conditions include dynamic and complex patterns of human flow, such as bidirectional or cross-traffic flows, circular crowd formations, or casual social gatherings, which are very challenging to navigation planning.

- Interpersonal scenario : Interpersonal missions require the robot to carry out close-distance interaction or sustained social behavior with an individual or more than one human. Typical functions in this realm include monitoring someone, walking with or guiding humans, trailing an individual as they moves around, or social approaching of groups.

	Name	Scenario Description	Topology	Human Behavior
Traditional Scenario	Static Obstacle Avoidance	A navigation inside environment with static obstacle (walls, furniture, shapes, ...) without any other agents.	All	None
Sparse Scenario	Frontal Approach	A frontal engagement between pedestrian and robot.	Corridor / Open Space	Point A to Point B
	Overtaking	The robot is positioned behind the pedestrian and, being faster, needs to overtake.	Corridor	Point A to Point B
	Intersection	A robot and a human cross paths at an intersection.	Junction	Point A to Point B
	Corner	A crossing at a corner where both agents detect each other only at the last moment.	Corner	Point A to Point B
	Narrow Passage / Doorway	A head-on encounter in a narrow passage that allows only one agent to pass at a time.	Corridor	Point A to Point B
Crowded Scenario	Parallel Crowd	The robot navigates through a crowd going in the same direction as it.	Open Space	Spot A to Spot B
	Perpendicular Traffic	The robot navigates in a crowd going in a direction perpendicular to it.	Open Space	Spot A to Spot B
	Circle Crowd	The robot must cross a circular crowd of humans.	Open Space	Point A to Point B
	Crowd Navigation	The robot navigates in a given environment with a lot of humans.	Given environment	Hangs out
Interpersonal Scenario	Following Human	The robot follows a designated human from behind.	All	Point A to Point B
	Leading Human	The robot leads a human to a given destination.	All	Following Robot
	Accompanying Human	The robot accompanies a human to a given destination (side by side).	All	Following Robot side by side
	Joining Group	The robot joins a group of static humans present in the environment.	All	Static

Table 1.4 – List of scenarios for evaluating human-aware robotic navigation methods. The first part corresponds to traditional scenario, the second concerns the sparse scenario with few humans involving a need for advanced decision-making, the third gives the crowd scenarios and the last one describe the interpersonal scenarios between the robot and humans.

Moreover, these scenarios are designed to assess various facets of the robot's navigation behavior. For instance, in conventional tasks like avoiding static obstacles, the emphasis will be placed on the robot's performance. In contrast, scenarios involving doorways or intersections will highlight the method's capacity to prioritize human movement. As described in [Francis et al. 2025], it is possible to expand certain scenarios such as narrow passages or intersections by allowing the robot and humans to communicate their intentions (allow passage, force passage) explicitly using gestures.

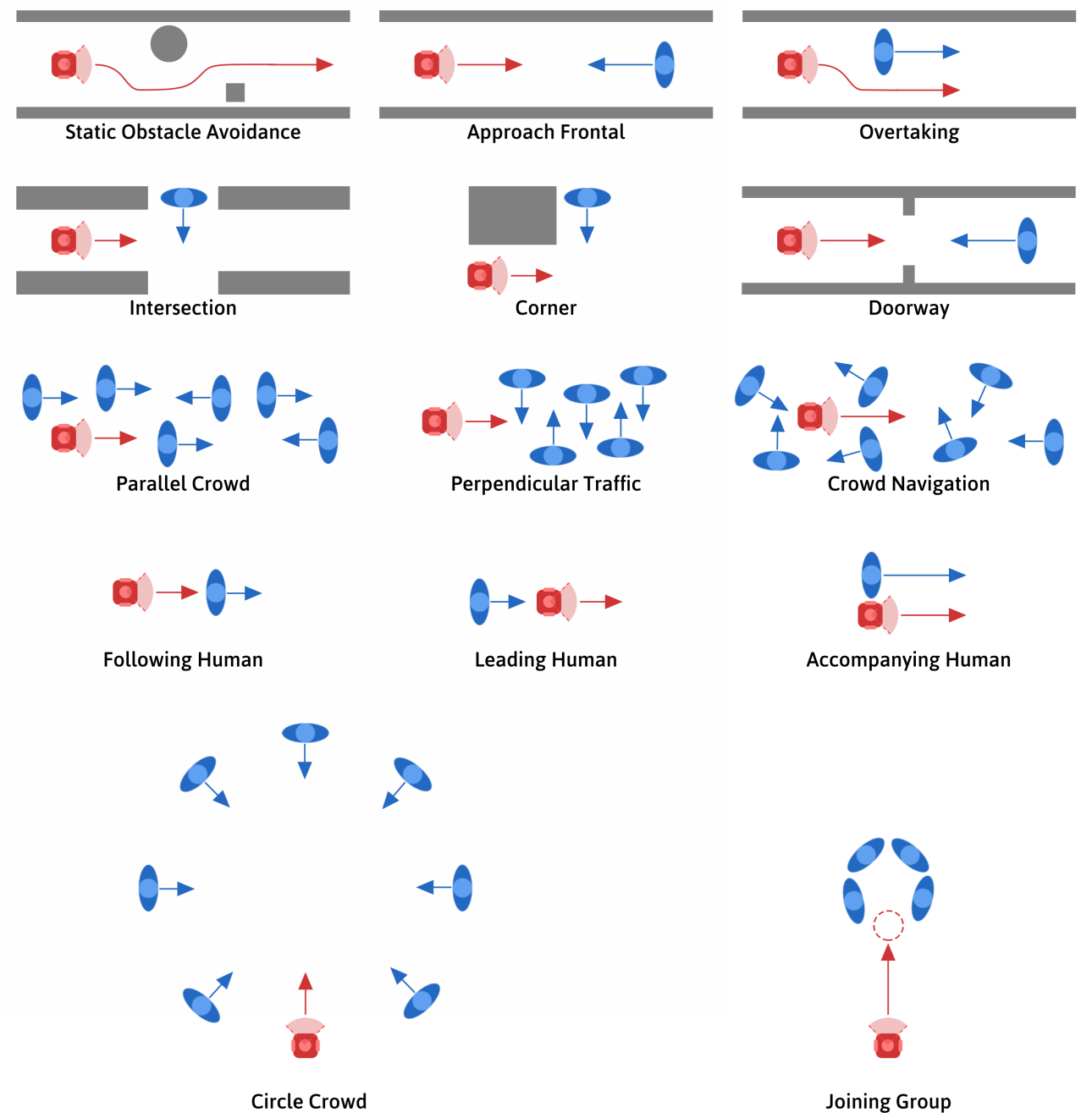


Figure 1.7 – Visual representation of the scenarios presented in Table 1.4. The robot and its intended direction are shown in red, humans in blue, and static obstacles in gray.

1.3.3 Benchmark

A benchmark is defined as a standardized methodology for evaluating HAN approaches. It typically comprises a set of predefined scenarios and quantitative performance metrics, enabling the systematic comparison of multiple navigation strategies. In a benchmark setting, robots are subjected to identical or equivalent conditions—most commonly the same set of scenarios—to ensure fair and reproducible evaluation across different methods. Our definition of a benchmark in the context of an evaluation of HAN methods is then the following: A benchmark is a simulation tool comprising several metrics and scenarios that can evaluate a robotic navigation method according to performance and social aspects. It is worth noting that the simulation could also be carried out in the real world, using a real robot, but in a controlled environment and easily reproducible setup. However, due to the cost of such experiments, existing benchmarks and the one proposed in this thesis rely on virtual simulations.

Table 1.5 presents a non-exhaustive overview of existing benchmarks designed for the assessment of HAN. For each benchmark, the table specifies the simulation platform employed, the underlying human behavior model, the types of scenarios included, the performance and social metrics used for evaluation, and whether the benchmark is compatible with the ROS framework. ROS compatibility ensures that the solution can be deployed on real robots, while excluding overly simplistic benchmarks, such as dots in a 2D area [Chen, Liu, Kreiss, et al. 2019], which lack sufficient accuracy.

It can be seen that the set of benchmarks is highly heterogeneous, whether in terms of scenario, metrics, or simulator used. Very often, benchmarks offer both method evaluation and tools for developing navigation methods. For example, Arena4.0 offers several interesting tools such as environment generation using an LLM model or a pipeline for training planning agents based on reinforcement learning approaches from stable baselines3[2]. The different benchmarks therefore do not all have the same usefulness or the same evaluation capacity. The most popular models of human behavior among benchmarks are the Optimal Reciprocal Collision Avoidance (ORCA) model and the Social Force Model (SFM). Another common scenario featured in many benchmarks involves navigating through a crowd of human agents within a defined environment.

[2] https://stable-baselines3.readthedocs.io/en/master/

Benchmark Name (year)	Simulator	Human Motion Model	Scenarios	Metrics	ROS
Arena4.0 [Shcherbyna et al. 2024]	Flatland / Gazebo / Isaac Sim / Unity3D	SFM	Crowd	TBD	✓
DynaBarn [Nair et al. 2022]	Gazebo	Custom Model	Crowd	S	✓
Habitat3.0 [Puig et al. 2023]	habitat-sim	TBD	Crowd, Following	S, SPL, C	✓
HuNavSim [Pérez-Higueras, Otero, et al. 2023]	Gazebo / Isaac Sim	SFM (with human reaction)	Frontal, Overtaking, Intersection, Following and Leading	All	✓
iGibson [Li, Xia, et al. 2021]	iGibson	ORCA	Static Obstacle, Crowd	S, SPL, DH	✗
SEAN2.0 [Tsoi et al. 2022]	Unity3D	SFM	Crowd, Joining Group	All	✓
SocialGAIL [Ling et al. 2024]	Python	Dataset	Crowd, Parallel Crowd, Perpendicular Crowd	S, FDE, Faithfulness	✗
Social-Gym2.0 [Sprague et al. 2023]	Python	TBD	Frontal, Intersection, Narrow Passage, Crowd	PL, C, TNM, V	✓
SocNavBench [Biswas et al. 2022]	SocNavBench	Dataset	Frontal, Intersection, Crowd	S, T, PL, TTC, DH, HC, A, J	✗

Table 1.5 – Overview of benchmarks used for evaluating human-aware robotic navigation, detailing key characteristics including the simulation platform, human behavior modeling approach, available scenarios, evaluation metrics, and ROS compatibility.

1.3.4 User Studies

User studies provide an insight into human-robot interaction in the real world. Through observation and analysis of human behavior, it is possible to identify patterns, preferences, and potential challenges in interactions with robots. For example, such studies can provide insights into how humans perceive robot movements, what distances they find comfortable, and how they interpret specific signals or gestures from robots. These observations have a direct impact on the design of navigation algorithms considering safety, predictability, and social norms. Moreover, user studies help to bridge human needs and technical capabilities. For instance, a robot may technically avoid all obstacles with precision but in a way that does not appear natural or courteous to humans. Thus, user studies help researchers to refine robotic behaviors through iterative testing and human feedback, such that the behaviors conform to human social norms and cultural expectations.

The Godspeed questionnaires series (GQS) [Bartneck et al. 2009] is one of the most popular and used sets of scales in the area of HRI and Human-Agent Interaction (HAI). The GQS measures five basic perceptual dimensions:

1. Anthropomorphism - the degree that the robot is perceived as being a human rather than a machine;
2. Animacy - the extent to which the robot is perceived as having life or being animated;
3. Likeability - the perception of friendliness or social appeal of the robot;
4. Perceived Intelligence - the extent to which one believes the robot has cognitive capabilities;
5. Perceived Safety - the observers emotional reaction to and perceived comfort and anxiety when interacting with the robot.

Developed from the Godspeed Scale and on social perception psychological theories, the Robotic Social Attributes Scale (RoSAS) [Carpinella et al. 2017] aims to offer a structured way of studying how people perceive the social character of robots. Factor analysis reveals three broad dimensions: warmth, competence, and discomfort. Discomfort is not just about unfamiliarity but relates more to value-based assessments of the interaction. The Perceived Social Intelligence (PSI) scales, introduced by [Barchard et al. 2018], provide a comprehensive framework for measuring robots' social intelligence. It looks at some of the critical capabilities actually necessary in understanding human emotions, behaviors, and cognition and categorizing persons and groups while presenting robots as desirable social partners for advanced and effective human-robot interaction. The scales are intended to be open to any specific robot instantiation and activity, and prioritize impressions of social competence over inherent intelligence. Studies have also been conducted on specific categories of people, such as the elderly [Cormons et al. 2020], to gather opinions on the robot's acceptability. A more extensive questionnaire specifically designed for HAN is

reported by [Pirk et al. 2022]. The questionnaire includes 4-5 questions per scenario to evaluate the following primary properties: Realism, Scalability, and Repeatability.

1.3.5 Challenges of Assessments

A first observation is that no common framework has yet been developed for benchmarking or comparing robotic social navigation methods. Existing benchmarks are highly heterogeneous regarding the simulators used, utilized metrics, and scenario definitions, usually not being detailed or being poorly described. Despite this diversity, there are some trends that can be observed. Indeed, many of the benchmarks studied have the ability to test the robot in crowd conditions with given environments using a common set of metrics (success rate, time, path length, human-robot distance). However, analysis of the metrics and scenarios in the last subsections indicates that it is difficult to find a single benchmark with an integrated assessment framework capable of addressing the full range of challenges in HAN. Even though many benchmarks revolve around crowd navigation, the lack of tests designed for less crowded topologically structured environments is highly noticeable. Especially in environments that require advanced decision-making, e.g., passing through a narrow passage or handling structured interactions between people. The other problem is that only half of the benchmarks are ROS compatible, just like the methods in the literature are not necessarily open source and ROS compatible. Given the existence of many different versions or architectures of ROS, reproducing the experimental part of some methods can be complex. Merging these problems, it significantly restricts the assessments which may be conducted among HAN systems.

Another significant challenge in assessing HAN methods is performing the evaluation under real-world scenarios. Although many user studies have been suggested in the literature, they are infrequently employed in the actual evaluation of the techniques. An important, yet limiting, aspect of these studies is the need for feedback to modify or take into account people's remarks and comments. The assessment thus becomes continuous, rather than a one-time evaluation as it is often the case. However, most of the evaluation frameworks used do not support feedback loops or adaptation with human intervention, making this type of evaluation difficult to formalize and replicate, and therefore difficult to compare multiple solutions.

1.4 Summary and Our Approach

Building on the state-of-the-art analysis presented earlier in the chapter, this section positions our approaches as responses to the identified limitations and open questions in Human-Aware Robot Navigation (HAN). HAN is the integration of traditional robotic navigation techniques with human-centric considerations to limit disturbance to humans in the proximity of the robot. The field is rich with diverse approaches, from model-based and machine learning techniques to hybrid techniques. Each approach comes with its own set of strengths

and limitations, making comparison essential. To evaluate these methods, Human-Robot Interaction (HRI) scenarios can be used to collect qualitative (trajectory visualization) and quantitative (various metrics) data. Benchmarks, combining these scenarios with appropriate evaluation measures, are useful for this purpose. Using simulators, they allow for systematic comparisons between a set of navigation methods. However, existing benchmarks suffer from limited sets of scenarios and/or metrics for holistic evaluation.

As key contribution, we introduce in Chapter 2 a new benchmark with all the components necessary for thorough evaluation. We use this benchmark for comparison of a range of state-of-the-art methods from various categories. The results highlight two key problems: (1) most current approaches excel in particular situations, but do not generalize well to varying circumstances; (2) the lack of anticipation of methods specifically in narrow environments. Our technical contributions then address the main shortcomings of current methods.

In Chapter 3, to address these challenges, we present a new architecture with a dedicated decision module. This module takes in as input the global path that is calculated by the global planner and provides a socially acceptable path that is in turn handed to the local planner for running, as depicted in Figure 1.8. By formulating the problem as a Markov Decision Process (MDP) and representing the world as a navigation graph, our solution outperforms comparable algorithms in the literature.

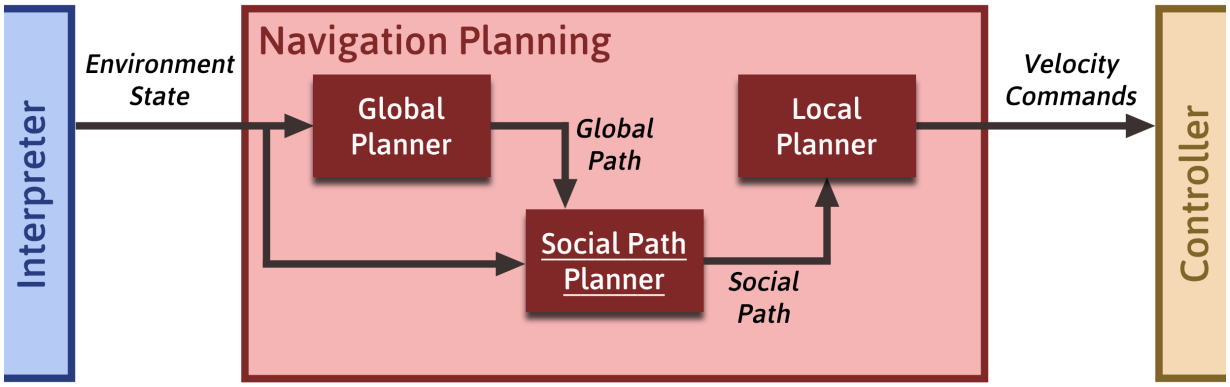


Figure 1.8 – Navigation planning system combining global, social, and local planning strategies. The Interpreter providing environment data, the Global Planner computes a path, the Social Path Planner adapts it for socially compliant navigation, and the Local Planner generates real-time motion commands, executed by the Controller.

Building on these promising results, we extend our baseline model to overcome two major limitations: the utilization of expert-supplied topological waypoints and the planning computational expense of the MDP. To address these problems, we reframe the problem in terms of a polygonal grid representation of the environment and solve the MDP using the Monte Carlo Tree Search (MCTS) algorithm, as described in the second part of the Chapter 3.

Thirdly, we seek to reconcile the advantages of Deep Reinforcement Learning (DRL) methods—no need for a precise model and fast inference—with the robustness of navigation across multiple environments, while maintaining the performance of our first two models. Therefore, we propose our own solution, which is more versatile across different environments

compared to DRL methods in the literature. This approach increases training efficiency and adaptability to a variety of environments by utilizing Curriculum Learning (CL) and combining multiple modalities (laser scans, robot state, and human information).

We test our three proposed methods in simulation and experimental settings with our benchmark system over a variety of scenarios, allowing us to compare them with existing solutions. We then deploy our algorithms into an end-to-end navigation system to execute on real-world robotic platforms. These experimental results are presented in Chapter 5.

Chapter 2

RobotSNAP: Robot Social Navigation Assessment Platform

Outline of the current chapter

Building on the state of the art of HAN review and its assessment in the previous chapter, we now turn our attention to addressing the identified gap: the lack of extensive benchmarks. The ideal benchmark allows repeatable evaluations, covers a wide range of situations, and provides evaluations regarding all aspects of robot navigation.

This chapter introduces a systematic evaluation framework for social navigation methods. It is organized in two parts. First, we present our Robot Social Navigation Assessment Platform (RobotSNAP), a standard for evaluating any social navigation method that can be coupled with the ROS middleware. Second, we use RobotSNAP to evaluate several state-of-the-art off-the-shelf methods, reporting quantitative metrics of their performance. We conclude this chapter by discussing the results obtained and identifying the shortcomings of existing solutions. RobotSNAP represents a key technical contribution of this thesis, aiming to bridge the gap in reproducible and comprehensive evaluation tools for HAN.

2.1 Overview of RobotSNAP

As seen in the previous chapter, there are numerous simulation benchmarks offering different ways to evaluate HAN methods. However, no benchmark offers metrics and scenarios that are sufficiently diverse to evaluate a method from different navigation perspectives. For this thesis, we propose a new benchmark promising to address the problems raised in the first chapter. Thus, we present RobotSNAP, a benchmark capable of evaluating any ROS-compatible solution (ROS1 and ROS2) using numerous metrics and on a large number of scenarios. RobotSNAP was initially derived from the SEAN2.0 benchmark, offering more metrics and sparse scenarios. We improved it by porting it to ROS2, developing more scenarios and metrics, as well as some additional features. In this section, we explain the full scope of RobotSNAP and the proposed methodology associated to the tool for evaluating HAN methods. RobotSNAP benchmark is available on github[1].

2.1.1 RobotSNAP Architecture

Regarding the architecture of RobotSNAP, we use Unity3D[2] as a simulator. Unity3D is a video game development software used mainly for creating video games but widely utilized as a high-performance simulation environment in research and industrial applications. Its capacity to render in real-time and being able to adapt versatile scripting makes it the perfect platform on which to simulate complex situations. For our benchmark, we prioritized the ability to evaluate a wide range of methods, which is why we chose communication via ROS, since it is the most well-known and used framework in the robotics community. With the help of a ROS TCP Connector[3], we establish communication between Unity3D and ROS as shown in Figure 2.1.

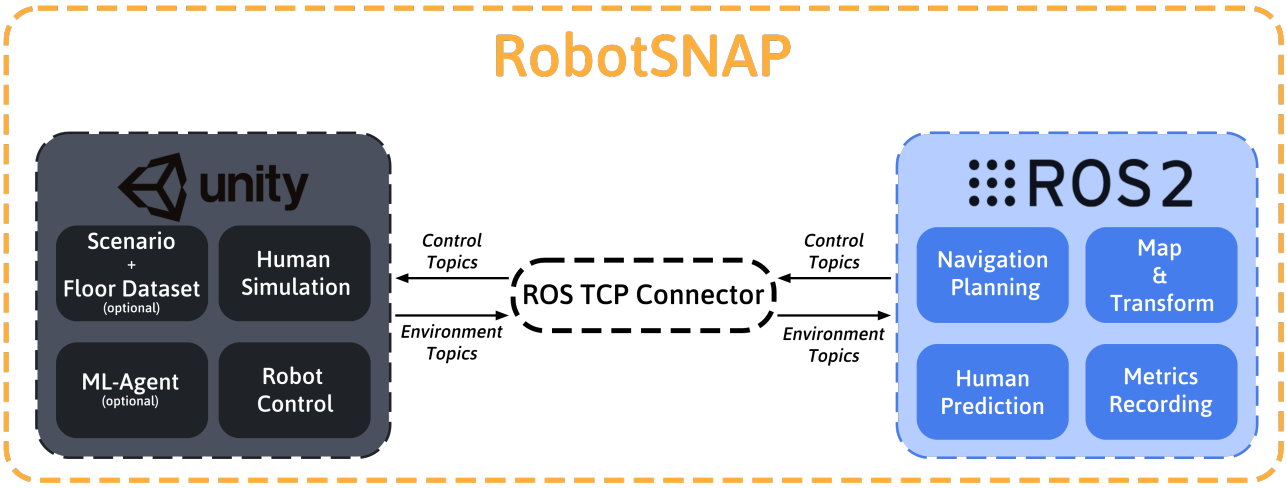


Figure 2.1 – RobotSNAP architecture with communication between Unity3D and ROS2 via ROS TCP Connector package.

[1] https://github.com/agouguet/social_benchmark_robotic_navigation.git
[2] https://unity.com/
[3] https://github.com/Unity-Technologies/ROS-TCP-Connector

The Unity3D simulator handles the robot and human physics, planning, and human motion control, as well as random scenarios. ROS2 manages the robot's planning by processing environmental information provided by the simulator. ROS2 includes all the software components necessary for the path planner of the robot being evaluated. This could be, for example, a trajectory prediction module, activity detection, cost map construction, etc. The simulator handles the sensor and control part of the robot's navigation while ROS2 manages the interpreter and path planner part. Additionally, a package gathers all necessary data (robot and human information) from Unity3D simulator through ROS2 for metric calculation and stores it in JSON and CSV formats for later analysis.

ROS-Bridge

Although ROS2-based methods can be directly evaluated, as illustrated in Figure 2.1, many existing approaches are still implemented using the original ROS1 framework. For a comparison and comprehensive evaluation, we employ the *ros_bridge*[4] package, which offers a communications bridge between ROS1 and ROS2. The bridge allows for simple passing of messages between both versions, and therefore we can add and compare ROS1-based methods in our ROS2 evaluation framework. Furthermore, our benchmark is platform independent, meaning that it can run under Windows, Linux, or macOS. This is done due to the use of Unity3D, which is available on all major operating systems, and through Docker containers that wrap the navigation methods and the ROS bridge, to ensure platform-interoperability. We then employ an architecture within ROS2, as illustrated in Figure 2.2.

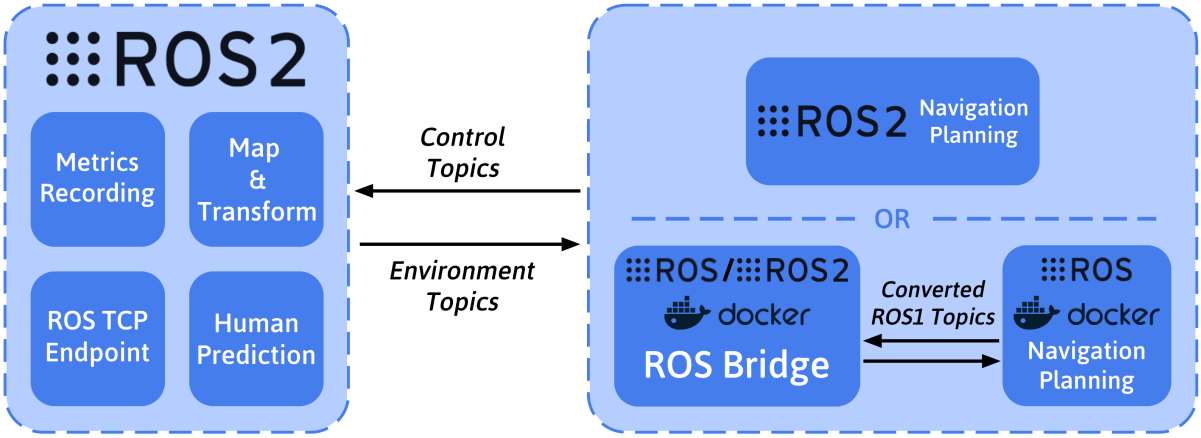


Figure 2.2 – Using Docker and *ros_bridge* for communication between ROS1 and ROS2.

Simulated Human Motion and Path Planning

To properly evaluate socially-aware navigation techniques, it is important to realistically model human motion in shared environments. A human motion model can produce inter-

[4] https://github.com/ros2/ros1_bridge

active and meaningful socially relevant scenarios, allowing the exploration of the robot's behavior alongside pedestrians that have plausible social dynamics in their trajectories.

We used Social Force Model (SFM) [Helbing et al. 1995], a popular model mostly used by the community to simulate human motion, especially in Pedestrian Simulations[5]. The SFM is a mathematical model used to simulate pedestrian flow by treating people as if they are under the influence of virtual social forces, as illustrated in Figure 2.3. Each person is drawn towards a target and repelled from other people and obstacles by repulsive forces. Attractive forces can also be included in some situations to represent social gatherings or points of interest. It is widely applied in crowd simulation, evacuation planning, and city planning to model and predict the movement of individuals in shared spaces.

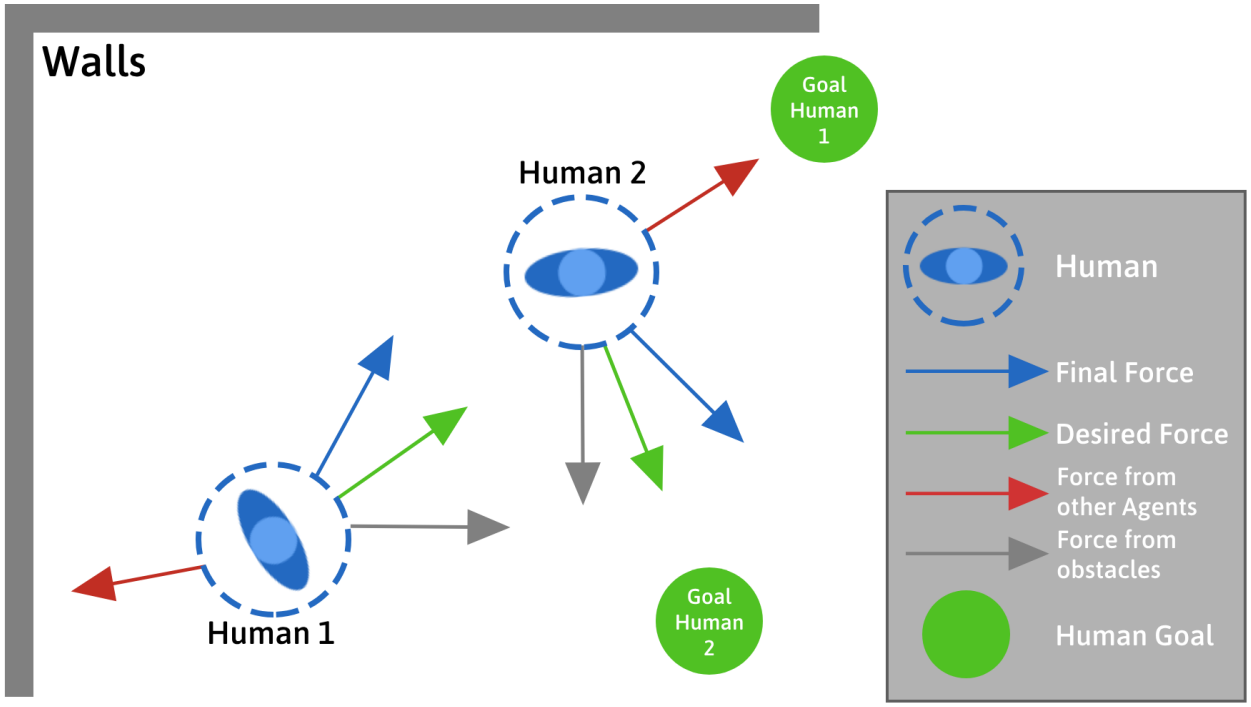


Figure 2.3 – Illustration of the SFM applied to two agents (Human 1 and Human 2) navigating toward their respective goals. Each human experiences a combination of forces: a desired force driving them toward their goal (green arrows), repulsive forces from other humans (red arrows), and avoidance forces from obstacles (gray arrows). The resulting final force (blue arrows) determines their movement direction.

A Navigation Mesh, or NavMesh, is a simplification of the walkable surfaces in a 3D environment, as shown in Figure 2.4. It outlines all the areas through which an agent can walk, excluding automatically any obstacles, walls, or other non-walkable areas. In Unity3D, we use the NavMesh and a path planning algorithm to calculate the most efficient path from one point to another in the environment, so agents can reach their destinations in the most optimal way. This global path serves as the agent's instruction and the Social Force Model provides the local path, allowing for intelligent and human-like navigation.

[5] https://github.com/srl-freiburg/pedsim_ros

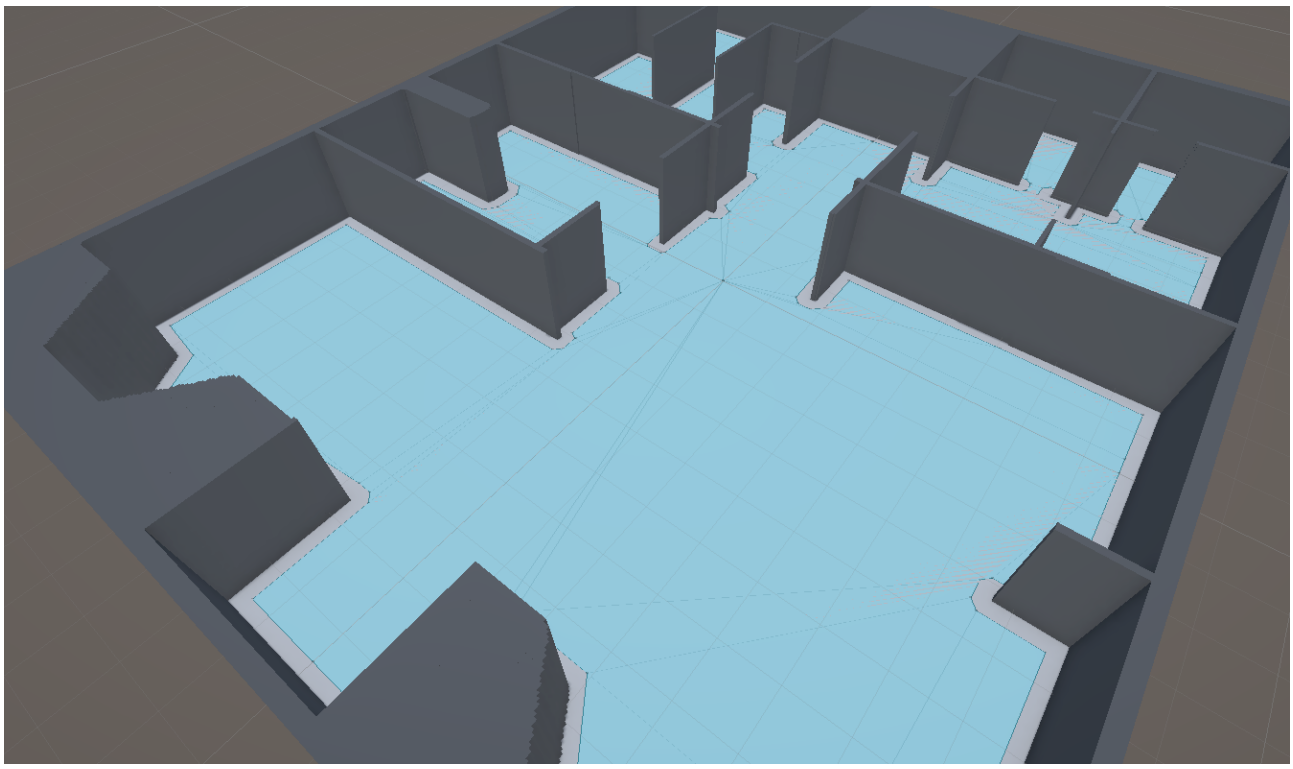


Figure 2.4 – Screenshot from Unity3D showing a navigation mesh (navmesh) used for pathfinding. The blue-shaded areas represent walkable surfaces where agents can navigate, while non-walkable areas are excluded from traversal.

2.1.2 Implemented Scenarios

We now introduce the scenarios implemented in our benchmark. Building on the identified scenarios in the state of the art, in the previous chapter, we developed the most fundamental ones, as illustrated in Figure 2.5. We chose to implement these scenarios first, because each scenario evaluates a different situation and we expect the robot to behave very differently between each scenario. For sparse scenarios, these include: frontal approach, narrow passage, corner, and intersection. For crowd-based scenarios, we selected: circle crowd and perpendicular crowd. A visual of what scenarios look like in RobotSNAP is shown in Figure 2.6.

- **Frontal Approach**: How the robot will manage a situation where it comes face to face with a moving pedestrian directly—testing personal space respect and yielding behavior.
- **Narrow Passage**: Check the robot's skill in passing through narrow places together with other agents, focusing on negotiation, timing, and social conventions (e.g., who passes first).
- **Corner**: The experiment demonstrates the robot's ability to anticipate and avoid collision in case of poor visibility—can the robot infer and respond to possible hidden agents.

- **Intersection**: Evaluates decision-making when multiple agents may converge at a point—balancing priority and trajectory prediction.
- **Circle Crowd**: It gives information about how the robot moves within a dense and symmetric crowd of people. Showing reactivity, smoothness, and crowd integration.
- **Perpendicular Traffic**: Assesses the ability to cross a dynamic flow of pedestrians moving perpendicular to the robot's goal direction—testing assertiveness and timing.

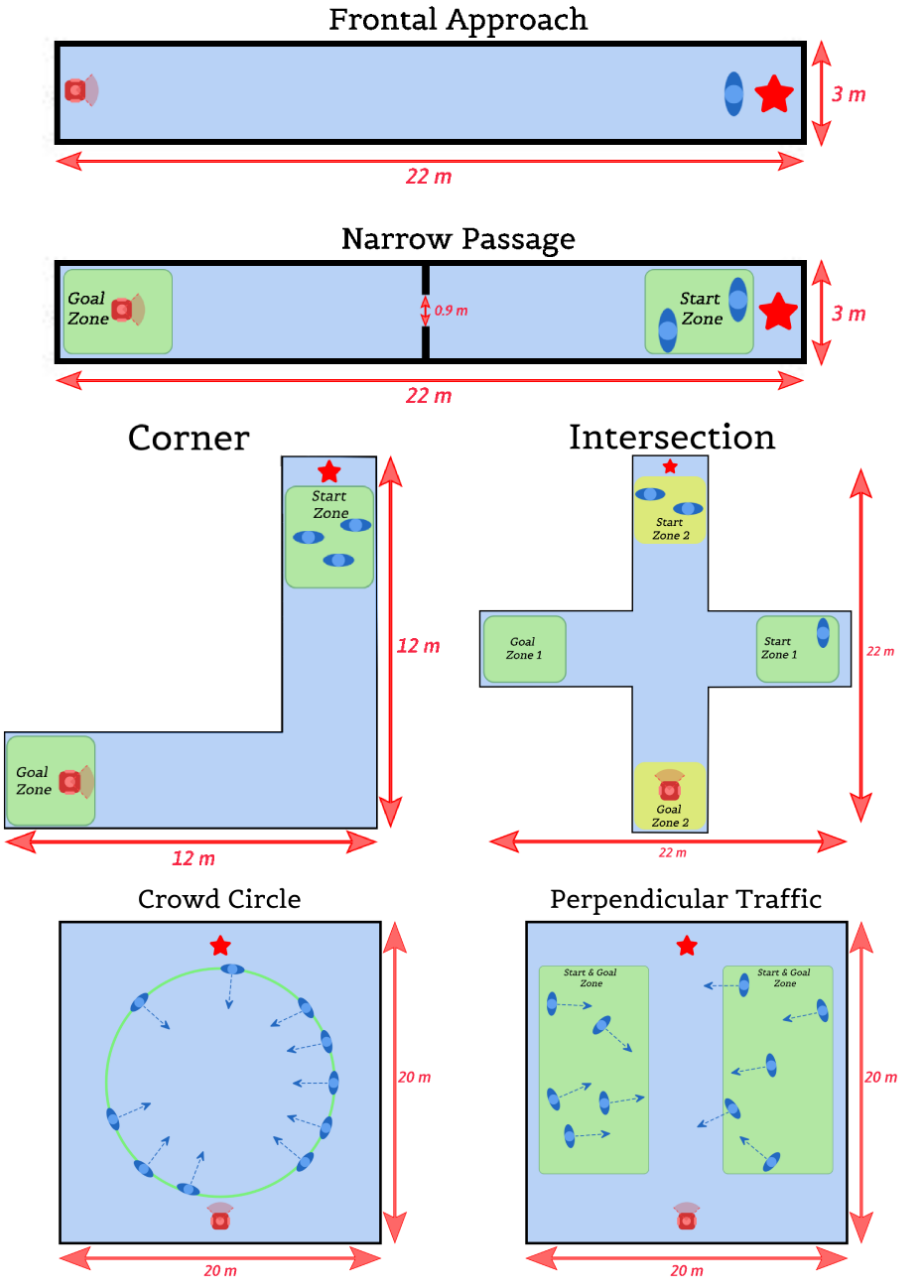
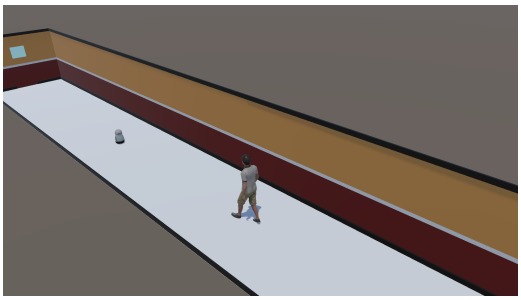
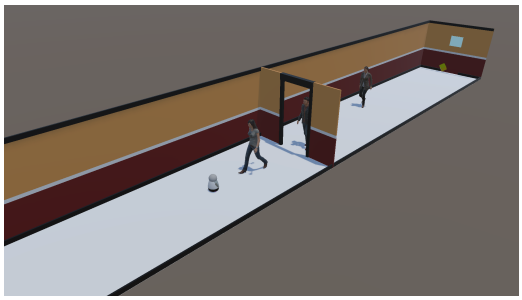


Figure 2.5 – Six commun scenarios to assess human-aware robotic navigation. Static obstacles (walls) are in black and navigation space in light blue. Robot in red and his goal is the red star. Humans are in dark blue and green zones correspond to spawn and goal zones of humans.

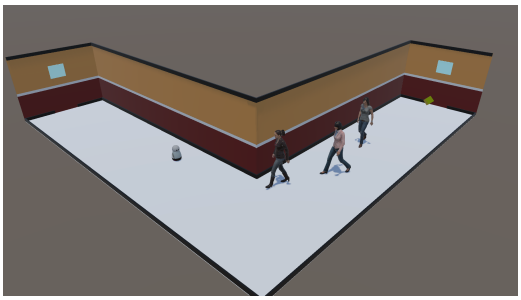
These six scenarios collectively allow the evaluation of a robot's capacity for local reactivity, social awareness, and safe motion planning in typical and challenging indoor situations.



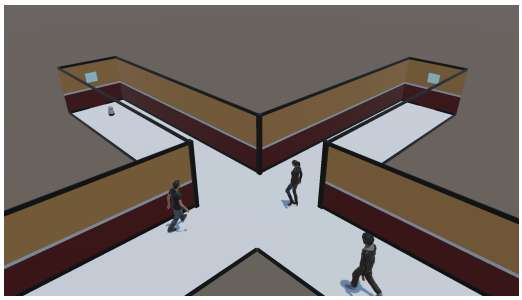
(a) Frontal Approach



(b) Narrow Passage

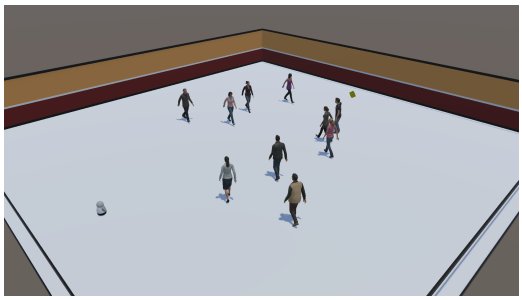


(c) Corner

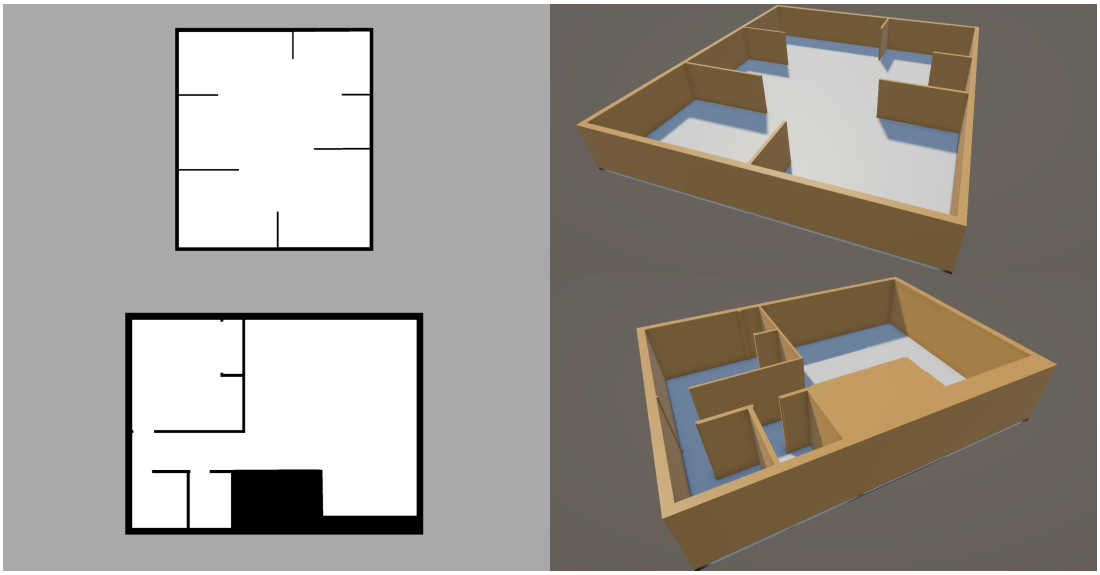


(d) Intersection

(e) Crowd Circle



(f) Perpendicular Traffic



(g) Examples of 2D floorplans from the HouseExpo dataset (left) and their corresponding 3D reconstruction in RobotSNAP (right). The black pixels in the 2D maps are interpreted as walls and extruded to generate realistic 3D indoor environments.

Figure 2.6 – Screenshot of the scenarios present in RobotSNAP.

In addition to the basic scenarios presented above, we implemented the model using the HouseExpo dataset. HouseExpo [Tingguang et al. 2019] is a dataset containing 35,126 2D plans and including 252,550 rooms in total. This dataset was created to simulate SLAM and robot navigation in unknown 2D environments. We reused it to create realistic test environments and training. To do this, we integrated these 2D plans into our Unity3D simulator, which takes a 2D plan and transforms it into 3D (see Figure 2.6g) by raising the black pixels (walls). Using these environments, we created random scenarios: randomly distributed starting and finishing positions of the agents. This allows methods to be evaluated in completely different contexts and environment topologies, unlike the fundamental scenarios presented before where the environment was unchanged.

RobotSNAP therefore contains well-defined pre-implemented indoor scenarios plus a database that is more than sufficient to evaluate social navigation methods. It is also possible to add your own maps if needed, either by manually defining these environments in Unity3D or by providing the occupancy grid, as with HouseExpo.

2.1.3 Selected Subset of Metrics

In Chapter 1 (Table 1.2), we introduced a broad set of metrics for the evaluation of HAN, both in terms of performance and social interaction. RobotSNAP supports most of these metrics. Specifically, we include classical metrics such as Success Rate, Collisions, Path Length, Time to Reach the Goal, Target Error ; performance metrics such as Linear Jerk, as well as social behavior metrics such as Human Collisions, Minimum Distance to Humans, and Minimum Time-to-Collision (TTC).

This selection was made to strike a balance between exhaustive, HAN relevance, and practical discriminatory power. In other words, we tried to select metrics that represent the essential dimensions required to evaluating a navigation method's success—whether in terms of achieving the goal, quality of trajectory, and interaction with humans—without overloading the evaluation with unnecessary or low output distinguishing metrics. Furthermore, RobotSNAP saves the entire trajectory and state information for all agents present in the environment, allowing for flexible post-hoc analysis; new or custom metrics can be computed later without restarting the simulation. This is particularly valuable because the benchmark is useful for real-time evaluation and extensible for more in-depth diagnostic or comparative studies. Moreover, it is possible to implement these own metrics a posteriori in the ROS package in question and then calculate them based on past experiments. In summary, the selected subset is sufficient to measure navigational quality and social compatibility for baseline methods, providing a comprehensive and useful evaluative framework used within RobotSNAP.

2.1.4 Comparison with existing benchmark, limitations and possible improvement

When contrasted with the existing HAN benchmarks summarized in Table 1.5, RobotSNAP can be described as one of the most extensive and analytically rigorous evaluation environments.

RobotSNAP supports a myriad of metrics such as traditional performance metrics (e.g., path length, success, jerk, time not moving) as well as social behavior metrics (e.g., human collisions, closest distance, time-to-collision) that can enhance detailed understanding of navigation behavior. While HuNavSim and SEAN2.0 both provide far-reaching metric coverage, others only measure success rate or count the rate of collisions as performance metrics, which cannot assess nuanced social behaviors. There is also a diverse set of social navigation scenarios provided in RobotSNAP, for example, frontal crossing, intersection, overtaking, and dense, crowds. This allows for focused benchmarking of controlled, repeatable scenarios—something that dataset oriented platforms like SocialGAIL, or SocNavBench lack. RobotSNAP is fully simulated and does not have to rely on existing real data which permits greater control in tuning parameters. One of the most notable strengths of RobotSNAP is that it has native support for both ROS1 and ROS2, and this allows it to be best suited for integration with a wide range of robotic platforms. Among the existing platforms, only two offer support for ROS2, and neither provides out-of-the-box compatibility with both versions.

While RobotSNAP offers many strengths, it also presents certain areas for improvement. It does not currently accommodate the utilization of actual human trajectory data or subjective human assessment protocols such as user preference tests or surveys. Nonetheless, RobotSNAP compensates, with high-fidelity, objective motion measurements that take in smoothness, responsiveness, and implicit social compliance—very critical factors to train and evaluate new learning-based navigation systems.

Finally, future improvements could explore generative tools for scenario or map creation using LLM, such as the Arena platform [Shcherbyna et al. 2024], to support greater diversity in test environments and richer social interaction models.

Overall, RobotSNAP fills a gap in the available set of benchmarks by offering a simulation-based, reproducible, and ROS-integrated system specifically designed for testing the social intelligence and motion quality of autonomous navigation agents. We demonstrate its evaluation and comparison capability on a wide range of different solutions in the next section.

2.2 Benchmarking Off-the-shelf Solutions

In this section, we evaluate several state-of-the-art approaches through our benchmark and using our plateforme, RobotSNAP. We first detail the selected 7 methods including heuristics based methods, deep-learning methods, model based methods and hybrid methods and then

present a scenario-by-scenario comparison of the results.

2.2.1 Selection of Representative Navigation Solutions

The selected methods are off-the-shelf methods—publicly available, pre-trained, and directly compatible with ROS—making them suitable for fair benchmarking using RobotSNAP. Despite these restrictions, seven different approaches are implemented in our benchmark tool. Here is the list of the evaluated methods:

- **Dynamic Window Approach (DWA) [Fox et al. 1997].** The DWA local planner in ROS Nav2 is a real-time trajectory generator used for local navigation and obstacle avoidance. It simulates and scores multiple velocity-based trajectories using configurable cost functions, selecting the safest and most efficient path toward the goal. DWA is flexible, plugin-based, and designed to work seamlessly in dynamic environments with differential or holonomic robots.

- **Human-Aware Timed Elastic Band (HATEB) [Singamaneni, Favier, et al. 2021].** It is a navigation planner capable of planning cooperative trajectories. HATEB offers co-navigation solutions by jointly calculating the trajectories of humans and robots using TEB[Roesmann et al. 2012; Rösmann et al. 2013]. To compute a human's trajectory, the local planner uses a human motion prediction module. A potential field is also applied to static humans to avoid their personal spaces. HATEB has already been compared to the Timed Path Follower solution [Kollmitz, Hsiao, et al. 2015] showing more socially acceptable behavior on sparse and crowd scenarios [Singamaneni, Favier, et al. 2021].

- **Collision Avoidance with Deep Reinforcement Learning (CADRL) [Everett et al. 2018].** CADRL leverages DRL to learn collision avoidance behaviors through experience, enabling more adaptive and socially compliant navigation. The training scenarios included pair-swap tasks and circular crowds, though the environments lacked static obstacles, limiting exposure to complex layouts.

- **Socially Attentive Reinforcement Learning (SARL)* [Li, Xu, et al. 2019].** SARL* model is an improvement of SARL method, which is proposed to rethink pairwise interactions with a self-attention mechanism and joint Human-Robot and Human-Human interaction model using DRL framework. Li, Xu, et al. showed that SARL* outperforms traditional SARL. For our evaluation of this method, we will use a pre-trained model given by the authors. In fact, SARL* requires a training step on the deployment scenarios to exploit its full potential. However, as the goal is to target generic and adaptable solutions in a 'off-the-shelf' philosophy, we chose to skip the training step.

- **Deep Reinforcement Learning-based Velocity Obstacle avoidance (DRL-VO) [Xie et al. 2023].** DRL-VO is a hybrid motion planning approach that combines the Velocity Obstacles (VO) framework with the adaptability of deep reinforcement learning. It enables

autonomous agents to navigate dynamic environments by learning collision-avoiding velocity commands through interaction, rather than relying solely on handcrafted rules. This decentralized method supports real-time, socially-aware navigation among moving obstacles or agents.

- **Bayesian Recursive Nash Equilibrium (BRNE) [Muchen Sun et al. 2024].** BRNE is a real-time crowd navigation algorithm that models interactions between robots and pedestrians as a mixed-strategy Nash equilibrium game. By iteratively updating probabilistic trajectory distributions using Bayesian inference, BRNE enables robots to anticipate and adapt to human movements, facilitating safe and efficient navigation in dense, dynamic environments. Its design allows for real-time computation on low-power embedded systems and seamless integration with existing navigation frameworks like ROS.

- **Model Predictive Control With One-Shot Energy-Based Multi modal Motion Prediction (MPC-EBM)[Zhang et al. 2025].** MPC-EBM incorporates multi modal motion prediction of dynamic obstacles and model predictive control for successful obstacle avoidance. The motion prediction element of MPC goes through an energy-based deep learning model that predicts multiple plausible future positions of neighboring agents. The MPC controller then uses these predictions to determine the safe, collision-free trajectories of the robot.

This selection of solutions has the advantage of being representative of the state of the art and the various approach families presented in section 1.2. We now present comparison results between these solutions evaluated with RobotSNAP.

2.2.2 Results

Table 2.1 summarizes both performance and social metrics across all scenarios. Using RobotSNAP, we conducted 50 experiments for each solution in the two crowd scenarios (2.6e, 2.6f) and 20 experiments in other scenarios (2.6a-2.6d). The execution time for each experiment was bounded, modeling a failure to reach the goal position within the expected time window. The limits were set to 120s in crowd scenarios and 200s in others.

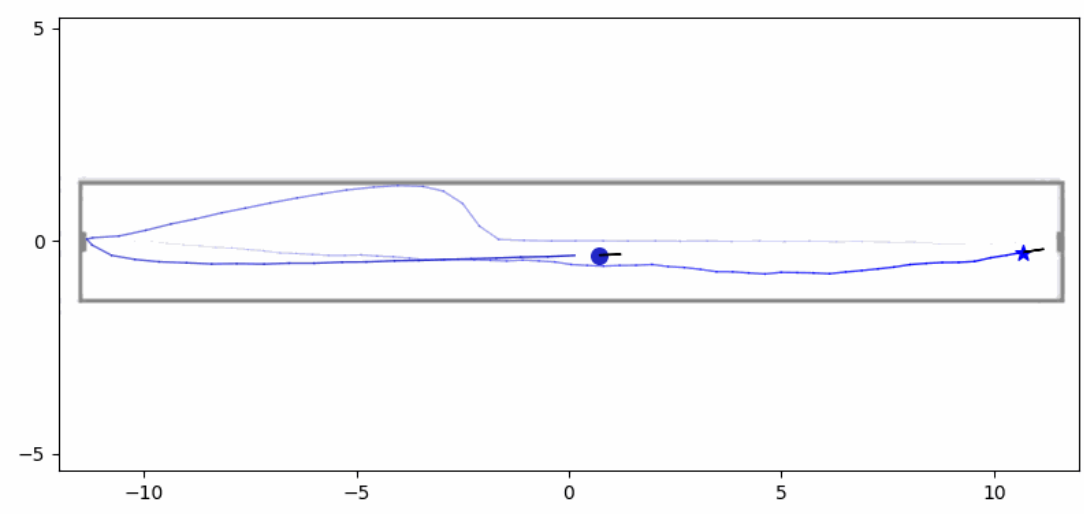
For each scenario, we also provided a visual result of an experiment showing the interesting navigation methods for the scenario, based on the metrics, with the robot and human trajectories. On the visuals, the robot is represented in blue and in the shape of a star and humans are represented with a random color in the shape of a circle. For humans and robot, past trajectories are also traced in time. Higher opacity of the trajectory means more recent positions, while lower opacity indicates older positions. This creates an easy-to-understand timeline of movement trajectory. The full visual results of all methods on all scenarios are available in the appendix A.

		Performance Metrics						Social Metrics		
Scenario	Baselines	Success Rate (%) ↑	Collision ↓	Path Length (m) ↓	Time To Reach (s) ↓	Target Error (m) ↓	Linear Jerk (m/s^3) ↓	Human Collision ↓	Min Human Distance (m) ↑	Minimum TTC (s) ↑
Frontal	DWB	100	0	22.84	63.45	0.26	0.34	1	0.64	10.39
	HATEB	95	0	22.91	92.75	0.21	0.24	0	0.85	21.04
	CADRL	0	0	2.47	200	21.37	0.01	0	0.69	41.63
	SARL*	18	0	32.76	192.9	13.02	0.06	0	0.72	9.56
	DRL-VO	100	0	22.14	63.1	0.86	0.35	0	0.83	9.83
	BRNE	0	20	12.32	199.5	10.87	0.07	0	1.11	6.57
	MPC-EBM	0	20	8.93	200	14.55	0.05	0	1.20	4.46
Narrow Passage	DWB	90	0	21.19	70.23	1.87	0.20	4	0.53	6.39
	HATEB	75	5	20.54	117.8	3.10	0.14	0	0.74	20.08
	CADRL	0	7	4.55	188.61	18.66	0.02	0	0.95	34.39
	SARL*	0	0	9.13	200	20.81	0.03	2	0.77	27.64
	DRL-VO	95	0	21.46	67.71	1.91	0.19	2	0.56	4.67
	BRNE	0	21	9.92	190	13.19	0.05	0	1.46	18.06
	MPC-EBM	5	19	14.10	183.8	13.11	0.06	3	0.71	12.78
Corner	DWB	80	5	16.21	77.71	1.72	0.16	3	0.67	12.00
	HATEB	40	8	12.37	148.1	5.45	0.09	0	0.85	17.12
	CADRL	0	0	7.91	200	9.19	0.06	0	1.07	32.45
	SARL*	0	0	7.48	200	12.88	0.01	0	1.01	18.64
	DRL-VO	95	1	16.85	91.5	1.24	0.18	2	0.84	10.22
	BRNE	0	20	3.06	200	11.30	0.02	0	0.89	18.62
	MPC-EBM	30	14	12.50	166.6	7.29	0.08	1	1.16	10.5
Intersection	DWB	100	0	20.78	58.1	0.27	0.37	4	0.74	1.18
	HATEB	100	0	21.42	83.1	**0.19**	0.32	**1**	0.77	3.98
	CADRL	35	0	13.91	150.05	8.69	0.14	8	0.63	4.81
	SARL*	0	0	7.42	200	20.80	0.02	2	0.69	2.39
	DRL-VO	100	0	20.09	54.45	0.85	0.16	3	1.01	3.78
	BRNE	0	1	6.20	200	14.91	0.05	1	1.05	5.58
	MPC-EBM	30	14	13.00	159.25	12.33	0.10	3	1.02	4.00
Circle Crowd	DWB	100	0	9.97	80	0.52	0.08	7	0.59	3.29
	HATEB	48	0	11.20	107	1.36	0.07	7	0.59	2.71
	CADRL	98	0	9.69	55	0.58	0.12	4	0.59	2.07
	SARL*	88	0	12.55	54	1.57	0.12	0	0.69	3.67
	DRL-VO	**100**	0	9.05	40	0.86	0.16	4	0.54	1.33
	BRNE	96	0	9.95	50	0.97	0.13	3	0.60	4.47
	MPC-EBM	-	-	-	-	-	-	-	-	-
Perpendicular Traffic	DWB	90	0	14.40	111.38	0.90	0.25	12	0.52	4.02
	HATEB	52	0	14.83	113.26	2.14	0.12	22	0.52	1.63
	CADRL	98	0	14.55	63.40	0.62	0.14	14	0.68	1.37
	SARL*	0	0	17.25	120.00	13.0	0.07	0	1.18	13.51
	DRL-VO	32	0	28.75	98.12	9.54	0.17	21	1.23	9.13
	BRNE	**98**	0	14.70	67.48	0.73	0.16	4	0.70	2.34
	MPC-EBM	-	-	-	-	-	-	-	-	-

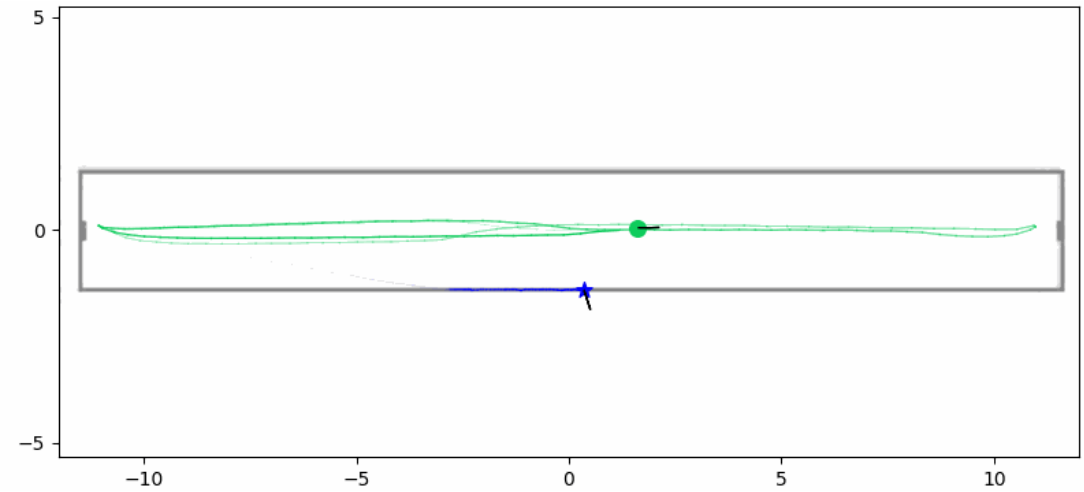
Table 2.1 – Results of the simulations on all scenarios for each method. For each scenario, the methods are ranked from the worst (dark) to the best (light). Metrics showing a poor difference between methods are underlined and those showing a significant difference are in bold. The arrows near the metrics show whether the metrics are better if the value is high (up arrow) or if the value is low (down arrow).

Frontal Approach

Despite high success rates from DWA and DRL-VO in this simple frontal approach setting, their performance hides major weaknesses. DWA, though reliable in reaching the goal, causes human collisions and maintains a worryingly low minimum human distance (0.64m), pointing to a lack of social caution. DRL-VO, while smooth and accurate, has the highest target error (0.86m) in this scenario, raising concerns about its goal precision. Furthermore, we can see in Figure 2.7a that the human feels obliged to shift first. Since the robot has a lack of anticipation of the human, the method does not show any social aspect in this scenario. The majority of learning-based methods (CADRL, SARL*, BRNE) completely fail here with 0% success, either freezing or producing unsafe paths. As shown in Figure 2.7b, BRNE shows a critical lack of adaptability even in simple tasks. MPC-EBM completely fails here, with 0% success and 20 collisions, indicating poor reactivity even in simple settings.



(a) DRL-VO

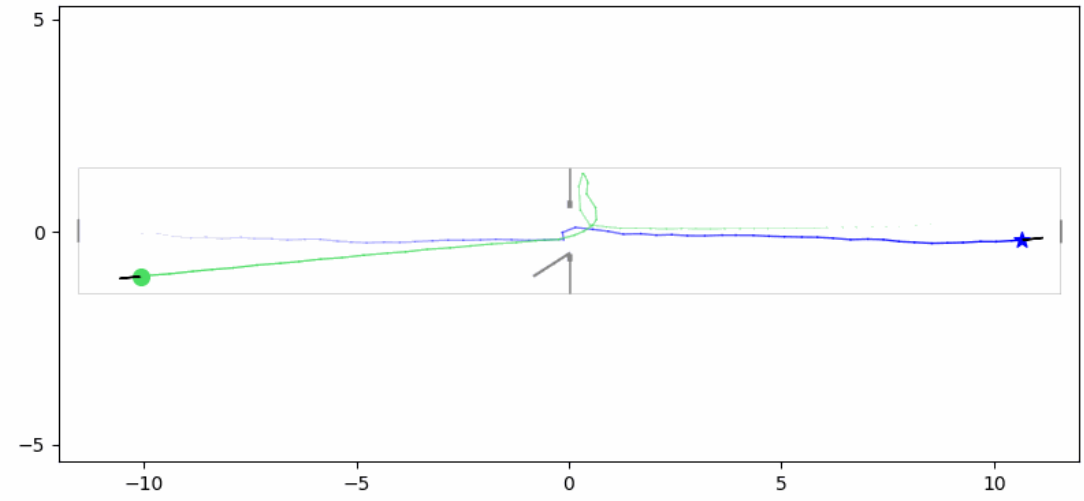


(b) BRNE

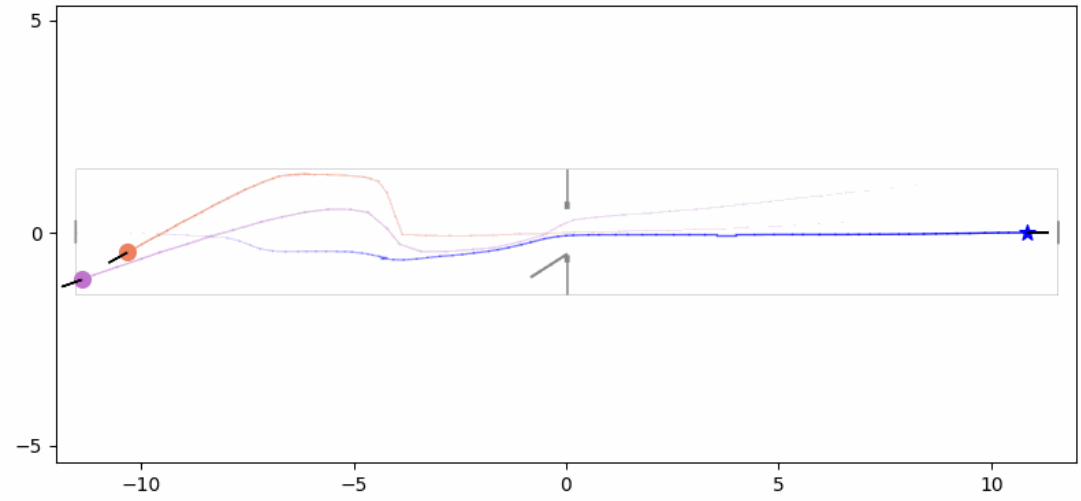
Figure 2.7 – Best (a) and worst (b) performances on the frontal approach scenario. The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Narrow Passage

In narrow spaces, performance degrades rapidly across all methods. DRL-VO and DWA are the only ones to navigate effectively, but DRL-VO still produces 2 human collisions and DWA peaks at 4. Furthermore, as shown in Figure 2.8a, both do not prioritize the human, forcing them to prioritize the robot. HATEB, while more socially responsible since it prioritizes humans (Figure 2.8b), is inefficient. It shows long travel times and a low success rate (75%) due to blocking moments, whether with humans or the difficulty of navigating the narrow passage. The rest — CADRL, SARL*, and BRNE — fail to demonstrate any capacity for constrained path planning, either stalling entirely or generating dangerously erratic trajectories. This indicates that most of these systems lack the nuanced spatial reasoning and crowd-aware behavior needed for tight environments. MPC-EBM manages only 5% success and causes 3 collisions, often pushing through instead of adapting to humans.



(a) DRL-VO

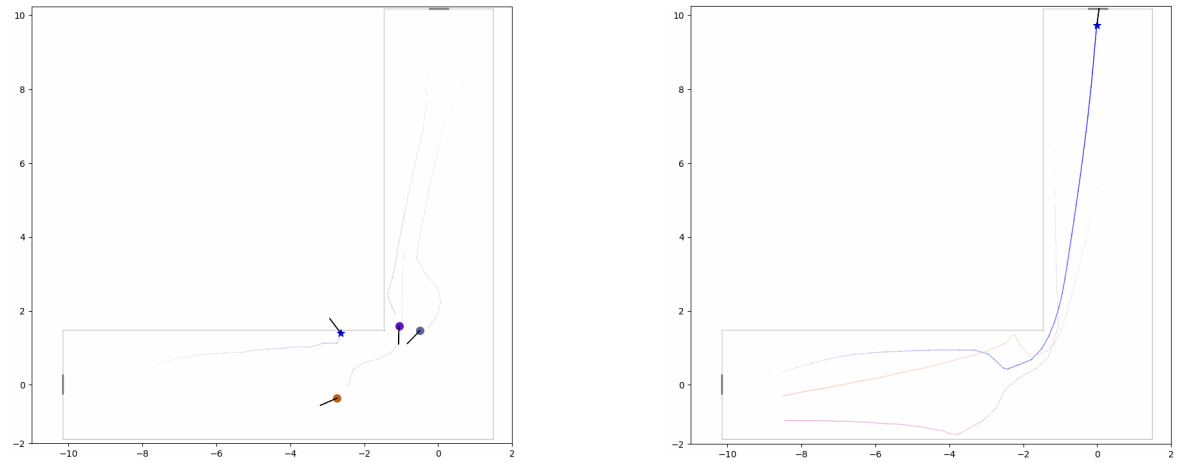


(b) HATEB

Figure 2.8 – Two random examples between 1 and 3 humans, a method with good metrics but no social aspect (a) and a good social method but with flaws in the narrow passage scenario (b). The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Corner

This scenario underlines a major limitation across all models: the inability to handle dynamic and occluded motion planning. While DRL-VO manages a 95% success rate, it still leads to 2 human collisions and 1 static collision (Figure 2.9a), and its proximity metrics remain mediocre. DWA drops to 80% success and causes 5 collisions, clearly struggling to adapt to curved trajectories and forcing humans to abruptly change trajectory as shown in Figure 2.9b. HATEB struggles significantly with only 40% success, despite low collision numbers, suggesting it sacrifices goal-directed behavior for social comfort. Learning-based approaches (CADRL, SARL*, BRNE) once again fail completely or perform poorly, with BRNE recording 20 collisions — an alarming outcome. With 30% success but 14 collisions, MPC-EBM struggles with turning and occlusions, leading to erratic paths. The inability of any method to reliably handle occlusions or reactive turning highlights a fundamental weakness in generalization.

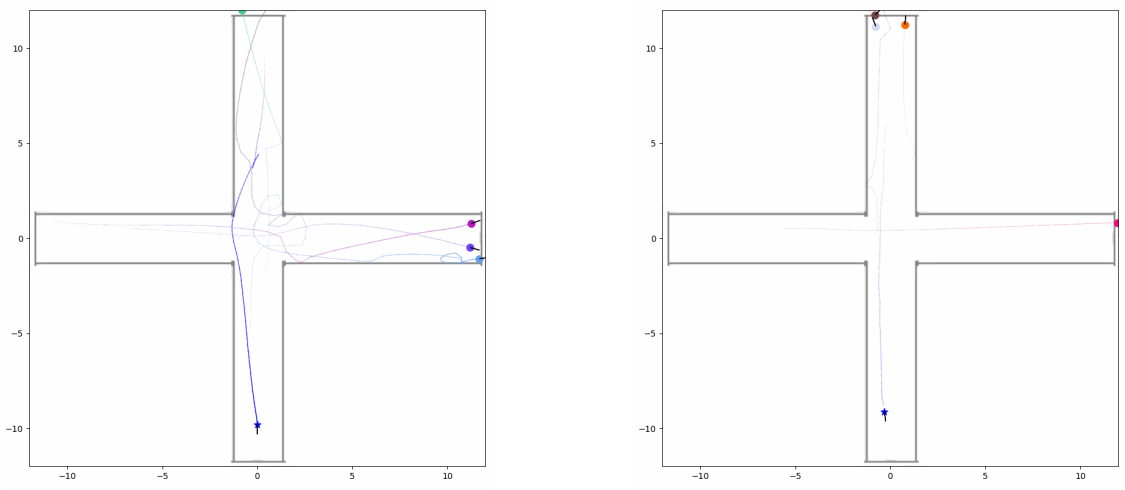


(a) DRL-VO (b) DWA

Figure 2.9 – The unique failure of the DRL-VO method (a) and a non-social planner example with DWA (b) on the corner scenario. The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Intersection

Intersections demand a balance of speed, safety, and anticipation — none of which are consistently achieved. DRL-VO, DWA, and HATEB achieve 100% success but diverge significantly in social safety. DWA causes 4 human collisions, DRL-VO causes 3, and even HATEB — the best in this regard — still records 1. CADRL, SARL* and BRNE are again unable to manage the scenario effectively, either failing to complete the task or generating risky and long paths. MPC-EBM achieves 30% success but lacks anticipation, with 3 collisions and unstable navigation behavior. This suggests that even the best-performing systems prioritize ego-centric success over genuine HAN, which would be critical in real-world intersection usage.

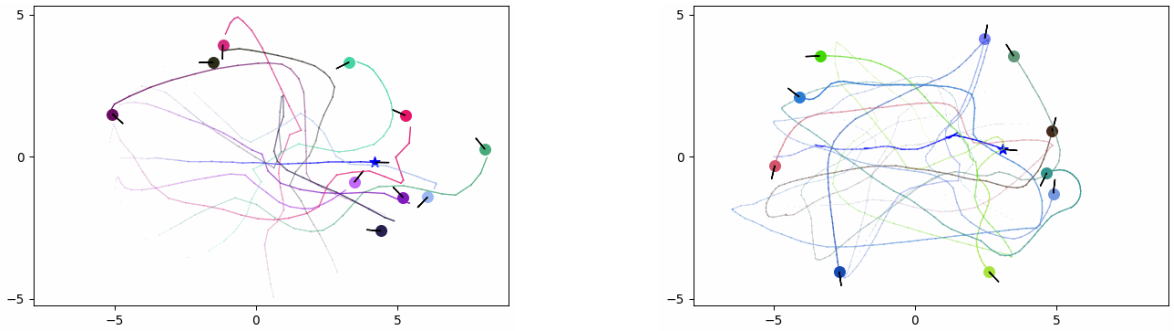


(a) HATEB (b) SARL*

Figure 2.10 – Best (a) and worst (b) performances on the intersection scenario. The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Circle Crowd

In dense crowds, all systems show critical shortcomings. Although DRL-VO and DWA reach the goal (100%), they do so by pushing through people: 4 and 7 human collisions respectively. HATEB is safer but fails in success (48%), essentially giving up when facing crowd density. CADRL, SARL*, and BRNE offer no real breakthrough, often reverting to suboptimal paths or stalling. SARL* manages 0 human collisions but at the cost of very poor path efficiency and success. MPC-EBM fails to produce valid trajectories in this dense scenario due to its high computational cost with many humans; being impossible to run the method on this scenario, we have no data. These results show that current methods are not capable of both understanding crowd flows and navigating them respectfully — a vital requirement for robots in public spaces.

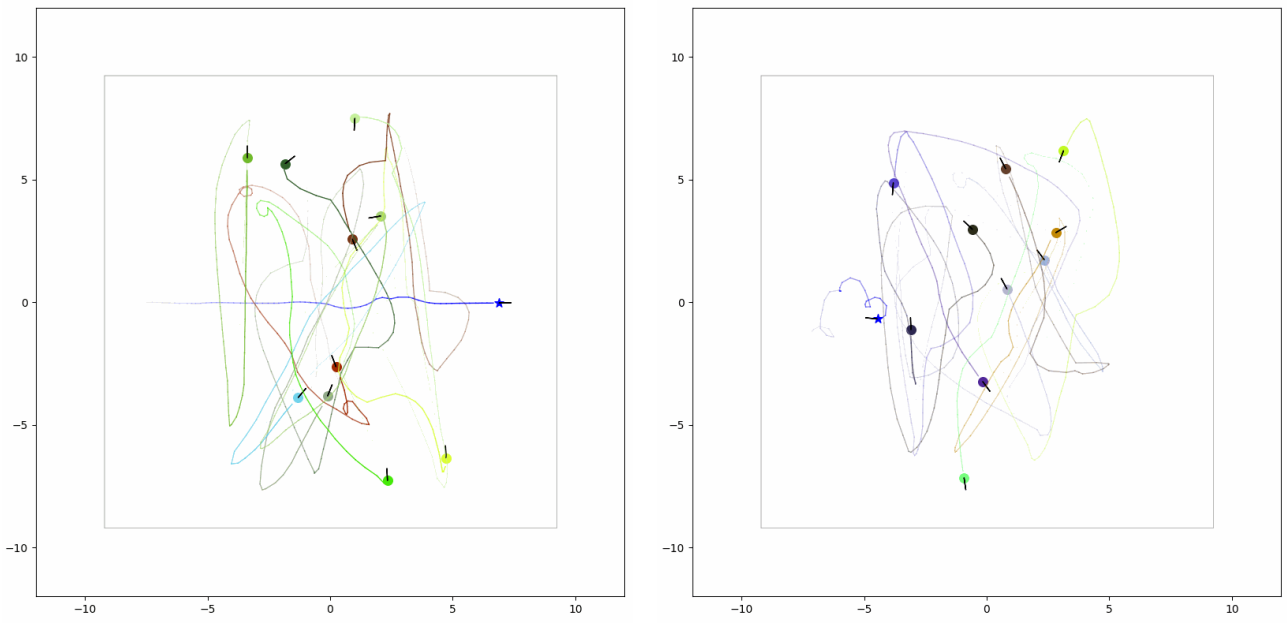


(a) DRL-VO (b) HATEB

Figure 2.11 – Best (a) and worst (b) performances on the circle crowd scenario. The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Perpendicular Traffic

This is where all methods show their worst side. DWA, which performed well in earlier scenarios, registers 12 human collisions — a completely unacceptable result in traffic crossing contexts. HATEB, despite its human-aware philosophy, causes 22 collisions, proving ineffective in dynamic multi-agent environments. DRL-VO and CADRL are no better, colliding with 21 and 14 humans respectively. Only BRNE achieves reasonable success and safety (98% success, 4 human collisions), but this still fails to meet acceptable social compliance standards. SARL*, once again, avoids collisions but cannot complete the task. For MPC-EBM, the situation is the same as in the previous scenario: excessive computational cost was required, making it impossible to run the method. This scenario reveals that none of these approaches are ready for truly interactive, unpredictable urban settings.



(a) BRNE (b) SARL*

Figure 2.12 – Best (a) and worst (b) performances on the perpendicular traffic scenario. The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

2.2.3 Discussions

The comparative assessment of navigation approaches shows that each solution has its own pros and cons. Classical methods like DWA have good path efficiency, and they are reliable in terms of goal reaching behavior. However, their lack of social consciousness leads to unsafe interactions at times. This confirms that purely geometric planning is insufficient in scenarios requiring nuanced social behavior.

MPC-EBM lacks scalability in complex or crowded situations, although it does have an offer of planning structure. In practice, the solution suffers from its trajectory prediction model which is too restrictive, encompassing space too widely. HATEB's behavior is too

conservative: it imposes a stop or retreat behavior if the robot enters the personal space (1.2 m) of any human. This often leads to deadlocks or immobility for long periods without any movement, especially in crowds.

Learning-based methods behave differently depending on the environments. CADRL works well in simple crowd scenarios, but does not preprocess measures for spatial perception (no costmap; no LiDAR scan input) which makes the model ill-suited to environments where static obstacles are present. SARL* exemplify the common limitation of reinforcement learning based methods: the training environments of the RL agents are often biased towards simple and impoverished simplified configurations, i.e., circular crowds. Thus, these policies do not apply well to more varied or structured situations. DRL-VO, nevertheless, shows that it is possible to design a DRL method capable of adapting to different environments.

Overall, these examples are useful in demonstrating the need for a combination of social reasoning, perception of the social space and understanding the environment. Simply avoiding collisions is not sufficient to accomplish high success rates: there must be actions taken to handle geometric constraints, uncertainties in human motion, and implicit social conventions.

2.3 Conclusion

Evaluating social navigation in dense indoor environments has distinct challenges that necessitate a variety of well-defined scenarios and thorough evaluation indicators. Existing benchmarks frequently lack scenario variety or metrics to capture the social impact of methods, as covered in Chapter 1. A comprehensive benchmark is therefore essential to evaluate current solutions, identify their limitations, and guide future research toward more efficient and more socially acceptable systems. In this chapter, we introduced RobotSNAP, a unified and modular benchmark to evaluate social navigation methods into a virtual simulation. By using a broad set of performance and social compliance metrics, RobotSNAP facilitates systematic and structural comparison between various approaches. Among existing benchmarks, it stands out above all for its native support of both ROS1 and ROS2, thus making it the most versatile tool available today. We leveraged RobotSNAP to compare a variety of off-the-shelf navigation systems, selected for ROS compatibility and generalizability across algorithm categories. The results show strong differences between how these systems handle social navigation:

- Classical planners like DWA operate reliably in controlled environments, but are insensitive to human presence.
- HATEB makes good decisions in conditions where there are few humans, but fails in conditions of high population density due to its radical withdrawal policy in dealing with the intimate space of a human subject (1.2 m), which leads to the robot freezing problem.

- Learning-based methods such as CADRL, SARL*, and BRNE show promise to be adaptive and fluid under crowd conditions but lack generalization (other environment). DRL-VO, for instance, efficiently reaches the robot's goals in all environments but usually prioritizes ego-motion over social safety (as in the narrow passage example).
- MPC-EBM, while combining robust multimodal trajectory estimation with MPC, is afflicted by over-conservative predictions. Its own predicted human motion envelopes are often too large, forcing the robot into overly complex avoidance behaviors at the cost of navigation efficiency and smoothness.

No method consistently meets the full set of demands for versatile and socially-aware navigation across all scenarios. The results highlight a critical gap between algorithmic promise and real-world readiness—especially in dense, dynamic, or ambiguous environments—emphasizing the need for more balanced approaches that integrate efficiency and adaptability.

Chapter 3

Markov Decision Process Based Social Navigation Solutions

Outline of the current chapter

3.1 Introduction

In the previous chapter, we conducted a comprehensive analysis of state-of-the-art navigation algorithms across a variety of scenarios, ranging from basic obstacle avoidance to complex crowd interactions. This evaluation revealed a significant limitation common to many existing approaches: their lack of robustness across diverse environments. In particular, a consistent failure (the robot does not reach the goal on time) has been observed in constrained settings such as doorway or corner scenarios, where these methods often exhibit suboptimal behavior or complete failure.

In response to these challenges, this chapter addresses the social navigation problem by introducing a novel approach based on the Markov Decision Process (MDP) framework [Bellman 1957]. The proposed method is designed to enable socially compliant navigation by explicitly accounting for human presence and comfort, while simultaneously ensuring goal-directed efficiency. This solution is based on a navigation graph [Kavraki et al. 1996] to explicitly map the strategic points of the environment. Rather than relying on data-driven or learning-based methods to solve the MDP, we exploit the nature of the modeled graph to construct a solution that is both interpretable and generalizable.

This design choice is motivated by the need for greater safety, reduced computational and data requirements, and the ability to deploy the navigation strategy in unseen environments without retraining. We demonstrate through experiments on the narrow passage scenario that this solution is more efficient and socially responsible than HATEB [Singamaneni, Favier, et al. 2021], the most effective state-of-the-art solution on this specific scenario according to our benchmarks presented in Chapter 2. HATEB is an interesting concurrent approach due to its capacity to handle a few humans in cluttered environments, compared to other approaches.

Building on these promising results, we further extend our approach to overcome two key limitations of the initial model: the reliance on expert-defined topological waypoints and the computational cost associated with solving the MDP. To address these issues, we reformulate the navigation problem using a polygonal grid representation of the environment, enabling fully automatic spatial discretization. Furthermore, we employ a Monte Carlo Tree Search (MCTS) [Kocsis et al. 2006] algorithm to efficiently solve the MDP, allowing the robot to plan socially-aware behaviors in complex and previously unseen environments, while maintaining computational tractability.

This chapter is structured as follows. Section 3.2 introduces the theoretical foundations of MDPs, detailing their formal definition, solving techniques, and their relevance to robotic planning tasks. Section 3.3 presents our proposed method for HAN, which combines the MDP framework with navigation graphs to effectively handle interactions in constrained spaces. Section 3.4 then describes our extended solution using polygonal grids and MCTS. Finally, Section 3.5 discusses a summary of key findings, the limitations of the approach and perspectives for future work.

3.2 Markov Decision Process

An Markov Decision Process (MDP) [Bellman 1957] provide a principled mathematical framework for modeling decision-making in situations where outcomes are partly random and/or partly under the control of a decision-maker. This makes them particularly suitable for robotic navigation tasks in dynamic or uncertain environments. In this section, we introduce the formal definition of MDPs and review common solution methods. We then discuss how this framework can be adapted to model HAN in constrained environments.

3.2.1 Markov Decision Process Formalism

An MDP is a common formalism for sequential decision models, so it is a stochastic model that describes an agent in its environment (the robot in our problem).

Definition 1 (Markov Decision Process) *A Markov Decision Process is defined by a tuple* $(\mathcal{S}, \mathcal{A}, T, R)$*, where:*

- $\mathcal{S}$ *is a finite set of states modeling the situations faced by the agent.*
- $\mathcal{A}$ *is a finite set of actions that the agent can perform.*
- $T(s^{t+1}|s^t, a)$ *is the state transition probability function, giving the probability of transitioning to state* s^{t+1} *from state* s^t *when action* a *is taken.*
- $R(s^t, a)$ *is the reward function, giving the expected reward received after taking action* a *in state* s^t*.*

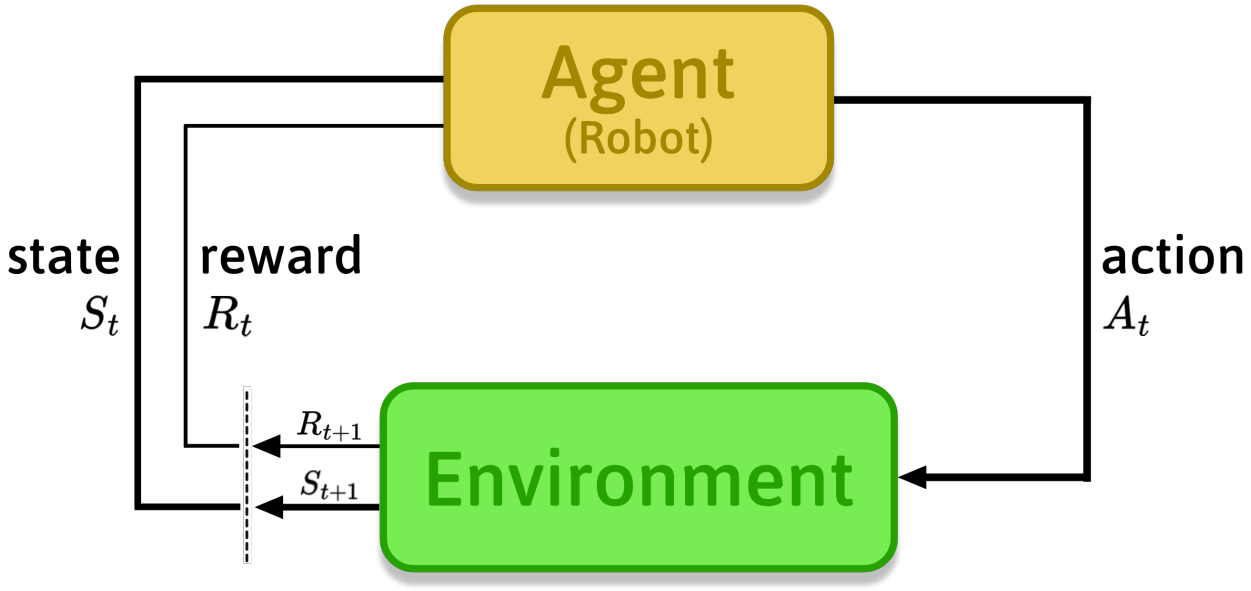


Figure 3.1 – Representation of the MDP model

The agent interacts with the environment over a sequence of time steps, as shown in Figure 3.1. At each time step t, the agent is in state $s^t \in \mathcal{S}$, chooses an action $a^t \in \mathcal{A}$, receives a reward $r^t = R(s^t, a^t)$, and transitions to the next state s^{t+1} (also denoted s') based on the transition probabilities $T(s^{t+1}|s^t, a^t)$. The MDP model uses the Markov Property, which states that the future can be determined only from the present state that encapsulates all the necessary information from the past. The action-selection mechanism is denoted by the policy π.

Definition 2 (Policy) *A policy π is a mapping from states to actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$, that specifies the action $a = \pi(s)$ the agent will take when in state s.*

To evaluate the efficiency of a given policy is, we define a value function that captures the expected long-term return from each state under that policy. In most settings, a discount factor $\gamma \in [0, 1]$ is introduced to give more or less weight to immediate rewards and ensure convergence of the return:

Definition 3 (Value Function) *The value function [Bellman 1957] $V^\pi(s)$ for a policy π is the expected cumulative discounted reward starting from state s and following policy π:*

$$V^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t))\right] \quad \forall s \in S \tag{3.1}$$

This function satisfies the recursive Bellman expectation equation [Bellman 1957]:

$$V^\pi(s^t) = R(s^t, \pi(s^t)) + \gamma \sum_{s^{t+1} \in \mathcal{S}} T(s^{t+1}|s^t, \pi(s^t)) V^\pi(s^{t+1}) \tag{3.2}$$

Definition 4 (Optimal Policy) *An optimal policy π^* is one that maximizes the expected return from every state. That is,*

$$V^{\pi^*}(s^t) = \max_\pi V^\pi(s^t), \quad \forall s^t \in \mathcal{S} \tag{3.3}$$

All optimal policies share the same optimal value function, denoted by $V^*(s)$:

$$V^*(s^t) = \max_{a \in \mathcal{A}} \left[R(s^t, a) + \gamma \sum_{s^{t+1}} T(s^{t+1}|s^t, a) V^*(s^{t+1}) \right] \tag{3.4}$$

3.2.2 Solving the Markov Decision Process

Once the framework of a Markov Decision Process is defined and the value function is introduced, the next goal is to compute the optimal policy π^* that maximizes the expected return of each state.

Two fundamental dynamic programming methods to compute π^* are **Value Iteration** [Bellman 1957] and **Policy Iteration** [Howard 1960]. Both rely on the Bellman optimality principle but differ in how they approach convergence to the optimal policy.

3.2.2.1 Value Iteration

Value Iteration [Bellman 1957] is an iterative algorithm (Algorithm 1) that successively approximates the optimal value function V^* by repeatedly applying the Bellman optimality update:

$$V_{t+1}(s^t) \leftarrow \max_{a \in \mathcal{A}} \left[R(s^t, a) + \gamma \sum_{s^{t+1} \in \mathcal{S}} T(s^{t+1}|s^t, a) V_t(s^{t+1}) \right], \quad \forall s^t \in \mathcal{S} \tag{3.5}$$

The process is repeated until the change in the value function across all states falls below a small threshold θ. Once V^* is approximated, the optimal policy π^* can be extracted by choosing actions that maximize the right-hand side of the Bellman equation (Equation 3.2). Value Iteration combines policy evaluation and improvement in a single step and is particularly effective in problems where convergence speed is important. The computational complexity of Value Iteration is approximately $O(|\mathcal{S}|^2 \cdot |\mathcal{A}|)$ per iteration.

Algorithm 1 Value Iteration

Input: State space $\mathcal{S}$, action space $\mathcal{A}$, transition probabilities $T(s^{t+1}|s^t, a)$, reward function $R(s^t, a)$, discount factor $\gamma \in [0, 1]$, threshold $\theta > 0$
Output: Optimal policy π^* (up to error θ)
Initialize $V(s) \leftarrow 0$ for all $s \in \mathcal{S}$
repeat
    $\Delta \leftarrow 0$
    for each $s^t \in \mathcal{S}$ **do**
        $v \leftarrow V(s^t)$
        $V(s) \leftarrow \max_{a \in \mathcal{A}} \left[R(s^t, a) + \gamma \sum_{s^{t+1} \in \mathcal{S}} T(s^{t+1}|s^t, a) V(s^{t+1}) \right]$
        $\Delta \leftarrow \max(\Delta, |v - V(s^t)|)$
    end for
until $\Delta < \theta$
for each $s^t \in \mathcal{S}$ **do**
    $\pi^*(s^t) \leftarrow \arg\max_{a \in \mathcal{A}} \left[R(s^t, a) + \gamma \sum_{s^{t+1} \in \mathcal{S}} T(s^{t+1}|s^t, a) V(s^{t+1}) \right]$
end for
return π^*

3.2.2.2 Policy Iteration

Policy Iteration [Howard 1960] is an alternative approach that explicitly separates the evaluation of a policy from its improvement. It proceeds in two alternating phases:

- **Policy Evaluation:** Given a fixed policy π, compute the value function V^π that satisfies the Bellman expectation equation (Equation 3.2). This step can be done iteratively or exactly (e.g., by solving a linear system).

- **Policy Improvement:** Given the value function V^π, update the policy by choosing

actions that yield higher expected returns:

$$\pi_{\text{new}}(s^t) = \arg\max_{a \in \mathcal{A}} \left[R(s^t, a) + \gamma \sum_{s^{t+1}} T(s^{t+1}|s^t, a) V^{\pi}(s^{t+1}) \right] \tag{3.6}$$

The algorithm repeats these steps until the policy stabilizes, i.e., no further improvement is possible. At convergence, the resulting policy is guaranteed to be optimal. The policy evaluation step determines the complexity, requiring the solution of a linear system of size $|\mathcal{S}|$, with an approximate complexity of $O(|\mathcal{S}|^3)$ for each iteration.

3.2.3 Markov Decision Process for Robotic Navigation

There are numerous contributions to robotic path planning using MDPs, such as representing the robot's environment with a Quad tree [Burlet et al. 2004], but few are effective for social navigation. For instance, the solution in [Kim, Lee, et al. 2018] uses both an MDP for global path planning and a Partially Observable MDP (*POMDP*) for local path planning. Nevertheless, it represents the robot's environment using an occupancy grid, leading to a very large model that is impractical for real-world scenarios with large maps, especially when modeling humans in the environment. Other approaches, like Cooperative Markov Decision Process (Co-MDP) [Smith et al. 2023], attempt to model human behavior, yet face limitations due to exponential state space growth with increasing agents or map size. Also, [Vanhée et al. 2022] introduce Anxiety-Aware Markov Decision Processes (AA-MDPs), where anxiety computation is directly based on a policy. AA-MDPs have been demonstrated on a 2D map navigation problem with obstacles toward a goal, but also confronts the exponential state space problem. To address this, we propose using graphs instead of grids, placing nodes strategically where the robot must make decisions. Navigation graphs have been used with MDPs to coordinate robot fleets [Lozenguez et al. 2013]. However, with multiple robots, coordination relies on intensive digital communication, which is not applicable to human–robot coordination. Other work [Jeanpierre et al. 2017] also integrates distributed autonomous decision-making using Markov decision processes and Petri-Net planning to develop software that allows a fleet of robots to cooperatively assist a group of people.

3.3 MBSN: The MDP-Based Social Navigation Approach

In this section, we present our social navigation method for mobile robots based on a Markov Decision Process (MDP) approach called MDP-Based Social Navigation (MBSN). We begin by highlighting the advantages of incorporating a navigation graph within the planning step, before defining the problem as an MDP. We then give the overall navigation architecture incorporating MBSN. The section concludes with a demonstration of MBSN and a comparison with HATEB [Singamaneni and Alami 2020] on the narrow passage scenario, as well as an overview of its strengths and weaknesses.

3.3.1 MDP Formalism for Human-Aware Robotic Navigation

As explained in Section 1.1.1, the robot uses multimodal sensor data—like LiDAR, inertial measurement units (IMU), and cameras—for navigation. This kind of raw sensory data is interpreted in an attempt to infer environmental properties such as obstacles or humans being nearby. However, no sense of social navigation considerations, such as the comfort of humans or intrusiveness of the robot's presence, are sensed by the system currently.

To address this limitation, we suggest an extended environmental representation as a navigation graph. The graph encodes additional semantic information, in which nodes are significant spatial locations that are likely to be influenced by or influence human-robot interaction. These locations include narrow passages, alcoves, doorways connecting various interior areas, and other socially sensitive areas. This more elaborate representation allows the robot to take into account potential social interference within its path planning and decision making.

Navigation Graph A navigation graph [Kavraki et al. 1996] is an abstract representation of the robot's environment in graph form, where the space is modeled as a set of discrete nodes connected by edges. While edges indicate traversable paths between these locations, nodes typically represent important locations or waypoints that the robot may need to reach (such as doorways, intersections, or open areas). By discretizing the continuous space into a structured model, this graphical representation facilitates planning, decision-making, and reasoning about potential movements.

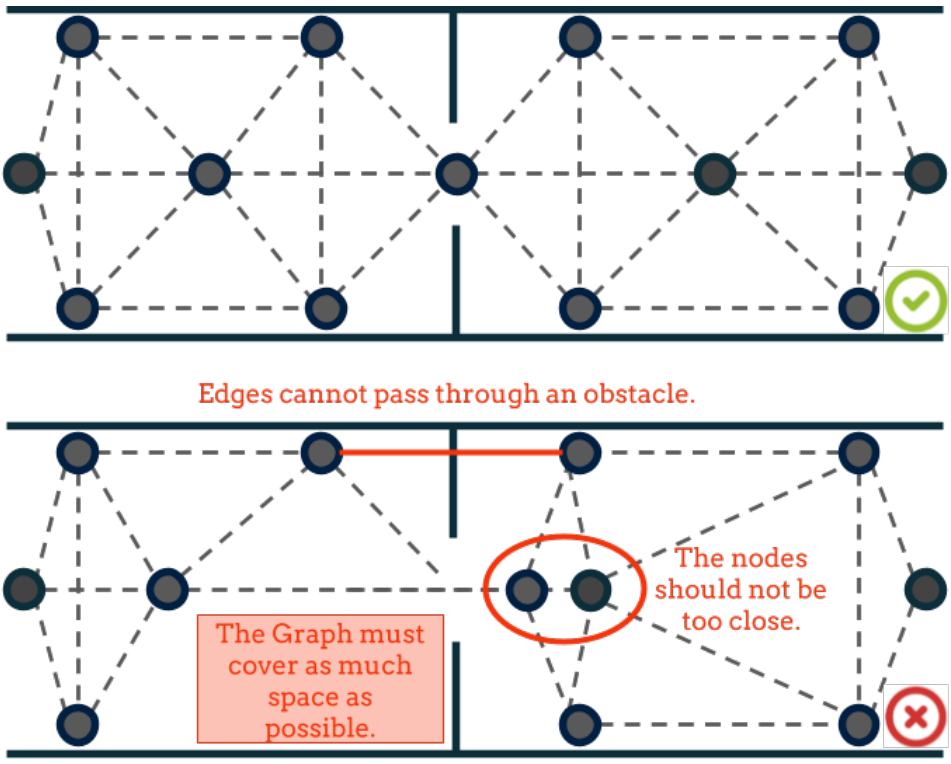


Figure 3.2 – Narrow passage case with two different graphs: A correct one (top) fulfilling all three constraints and an incorrect one (bottom).

The graph can be constructed manually by an expert or using an algorithm, from a map of the environment and an occupancy grid, generally a grid that describes the occupancy status of the space. Social navigation rules are embedded into the graph (cf. Figure 3.2) through the following construction rules:

1. nodes must be at least 1.20 meters apart (human personal space distance);
2. two nodes are connected if there are no obstacles between them, allowing the robot to navigate freely;
3. the graph must cover as much space as possible.

As described just before, we represents the robot's environment as a graph $G = (N, V)$, where nodes N indicate key navigation points and edges V represent possible movements, with nodes occupied by humans marked accordingly. We propose a control architecture that uses this graph-based model to represent the robot's environment, and a deliberative framework based on an MDP to compute the action policy (navigation plan).

States We define a state s of the MDP as a tuple of the robot position, occupied nodes and the goal node. Formally, $s = (n_s, g_s, O_s)$ with:

- n_s the current node where the robot is (nearest),
- g_s the node matching the goal position for the robot to reach,
- O_s the set of all nodes which are occupied by humans.

To determine node occupancy, we consider the current presence of a human and predict future occupancy based on estimated trajectories. The *Interpreter* module calculates future human positions using their current location, orientation, and velocity, as shown in Figure 3.3 and described by:

$$\text{Traj}(h) = h_{pos} + t_i * h_{vel} \tag{3.7}$$

Where h is the human information, h_{pos} is the human position, t_i is the time in seconds in the future and h_{vel} is the linear velocity of the human h.

Additionally, we define a constant t which represents the estimated occupancy duration of a given node. Thus, a node can be considered busy for three reasons: it is currently occupied by a human, it is predicted to be occupied soon, or it was recently occupied by a human. Representing busy nodes addresses the need to account for human presence and comfort.

Actions When the robot occupies a node, it has the option to either remain on it or move to a neighboring node. Thus, in the current state, the set of available actions $a = n_a \in A(s^t)$ (where a is the action to move to node n_a) is given by:

$$A(s^t) = \{n_{s^t}\} + \{n_i | n_i \in \text{Neighbor}(n_{s^t})\} \tag{3.8}$$

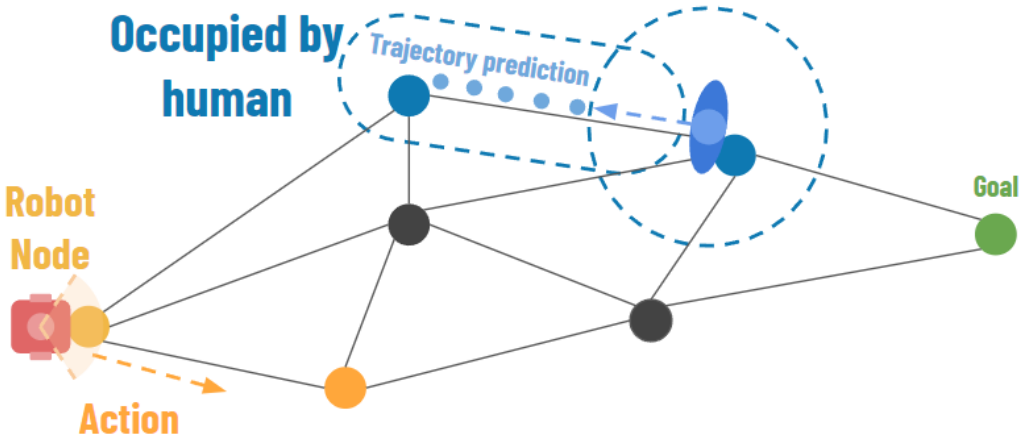


Figure 3.3 – Example of trajectory prediction: The orange node represents the robot's current position, the green node is the goal, and all blue nodes indicate occupied nodes.

Rewards Function We define the reward function as a cost function, where the robot is penalized based on human safety and comfort. The robot incurs a penalty proportional to the minimum distance between its current node and the nearest occupied node. If the robot collides with a human (i.e., occupies the same node), it will receive a significant penalty. To account for path planning efficiency, we introduce a distance-based penalty between nodes. Thus, the **reward function** is defined as:

$$R(s^t, a, s^{t+1}) = goal(n_{s^{t+1}}, g_{s^{t+1}}) - proximity(n_{s^{t+1}}, O_{s^{t+1}}) - dist(n_{s^t}, n_{s^{t+1}}) \tag{3.9}$$

The $goal(n_{s^{t+1}}, g_{s^{t+1}})$ function is triggered once and rewards the robot upon reaching its goal position ($n_{s^{t+1}} = g_{s^{t+1}}$). The function $dist(n_x, n_y)$ returns the shortest distance using the A* algorithm between node n_x and node n_y.

$$proximity(n, O) = \begin{cases} \alpha, & \text{if } \min_{o \in O}(\text{dist}(n, o)) = 0 \\ \frac{\alpha}{\min_{o \in O}(\text{dist}(n, o))} & \text{otherwise} \end{cases} \tag{3.10}$$

The *proximity* function (Equation 3.10) gives a penalty based on the distance to the nearest occupied node, α is a constant representing the collision penalty between the robot and the human. By considering proximity between the robot and the human, we not only respect the human's personal space as much as possible but also anticipate their movement, ensuring comfort in future steps.

Transition function To validate our MDP-based social navigation approach using a navigation graph representation, we make strong assumptions about the transition function definition.

- The first assumption is that the transition function (i.e., $p(s^{t+1}|s^t, a)$, the probability of reaching state s^{t+1} given the current state s^t and action a) can be factorized into three

independent partitions:

$$T(s^t, a, s^{t+1}) = P(n_{s^{t+1}}|n_{s^t}, n_a) \times P(g_{s^{t+1}}|g_{s^t}) \times P(O_{s^{t+1}}|O_{s^t}) \tag{3.11}$$

Here, $P(n_{s^{t+1}}|n_{s^t}, n_a)$ represents the probability of the robot reaching the next node, depending only on its last position and action; $P(g_{s^{t+1}}|g_{s^t})$ represents the probability of the next goal, depending only on the current goal; and $P(O_{s^{t+1}}|O_{s^t})$ represents the probability of future occupied nodes, depending only on the occupied ones in the current state. In this model, a robot and a human can share the same node, but this incurs a significant penalty, as defined in the reward function.

- The second assumption states that robot movements and goals are deterministic. An action always leads to its intended goal. Thus, the probability $P(n_{s^{t+1}}|n_{s^t}, n_a)$ is 1 if $n_{s^{t+1}}$ equals n_a. Additionally, the goal remains unchanged throughout the mission, meaning $(g_{s^{t+1}} = g_{s^t})$ until the goal g_{s^t} is achieved.
- The third and final assumption treats reachable configurations as equiprobable:

$$P(O_{s^{t+1}}|O_{s^t}) = \frac{1}{|\text{future}(O_{s^t})|} \quad if \quad O_{s^{t+1}} \in \text{future}(O_{s^t}) \tag{3.12}$$

The set of future possible occupied nodes (future(O_{s^t})) consists of all combinations of currently occupied nodes and their neighboring nodes, as shown in Figure 3.4.

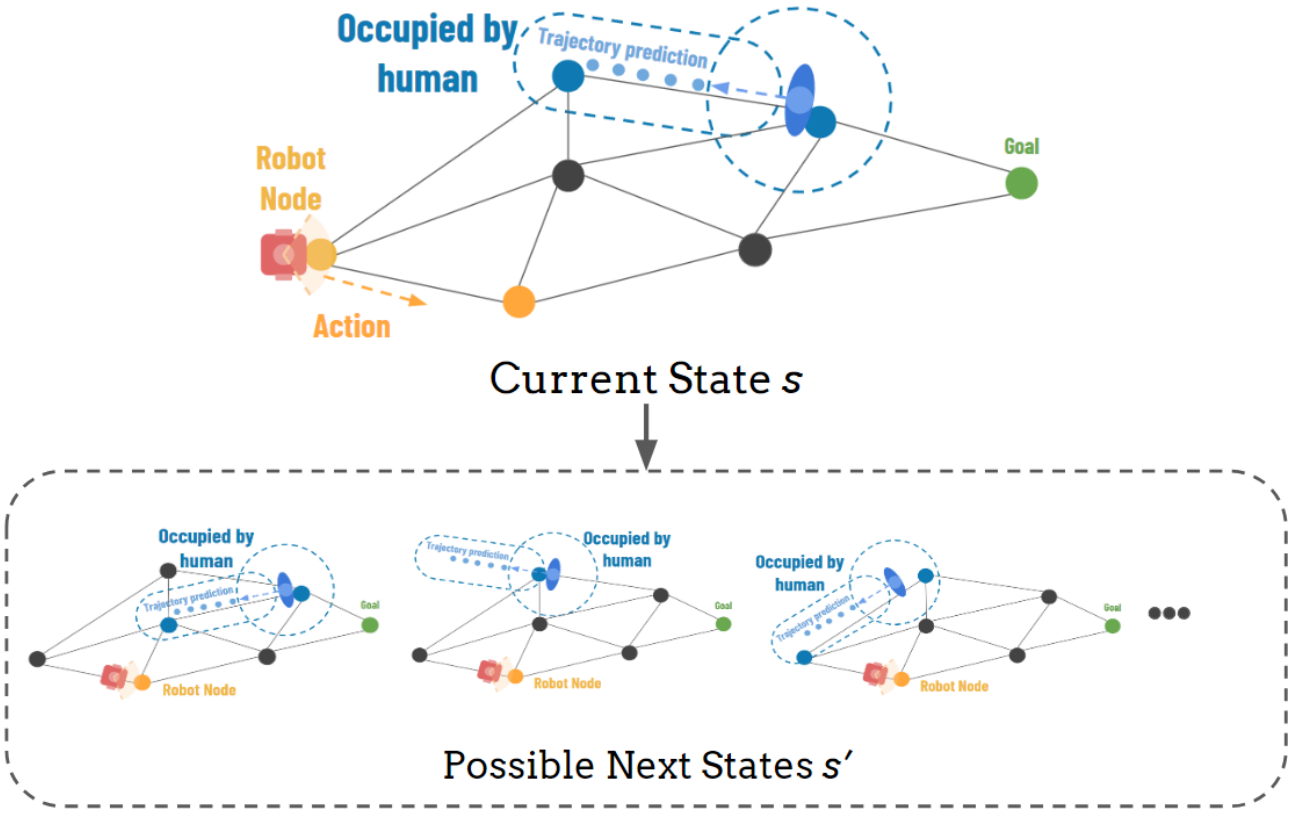


Figure 3.4 – Possible future states resulting from a robot's action.

MBSN version 1.0 Overall, the more accurately the state definition and transition probabilities reflect the environment, the better is the policy generated by the MBSN approach. The proposed model is based on strong assumptions and heuristics to reduce the state space and avoid the need for a learning phase. The idea is for our version 1.0, that if *MBSN* performs well with imprecise state definition and transition probabilities, it will work even better with fine evaluations of possible futures.

3.3.2 MBSN Integration in Robotic Navigation Pipeline

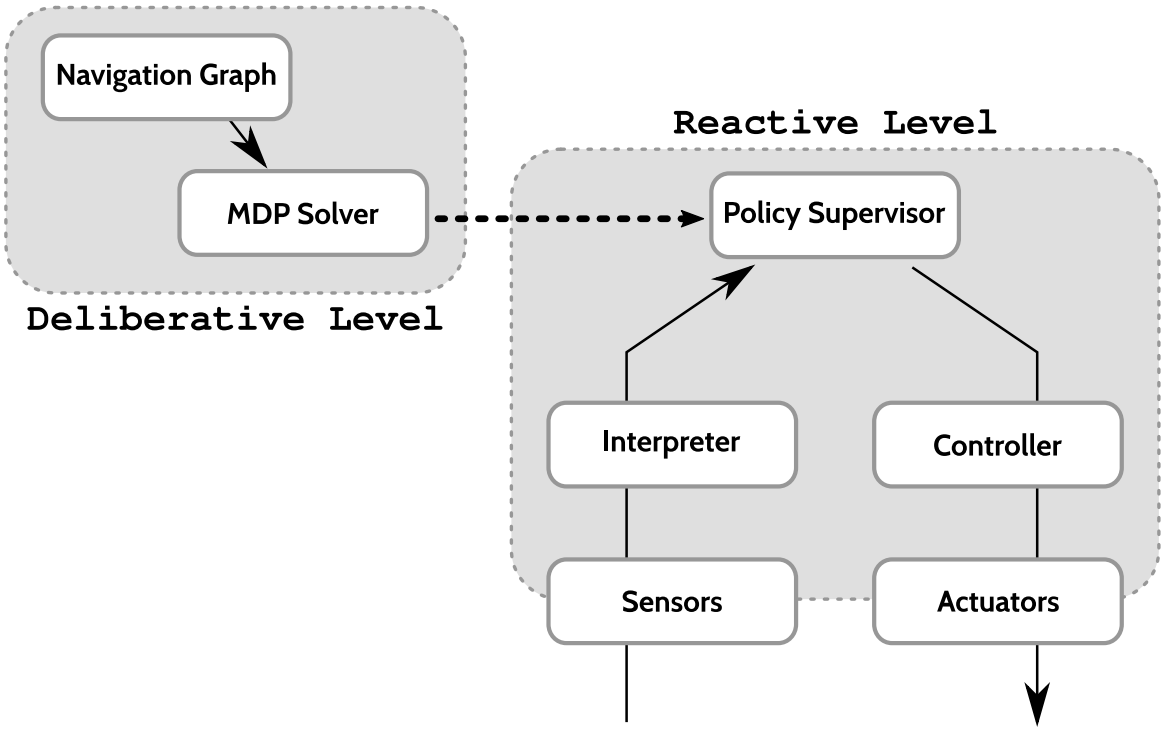


Figure 3.5 – The MBSN architecture of the robot process.

Based on the Robot Navigation Pipeline presented in section 1.1.1, the Figure 3.5 provides an overview of the proposed architecture, where we first compute the optimal policy by solving the MDP, with the graph serving as input to provide information about the nodes. This constitutes the deliberative level, where the policy is pre-calculated prior to execution. During execution of the policy, in reactive level, the robot gathers information from its environment via sensors and converts it into a state, allowing the optimal policy to determine the best action to take. With the optimal action identified, the robot can define the sub-goals and pass them to the controller.

Interpreter/Controller components Although the MDP provides the shortest path while maximizing human comfort, an *Interpreter* is required to provide the policy supervisor with a state, and a *Controller* is needed to translate actions into effective commands for the robot's actuators.

The *Interpreter* estimates the robot's state by considering its position within the navigation

graph and the occupancy of the environment by other entities, such as humans in our case study. It also employs classical localization techniques along with the capability to detect and locate other entities within the environment.
The *Controller* computes the robot's velocity and orientation to follow the designated path. For this, we use the *DWA* [Fox et al. 1997] approach motion controller that generates plausible speed commands, evaluates them based on various metrics, and selects the one with the best score, enabling the robot to proceed to its goal, i.e., the next node provided by the MDP.

3.3.3 MBSN-HATEB comparison on narrow passage scenario

To provide an initial evaluation of the MBSN approach, a first experiment is conducted in a simulated environment using RobotSNAP. We present a qualitative comparison with HATEB [Singamaneni, Favier, et al. 2021] on a narrow passage scenario—a highly challenging setting for socially-aware navigation due to spatial constraints and the risk of human-robot misunderstanding. HATEB has proven to be the most effective solution in this scenario according to our comparison presented in Chapter 2. Our goal here is not an exhaustive comparison of performance (which will be given in Chapter 5), but an illustration of MBSN's abilities in a common scenario where HATEB and state-of-the-art methods tends to fail.

Figure 3.6 shows the trajectories of the human and the robot in a narrow doorway scenario. With HATEB (Figure 3.6 A), the robot attempts to pass through the doorway at the same time as the human, so the two trajectories meet at the point of highest constriction. This leads to inefficient behavior and potential intrusion into the personal space of the human, measured by trajectory overlap and the robot's evasive behavior to avoid collision. The MBSN (Figure 3.6 B) solution, on the other hand, is more socially respectful. Anticipating the human's path, the robot decides to wait before stepping through the small doorway. A smoother and more respectful interaction results from the intentional pause, which lets the human pass through naturally with less regard for the robot. The robot's trajectory is also cleaner, straighter, without abrupt turns or last-minute avoidance around the door. Although these results show the potential of MBSN in narrow passage, they may not accurately represent the unpredictable human responses in real-world situations because they are based on simulated humans. Real-world experiments are presented later in Chapter 5.

3.3.4 Advantages/Limitations

The set of possible actions enables the robot to navigate between the nodes of the graph, as well as to stop when the policy determines that waiting is necessary and advantageous. The definition of occupied nodes, along with its inclusion in the reward function, ensures respect for the human's personal space and preserves their comfort in future states. The stochastic nature of the model and the transition function allows the consideration of possible future human positions when calculating the optimal policy.

One of the strengths of the MBSN approach lies in its ability to generate socially-aware

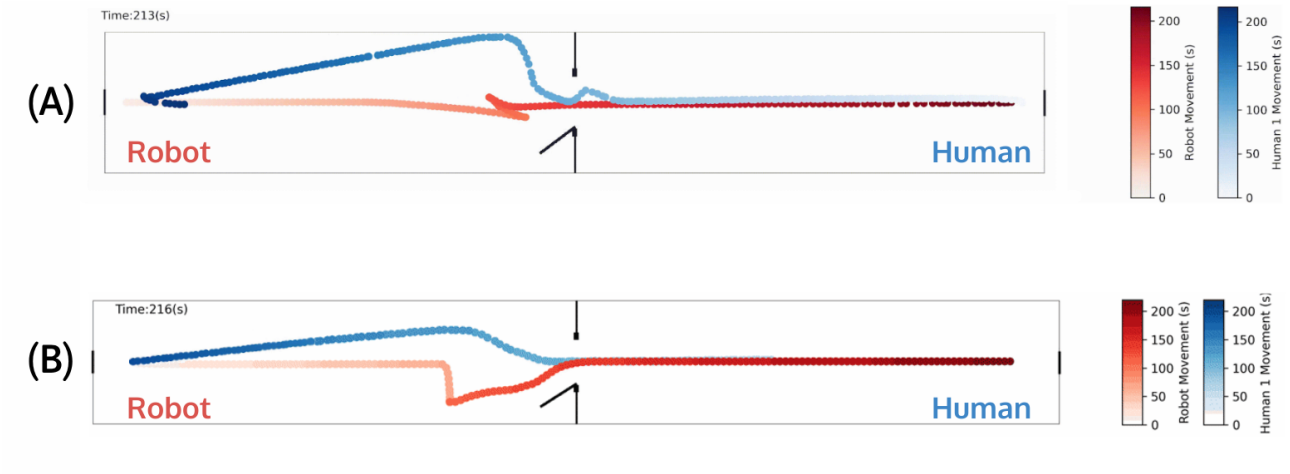


Figure 3.6 – Results of the HATEB (A) and MBSN (B) methods on the door-passing scenario in simulation. The red trajectory corresponds to the robot and the blue one to the human.

behavior without requiring precise predictions of human goals. However, this modeling choice comes with a major drawback: the exponential growth of the state space. Specifically, for a graph with n nodes, the robot may occupy any of the n nodes, aim for any of the n goal nodes, and face 2^n possible combinations of occupied nodes. This results in a total state space of size $n^2 + 2^n$, which may become intractable for large environments.

Regarding the MBSN architecture (Figure 3.5), both the deliberative level and the reactive level are expected to be processed online, at low and high frequencies respectively. However, in our experimental setup, which relies on a classical *Value Iteration* MDP solver, the deliberative level must be processed offline before the experimental simulation begins.

The other limitation arises due to the spatial abstraction by graph representation. Since the robot reasons at the node level, we need to define on what basis a human is considered to "occupy" a node. In our simulation, a node is occupied if a human is within a 1.2-meter distance—consistent with the conventional definition of human personal space. The threshold, however, gives rise to fuzzy cases. If a human is slightly above this range, he or she may not be accounted for, although he or she may affect the movement of the robot. Conversely, if the system merely marks the closest node to each human as occupied, two humans standing near each other can result in an overestimation of occupied space and therefore overly cautious behavior, or the so-called freezing robot problem.

These issues highlight a critical trade-off in MBSNv1: while the navigation graph provides interpretability and structure, it introduces approximations that can affect the accuracy of social reasoning in dense or complicated scenarios.

3.4 Enhancing MBSN via Polygonal Grids and Monte Carlo Tree Search

In the previous section, we demonstrated that representing social robotic navigation as an MDP enables effective anticipation and decision-making. However, the MBSNv1 formulation made radical assumptions: an expert-defined navigation graph with a fixed and often coarse discretization. Additionally, solving the MDP using the value iteration algorithm had too high a computational time complexity for large environments.

To overcome these limitations, this section introduces MBSN version 2 with two key improvements: (1) a new MDP definition based on uniform spatial discretization in the form of a polygonal grid (Figure 3.7), and (2) an online running of the MDP with the application of the Monte Carlo Tree Search (MCTS) [Kocsis et al. 2006] algorithm, improving scalability and responsiveness in dynamic environments.

We first present the polygonal grid-based MDP representation and then describe the planning algorithm based on MCTS. We discuss the impact of the trajectory prediction models used and the exploration heuristics by presenting some results. A conclusion is presented with a discussion of the limitations of the method and its possible extensions.

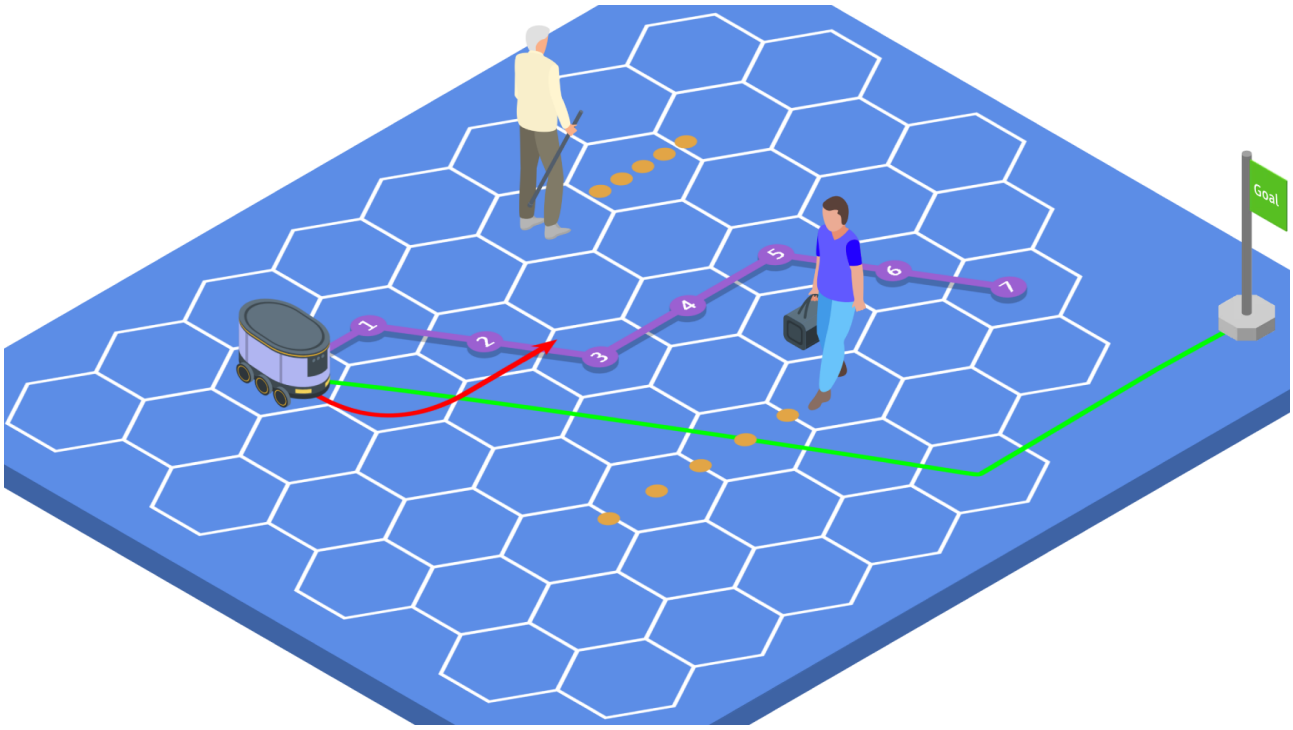


Figure 3.7 – The robot avoids disturbing humans by following a local social path instead of the global planner. Future positions of the humans are shown in orange, the global path in green, our local social path in purple and the velocity command in red.

3.4.1 MDP Polygonal Grid Representation

Like section 3.3, we define the social navigation problem as an MDP. Our approach aims to compute a socially acceptable path within the robot's local scope. In other words, we focus on the computation of a path between the robot and a sub-goal in the direction of the global long-term goal while maintaining social rules. The global architecture is presented in Figure 3.8.

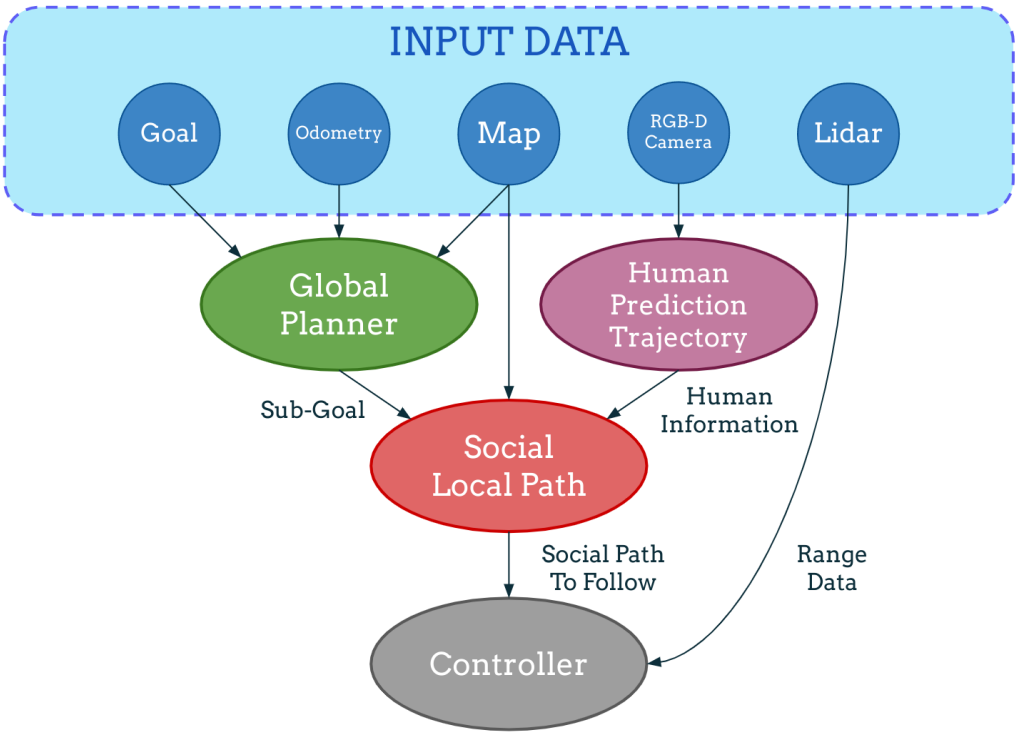


Figure 3.8 – The architecture of our method, including the data flow and different software components enabling navigation.

Compared to MBSNv1, MBSNv2 is intended to be deployed directly on the robot during long-term missions, which involve successive situations such as corridors, doors, and open spaces. Given the high uncertainty of human behaviors, the social path is meaningful only in the robot's local range, where humans can be detected by the robot itself and their movements may have an impact in the near future. To address this, the *Social Local Path Planner* is introduced as a component positioned between the *Global Planner* and the robot *controller*. As in MBSNv1, it is fed with predicted human trajectories.

3.4.1.1 State

Polygonal maps (also called vector or mesh maps) represent the environment as traceable polygonal cells. A polygonal map can be interpreted as a graph where the nodes correspond to the polygons and the edges represent possible movement between two adjacent polygons.

Using this framework and considering the robot environment with $N \in \mathbb{N}$ humans, we define p_r and q_r as the robot's position and orientation, respectively. The task assigned to

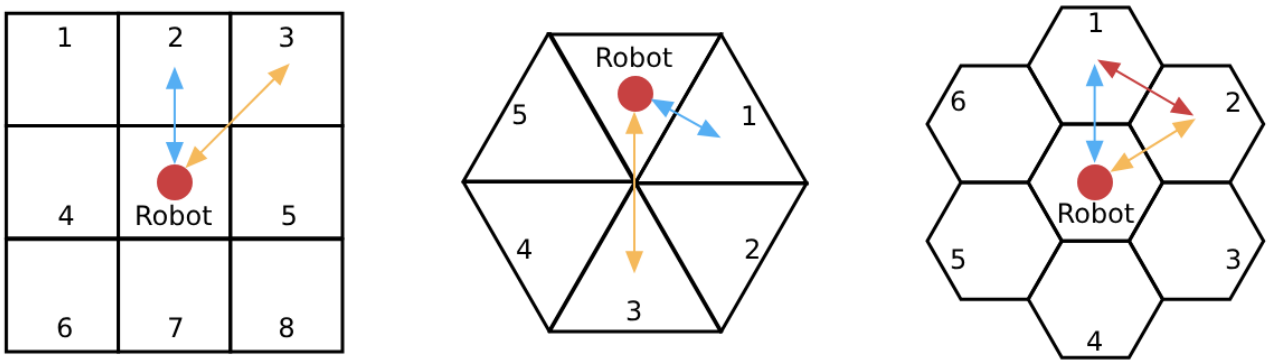
the robot is defined by giving the position of the goal to the robot g. Human information encompasses past and current positions and orientations within a time window: $H^{T_p:T_0} = \{(p^{T_p}, q^{T_p}), ..., (p^{T_0}, q^{T_0})\}_{n=1}^{N}$, where T_0 is the present time and p the number of desired history steps. Thus, we define a $s^t \in S$ the state of the MDP:

$$s^t = [p_r^t, q_r^t, g, H^{T_p:T_t}] \tag{3.13}$$

Therefore, we discretize the environment into a polygonal grid while maintaining continuous robot and human data for greater precision. This enables future human position estimation and allows the robot to consider social and performance aspects in navigation.

3.4.1.2 Actions

The possible actions for the robot at a given time are to navigate towards a neighboring polygon to which it is located, or to not move. For convenient reason, the robot environment is generally represented in regular polygonal form. Square grid maps, triangle grid maps or hexagonal grid maps are commonly used due to their simplicity of visualization and ease of implementation. As shown in Figure 3.9, hexagonal grid maps provide smoother and more consistent movement, particularly in scenarios where precise positioning and equal distances are critical [Uher et al. 2019], such as in social robotic navigation [Samadi et al. 2013; Duszak and Siemiątkowska 2020; Duszak, Siemiątkowska, and Więckowski 2021].



(a) Square Grid
d1 ≠ d2

(b) Triangle Grid
d1 ≠ d2

(c) Hexagonal Grid
d1 = d2 = d3

Figure 3.9 – As in [Uher et al. 2019], the figure shows the differences in properties between square (a), triangle (b) and hexagonal grid (c), number correspond to possible movement actions

In fact, the polygon shape used does not affect the ability to solve the problem as long as tailored actions and transition definitions are provided. They do, however, influence the quality of the trajectory followed by the robots. Additionally, each polygon area should be larger than the size of any robot or human to ensure that a mobile entity stays primarily within a single polygon or node, yet small enough that it cannot encompass two entities at once.

Hexagonal grid maps are used in the rest of this work. Strong and reliable assumptions about actions, transitions, and reward definitions are made possible by their regularity. Actions can be defined consistently without the need for special-case handling because every hexagon is the same. Furthermore, transition modeling and trajectory evaluation are made simpler by the equal distances between nearby polygons.

Thus, in a state s, the set of available actions $a = p_a \in A(s)$ (where a is the action to move to polygon p_a) is given by:

$$A(s) = \{p_s\} + \{p_i | p_i \in Neighbors(p_s)\} \tag{3.14}$$

With p_s the current position of the robot and $Neighbors(p)$ a function giving the position of the adjacent polygons (centroid) at position p.

3.4.1.3 Transition Function

The transition function models the dynamics of the environment according to the applied actions. This dynamic relies on the effective movements of robots and humans. However, learning this function is beyond the scope of MBSN. An arbitrary definition is proposed based on strong assumptions, as initial MBSN transition function definition (3.3.1). As a reminder, the first assumption is that the transition probability can be factorized into independent components:

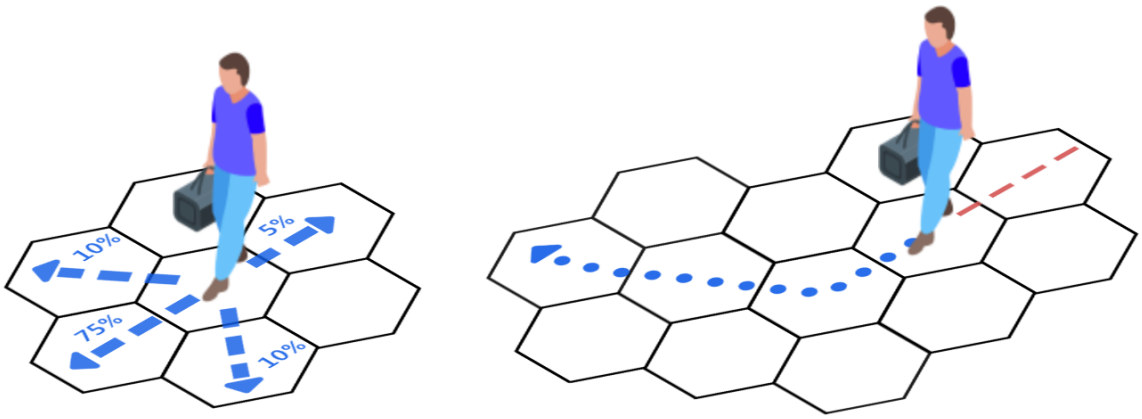
$$T(s^t, a, s^{t+1}) = P(p_r^{t+1} \mid p_r^t, a) \times P(g^{t+1} \mid g^t, a) \times P(H^{T_{p+1}:T_{t+1}} \mid H^{T_p:T_t}, a) \tag{3.15}$$

where p encodes the robot polygon, g the goal, and H the human states.

Second, robot and goal dynamics are largely deterministic. The robot reaches its intended next polygon with high probability: $P(p_r^{t+1} \mid p_r^t, a) = 0.9$ if p_r^{t+1} corresponds to action a, and 0.1 to remain in place (to capture possible delays). The goal is assumed to remain fixed, giving $P(g^{t+1} \mid g^t, a) = 1$.

Humans bring the most stochastic side of the MDP. Human information in the current state s includes the positions and orientations of the past and present within a history window $H^{T_p:T_t}$. A trajectory prediction model uses this brief history to predict future human states $H^{T_{p+1}:T_{t+1}}$. As a result, the robot's perception module predicts the potential trajectories of humans in addition to their current poses.

Two distinct prediction models are used in this study (Figure 3.10): (a) a simple model that generates very short-term probability estimates without considering position or orientation history, and (b) a learning-based goal prediction model that requires the last n observed positions to estimate the most likely future trajectories toward an inferred goal. This transition function's goal is to offer testing scenarios that are both realistic and difficult, rather than to replicate human behavior exactly.



(a) Basic Movement Prediction (b) Goal Prediction

Figure 3.10 – The two prediction models chosen for the transition function, in blue the possible future trajectories of the human and in red his past trajectory. (a) Simple general prediction with handwritten probabilities. (b) Goal prediction, with a trajectory estimation between the current position and the predicted goal.

3.4.1.4 Reward Function

We define a reward function to guide the robot toward socially acceptable and efficient paths. The function penalizes unsafe or inefficient behavior and rewards progress toward the goal. A severe penalty CP (collision penalty) is incurred if the robot collides with a human (i.e., if it occupies the same polygonal space). To account for the efficiency of trajectory planning, a penalty DP (distance penalty) is applied proportionally to the distance traveled by the robot. Additionally, a constant penalty TP (time penalty) for each action is introduced to discourage the robot from remaining stationary, waiting for humans to leave its vicinity. To guide the robot toward its objective, we provide a reward GR (goal reward) when it successfully reaches the goal, and also grant a reward based on whether the robot moves in the direction of the goal. Consequently, the reward function is defined as:

$$R(s^t, a, s^{t+1}) = TP + DP(s^t, a, s^{t+1}) + GR(s^t, a, s^{t+1}) + CP(s^t, a, s^{t+1}) \tag{3.16}$$

The distance penalty is proportional to the displacement of the robot:

$$DP(s^t, a, s^{t+1}) = -dist(p_r^t, p_r^{t+1}) \tag{3.17}$$

where p_r^t and p_r^{t+1} denote the robot positions before and after applying action a. Distances are computed between polygon centroids. The goal reward encourages progress toward the assigned objective:

$$GR(s^t, a, s^{t+1}) = \begin{cases} \text{Goal Reward}, & \text{if } p_r^{t+1} = p_g \\ dist(p_g, p_r^t) - dist(p_g, p_r^{t+1}), & \text{otherwise} \end{cases} \tag{3.18}$$

where p_g is the polygon containing the goal and Goal Reward is a high positive constant. Finally, collisions with humans are heavily penalized:

$$CP(s^t, a, s^{t+1}) = \begin{cases} \text{Penalty Collision}, & \text{if } \exists h \in H^{T_p:T_t} : p_r^t = p_h^t \text{ or } p_r^{t+1} = p_h^{t+1} \\ 0, & \text{otherwise} \end{cases} \tag{3.19}$$

with Penalty Collision a high negative constant and p_h^t the polygon occupied by a human h at time t in the history $H^{T_p:T_0}$. At the current time step t or next time step $t+1$, this collision penalty is intended to strongly prevent the robot from occupying the same polygon as any human.

3.4.2 Monte Carlo Tree Search to Solve MBSN

Solving the MBSN formulation online requires a planning method capable of handling stochastic human behavior and the need for real-time decisions. Exact dynamic programming methods, such as value iteration used in the first version of MBSN, are computationally difficult to leverage in this context because the state space grows exponentially with the number of humans. We therefore rely on Monte Carlo Tree Search (MCTS) [Kocsis et al. 2006], a simulation-based algorithm that progressively builds a search tree guided by statistical deployments.

MCTS simulates the futures by modeling the potential movements of both humans and robots in the surrounding environment. An estimate of the effectiveness of a particular action is provided by each simulation, and these estimates improve with each additional simulation. The algorithm determines a balance between exploitation (favoring actions that already appear promising) and exploration (trying out actions that haven't been tested much).

Concretely, MCTS iterates over four phases:

1. **Select**: During the selection process, a tree policy is used to search a node from the tree that is not fully expanded, meaning at least one of its child nodes has not been explored. Typically, the strategy selection is based on the Upper Confidence Bounds applied to Trees (UCT) algorithm [Kocsis et al. 2006], which balances state exploration and exploitation :

$$a^* = \arg\max_a \left(\bar{X}_a + C\sqrt{\frac{\ln N}{n_a}} \right) \tag{3.20}$$

   where:

   - $\bar{X}_a$ is the average reward of action a,
   - N is the total number of simulations from the parent node,
   - n_a is the number of times action a has been selected,
   - C is a constant controlling the trade-off between exploration and exploitation.

2. **Expand**: Expand the selected node by applying an available action (as defined by the MDP) to generate a new child node.

3. **Simulate**: Perform a complete random simulation of the MDP from one of the outcomes of the expanded node to a terminal state. This assumes the simulation is finite, although versions of MCTS exist where the simulation runs for a fixed time and then estimates the result.

4. **Backpropagate**: The value of the simulated node is then backpropagated up to the root, updating the value of each ancestor node along the path based on the expected value.

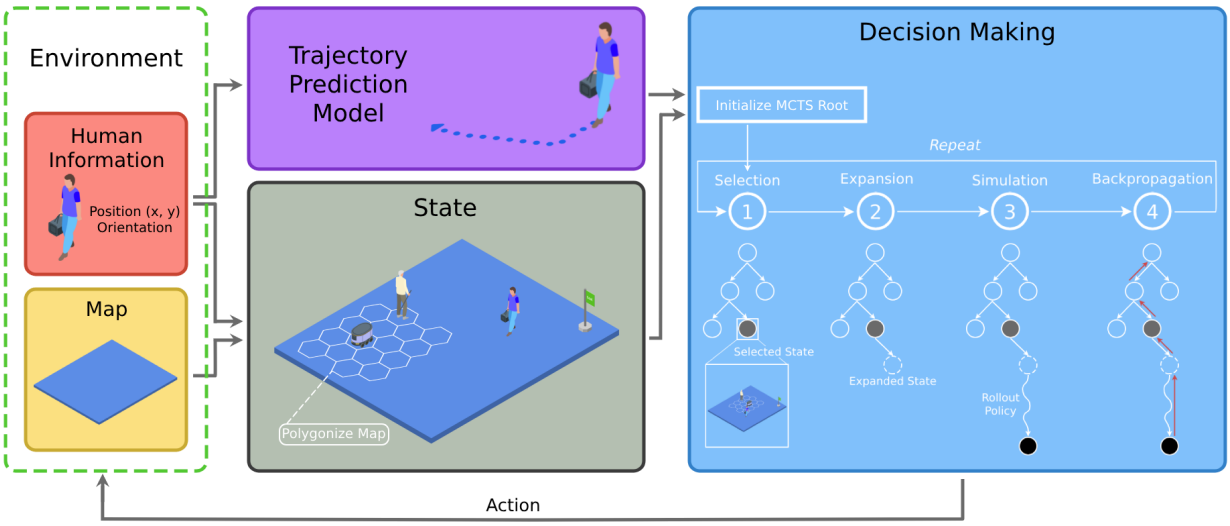


Figure 3.11 – Overview of the proposed solution: The map data and human information are used to define the MDP state. A prediction module processes human positions and orientations to output potential trajectories with probabilities. Using the environment state and predicted trajectories, the MCTS tree's root is initialized. The search iterates through MCTS steps until a *timeout* or sufficient state evaluations. Finally, the best action to perform in the input state is selected and returned to the agent.

Applying MCTS to solve MDPs online is non-trivial and presents several challenges. Encountering unvisited states triggers numerous rollouts (simulations) that should be treated within a short time to find a good enough policy. However, by refining the algorithm—such as avoiding duplicate states and applying heuristics during rollouts—this performance can be improved.

During the simulation phase of MCTS, a randomized exploration of the MDP is performed until a terminal state is reached. At each decision point, an action is randomly chosen from the available options, with transition probabilities P used to determine the outcome.

However, as mentioned earlier, purely random action selection slows down the process and produces poor, unrealistic simulations. To improve performance, we explore two heuristics: one that restricts actions based on human trajectories and another that assigns scores to social rules.

3.4.2.1 Action Restriction-Based Heuristic

The first heuristic is straightforward: it limits the set of actions to those that avoid current or future human positions. Once filtered, the A* algorithm is applied to compute the shortest path, and the first action along this path is selected.

Algorithm 2 Action Restriction-Based Heuristic

Input: $< S, A, R, T >$ - Markov Decision Process, s - current state
Output: a - best action to choose
if $H_s == \emptyset$ **then**
    return $A\_star(s, A(s))$
end if
$actions \leftarrow A(s)$
for all $h^{T_p:T_0} \in H^{T_p:T_0}$ **do** ▷ for each human with history
    $predicted\_traj \leftarrow prediction\_model(h^{T_p:T_0})$
    for all $pos \in predicted\_traj$ **do**
        if $p_{pos} \in actions$ **then**
            $actions \leftarrow actions \setminus p_{pos}$
        end if
    end for
end for
return $A\_star(s, actions)$

3.4.2.2 Score-Based Heuristic

The second heuristic calculates scores based on a given state s. The objective is to assign a score to each social rule on a scale from 0 to 1, where 1 represents full compliance and 0 represents non-compliance. The goal is to be able to propose a method that can integrate desirable social rules and to identify the most balanced action that respects these rules. Below, we outline the various social rules and explain how their scores are calculated, before detailing the method for selecting the appropriate action.

Goal Score. The goal score is assigned on a scale from 0 to 1, with a score of 1 given to the action that brings the agent closest to the goal, and 0 to the action that takes it the furthest away. To measure how close an action is to the goal, we calculate the distance between the polygon corresponding to the action and the polygon corresponding to the goal. By applying the min-max normalization to the set d_g, which represents the distances between each action and the goal (g). Formally, the goal score for a given action a is calculated as follows:

$$score_g(a) = 1 - \frac{d_{g_a} - d_{g_{min}}}{d_{g_{max}} - d_{g_{min}}} \tag{3.21}$$

Here, d_{g_a} corresponds to the distance between action a and the goal. Additionally, $d_{g_{min}}$ is the minimum distance between any possible action and the goal, while $d_{g_{max}}$ is the maximum distance between a possible action and the goal. The result is inverted by subtracting it from 1, as the goal is to minimize the distance.

Movement Score. This score represents the effort required for the robot to reach the next subgoal (polygon). Similarly to the goal score, the movement score uses min-max normalization to the set d_m, which represents the distances between the current robot position and the position represented by the action. The movement score is therefore calculated as follows:

$$score_m(a) = 1 - \frac{d_{m_a} - d_{m_{min}}}{d_{m_{max}} - d_{m_{min}}} \tag{3.22}$$

With, d_m the set of robot-action distances, d_{m_a} the distance between action a and the robot, $d_{m_{min}}$ the minimum distance in d_m and $d_{m_{max}}$ the maximum distance in d_m.

Passing Side Score. We chose to implement a social rule that favors passing an obstacle or agent on the right or left, depending on the cultural context. In our case, we favor going to the right [1]. To assign a score, we look at whether the chosen action is located more to the right or left of the agent closest to the robot. To calculate the score of an action, we examine the vector product between the $\vec{RA}$ vector (Robot-Action) and the $\vec{RH}$ vector (Robot-Human). If the result is greater than zero, the action is to the right of the human; if less than zero, the action is to the left; and if equal to zero, the action is directly in front of the human. We thus assign a score for each case:

$$score_s(a) = \begin{cases} 0, & \text{if } \vec{RA} \times \vec{RH} = 0 \\ 0.99, & \text{if } \vec{RA} \times \vec{RH} < 0 \\ 1, & \text{if } \vec{RA} \times \vec{RH} > 0 \end{cases} \tag{3.23}$$

Since our goal is to avoid other agents without disturbing them, a score of 0 is assigned when the robot moves directly toward the human. A slight preference is given between left and right when other scores are equal for two actions. This score helps differentiate between symmetrical actions and guides the decision-making process in such situations.

Social Space Score. The social space score corresponds to the rule of respecting the personal space of humans (HPS). If the robot respects the social space, the score will be 1. Otherwise, as the robot gets closer to the human, the score decreases and approaches 0. We can therefore define the social space score as being:

$$score_{near}(a) = \begin{cases} \frac{d_{near_a}}{HPS} * \exp^{-\alpha * (HPS - d_{near_a})^2}, & \text{if } d_{near_a} < HPS \\ 1, & \text{otherwise} \end{cases} \tag{3.24}$$

With d_{near} representing the set of distances to the nearest agent for each action, d_{near_a} corresponds to the distance to the nearest agent if action a is taken. The human personal space (HPS) is defined as a range from 0.45 to 1.2 m. The closer the robot is to the center of this space, the lower the score assigned. Parameter α controls how rapidly the score decreases as the robot enters the human personal space. Larger values of α produce a steeper decline,

[1] This is to respect the French social convention but it could be adapted to the left.

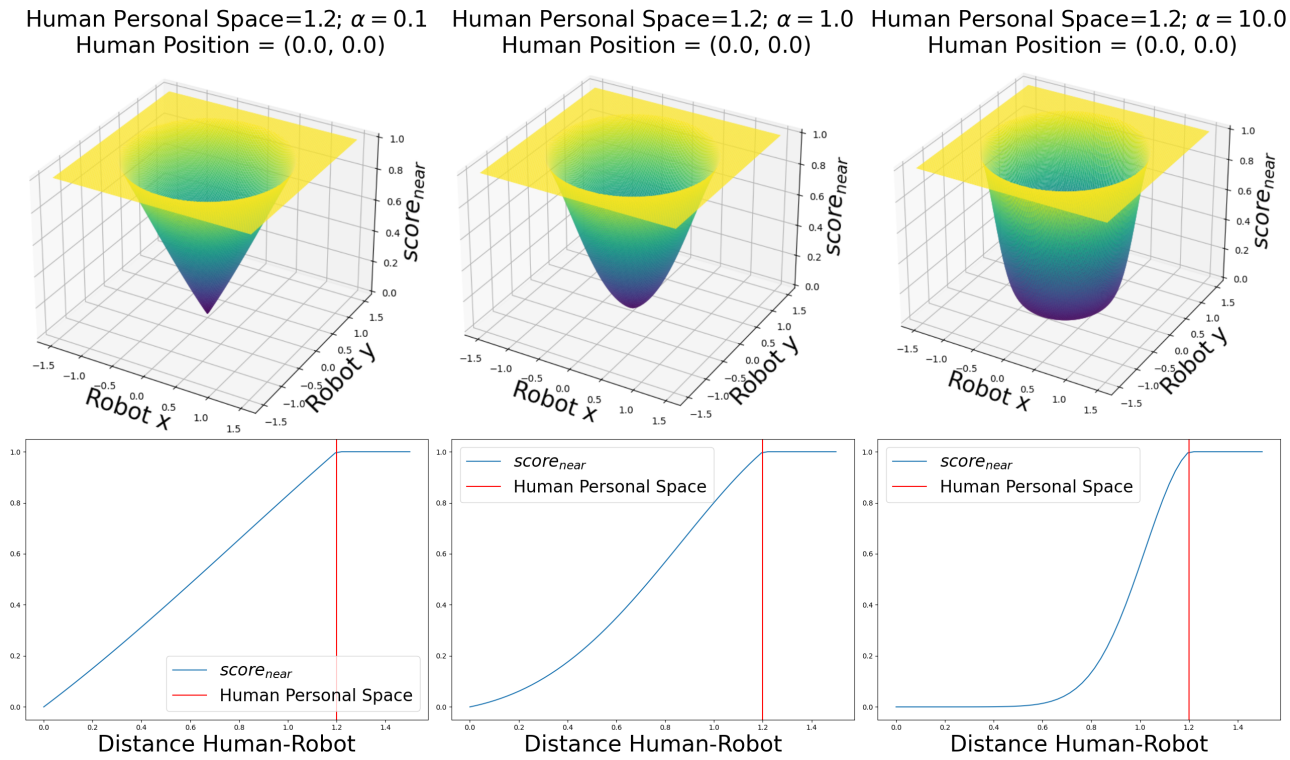


Figure 3.12 – An example of different α values when calculating $score_{near}$, ranging from 0.1 to 10, is shown from left to right.

meaning the score drops more quickly for small violations of HPS (see Figure 3.12).

Action Selection. With the different scores defined previously, we formalized the following four social rules: achieve the set goal, reduce effort as much as possible, prioritize moving to the right and finally respect for social space.

To determine an overall measure of rule compliance, we compute a weighted average of the scores for each potential action. However, we are unable to determine whether an action is balanced across all rules based solely on the average. In order to quantify the imbalance between the various criteria, we also compute the weighted standard deviation of the scores. A low standard deviation denotes a more balanced decision, whereas a high standard deviation shows that an action strongly favors some rules while ignoring others.

Finally, the heuristic policy based on the social-scores (π^{scores}) selects the action that maximizes the difference between the weighted average and a penalty proportional to the standard deviation:

$$\pi^{scores}(MDP, s) = \underset{a \in A}{\text{argmax}}(\overline{\text{score}}_a - k * \sigma_{\text{weighted}_a}) \tag{3.25}$$

where the trade-off between minimizing imbalance and maximizing the mean score is controlled by k.

The weighted average and weighted standard deviation are defined as:

$$\overline{\text{score}} = \frac{\sum_{i=1}^{n} w_i \cdot \text{score}_i}{\sum_{i=1}^{n} w_i} \qquad \sigma_{\text{weighted}} = \sqrt{\frac{\sum_{i=1}^{n} w_i \cdot (\text{score}_i - \overline{\text{score}})^2}{\sum_{i=1}^{n} w_i}} \tag{3.26}$$

where w_i are the weight assigned to each social rules.

Illustrative Example. Figure 3.13 shows a navigation scenario where a robot must select the best action in an environment with humans. **(a)** The goal is shown in gray star, the robot in gray diamond, the humans in gray triangle, and the robot's potential actions (3) in blue, orange and green. **(b)** The individual social rule scores for each action—goal proximity, movement effort, passing side, and respect for personal space—are displayed in the radar plot. **(c)** The weighted mean and the final score following the use of a standard deviation penalty are displayed in the bar plot, which compares the global scores of each action.

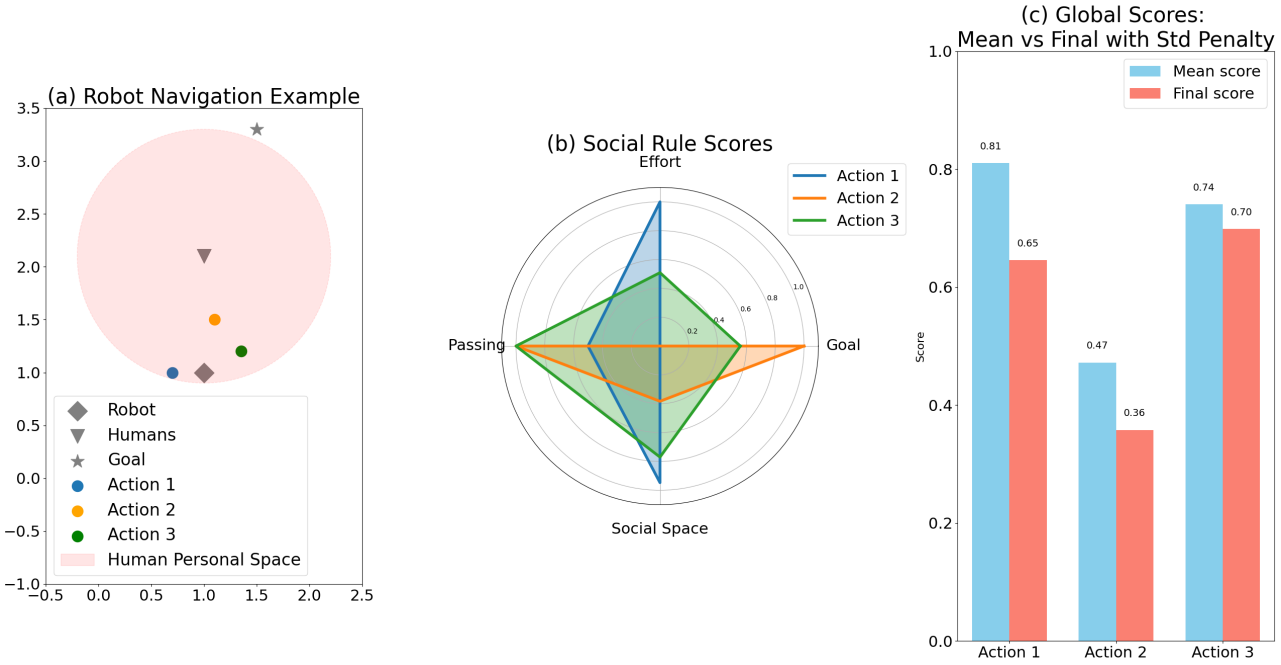


Figure 3.13 – Social navigation example: **(a)** Robot (blue), humans (red), goal (green), and possible actions (orange) with labels. **(b)** Radar plot of social rule scores for each action: goal proximity, effort, passing side, and personal space. **(c)** Bar plot comparing mean and final global scores (with standard deviation penalty) for each action.

This example illustrates the impact of the standard deviation term in directing socially conscious navigation by showing how an action with a high average score can be penalized if it violates the balance across social rules. Three options are available in this scenario, where the robot has just entered a human's personal space:

1. Action 1 (blue, Figure 3.13(a)): This action respects the human's personal space the best because it deviates from the human the most. It is less effective at achieving the objective, though.

2. Action 2 (orange, Figure 3.13(a)): This action is the most effective since it gets the robot

the closest to the goal, but it ignores the human.

3. Action 3 (green, Figure 3.13(a)): This action is a compromise, while trying to stay away of the human, it moves the robot closer to the objective.

Figure 3.13(b) where the radar plot compares their scores across the four criteria, makes the trade-offs between these actions very evident. Finally, we can see that Action 1 has the best mean score when we examine the global scores in Figure 3.13(c). However, after the standard deviation penalty is applied, the Action 3 obtains the highest final score, demonstrating a better compromise between effectiveness and human-aware behavior.

3.4.3 First Tests With MCTS

To evaluate our method, we first perform multiagent simulations, with an agent who use our method to navigate (robot) and other agents (humans) using simple A* path planning to navigate. For these experiments, here are the values of the parameters that we fixed:

- In the calculation of $score_{near}$, we are considering the Human Personal Space (HPS) equals to 1.0 meter and the parameter α equals to 0.8.
- For the calculation of the *score_based* heuristic, the parameter k was equal to 0.5 and the weights of the scores were respectively equal to $w_{goal} = 1.0$, $w_{movement} = 0.1$, $w_{side} = 1.0$ and $w_{near} = 1.5$.
- When utilizing MCTS, the simulation part duration was constrained, setting the $timeout$ to 1.0 seconds, to keep the online usable aspect of the solution and $discount\_factor$ to 0.99.
- There is a distance of 1 meter between each hexagon center. The polygon must be large enough to fit an agent (0.4m here) but not too large to avoid losing detail.

Two scenarios tested in this experiment. A first scenario in a multi-branch corridor setting, designed to compare and observe the different combinations of the trajectory prediction model, the heuristics applied and the use or not of the MCTS algorithm. A second scenario in an open space simulating navigation in a crowd of people, showing the effectiveness of the proposed method in such environments.

3.4.3.1 Multi-branch Corridor Scenario

In this scenario, we simulate a situation with the robot and the human face to face in a corridor with several exits, the human then has several possible goals, this scenario is described in Figure 3.14a. The objective of this experiment is twofold: first, to determine if the robot can navigate without disturbing the human in a deterministic frontal navigation scenario. Second, to examine the resilience of the solution when the human deviates from the trajectory predicted by the robot's prediction model. This scenario is described in Figure 3.14b.

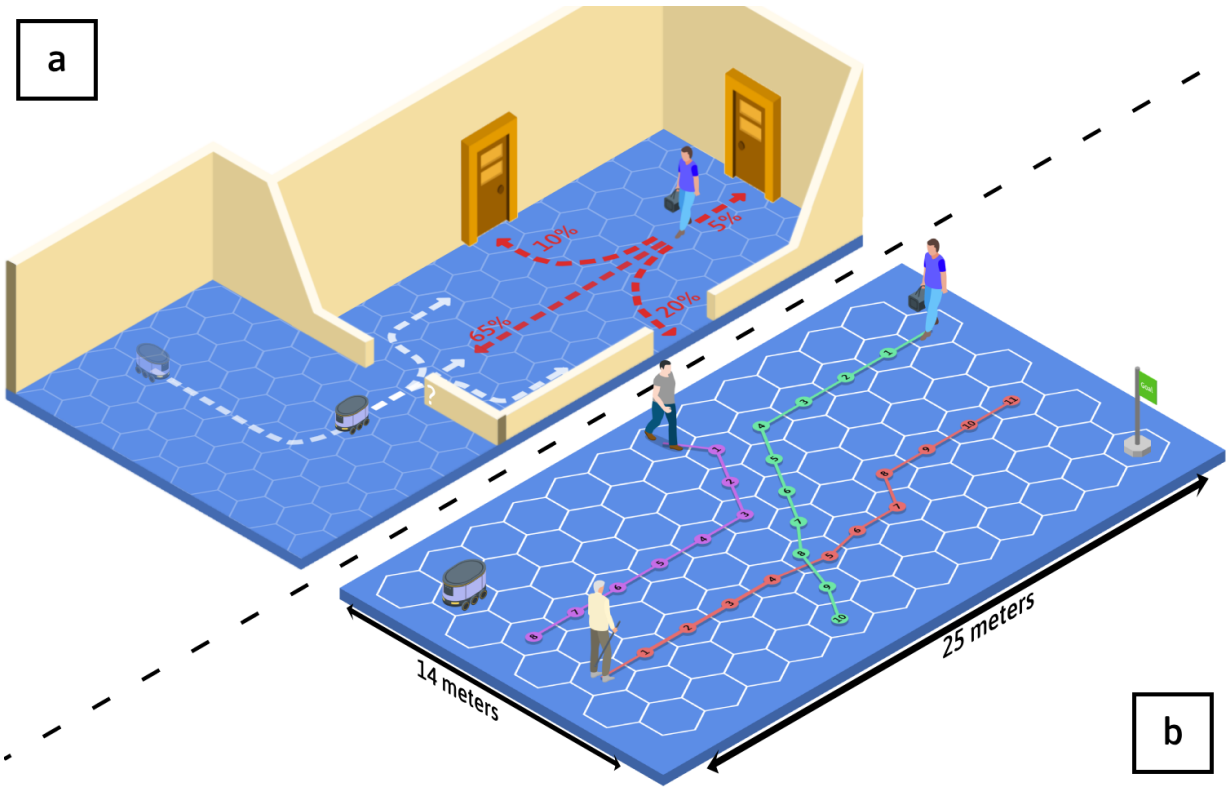


Figure 3.14 – Representation of the two scenarios for simulation experiments: (a) multi-branch corridor scenario and (b) open space scenario with several humans.

Deterministic Scenario. In this scenario, the human's goal is predetermined, their trajectory is completely predictable. Therefore, the evaluation focuses on comparing the robot's behavior under various decision strategies, including (i) using the heuristic alone, (ii) using MCTS guided by the heuristic, and (iii) changing the prediction model used. In a controlled and deterministic environment, this configuration enables us to separate the effects of the prediction model and the planning method (heuristic vs. MCTS).

The results are illustrated in Figure 3.15. We observe that across all experiments, no collisions occurred between the human and the robot, i.e., they were never on the same hexagon simultaneously. The score-based heuristic adheres to passing on the right side, unlike the heuristic that restricts actions to only those corresponding to the human's predicted future positions. Predicting the human's goal is more effective than local prediction, as foreseeing the human's path allows the robot to plan more efficiently. Integrating MCTS leads to slightly more cautious robot behavior and improved human anticipation compared to approaches that omit it, addressing a limitation noted in other solutions (2.2).

Unpredictable Trajectory Scenario. It has been shown that the robot performs better when using goal prediction rather than relying on local prediction. However, this section focuses on scenarios where the human's trajectory is unpredictable, or when the goal prediction model fails. As shown in the Figure 3.16a, the robot performs well when the prediction model is accurate. However, when the model is incorrect (Figure 3.16b), the robot's performance deteriorates, leading to instances where the robot encroaches on the human's path and cuts them off.

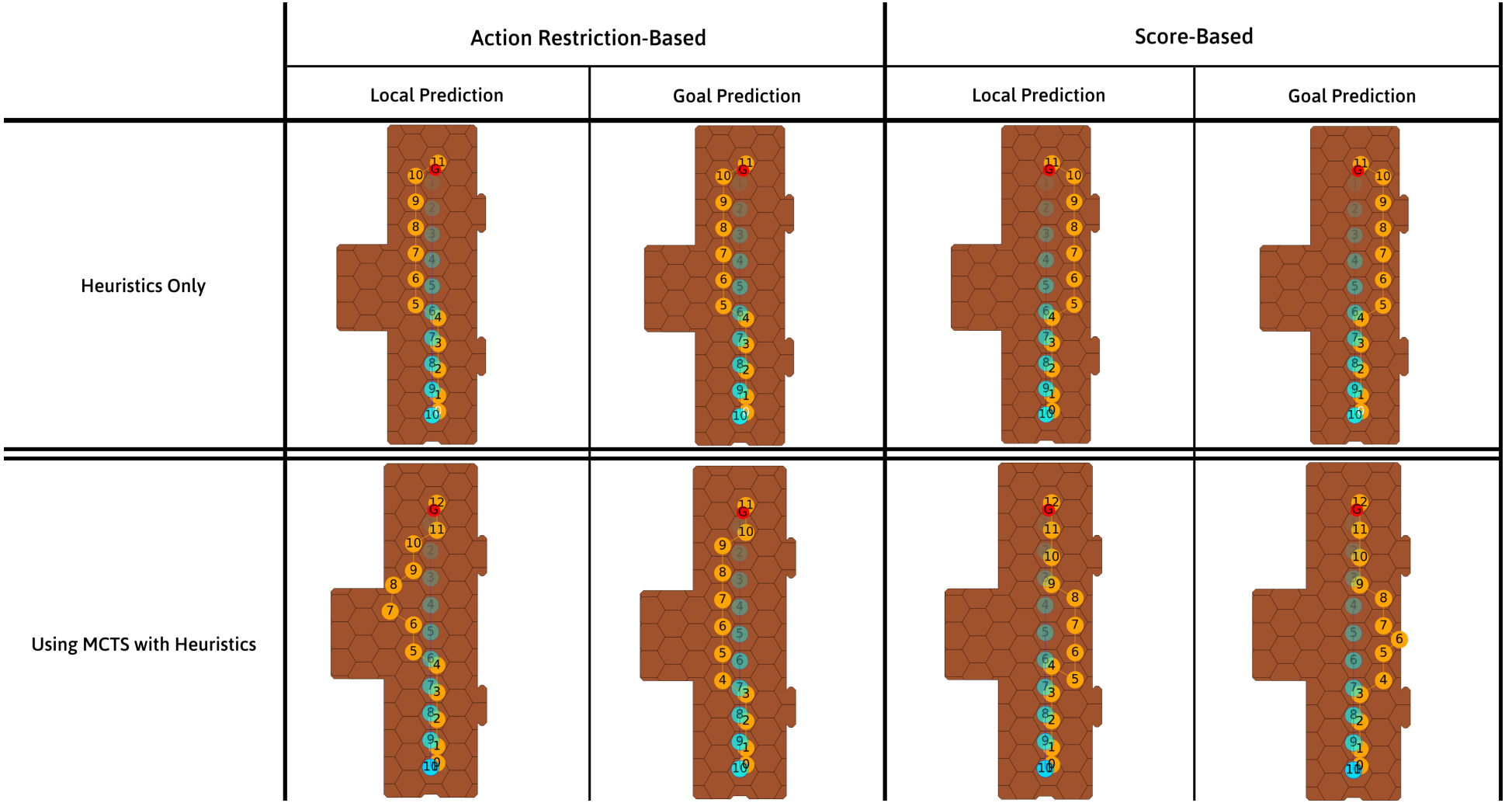
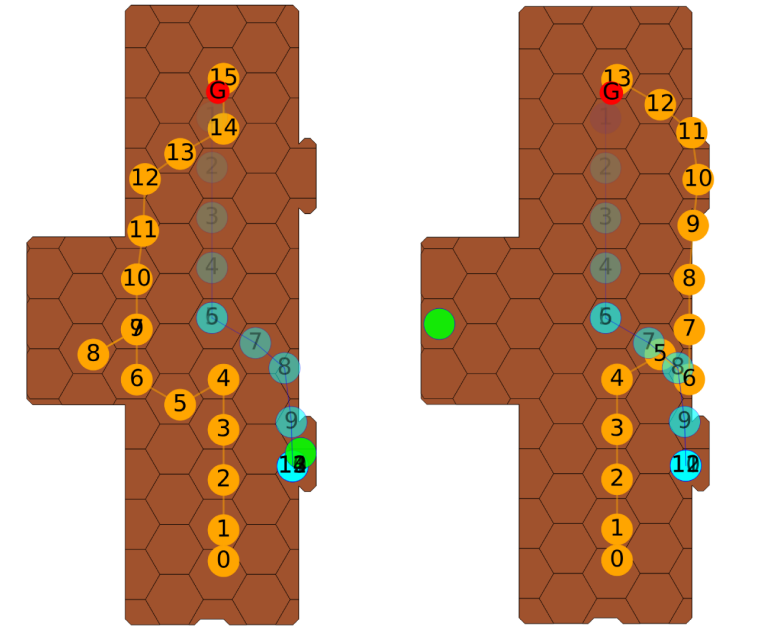


Figure 3.15 – Table of results of the corridor scenario according to the use of heuristics with or without MCTS and according to the prediction method used. The paths taken by the robot and the human are respectively orange and blue, the nodes correspond to the position at the time step written inside the node and in red the goal of the robot (G).



(a) Result when human path predicted by the robot is correct

(b) Result when human path predicted by the robot is incorrect

Figure 3.16 – Example of a scenario where (a) the goal prediction is correct and (b) where it is incorrect. With the path taken by the robot in orange, the path taken by the human in blue and the green circle corresponding to the predicted goal.

3.4.3.2 Open Space Scenario

In this scenario, we introduce an open space environment (a 14 by 25 meter room) where the robot navigates within a crowd, as shown in Figure 3.14b. To create interesting scenarios, the agent's starting position and goal are set far enough apart to require it to cross through the entire room. For this scenario, we used MCTS, applying a score-based heuristic along with a local prediction module. Our goal is to assess the scalability of using the MCTS algorithm and determine whether there are limitations based on the number of humans present in the robot's surroundings.

We varied the number of humans (h) in the scenario and analyzed both the minimum distance between the robot and the humans, as well as the number of collisions recorded throughout the experiments. The results are presented in Table 3.1. To comply with Human Personal Space (HPS) requirements, the robot must maintain a minimum distance from humans, ranging from 0.45 to 1.2 meters, depending on the context. To give an intuitive measure of crowd density, we also report the proportion of the total HPS area relative to the room's surface, assuming that the HPS regions of all humans do not overlap. In terms of personal space, this ratio gives an idea of how much of the space is "occupied" by people.

The results indicate that our agent maintains appropriate social distancing with up to 10 humans in the environment. However, performance degrades with more than 10 humans. Specifically, with 20 humans, we observed 9 collisions out of 50 scenarios, representing nearly 20% of the trials. This illustrates the increasing difficulty of navigating safely while respecting HPS as crowd density grows.

	h=3	h=5	h=10	h=20
Sucess Rate (%)	100	100	100	86
Average Minimal Distance Human-Robot (m)	2.81	2.36	1.22	0.84
Number of collisions	0	0	1	9
Proportion HPS (% of room)	2.7	4.5	9.0	18.0

Table 3.1 – The results per scenario, launched 50 times for each, where h is the number of humans present in the scenario.

We tracked the number of MCTS rollouts carried out for each decision step in order to assess scalability in relation to the number of humans. Figure 3.17 illustrates how the number of rollouts rapidly declines as the human population grows. The algorithm can only evaluate approximately one rollout per second (depending on the $timeout$) when there are more than three humans, which drastically lowers the number of actions assessed.

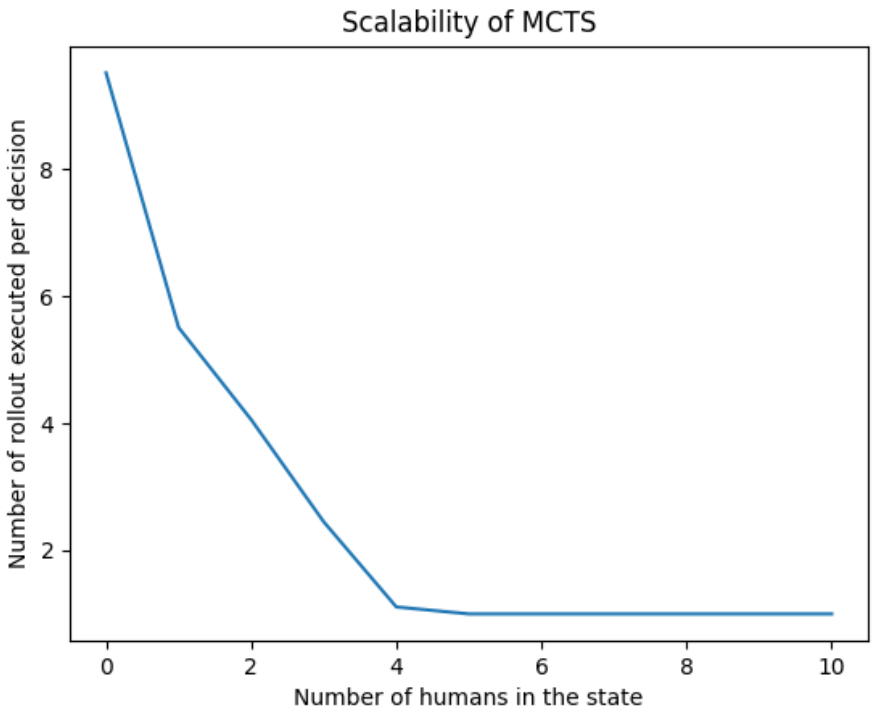


Figure 3.17 – Graph showing the scalability of the solution, giving the number of MCTS cycles per decision-making according to the number of humans present in the state.

In other words, scenarios involving more than three humans effectively force MCTS to rely almost exclusively on the heuristic when exploring with a one-second time budget. Instead of conducting a comprehensive search of the action space, the exploration heuristic plays a major role in determining the selected action because the algorithm only has time for one exploration.

3.4.4 Advantages/Limitations

This section introduced MBSNv2, a novel approach to online social navigation via polygonal MDPs and Monte Carlo Tree Search (MCTS). First, discretization on a polygonal grid, and more precisely a hexagonal one, allows for more consistent and accurate spatial discretization with a decreased dependence on an expert modeling. The homogeneous representation facilitates planning in diverse environments while providing more natural movement transitions for the robot even in dynamic, partially predictable environments. The use of the MCTS algorithm to solve the associated MDP online allows the agent to react dynamically to unforeseen events, such as the presence or movement of pedestrians.

However, this approach presents certain limitations. The main obstacle lies in the computational complexity of MCTS, which increases rapidly with the number of human agents to be considered. Beyond three or four humans, the number of simulations that can be performed in real time decreases sharply, reducing the value of tree-based search and making the agent highly dependent on exploration heuristics. Additionally, the transition model depends on strong, unlearned assumptions, and the quality of generated behaviors is still susceptible to human motion prediction errors. Lastly, social rules weighting parameters are manually determined and might not apply in different situations.

In order to overcome these constraints, we also present Heuristic-Based Social Navigation (HBSN), which builds upon the same polygonal discretization but substitutes a direct heuristic evaluation for the MCTS-based solver. Although this eliminates MCTS's planning capabilities, it significantly lowers computational requirements while, in many cases, still generating socially acceptable behaviors. To put it another way, HBSN can be thought of as a lightweight substitute for MBSNv2: less effective in theory, but competitive in practice in limited environments and more appropriate for real-time deployment.

3.5 Discussion and Limitations

In this chapter we focused on modeling the HAN problem using the MDP framework. The MBSN methods use a navigation graph and estimated human trajectories to generate a model of possible movements for the robot. The development of MBSN comes in two versions, and both demonstrate the potential of representing social navigation as an MDP to achieve interpretable, goal-oriented, and socially compliant robot behaviors. By explicitly modeling both the structure of the environment and the dynamics of human motion, our methods address several shortcomings identified in state-of-the-art techniques, particularly in constrained scenarios such as narrow passages.

The first proposed solution, MBSNv1, leverages an expert-defined navigation graph to reduce the complexity of planning and to exploit strategic waypoints in the environment. This representation proved effective in producing socially aware behaviors without requiring explicit predictions of human goals. However, its reliance on manual graph construction

limits adaptability to new environments. The MDP is solved using the Value Iteration algorithm, which guarantees convergence to an optimal policy in the discrete, finite setting considered. This formulation remains highly sensitive to the exponential growth of the state space with the number of nodes (n), which severely impacts scalability. The graph-based reasoning also introduces approximations in human–robot interaction modeling, which can result in overly cautious or, conversely, insufficiently cautious behavior in borderline cases.

The extended MBSNv2 addresses some of these limitations by replacing the manual graph with a regular polygonal grid (hexagonal grid), which modifies the MDP definition, and by computing online local policies using MCTS. This formulation provides uniform spatial resolution and enables smoother, more flexible robot motion. The online search further allows reactive adaptation to unforeseen events, such as sudden pedestrian movements. However, these advantages come at the cost of increased computational demands as the number of humans in the scene grows. By considering that three or four humans are present, the number of simulations per planning step is significantly reduced. Beyond four humans, the MCTS algorithm becomes less effective than relying solely on the exploration heuristic.

Furthermore, both version remain prone to errors in human trajectory prediction models and require manually defined parameters to control the trade-off between efficiency and social convenience. The other restriction that is shared by both of these algorithms is computational complexity. Despite the difference in representation, our experiments have shown that using conventional techniques such as Value Iteration (offline) or MCTS (online) remain complicated in real-world applications while the number of possibilities grows exponentially. However, even with this limitation, experiments on the *Heuristic Based Social Navigation*, HBSN, have shown that using only the heuristic based on social rule scores yields good results.

Finally, the current evaluation was primarily conducted in simulation and over a limited set of scenarios. Complementary results on additional scenarios, as well as comparisons with the state of the art using evaluation metrics, are presented in Chapter 5. Before that, the promising results and the limitations highlighted for MDP-based social navigation methods in this chapter point to the need for more adaptive and scalable planning. In particular, computational complexity and vulnerability to hand-crafted models motivate the exploration of learning-based approaches.

Chapter 4

MDHC: Multi-branch Deep Reinforcement-learning for Human-aware Control

Outline of the current chapter

4.1 Introduction

In Chapters 1 and 2, the current state of the art in social robot navigation demonstrates how Deep Reinforcement Learning (DRL) can achieve good performance in crowded scenarios. This approach allows robot-agents to learn socially acceptable behavior through experiences with human presence. However, existing approaches are generally too limited in their generalization: beyond training scenarios, the performance drops significantly.

In our previous work, we used model-based decision-making approaches with value iteration and MCTS algorithms. Although theoretically capable of handling a wide range of situations, these two approaches have proven computationally expensive, making their real-time application unlikely for negotiating dynamic social environments.

These factors justify the use of DRL as a central component of our next proposal. DRL provides the capability of learning policies that can function in high-dimensional and partially observable spaces while attaining optimal computational efficiency at runtime by utilizing the representational power of deep neural networks. Moreover, reinforcement learning allows us to tackle the problem by directly connecting to the actuators, thereby simplifying the overall navigation control architecture.

In this chapter, we present a novel framework for versatile social navigation. Our contributions are threefold:

1. We leverage HBSN heuristic to provide socially sub-goals within the robot's observations. Enabling the robot to reason over socially meaningful sub-goals rather than raw coordinates, which improves interpretability and decision-making.

2. We design a novel multi-branch neural architecture integrating complementary information: the robot's state, LiDAR perception, and two dedicated social modules. The first, our proposed Social Angular Field (SAF) module, is a novel contribution of this work, providing a structured representation of Human-Robot Interaction (HRI). The second, a Spatio-Temporal Graph (ST-Graph) to model Human-Human Interaction (HHI) over time, allowing the network to better anticipate human movements in dynamic social environments. This unified design progressively fuses complementary information across modalities through our own fusion mechanism, allowing the agent to reason jointly about its own state, the environment, and human dynamics for safe and socially compliant navigation.

3. We implement a novel Curriculum Learning (CL) strategy tailored to social navigation with the RobotSNAP simulator. By progressively increasing the number of humans and task difficulty, our approach is designed to reduce learning time, to facilitate stable policy learning and to improve generalization to unseen scenarios.

The chapter is organized as follows. First, we review the fundamentals of Deep Reinforcement Learning (DRL) and examine in details the limitations of current DRL social navigation

solutions. Next, we present our approach: Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC), describing its structure, the incorporation of the HBSN heuristic and the specific curriculum learning strategy. We then analyze various network variations to determine each component's contribution and highlight the interest of merging them. We conclude with a discussion about the findings, pointing out existing shortcomings, and discussing possible future works.

4.2 Fundamentals of Deep Reinforcement Learning

The Markov Decision Process (MDP) offers a formal model for sequential decision making, as described in Chapter 3: defining the state space S, the action space A, the transition probabilities $T(s'|s,a)$, and the reward function $R(s,a)$. Reinforcement Learning (RL) expands on this model by enabling an agent to interact with its environment in a trial-and-error manner and find the best policy π^* without being informed of the transition or reward function beforehand. The agent observes the current state s at each time step, selects an action a according to its current policy π, receives a reward r, and advances to the next state s'. The objective is to maximize the expected return, which is the sum of the discounted rewards. Two of the key ideas in reinforcement learning are the state-value function $V^\pi(s)$ (definition 3) and the action-value function $Q^\pi(s,a)$.

Definition 5 (Action-Value Function) *The action-value function $Q^\pi(s,a)$ for a policy π gives the expected cumulative discounted reward obtained by taking action a in state s and then following policy π thereafter:*

$$Q^\pi(s,a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)\right] \tag{4.1}$$

Where γ is the discount factor, the importance of short-term rewards compared to long-term ones.

In Chapter 3, we presented different planning methods, such as Value Iteration and MCTS, both assume full knowledge of the transition dynamics T and the reward function R. RL assumes that these elements are unknown and that the agent learns by interacting with the world. Deep Reinforcement Learning (DRL) avoids the constraints of high-dimensional state spaces by approximating policies and/or value functions with deep neural networks, allowing efficient learning in complex environments.

4.2.1 Model-Based vs Model-Free

Depending on whether they explicitly learn an environment model, reinforcement learning algorithms can be classified into the following categories:

- **Model-Based.** The agent learns then uses a model of the transition dynamics T and reward function R to simulate interactions and plan ahead, similar to the planning approaches of Chapter 3.

- **Model-Free.** The agent does not attempt to learn the model T or R explicitly, but directly estimates value functions or policies from experience. This is often more practical in complex or partially observable environments.

Model-free approaches rely on learning a simpler object (typically the action-value function $Q(s,a)$ rather than the transition function $T(s,a,s')$), directly targeting an efficient policy. In contrast, model-based approaches offer theoretical guarantees regarding policy computation. With an explicit model, policy computation is generally faster than policy learning, and optimality can be guaranteed in small or tractable instances. In the best case, model-free policy learning is proven to converge to optimality only in infinite time. However, learning a model from scratch remains challenging. Consequently, the majority of the literature, as our proposal, focuses on model-free methods.

4.2.2 Value-Based vs Policy-Based

In model-free reinforcement learning, a distinction can be made based on what is learned.

- **Value-Based** Using techniques like Q-learning [Watkins et al. 1992] and deep Q-networks (DQNs [Mnih, Kavukcuoglu, et al. 2013]), the agent learns an approximation of the optimal action-value function $Q^*(s,a)$ and determines the policy by choosing the action that maximizes $Q^*(s,a)$.

- **Policy-Based** The agent directly parametrizes and optimizes the policy $\pi(a|s)$, frequently with the help of gradient-based optimization techniques like REINFORCE [Williams 1992] or Actor-Critic algorithms (A2C/A3C [Mnih, Badia, et al. 2016], SAC [Haarnoja et al. 2018], PPO [Schulman et al. 2017]).

Value-based approaches are commonly employed in discrete action spaces, whereas policy-based approaches naturally extend to continuous or stochastic action spaces. Because they combine the benefits of both paradigms, actor-critical approaches are mostly used in HAN solutions [Xie et al. 2023; Liu, Xia, et al. 2024].

4.3 Limitations in Existing DRL Solution

In chapters 1 and 2, we discussed the limitations of learning methods in terms of their ability to generalize to different environments. In this section, we further detail why these limitations exist in DRL methods, which supports the design choices for our solution presented in the following section.

4.3.1 Static Obstacles Consideration

One of the key limitations raised in the literature is that few studies train agents in a variety of environments with different topologies. Conversely, most studies usually focus on relatively

simple scenarios, for instance, open spaces with few or no obstacles, leading to overfitting and a lack of generalizability to more complex setups.

As discussed in [Liu, Xia, et al. 2024], the environment interpretation is also very important for learning because it consists of the structure of the observation received by the agent. We can classify the studies into four categories:

- **Dynamic Agent Only.** Early work on HAN solutions does not consider static obstacles in the scene at all, considering only dynamic agents [Everett et al. 2018; Li, Xu, et al. 2019].
- **Shape Obstacles.** Some solutions take into account static obstacles in addition to humans, providing their positions in the environment [Liu, Chang, et al. 2021; Chen, Wang, et al. 2024; Liu, Xia, et al. 2024], but only obstacles with specific geometric shapes (circle, rectangle). It is very difficult to capture and represent real world environments (walls, furnitures, ...) with such simple geometric shapes.
- **Raw Sensor Data.** Alternative learning-based approaches leverage raw sensory inputs, such as RGB images [Dugas et al. 2022] or LiDAR point clouds [Pérez-D'Arpino et al. 2021], to encode a global representation of the environment. Nonetheless, achieving robust generalization to real-world deployments typically requires access to high-fidelity simulation environments and/or extensive, carefully annotated datasets.
- **State Information.** Other work transforms raw information into processed state information, such as occupancy maps [Xie et al. 2023]. However, compared to human circles or point clouds, this greatly expands the state space, reducing learning efficiency.

Ultimately, only methods that take raw data or transform them into a state of information are able to have realistic observations and, therefore, have possible simulation to real capability. That is why we designed our solution to directly take as input LiDAR-type and human-like information. And our simulation platform RobotSNAP can provide realistic values for both these information across multiple environments.

4.3.2 Goal Reward Shaping

A frequently overlooked constraint involves the definition and implementation of the reward function. In most social navigation research, the progress reward towards the goal, represented as r_g, is conventionally designed as:

$$r_g^t = \begin{cases} r_{goal}, & \text{if } d_{goal}^t \leq g_d \\ r_{fail}, & \text{if } t > t_{limit} \\ d_{goal}^{t-1} - d_{goal}^t & \text{otherwise} \end{cases} \tag{4.2}$$

where $r_{goal} > 0$ is a positive reward for achieving the target, $r_{fail} < 0$ is a penalty for failure (often $r_{fail} = -r_{goal}$), and d_{goal}^t is the distance between the robot's location and the goal at

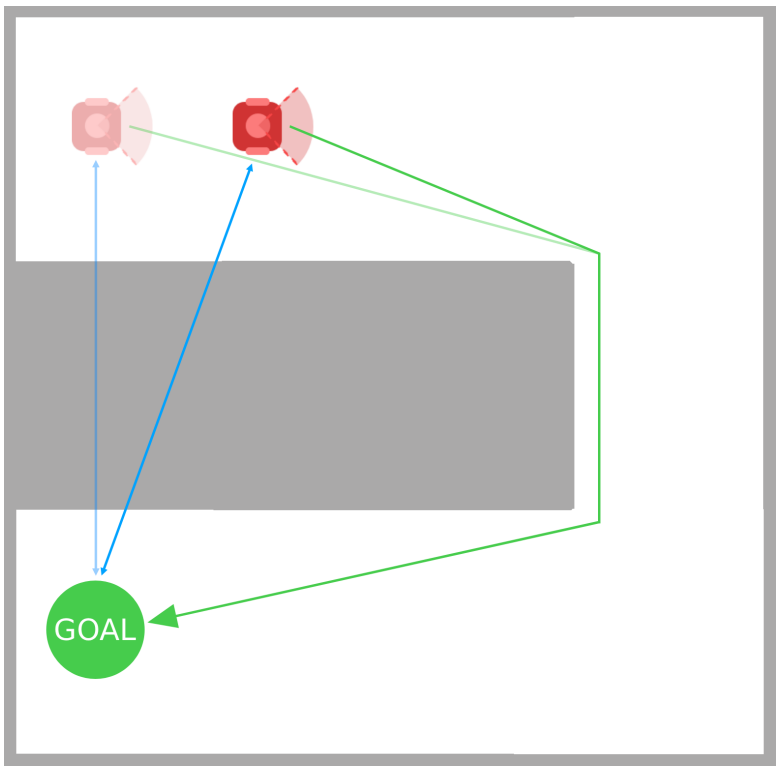


Figure 4.1 – Navigation scenario in a U-shaped environment. The Euclidean distance to the goal (blue) incorrectly increases from time $t-1$ to t, even though the robot has no alternative path. In contrast, the shortest-path distance (green) correctly reflects progress toward the goal.

timestep t. This model of shaping rewards the robot for getting closer to its objective and penalizes it for moving away from it.

Although this formulation is popular and has been proven effective, it typically depends on the Euclidean distance from the robot to the goal [Liu, Chang, et al. 2021; Xie et al. 2023; Liu, Xia, et al. 2024]. The robot is then penalized when it must temporarily deviate from the direction toward the goal to reach it, as in a U-shaped topology shown in Figure 4.1. This is what causes the collisions recorded in corner or narrow passage scenarios in the benchmark with DRL methods (2.2). Few studies address this problem by measuring progress using the global path distance in the environment [Pérez-D'Arpino et al. 2021], which better reflects the true cost of navigation.

4.4 Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC)

We discussed the challenges of enhancing the robustness DRL approaches for the HAN problem in the previous section. To adresse these limitations, we now introduce our DRL solution: Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC). This solution is designed to be reliable and adaptable to a wide variety of topologies and environments.

We now outline the key contributions of our approach, which are subsequently detailed in this section. First, we leverage the HBSN heuristic to generate social sub-goals, provided to the agent as observations. Second, we propose a multi-branch model that integrates: social interactions via Social Angular Field (SAF), spatio-temporal dynamics via Spatio-Temporal Graph (ST-Graph), robot state, and environmental perception to better understand human behavior. By combining these complementary observed data, our approach captures social and environmental topology, thus enabling more informed and socially responsible robotic decision-making. Finally, using RobotSNAP as a simulator, we implement a Curriculum Learning (CL) approach, progressively increasing complexity through realistic building layouts, varying the number of humans, and scenario-specific tasks (e.g., frontal approach, corners, intersections) to train the agent in increasingly challenging and socially aware environments. Our architecture is described in Figure 4.2.

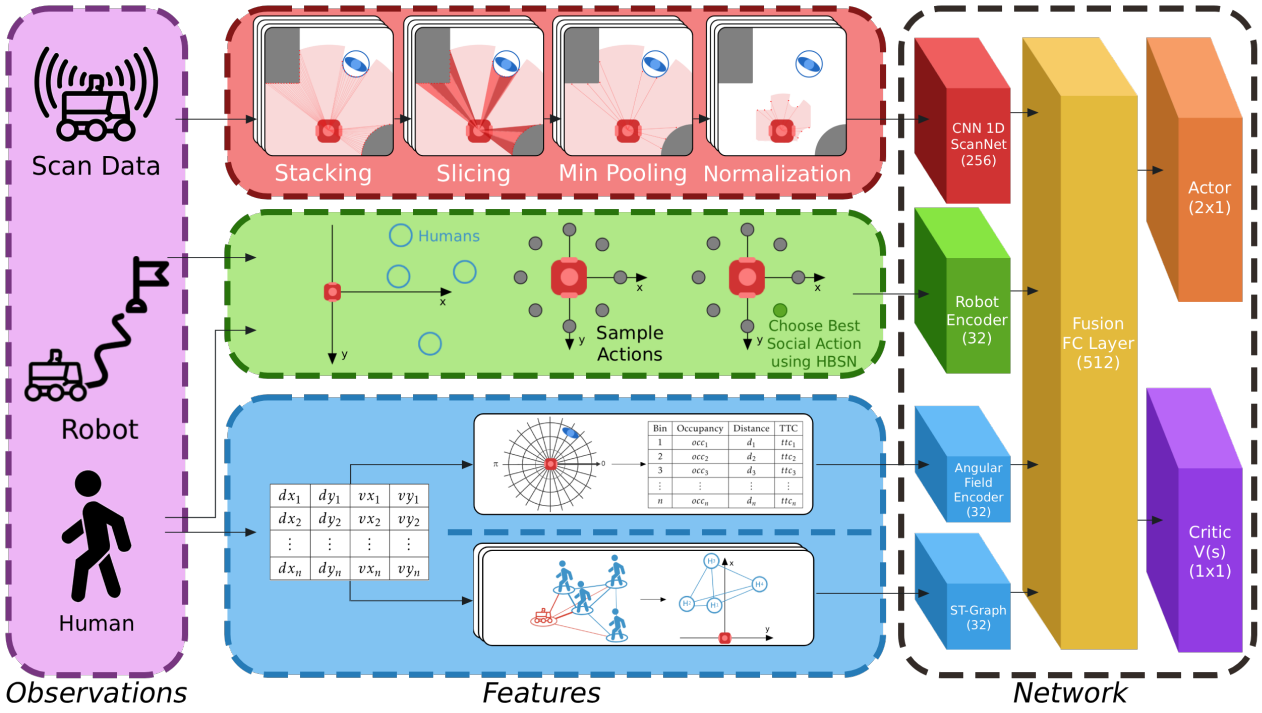


Figure 4.2 – Overview of the proposed deep reinforcement learning architecture for socially-aware robot navigation. The model processes scan data, robot state, and human motion features through multiple encoders: a 1D CNN for laser scans, a robot encoder, a Social Angular Field (SAF) encoder for HRI, and a Spatio-Temporal Graph (ST-Graph) for human interactions. The encoded features are fused in a fully connected layer, from which the actor outputs navigation actions and the critic estimates the state value $V(s)$.

This section is structured as follows: (1) We formulate the HAN problem as a tuple of states, actions and rewards ; (2) We explain in detail the network architecture of our solution; (3) the learning algorithm used; and finally (4) our learning setup and the use of CL.

4.4.1 Formulation

In the context of DRL, we maintain our formulation of the HAN problem as an MDP, but with an unknown transition function. Training an agent that learns a policy to navigate effectively toward its goal while adhering to social constraints imposed by human presence is the aim. The agent chooses an action from a continuous action space that governs its navigation dynamics at each timestep, receives an observation that describes the social and environmental context, and receives a reward that strikes a balance between social compliance and task efficiency. In the following subsections, we detail the components of this MDP, introducing the agent's state space, its action space, and the reward function.

4.4.1.1 State Space

Compared to the MBSN method presented in the previous chapter, by using model-free learning, it is possible to leverage the need for an explicitly defined transition function. As a result, it is possible to define a state space with all the elements at our disposal characterizing a situation. In HAN, the agent must reason about its own state, the environment, and the behavior of nearby humans, with an integration of those information as close as possible to sensor capacity. For this reason, we divide the state space into three main components: Robot, Scan and Human.

Robot Two main features are represented in the Robot component:

- Velocity: The current linear v and angular w velocity of the robot. These features allow the agent to account for its own dynamics when planning future actions and ensure smooth motion.
- Sub-goal Angle: Rather than including the goal position or absolute distance to the global goal, we compute a local waypoint in the robot's frame, like in [Xie et al. 2023], and encode the angle toward this sub-goal:

$$\theta_{\text{local}} = \frac{\arctan 2(y_{\text{local}}, x_{\text{local}})}{\pi} \tag{4.3}$$

  In more detail, the *local* waypoint is formed as a social sub-goal using our HBSN solution. To do so, we sample several points around the robot in a circle and apply the heuristic to this set, along with the robot's current position, to obtain the sub-goal that best adheres to social conventions while performing the task (Figure 4.3). Once this sub-goal is obtained, we calculate the angle as above. This normalized angle provides directional guidance to the agent without needing the explicit distance, helping it align its motion toward the next sub-goal efficiently.

Together, these features form the robot state vector $[v, w, \theta_{\text{local}}]$. By focusing on velocity and local sub-goal orientation, the agent has compact yet sufficient information to plan motion while keeping the state vector concise.

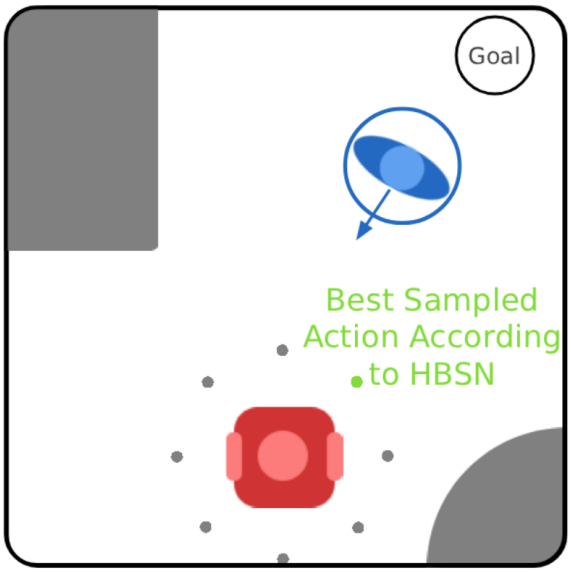


Figure 4.3 – Sampled actions (in gray) and selection of the best sub-goal (in green) according to HBSN.

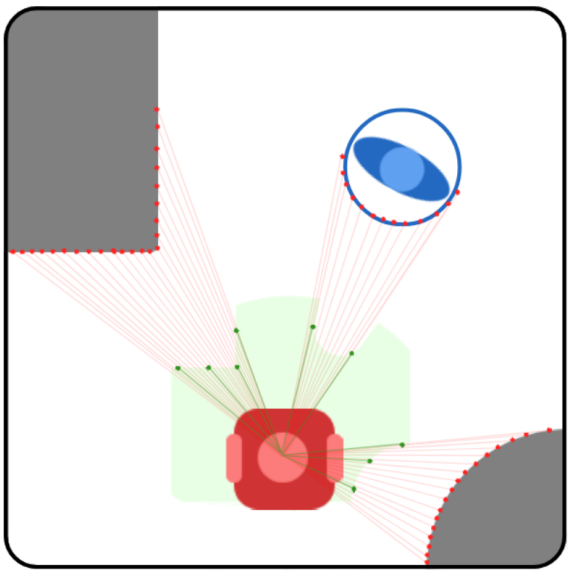


Figure 4.4 – Showing raw scan data (in red) and after pre-processing (in green).

Scan To capture the static layout of the environment (walls, furniture, free space), we use a 2D LiDAR scan as input. The raw LiDAR provides N range measurements $\{s^1, s^2, \ldots, s^N\}$ uniformly distributed over 360°. To improve robustness and reduce dimensionality, we therefore process it as follows:

- Stacking: Consecutive sensor frames are stacked to provide temporal context, allowing the agent to infer motion of dynamic obstacles.
- Slicing: To reduce dimensionality and highlight nearby obstacles, the stacked data is cropped or sliced to focus on the most relevant field of view.
- Pooling: Min pooling is an example of an aggregation operation that compresses data while maintaining spatial patterns.
- Normalization: To improve learning stability and ensure that various sensors contribute similarly to the agent's decisions, normalization is achieved by scaling features to a common range.

Raw sensor data is converted by this process into a condensed representation that can be used for learning. The converted data are represented in Figure 4.4. The pipeline is based on [Xie et al. 2023], where scan stacking and normalization have demonstrated better results.

Human Since humans are the main dynamic agents in the environment, socially conscious navigation requires precise modeling of their state. We represent each human with a vector containing $[d_x, d_y, v_x, v_y]$, with d_x and d_y the relative position of the human with respect to the robot's current location, and v_x and v_y the relative velocity of the human in the robot's reference frame.

Using relative coordinates has several advantages: (1) The agent's decisions are invariant to global position, focusing on spatial relationships; (2) Temporal features such as predicted collisions or time-to-collision can be computed directly from these vectors if needed.

Explicit inclusion of distance or other derived quantities is optional because the agent can infer them from d_x and d_y, reducing redundancy while keeping the state compact. Moreover, this representation naturally scales to multiple humans by concatenating vectors or using an attention mechanism to handle variable numbers of humans.

4.4.1.2 Action Space

The agent selects actions in the form $[v, \omega]$, where $v \in [0.0, 1.0]$ m/s is the linear velocity and $\omega \in [-2, 2]$ rad/s is the angular velocity. Stable learning in continuous control is made possible by normalizing actions using a maximum absolute value scaling.

4.4.1.3 Reward Function

The robot is guided toward its goal by the reward function, which also ensures social compliance and safety. The total reward is defined as follows at each timestep t:

$$R_t = R_{\text{goal}}(t) + R_{\text{collision}}(t) + R_{\text{heading}}(t) + R_{\text{human}}(t), \tag{4.4}$$

where each term encodes a specific objective of the navigation task.

Goal progress and arrival reward. As discussed in the previous section, unlike standard approaches that compute the distance to the goal using the direct Euclidean metric [Xie et al. 2023], we evaluate the progress towards the goal along the precomputed global path. This prevents the agent from exploiting locally shorter but infeasible trajectories (e.g., crossing walls), and mitigates local minima in the reward. Let $\mathcal{P} = \{p^0, p^1, \ldots, p^N\}$ denote the sequence of waypoints defining the global path from the robot to the goal. The distance-to-goal at timestep t is defined as the cumulative Euclidean distance along the path:

$$d_t = \sum_{i=k}^{N-1} \|p^{i+1} - p^i\|_2, \tag{4.5}$$

where k is the index of the waypoint closest to the current robot position. The goal reward is then defined as:

$$R_{\text{goal}}(t) = \begin{cases} r_{\text{goal}}, & \text{if } d_t \le r_{\text{threshold}}, \\ -r_{\text{goal}}, & \text{if } t \ge T_{\max}, \\ r_{\text{progress}} \cdot (d_{t-1} - d_t), & \text{otherwise}, \end{cases} \tag{4.6}$$

with $r_{\text{goal}} = 100$ and $r_{\text{progress}} = 5$.

As a result, the agent gets a dense progress reward that is proportionate to the path distance decreasing at each step, a positive terminal reward when it reaches the goal region,

and a penalty when it doesn't within the time horizon. The model places a high priority on completing goals successfully while providing detailed feedback to direct learning during navigation by setting $r_{\text{goal}} \gg r_{\text{progress}}$.

Collision and obstacle proximity penalty Safety is enforced by penalizing collisions and near-collisions with static obstacles. At each timestep t, the robot perceives its surroundings through a 2D LiDAR scan, represented as a set of range measurements:

$$\mathcal{S}_t = \{s_t^i \mid i = 1, \ldots, M\}, \tag{4.7}$$

where M is the number of scan rays and s_t^i is the measured distance along ray i at time t. The minimum distance to obstacles is then defined as:

$$\delta_t = \min_i s_t^i. \tag{4.8}$$

The collision penalty is given by:

$$R_{\text{collision}}(t) = \begin{cases} r_{\text{collision}}, & \delta_t \leq r_{\text{robot}}, \\ r_{\text{scan}} \cdot \left(\alpha r_{\text{robot}} - \delta_t\right), & \delta_t < \alpha r_{\text{robot}}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.9}$$

where r_{robot} is the robot radius, $\alpha > 1$ is a scaling factor controlling the safety margin, $r_{\text{collision}} = -100$ the collision penalty and $r_{\text{scan}} = -1.5$.

Heading Alignment To encourage efficient trajectories, the robot is rewarded for aligning its heading θ_t with the direction of the local sub-goal. Let $\theta_t^* = \arctan 2(g_y - p_{t,y}, g_x - p_{t,x})$ denote the desired heading. The alignment term is:

$$R_{\text{heading}}(t) = r_{\text{angle}} \cdot \max\left(0, \theta_{\text{thresh}} - |\theta_t^* - \theta_t|\right), \tag{4.10}$$

where $\theta_{\text{thresh}} = \pi/6$ defines an angular tolerance and $r_{\text{angle}} = 0.2$.

Human Proximity Inspired by proxemics [Hall 1969], the robot is penalized for intruding into the personal space of humans. For each human j at a distance of d_t^j from the robot:

$$R_{\text{human}}(t) = -\sum_j r_{\text{human}} \cdot \max\left(0, d_{\text{safe}} - d_t^j\right), \tag{4.11}$$

where $d_{\text{safe}} = 1.0$ represents the minimum socially acceptable distance and $r_{\text{human}} = 10.0$.

4.4.2 Multi-Branch Network Architecture

To process the heterogeneous observations described in the previous subsection, we design our solution as a multi-branch neural network that integrates social, spatio-temporal and environment information. The overall network architecture is shown in Figure 4.5.

Each branch specializes in a particular modality: the Social Angular Field (SAF) encodes the immediate geometry of human-robot interaction, the Spatio-Temporal Graph (ST-Graph) captures the Human-Human Interaction (HHI), the robot state branch encodes the robot's own dynamics and sub-goal information, and the perception branch process and encodes the LiDAR environmental context. The resulting fused representation is then passed through a final fully connected layer before being used by the actor head to predict continuous navigation actions.

As discussed in Section 4.3, recent approaches in Human-Aware Robot Navigation (HAN) using Deep Reinforcement Learning (DRL) suffer from one or more limitations: Some focus solely on crowd situations without considering static obstacles, such as NaviSTAR [Wang, Wang, Mao, et al. 2023] or Her-DRL [Zhou et al. 2024], or an unrealistic consideration, such as HEIGHT [Liu, Xia, et al. 2024], which requires providing the number of obstacles present in the robot's local environment and their bounding boxes. Others incorporate human positions but treat humans independently, failing to capture HHI, such as DRL-VO [Xie et al. 2023]. Without knowledge of HHI, the robot makes decisions that are less consistent with social norms, such as crossing a group of people or mispredicting human trajectories when moving in groups.

On the other hand, our multi-branch architecture overcomes these limitations by:

1. Capturing proxemic and social context through the explicit modeling of HRI (SAF);

2. Predicting collective human movements by encoding HHI over time (ST-Graph);

3. Preserving a branch dedicated to the robot state and one for LiDAR observations, thus preserving realistic robot-specific data: self-awareness, sub-goal information, and the surrounding environment;

4. Combining various modalities in an organized manner to make sure that data on social, temporal, environmental, and self-state factors all work together to inform value estimation and policy.

The limitations of previous methods that ignored interactions, lacked temporal modeling, or performed ad hoc data fusion are overcome by this design, which allows the robot to navigate in a socially aware, contextual, and anticipatory manner.

In this subsection, we provide a conceptual overview of each branch and the fusion mechanism, giving the rationale behind each design choice. The full mathematical details are reported in Appendix B.1.

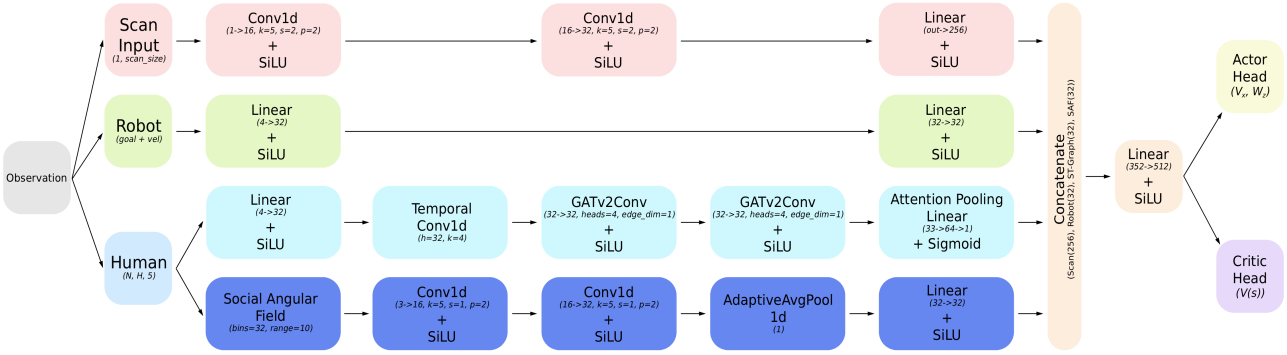


Figure 4.5 – Architecture of MDHC, a multimodal policy network. Observations are decomposed into three modalities: LiDAR scans (red), robot state (green), and human states (blue). Human information is processed through two complementary encoders: the ST-Graph for modeling human dynamics and interactions (light blue), and the proposed SAF, which summarizes human-robot interaction (dark blue). Features from all branches are concatenated and passed to actor and critic heads for policy and value estimation.

4.4.2.1 Robot State

The robot state branch encodes the agent's internal information, which consists of linear and angular velocities and the angle towards the social sub-goal. This low-dimensional vector is written as:

$$s_r = [v, \omega, \theta_g], \tag{4.12}$$

and processed through a Fully Connected (FC) network with ReLU (Rectified Linear Unit) activations:

$$h_r = \phi(W_r s_r + b_r), \tag{4.13}$$

where ϕ is the non-linear activation. The resulting embedding h_r is then passed to the fusion module.

4.4.2.2 LiDAR

The LiDAR branch treats the pre-processed scan vector $s_l \in \mathbb{R}^n$ described in the section 4.4.1.1. A 1D-CNN extracts higher-level features by capturing local spatial patterns:

$$h_l = \text{CNN}(s_l), \tag{4.14}$$

Since the input already encodes short-term temporal dynamics, spatial proximity, and normalized distances, the network focuses on extracting higher-level features that summarize the environment.

4.4.2.3 Social Angular Field (SAF)

The Social Angular Field (SAF) branch is one of our key contributions to this model, a new representation designed to encode the spatial distribution of humans nearby relative to

the robot in an angular format. Similar to polar histograms used in classical navigation [Borenstein et al. 1991; Ulrich et al. 1998], the SAF discretizes the surrounding space into K angular bins (i.e., fixed-width angular sectors) radiating from the robot, in contrast to previous works that model interactions through Cartesian occupancy grids [Everett et al. 2018; Li, Xu, et al. 2019; Pérez-D'Arpino et al. 2021] or pooling mechanisms like social pooling [Xie et al. 2023]. Each angular bin summarizes three key aspects: occupancy, distance, and time-to-collision (TTC). By explicitly capturing both spatial distribution and imminent risk of interaction, this design goes beyond standard occupancy-only encodings. A lightweight 1D CNN then encodes these bins into a latent vector h_{saf}, allowing the network to reason about crowding and immediate risk in a socially aware manner.

To summarize, SAF provides a learnable, compact, and robot-perspective representation of human-robot interactions, which to our knowledge has not been studied in state-of-the-art social navigation techniques.

4.4.2.4 Spatio-Temporal Graph (ST-Graph)

While SAF captures the "instantaneous" HRI, it ignores how humans move over time. We construct a Spatio-Temporal Graph (ST-Graph) to complements it by explicitly modeling motion dynamics and human-human interactions. The use of ST-Graph in the HAN problem is not new [Mohamed et al. 2020; Wang, Wang, Mao, et al. 2023], but is often considered without obstacles other than humans.

For each human in proximity to the robot, we consider its trajectory over the last H time steps. An MLP followed by a 1D temporal convolution encodes short-term movement patterns into compact embeddings. At the current time step, humans are represented by nodes in a fully connected spatial graph, where edges encode pairwise Euclidean distances. We then propagate the information across the graph using a graph attention network (GATv2) [Brody et al. 2021], allowing each node to refine its embedding based on its own movements and interactions with nearby humans. Finally, the node embeddings are aggregated into a single vector using an attention-based clustering mechanism, which highlights humans that are nearby or more relevant in terms of collision risk. This results in the ST-Graph embedding h_{st}, which captures temporal movement patterns and social interactions, and is passed to the multimodal fusion module for decision making.

4.4.2.5 Fusion Module

The embeddings produced by each branch, namely h_r, h_l, h_{saf}, and h_{st}, are concatenated and passed through a fully connected (FC) layer to obtain a unified representation. This operation projects the modality-specific embeddings into a common latent space, ensuring that information from the different inputs is jointly exploited. Finally, the fused vector h_f captures complementary information from all modalities in a compact form and is then used as the input to the policy network.

4.4.3 Learning Algorithm

Our multi-branch network is trained using Proximal Policy Optimization (PPO) [Schulman et al. 2017], a policy-gradient reinforcement learning algorithm well-suited for continuous action spaces. We implement PPO using the Stable Baselines3 (SB3)[1] library [Raffin et al. 2021], which provides stable and efficient RL components.

During training, the network observes the concatenated embeddings from the robot state, LiDAR, SAF, and ST-Graph branches, and outputs continuous linear and angular velocities $[v, \omega]$ for navigation. The agent receives rewards based on goal progress, collision avoidance, heading alignment, and human proximity, as defined in the reward function section. A value network (critic) is jointly optimized to reduce variance in the policy gradient updates, while the policy network (actor) is updated via PPO's clipped surrogate objective.

To handle the complexity of social navigation, training follows a curriculum-based approach, which is detailed in the next subsection.

4.4.4 Training Setup and Curriculum Learning

To train the agent, we design a custom Gymnasium[2] [Towers et al. 2024] environment that encapsulates a ROS2 node to interface with our benchmark RobotSNAP. This integration allows us to simulate realistic robot dynamics and human interactions while remaining compatible with standard reinforcement learning pipelines. Furthermore, we enable parallel training by running 10 environments simultaneously, which significantly increases sample efficiency and reduces training time.

RobotSNAP provides a diverse set of maps (map set) and scenarios, making it well-suited for curriculum-based training. Curriculum Learning (CL) is a training strategy where task complexity is progressively increased to improve stability and convergence. In our case, the curriculum is defined by gradually varying two factors: (i) the number of humans present in the environment, and (ii) the complexity of the maps used from the HouseExpo dataset [3][Tingguang et al. 2019] (introduced in Chapter 2). As shown in Figure 4.6, this progression allows the agent to first learn basic navigation skills in simple scenarios (frontal approaches, corner, dead-end, ...) before being exposed to dense and more challenging social environments. The overall learning schedule of MDHC is given in Appendix B.4.2.

Since HouseExpo contains thousands of floorplans with varying complexity, we design a scoring function to automatically categorize maps into three difficulty levels: easy, medium and hard. To do so, we compute a weighted difficulty score for each map and thresholds (based on mean and standard deviation) are used to split maps into the three levels. We randomly sample a fixed number of maps per level into subsets (up to 100), ensuring diversity while maintaining balance between difficulty classes. The implementation of this procedure,

[1] https://stable-baselines3.readthedocs.io
[2] https://gymnasium.farama.org/
[3] https://github.com/TeaganLi/HouseExpo

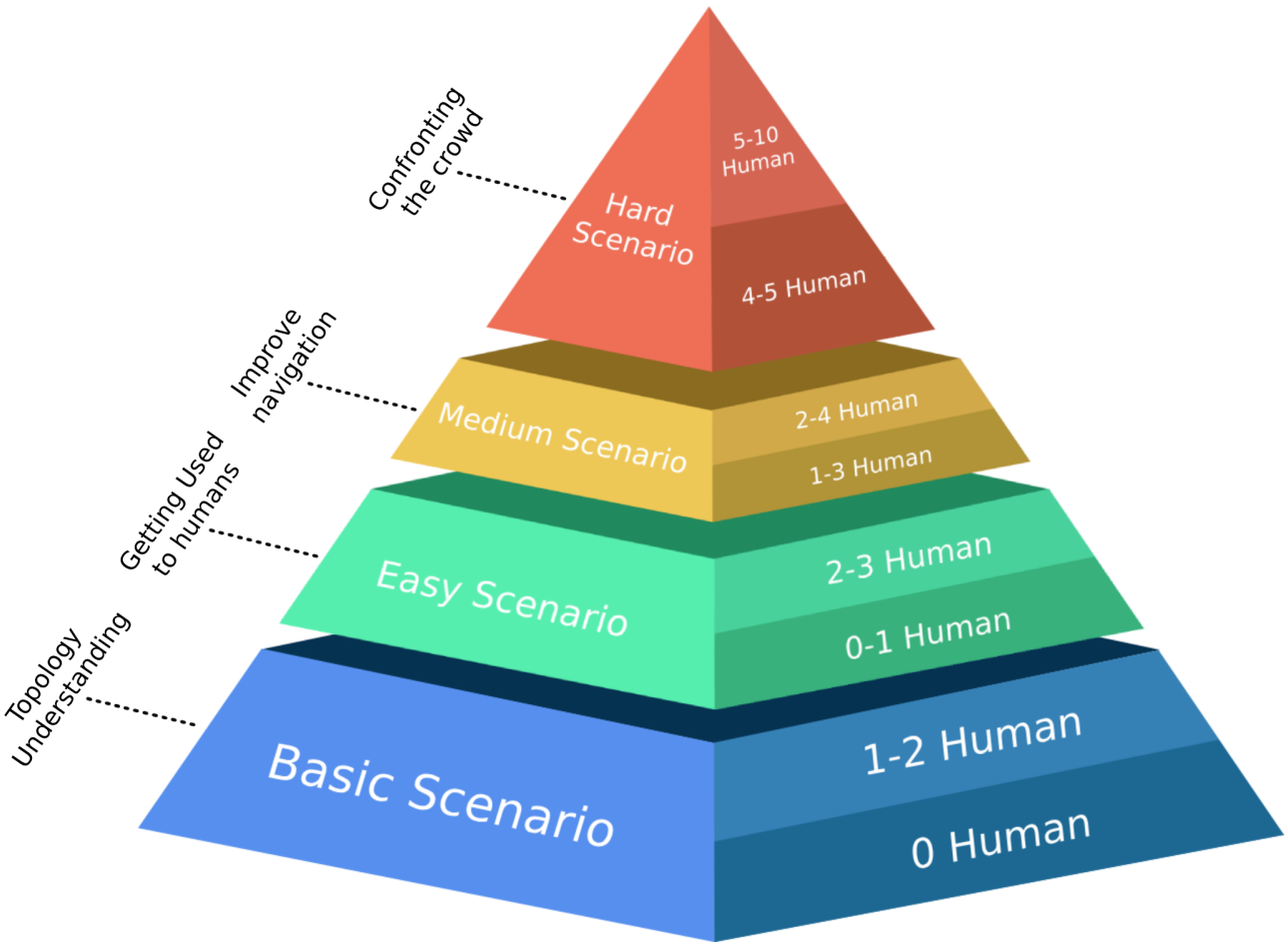


Figure 4.6 – Training difficulty evolution. The agent starts at the base with simple scenarios and few humans, ending in difficult environments with crowds.

including the difficulty scoring and sampling, is provided in Appendix B.3.

With this setup, we ensure that the agent experiences a wide variety of navigation challenges, enabling a fair comparison of different network variants on both simple and crowded scenarios, as presented in the next section.

4.5 Training Comparison of Network Variants

Before training the overall model, we assessed the advantages of our architecture in relation to its component parts (SAF, ST-Graph). By comparing results with and without the use of CL, we also demonstrate the impact of this architecture in this section.

4.5.1 Impact of Components on Learning

To assess the contribution of each architectural component, we trained and compared four network variants:

- **Baseline:** a minimal architecture without SAF or ST-Graph where humans are encoded with an MLP + Transformer and attention pooling,

- **SAF-only:** an architecture using the LiDAR, robot state and the SAF branch,
- **ST-Graph-only:** an architecture using the LiDAR, robot state and the ST-Graph branch,
- **MDHC:** the complete multi-branch architecture.

Training was performed on a basic set of scenario maps including frontal approach, corner, intersection, and dead-end situations. The positions and tasks assigned to the agents (robots and humans) were randomly assigned from a manually created realistic pool. In total, we had 16 different scenarios spread across four different maps. While the map set was fixed throughout training, the number of humans was progressively increased following a curriculum scheme:

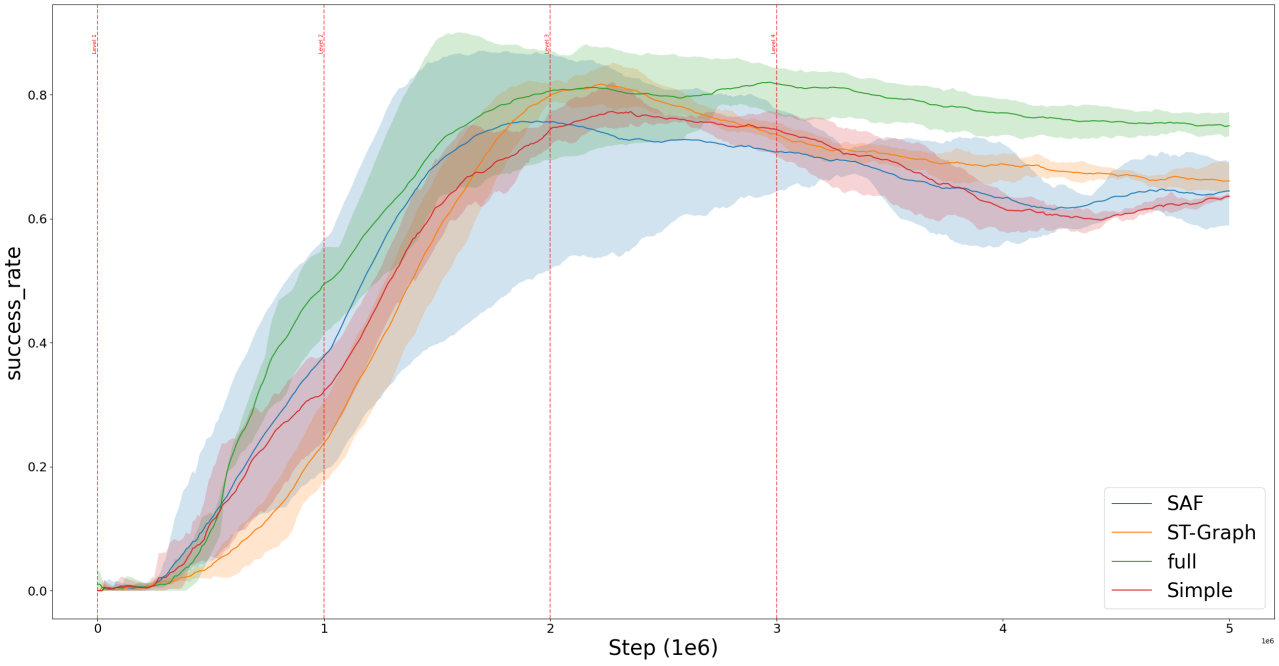
1. Steps 0–1,000,000: no humans,
2. Steps 1,000,000–2,000,000: 0 to 1 human,
3. Steps 2,000,000–3,000,000: 0 to 3 humans,
4. Steps 3,000,000–5,000,000: 0 to 5 humans.

This incremental curriculum allowed the agents to first master basic goal-reaching behaviors in empty environments before gradually learning HAN. Training was stopped after 5 million steps, which was sufficient to reach convergence and observe a stable performance plateau. All hyperparameters and implementation details are provided in Appendix B.

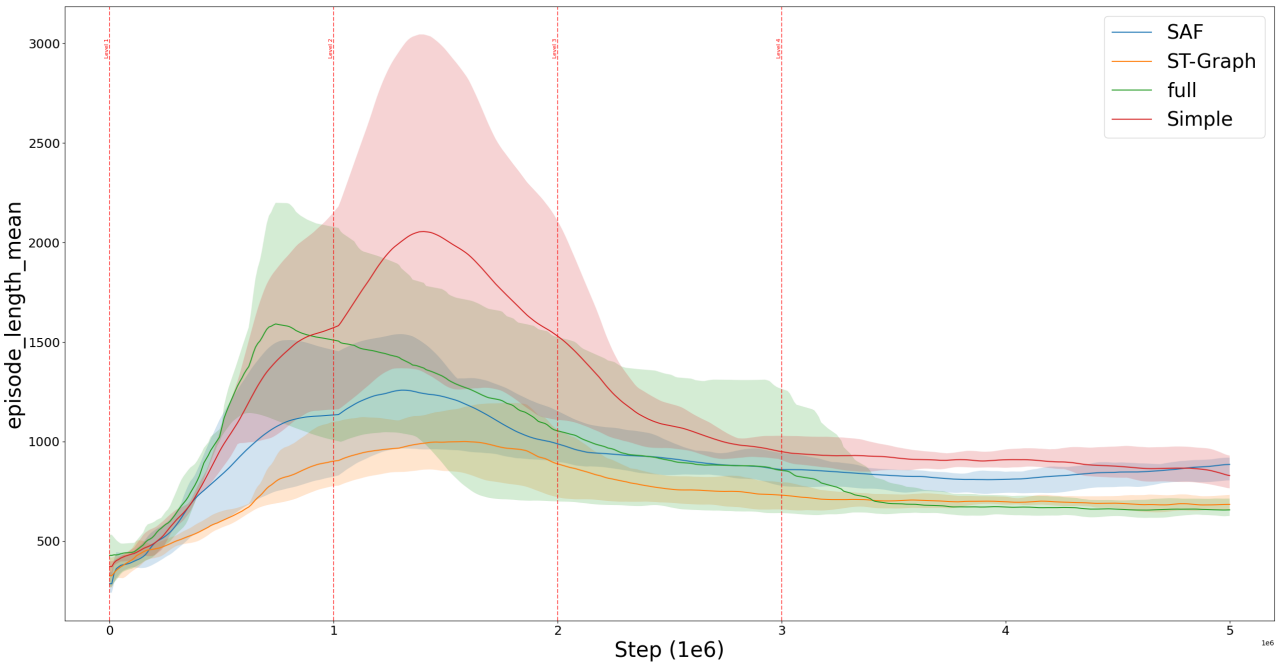
Results The training performance of the different network variants is displayed in Figure 4.7a and 4.7b showing the success rate and average episode length. CL levels are indicated with the vertical dashed lines.

The simple baseline has the lowest success rate (0.6) and generates extremely lengthy and unstable episodes, which are indicative of ineffective exploration. Though they still fall short of the full model, SAF-only and ST-Graph-only produce more consitent behaviors and modest gains (0.65-0.7 success rate). All variations are clearly outperformed by the full architecture, which combines SAF and ST-Graph. It consistently benefits from each CL stage, converges to shorter, more efficient episodes, and achieves the highest success rate (>0.75). We can still conclude that ST-Graph has a stronger positive contribution than SAF in the architecture. However, with ST-Graph in the architecture, the training time is doubled; for example, for 5 million steps, the training takes from 5 to 10 hours.

These findings demonstrate that although each element has a role, integrating them into the entire model is crucial for achieving both better success rate and quicker convergence.



(a)

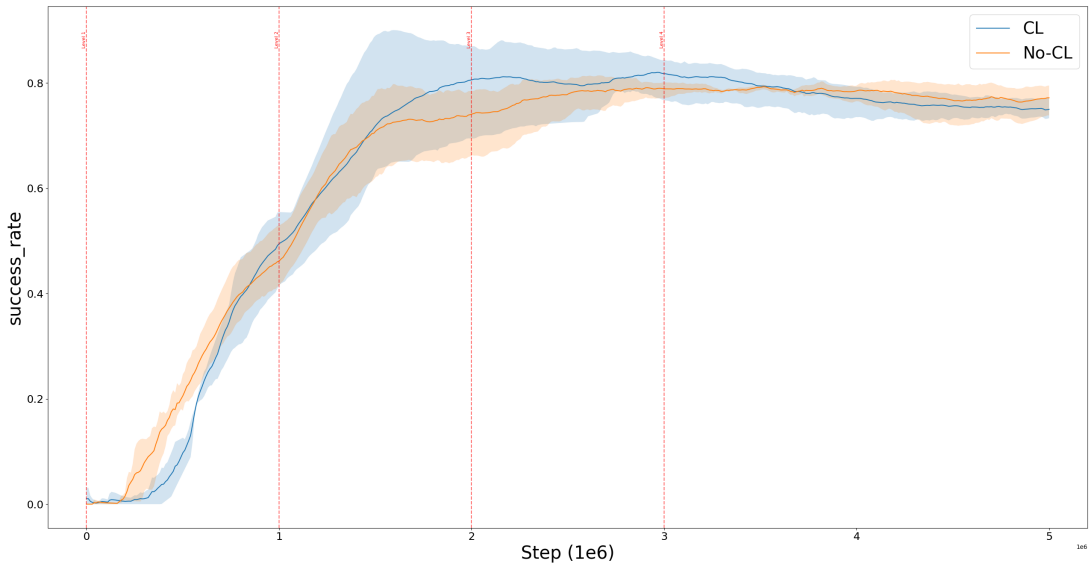


(b)

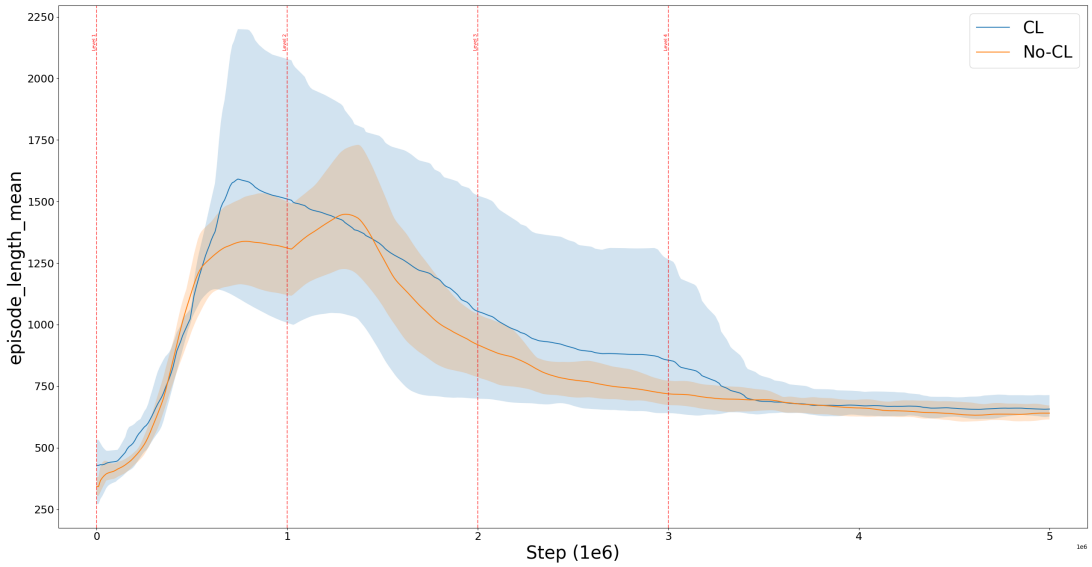
Figure 4.7 – (a) success rate (success_rate) and (b) Mean episode length (len_mean) during training for different network variants. Vertical dashed lines indicate CL levels.

4.5.2 Curriculum Learning Impact

To demonstrate the interest of CL, we experimented under the same environment and the same architecture (MDHC) with and without CL. Like the previous subsection, we took the basic scenario map set and ran a learning with CL (increasing number of humans) and without CL (directly with the maximum number of humans). The experimental results are presented in Figures 4.8a and 4.8b.



(a)



(b)

Figure 4.8 – (a) success rate (success_rate) and (b) Mean episode length (len_mean) during training with CL and without CL. Vertical dashed lines indicate CL levels in (a).

Both approaches eventually converge to a comparable performance level in terms of success rate (Figure 4.8a). Regarding the average episode length (Figure 4.8b), the CL agent first exhibits more exploratory behavior by reaching longer episodes before settling into shorter and more stable trajectories.

4.6 Discussion and Limitations

In this chapter, we introduce a new solution that addresses two key challenges: the high computational cost of model-based approaches discussed in Chapter 3, and the robustness limitations commonly observed in DRL methods from the literature.

The proposed multi-branch architecture MDHC explicitly accounts for both static obstacles, using LiDAR, and human interactions by separating human–robot interactions through SAF and human–human interactions through an ST-Graph. The robot can simultaneously reason over social context and environmental constraints due to this structured representation, which is rarely achieved in existing work. In particular, the SAF module offers an innovative approach to modeling interactions that are directly related to navigation decision-making, making it a unique contribution to the field of HAN. By modeling Human-Human Interaction (HHI), the ST-Graph branch enhances this and enables the agent to more accurately predict human behavior in dynamic environments.

We employed a Curriculum Learning (CL) approach to increase learning effectiveness and generalization. Stable policy learning and improved adaptation to unseen environments are made possible by progressively learning the robot to manage increasingly difficult situations by increasing the number of humans and the complexity of the scenarios.

We also experimented with a more sophisticated fusion strategy based on a multi-layer cross-modal transformer (Appendix B.2). While this approach allows the network to explicitly model inter-modality dependencies, in our experiments it did not yield consistent improvements and proved less robust than the simpler concatenation-based fusion, highlighting that sometimes simpler structured architectures are more effective in practice.

Preliminary results showed that the CL did not actually lead to faster convergence or better stability, but it did achieve the same results. While this doesn't result in better performance in the end, starting with simple scenarios makes it easier to solve complex scenarios, which is useful when practicing learning in real-life situations. The robot could then learn with few or no human input, reducing the learning time required in crowds.

Further results indicate that the proposed architecture, combined with CL, achieves faster convergence and better policy stability compared to naive baselines. The observed success rate (~75%) is still below expectations, perhaps showing a lack of parameter tuning and insufficient learning. It is unclear whether this limitation results from the method itself, from the lack of training time and parameter tuning, or from the intrinsic difficulty of the tasks assigned, as the robot is evaluated in a wide range of environments.

These results pave the way for the next chapter, in which we systematically compare the MDHC method with state-of-the-art approaches, providing a quantitative assessment of its effectiveness and identifying areas for further improvement.

Chapter 5

From Simulation to Reality: Assessing and Positioning Contributions

Outline of the current chapter

5.1 Introduction

In the previous chapters, we studied state-of-the-art solutions (Chapter 2) using RobotSNAP, providing an initial insight into the performance of existing methods. In Chapter 3, we introduced two model-based solutions: MBSN based on Monte-Carlo Tree Search and HBSN relying solely on heuristic decisions. Finally, in Chapter 4, a model-free solution was defined using DRL (MDHC).

This chapter's objective is to offer a comprehensive evaluation of these contributions. Through a structured set of experiments, we highlight the strengths and weaknesses of each approach, and position them with respect to the state of the art. This evaluation follows a progressive methodology: initially, we examine MBSN in reproducible, controlled settings intended to isolate particular facets of social navigation. We then extend the study to HBSN

and MDHC, which are evaluated on more extensive randomized benchmarks, enabling us to look into generalization, scalability, and robustness. In conclusion, we provide an initial validation of our methods in a real world scenario and with real robots, illustrating both the transferability of simulated results and the challenges of moving from simulation to reality.

Simulation provides the advantage of scalability and statistical robustness, while real-world experiments reveal practical difficulties such as sensing noise, dynamic unpredictability, and hardware limitations. Together, these results offer a dual perspective: detailed insights into social reasoning in controlled settings, and evidence of feasibility in real-world navigation. We first present the results obtained in simulation, before turning to real-world experiments.

5.2 Benchmarking in Simulation

We begin by evaluating our contributions in simulation, comparing them against the state-of-the-art methods that showed the most consistent results in Chapter 2.

Our first contribution (MBSN) was evaluated under controlled, reproducible scenarios involving a single human. This choice was motivated by the exploratory nature of this method: we aimed to isolate and analyze the key aspects of socially-aware decision-making in a simplified setting.

For our subsequent approaches (HBSN and MDHC), the experimental framework and benchmark were extended, allowing us to perform large-scale evaluations with multiple randomized scenarios, and dozens of repetitions.

These two experiments are carried out in simulation with RobotSNAP[1]., evaluated according to the same metrics used in Chapter 2: success rate, collision with static obstacle, path length, time to reach (the goal), error distance (to the goal), linear jerk, human collision, minimum distance to human and minimum time-to-collision.

While the first evaluation emphasizes fine-grained analysis under controlled conditions, the later experiments provide statistical robustness and generalizability. Together, they offer a complementary validation of our contributions.

5.2.1 MBSN Evaluation in Sparse Scenario

The MDP-Based Social Navigation (MBSN) method, introduced in Chapter 3, was our first contribution in the context of social awareness in robotic navigation. Its design focuses on explicit modeling of human behavior and decision-making through a dedicated planning module.

To evaluate this approach, we focused on sparse scenarios: frontal approach, narrow passage, corner and intersection; involving a single human with fixed start and goal positions. This choice was motivated by the exponential computational cost of the planning process,

[1] https://github.com/agouguet/social_benchmark_robotic_navigation.git

which makes large-scale evaluation with multiple humans infeasible. The sparse setting, nevertheless, allowed us to analyze in detail the social reasoning of the method under controlled and reproducible conditions.

For the comparison baseline, we selected HATEB, as it is the state-of-the-art method that exhibited the strongest social behaviors in such scenarios, despite achieving lower overall success rates than DRL-VO or DWA approaches. This makes it a particularly relevant point of comparison for assessing the social dimension of MBSN. The results of this comparison are reported in Table 5.1.

The MBSN-related parameters defined for these experiments are:

- The collision penalty in the reward function is 50.
- The discount factor used in the Value Iteration algorithm is 0.99.

	Metrics	Frontal		Corner		Intersection		Door Passing	
		HATEB	**MBSN**	HATEB	**MBSN**	HATEB	**MBSN**	HATEB	**MBSN**
Performance	Success	✓	✓	✓	✓	✓	✓	✓	✓
	Path Length	**23.8**	27.9	**18.8**	24.0	**21.4**	23.3	**22.5**	23.2
	Time To Reach	**158**	174	182	**158**	210	**168**	186	**169**
	Time Not Moving	37	**15**	59	**18**	57	**37**	46	**31**
	Error Distance	0.27	0.27	0.61	**0.24**	0.24	0.24	0.28	**0.25**
	Collision	0	0	0	0	0	0	0	0
	Linear Jerk	**0.062**	0.06	0.07	**0.05**	0.07	**0.04**	0.05	**0.03**
Social	Min Distance Human	1.12	**1.65**	0.54	**0.90**	0.79	**2.32**	1.20	**1.87**
	Collision Human	0	0	0	0	0	0	0	0
	Min TTC	5.02	**9.32**	1.77	**10.11**	1.95	**9.71**	6.12	**22.31**

Table 5.1 – Performance and social metrics of MBSN & HATEB methods on sparse scenarios through RobotSNAP. The best results are shown in bold.

While HATEB generates shorter paths across all scenarios, there is no significant difference in performance metrics. The results remain within the same order of magnitude, so we cannot definitively conclude which method is superior in terms of performance. However, the social metrics reveal key differences.

Our MBSN method maintains a greater minimum distance from humans (±1.7 versus ±1.3). Only in the corner scenario do both solutions fall short, as neither anticipates the appearance of a human from a blind spot. The Minimum Time To Collision metric shows that the MDP-based approach places the robot in less disruptive and safer positions. Although the MBSN method allows the robot to stop and wait, we expected a higher value of Time Not Moving metric, but this was not observed. This demonstrates that waiting is advantageous in specific situations, such as at intersections or doorways. Overall, MBSN outperforms HATEB in terms of social metrics, trading slightly longer paths for safer and more socially acceptable navigation.

However, these results should be interpreted with caution: the method is highly dependent on the structure of the input graph, and the exponential complexity of MDP solving limits scalability. In scenarios with multiple humans, the graph can frequently become fully occupied, which may lead to the so-called "freezing robot" problem, where the robot cannot find a valid path and remains stationary. Addressing this limitation is crucial for applying MBSN in more crowded or complex environments. The simple, small-scale scenarios tested here did not highlight these limitations, but addressing them remains a crucial step toward broader applicability.

In other words, MBSN is an interesting alternative only when the robot must repeat movements within a well-known area crossed by only a few humans at a time. These constraints make policy computation feasible but remain very restrictive in real-world scenarios. The next set of results focuses on HBSN and MDHC, designed to relax these constraints.

5.2.2 HBSN and MDHC Evaluation in Various Scenarios

While MBSN provided valuable insights into the role of decision-making in socially-aware navigation, its evaluation was intentionally restricted to simple scenarios with a single human agent. To assess robustness and generalizability, we now extend our analysis to a broader benchmark comprising multiple randomized scenarios with varying numbers of humans. This allows us to evaluate scalability, statistical consistency, and adaptability to diverse environments. In this section, we compare HBSN and MDHC method against selected state-of-the-art baselines (DWA, HATEB, and DRL-VO). The goal is to investigate how these approaches perform when exposed to richer and less controlled environments, and also to identify the trade-offs between model-based and model-free strategies.

5.2.2.1 Setup

As in Chapter 2.2, the scenarios are as follows: frontal approach, narrow passage, corner, intersection, circle crowd and perpendicular traffic. We conducted 50 simulations (each, up to 120 seconds) per method in the two crowd-based scenarios, and 20 simulations (each, up to 200 seconds) in each of the other scenarios. Overall, this totaled over 900 simulations completed in slightly over 30 hours. The experiments were run on a dedicated workstation: an Intel Xeon W-2245 CPU, 32 GB RAM, and an RTX A5000 GPU. And the specified configuration parameters:

- HBSN parameters:
  - In the calculation of $score_{near}$, the Human Personal Space (HPS) is equal to 1.0 meter and the parameter α is equal to 0.8.
  - For the calculation of the $social\_heuristic$, the parameter k is equal to 0.5 and the weights of the scores are respectively equal to $w_{goal} = 1.0$, $w_{movement} = 0.1$,

$w_{side} = 1.0$ and $w_{near} = 1.5$.

- There is a distance of 0.9 meter between each hexagon center. The polygon must be large enough to fit the robot (0.4m here) but not too large to avoid losing details.

- MDHC parameters:

  - For the training, we followed the training setup presented in Appendix B.4.

- In the simulator, the robot can detect humans at 360 degrees, but obstacles prevent human detection (occultation).

5.2.2.2 Result

Table 5.2 presents the results of the performance measures and social indicators for each of the five methods in the six scenarios. In order to visually represent the overall performance of each method within each scenario, the rows are color-coded. Lighter shades of gray indicate better overall performance, while darker shades indicate worse overall performance. The method with the lightest row achieves the best balance across all indicators. This ranking takes into account both performance metrics and social metrics. Since these are the most important factors for safe and socially acceptable navigation, the success rate, human collisions, and minimum human distance are the main metrics influencing the ranking.

The following paragraphs provide a more detailed examination of the HBSN and MDHC approaches' performance in comparison to the baselines.

HBSN results. HBSN exhibits strong robustness and reliable performance in every scenario. It consistently records no human collisions and achieves extremely high success rates, frequently near 100%. It maintains greater minimum distances to humans than most baselines, combining short path lengths with socially acceptable distances in sparse settings (e.g., frontal approach, narrow passage, and corner). It is the only approach to achieve a 100% success rate in the narrow passage scenario, with zero human collisions and an average minimum human distance of 0.80 meters. Figure 5.1 shows the comparison between DRL-VO (the second best-performing method in narrow passage scenario) and HBSN solution. We can see that DRL-VO actually doesn't give priority to the human, forcing them to move aside and wait for the robot to pass through the narrow passage. HBSN method does the opposite: the robot lets the human pass and waits its turn, even with multiple humans.

While delivering more stable results than DRL-VO, HBSN outperforms traditional techniques like DWA and HATEB in crowded scenarios (circular flow, perpendicular flow) in terms of performance and social metrics. Avoiding frozen or dangerous behaviors, frequently observed in reactive planners, is a significant advantage. However, in some situations, the trade-off results in longer times to reach the goal and slightly longer paths, reflecting cautious navigation.

		Performance Metrics						Social Metrics		
Scenario	Baselines	↑ Success Rate (%)	↓ Collision	↓ Path Length (m)	↓ Time To Reach (s)	↓ Target Error (m)	↓ Linear Jerk (m/s^3)	↓ Human Collision	↑ Min Human Distance (m)	↑ Minimum TTC (s)
Frontal	DWB	100	0	22.84	63.45	0.26	0.34	1	0.64	10.39
	HATEB	95	0	22.91	92.75	0.21	0.24	0	0.85	21.04
	DRL-VO	100	0	22.14	63.1	0.86	0.35	0	0.83	9.83
	HBSN	100	0	22.87	70.60	0.39	0.33	0	**0.80**	**13.53**
	MDHC	95	1	21.84	60.2	0.19	0.01	0	0.72	3.09
Narrow Passage	DWB	90	0	21.19	70.23	1.87	0.20	4	0.53	6.39
	HATEB	75	5	20.54	117.8	3.10	0.14	0	0.74	20.08
	DRL-VO	95	0	21.46	67.71	1.91	0.19	2	0.56	4.67
	HBSN	**100**	0	23.15	**79.1**	0.33	0.18	0	**0.86**	**9.22**
	MDHC	85	1	20.07	54.30	1.81	0.01	2	0.75	2.43
Corner	DWB	80	5	16.21	77.7	1.72	0.16	3	0.67	12.00
	HATEB	40	8	12.37	148.1	5.45	0.09	0	0.85	17.12
	DRL-VO	95	1	16.85	91.5	1.24	0.18	2	0.84	10.22
	HBSN	**100**	0	17.97	66.8	0.33	0.21	**0**	0.58	10.30
	MDHC	90	0	18.62	**60.8**	1.24	0.01	2	0.70	1.75
Intersection	DWB	100	0	20.78	58.1	0.27	0.37	4	0.74	1.18
	HATEB	100	0	21.42	83.1	**0.19**	0.32	**1**	0.77	3.98
	DRL-VO	100	0	20.09	54.45	0.85	0.16	3	1.01	3.78
	HBSN	100	0	20.80	61.1	**0.34**	0.34	3	0.76	3.35
	MDHC	85	3	18.27	44.69	1.32	0.01	1	0.87	6.32
Circle Crowd	DWB	100	0	9.97	80	0.52	0.08	7	0.59	3.29
	HATEB	48	0	11.20	107	1.36	0.07	7	0.59	2.71
	DRL-VO	**100**	0	9.05	40	0.86	0.16	4	0.54	1.33
	HBSN	98	0	11.83	61	0.54	0.11	4	**0.63**	3.39
	MDHC	**100**	0	16.57	46	0.12	0.01	**0**	**0.88**	2.55
Perpendicular Traffic	DWB	90	0	14.40	111.38	0.90	0.25	12	0.52	4.02
	HATEB	52	0	14.83	113.26	2.14	0.12	22	0.52	1.63
	DRL-VO	32	0	28.75	98.12	9.54	0.17	21	1.23	9.13
	HBSN	**96**	0	15.49	74.84	0.85	0.15	**3**	0.60	2.13
	MDHC	78	0	19.79	50.22	2.11	0.01	11	0.77	2.33

Table 5.2 – Results of the simulations on all scenarios for each method, including HBSN and MDHC. For each scenario, the methods are ranked from the worst (dark) to the best (light). Metrics showing a poor difference between methods are underlined and those showing a significant difference are in bold. The arrows near the metrics show whether the metrics are better if the value is high (up arrow) or if the value is low (down arrow).

MDHC results The behavioral profile of the MDHC approach is significantly different. It achieves exceptional success rates (up to 100%) and even better social indicators in specific situations, such as circling crowds, particularly for maintaining the greatest human distances in crowded areas compared to other baselines.

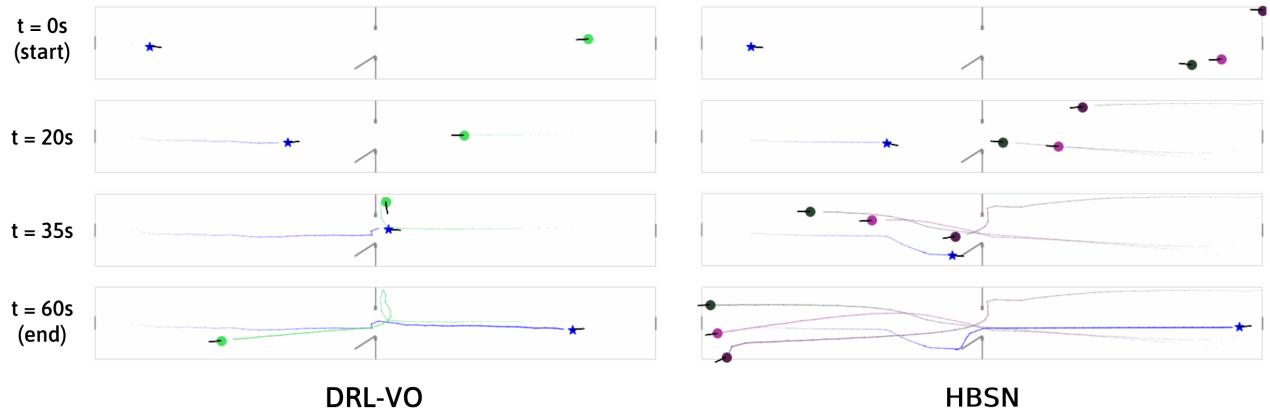


Figure 5.1 – Example of the DRL-VO and HBSN methods over time in the narrow passage scenario. Colored circles denote humans, the blue star indicates the robot, black small lines represent their respective orientations, and colored lines represent associated trajectories.

Its performance is, however, much less reliable; in other situations, such as turning or perpendicular traffic, its success rates are lower and human collisions are more frequent than those of the HBSN approach. Compared to DWA and HATEB, MDHC generally produces more efficient paths and better social indicators, but it lacks the predictable stability of DRL-VO in sparse scenarios. Jerk values close to zero suggest smoother instantaneous trajectories and no slowdowns, but this sometimes comes at the expense of sudden failures. The MDHC approach's lack of stability and reproducibility currently prevents it from being as reliable as the HBSN approach, despite its overall promising potential in handling dynamic and crowded environments.

Taken together, these results demonstrate that HBSN offers a scalable and reliable model-based approach to socially conscious navigation, whereas MDHC shows the promise of learning-based strategies but also emphasizes the need for more reliable training and generalization.

5.2.2.3 Discussion

The experimental findings confirm the suggested methods complementarity. As a model-based approach, HBSN achieves an impressive degree of reproducibility and robustness in every situation. Its ability to function consistently in crowded or uncertain environments emphasizes the value of explicitly simulating social heuristics. However, tradeoffs exist: the hexagonal discretizations introduces structural dependency that may restrict adaptability in more complex environments, and the conservative decision-making approach results in longer paths and travel times.

On the other hand, MDHC shows how data-driven approaches can be used for socially conscious navigation. Its capacity to effectively accomplish objectives and maintain great interpersonal distances in crowd situations indicates that reinforcement learning may be able to identify implicit patterns of human-robot interaction that are challenging to manu-

ally encode. Important limitations are revealed by its instability across scenarios, though: learning-based policies are extremely sensitive to training conditions. The generalization of MDHC to different scenarios is similar to DRL-VO, but conversely, better in crowds and worse in sparse scenarios. The use of CL, which progressively increases the scenario difficulty and ends with environments with dense crowds, could be responsible for this behavior. This can result in decreased stability and occasional failures in sparse or less crowded scenarios, as the agent may partially forget earlier, simpler scenarios with fewer humans.

A trade-off must be found between safety and adaptability when comparing the HBSN and MDHC approaches. While MDHC can occasionally produce more efficient and natural behaviors, there are no guarantees of robustness. In contrast, HBSN prioritizes safe, socially acceptable behaviors at the expense of efficiency. This conflict highlights a major issue in HAN: achieving a balance between dependability and adaptability.

5.3 Real-World Experimentation

We are now interested in putting these methods into practice in the real world. The uncertainties and dynamics of human-robot interaction in natural settings cannot be well captured by simulation, despite the fact that it offers a controlled and repeatable framework for methodical evaluation. In order to verify the transferability of our techniques and identify issues that do not arise in simulation, including sensor noise, actuation delays, or the unpredictable nature of human behavior, real-world experimentation is consequently crucial.

In this section, we present real-world tests conducted with our robotic platform. Given the computational cost of MBSN and its limited applicability, we restrict its evaluation to the narrow passage scenario, where its decision-making capabilities are most relevant. We also include HBSN, which was specifically designed to overcome some of the scalability limitations of MBSN and thus represents a more practical candidate for real-world navigation.

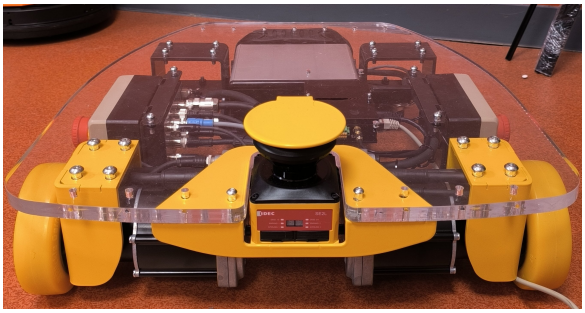
In contrast, we do not include MDHC in these real-world experiments. Despite promising simulation results, its high variability across environments, combined with safety concerns in uncontrolled human-populated spaces, makes it unsuitable for direct deployment without further safety guarantees. This would require a significant real-world learning phase.

The objective of this section is therefore twofold: (1) to assess whether the trends observed in simulation for MBSN and HBSN carry over to the real world, and (2) to highlight the challenges and discrepancies that arise when moving from simulation to real deployment. We first present the robotic platforms used and their setup.

5.3.1 Robot Platform

During the thesis, we used two different robotic platforms, shown in Figure 5.2: The first, the RobuLAB, was used for testing the MBSN method, and the second, the recently acquired Bibus, was used for testing the HBSN method. The use of two different robotic platforms reflects the historical progression of the thesis rather than a methodological choice.

(a) RobuLab



(b) Bibus

Figure 5.2 – The two robotic platforms that were used for these experiments: (a) RobuLAB and (b) Bibus

Sensors Both platforms were equipped with a Light Detection And Ranging (LiDAR) and an RGBD camera (Realsense D435I). The LiDAR was used to detect static obstacles around the robot, and the camera was used to detect humans and obtain their information.

Software Architecture We developed a vision module for human detection using the yolov8 model[2] and estimated human body orientation using the method proposed in [Wu et al. 2020]. The 3D human position was derived from the camera's extrinsic and intrinsic parameters providing all necessary information, such as human position, orientation, and speed (Figure 5.3). This module integrates ROS2 to send human's information to the program responsible for calculating the intermediate social path (MBSN/HBSN).

Figure 5.3 – Screenshot of what the camera sees with human detection and segmentation. The mask is used to obtain human information (position, orientation, speed).

[2] https://github.com/ultralytics/ultralytics

5.3.2 MBSN version 1

To determine whether the conclusions from the simulation benchmark are applicable to real-life scenarios, we conducted an experiment using the door passing scenario. Figure 5.4 shows the scenario with the start and end positions and the graph manually defined on the map. The robot used was a modified RobuLab, equipped with a camera installed at human height to ensure effective human detection, as shown in Figure 5.2a.

A visual analysis can be seen in Figure 5.5. MBSN method exhibits a similar behavior in both simulation and real life: when the robot detects the human and predicts that he is going in its direction, the robot parks itself on a node where it will not interfere (on the side) and waits for the human to pass the narrow passage before engaging. We observed that the robot detects the human intermittently due to the camera's 90° viewing angle. Using a camera with a wider field of view or extending the duration of node occupation could help address this limitation. Another solution could be for the robot to follow the human with its gaze, but this could create a feeling of discomfort in the human.

5.3.3 HBSN

As for MBSN, we conducted experiments using HBSN with a real robot to demonstrate the effectiveness of our proposed methodology in a real environment. On the software side, we used the same ROS2 stack detailed in Section 3.4 used for simulation experiments. We conducted the corner and narrow passage scenarios with a single human in our laboratory. Figure 5.6 illustrates the narrow passage scenario, where the results matched those observed in simulation, with 5.6a a picture of the scenario and 5.6b results path of the robot and the human.

Compared to MBSN version 1, we achieve similar robot behavior by providing only the environment map and no upstream computation. The main distinction lay in defining the map as a hexagonal grid. As a result, the robot's behavior became more decomposed, highlighting the benefits of regular spatial discretization. HBSN, the heuristic variant of MBSN version 2, leverages the increased number of possible social paths made available by the denser navigation graph.

5.3.4 Sim2Real Difficulty

Transferring methods from simulation to the real world is rarely straightforward. Even though simulation enables extensive testing across a wide range of scenarios with precise ground-truth information, the real world introduces sources of variability that are difficult to model. Our experiments highlight several key differences and complications that affect performance:

- **Sensors Uncertainty.** Human orientations and locations are well established in simulation. Perception in real environments depends on sensors (such as LiDARs and

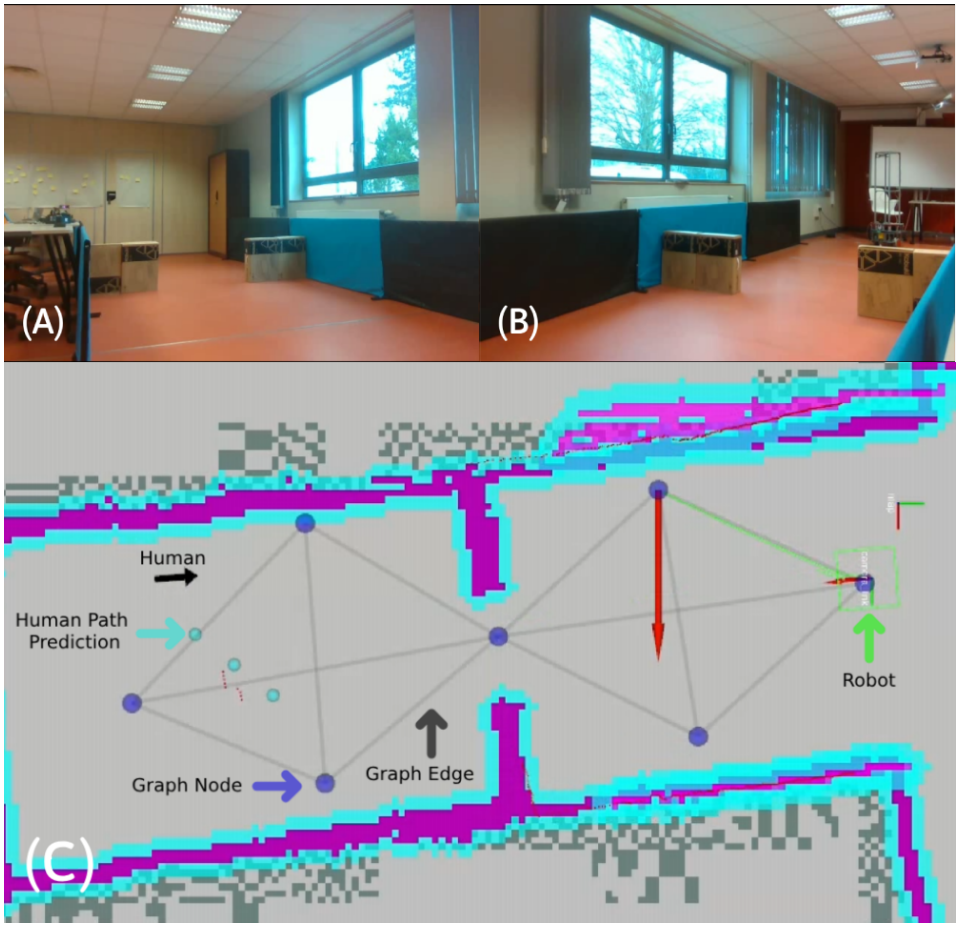


Figure 5.4 – (A) Robot's initial position viewpoint. (B) Goal position viewpoint. (C) Visualization of the navigation graph in RViz.

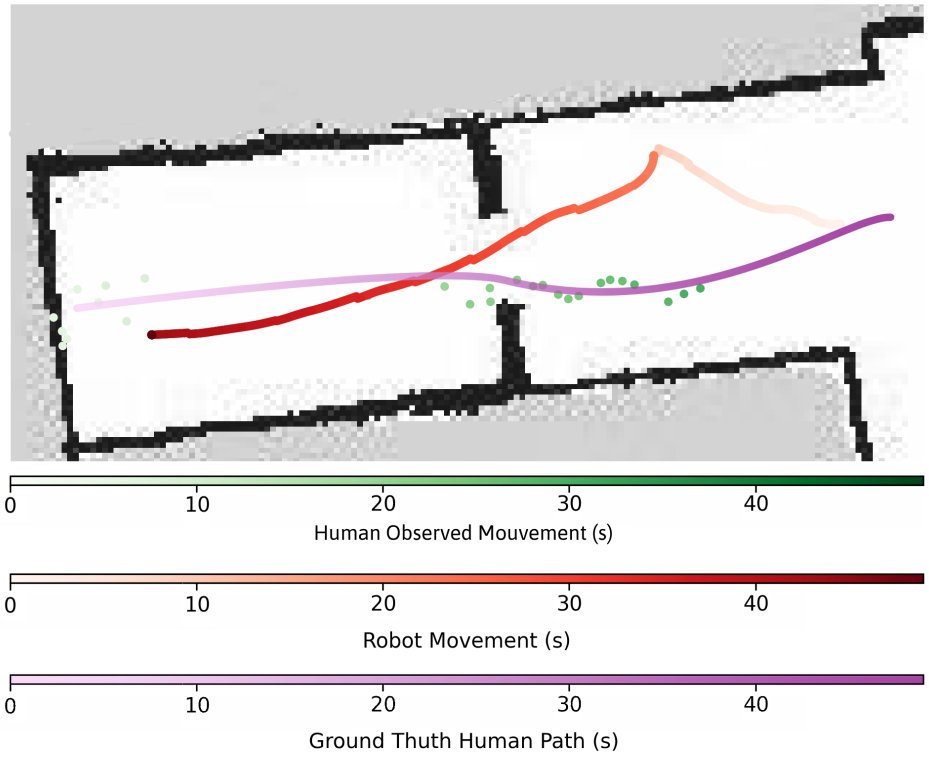
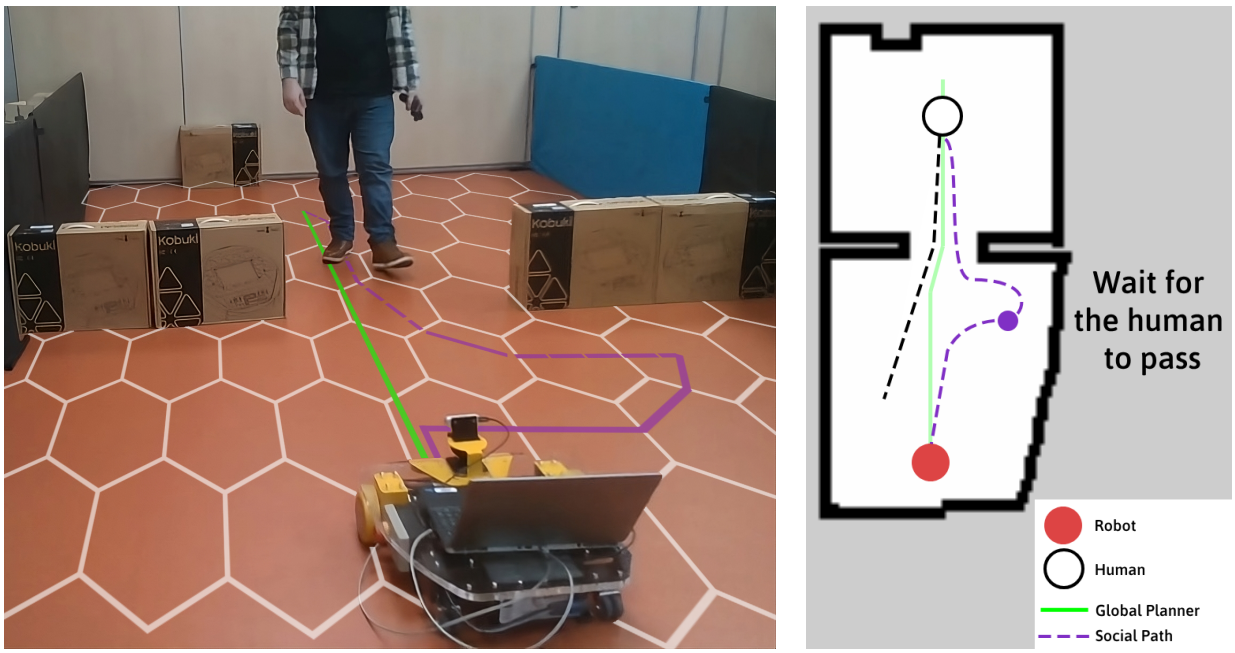


Figure 5.5 – Paths resulting from real-life experimentation. Robot (red) and human (green) positions over time. The purple path represents the ground truth of the human's trajectory.



(a) (b)

Figure 5.6 – Overview of the narrow passage scenario. (a) Picture of the scene from the robot's starting position. (b) Human and robot final paths.

cameras) that can be noisy, leading to inaccurate estimation of human position or even momentary loss of tracking.

- **Human Behavior.** In simulation, humans follow a very specific movement pattern, often SFM[Helbing et al. 1995] or ORCA[Berg et al. 2011]. Such patterns may fail to reflect actual human movements in real experiments, which can be unpredictable and at times influenced by the presence of the robot. This creates trial variability and complicates one-to-one comparisons between solutions and with simulation outcomes.

- **Robot dynamics and control.** Low-level execution in simulation is generally idealized: actions are applied instantly and without slippage. In practice, deviations from the intended trajectory can be caused by actuation delays, wheel slippage, and controller faults, particularly in confined spaces where precision is crucial.

- **Environment Complexity.** Typically, the only simulated elements are static obstacles and humans. Real environments are richer and less predictable because they include unmodeled elements that affect navigation and perception, such as furniture, reflections, and occlusions.

- **Safety Consideration.** In simulation, collisions with humans or walls are harmless. In reality, even small misbehaviors pose risks, which led us to limit real-world testing to selected scenarios (mainly narrow passage) and to exclude DRL-based methods without additional safety layers.

5.4 Conclusion

We carried out a thorough assessment of our methods presented in this chapter, both in simulation and in real-world settings.

Simulation results showed that:

- MBSN explicitly models social navigation interactions and offers safer and more socially acceptable navigation than HATEB, but at the cost of scalability due to exponential complexity.
- This restriction has been relaxed by HBSN, which outperforms state-of-the-art techniques and demonstrates versatility and adaptability to sparse and crowded scenarios.
- MDHC performed well in terms of adaptability, but often lacked consistency compared to model-based approaches.

Overall, the results demonstrate that, despite the potential of learning-based approaches, the heuristic solution HBSN continues to be the most dependable. While MDHC performs exceptionally well in crowded scenarios, it struggles to maintain its robustness in highly diverse environments. This variability, which is introduced by the need to learn over a large number of scenarios, can result in inconsistent performance even with Curriculum Learning (CL).

HBSN, on the other hand, has a number of advantages, such as being versatile, straightforward, and understandable, requiring only a moderate number of parameters, and not requiring a lot of training. Because of its structure, which makes it easy to add more social norms or contextual behaviors, it is very adaptable to new circumstances or future extensions. Despite their apparent simplicity, heuristic approaches remain to provide dependable and socially acceptable navigation, providing a strong basis for hybrid approaches that might eventually include learning-based enhancements.

While real-world experiments demonstrated the gap between simulation and deployment, they also validated some of these trends. MBSN and HBSN reproduced socially human-aware behaviors as in simulation, validating their applicability in practice. However, problems not found in simulation were brought about by sensor noise, human unpredictability, and robot dynamics.

The importance of connecting simulation and actual deployment is emphasized by these complementary insights, which open up possibilities for hybrid approaches.

Conclusion

One of the main challenges in robotics remains the Human-Aware Robot Navigation (HAN), which calls for the robot to navigate through shared environments with people in a safe, effective, and social compliance. In contrast to traditional robotic navigation problems, this framework takes into account human behavior, social norms, and the inherent uncertainty of human movement in addition to static obstacles and robot dynamics.

Due to its complexity, this research topic has received a lot of attention in recent years. Although HAN may appear simple and natural, the abundance of research and diverse methodologies—from DRL to model-based planning to strategies powered by large language models—highlight the difficulty of having robots replicate socially acceptable behaviors.

Resolving such problems is especially difficult since it calls for achieving a balance between behavioral sophistication—which is necessary to guarantee socially acceptable interactions—and computational efficiency, which is required for real-time deployment. Additionally, it is anticipated that the robot will be able to navigate in a wide variety of environments, from crowded open spaces to small, sparsely populated corridors, just like humans. Purely model-based approaches often encounter scalability issues when reasoning about large numbers of humans, while purely learning-based strategies often lack robustness and generalization beyond training environments.

Our objective was to design robot navigation frameworks capable not only of anticipating human behavior and respecting social constraints, but also of remaining efficient and adaptable to a wide range of scenarios, from sparse environment to dense crowds. To overcome these limitations, this thesis proposes a set of complementary solutions:

1. An evaluation platform dedicated to the systematic assessment of HAN in simulation;
2. Model-based solutions, including both a planning approach relying on Monte Carlo Tree Search and a heuristic approach designed for scalability and interpretability;
3. A learning-based approach leveraging DRL to jointly reason about the robot state, the environment, and human dynamics.

The Robot Social Navigation Assessment Platform

In Chapter 1, we analyzed the state of the art in HAN and observed that existing tools have limitations regarding their evaluation metrics and the variety of scenarios they cover. To

address this gap, we decided to propose our own tool in Chapter 2: RobotSNAP, a dedicated benchmark for Human-Aware Robot Navigation (HAN), composed of three main elements:

- Unity3D, used to simulate realistic environments and human behaviors with controllable levels of density, dynamics, and variability.
- ROS2, which provides the interface to the navigation methods, ensures that each approach can be evaluated in exactly the same way as it would be when deployed on a real robot.
- A set of evaluation metrics, covering both performance (success rate, time, efficiency) and social (human distance, collisions, comfort) metrics, as well as a library of pre-implemented scenarios reproducing common navigation challenges.

As a result, RobotSNAP enables two complementary functions:

1. It makes it possible to compare approaches directly and systematically under the same, repeatable conditions.
2. It serves as a diagnostic tool, pointing out the advantages and disadvantages of each approach.

No single method works consistently in every situation, according to a comparative study using a several number of off-the-shelf methods. Some work well in dense crowds situations but poorly in sparse scenarios, while others behave in the opposite way. This reaffirmed the necessity of proactive and flexible solutions, which motivated the creation of our later contributions.

Model-Based Solutions: MBSN and HBSN

In Chapter 3, we created MDP-Based Social Navigation (MBSN), a model-based technique that uses a Markov Decision Process (MDP) to represent the robot and human states, in order to address the lack of anticipation seen in state-of-the-art methods. With a graph representation of the environment, MBSN could predict human trajectories and select socially acceptable actions. The robot anticipates human movements and reacts accordingly in scenarios such as narrow hallways. This method worked extremely well. However, the high computational cost for larger environments and the requirement for manually defined navigation graphs limit its practical applicability.

We first examined a method based on Monte Carlo Tree Search (MCTS) to reduce computational costs, as well as eliminate the need for expert-defined graphics by incorporating a polygonal grid instead. In theory, MCTS can explore multiple future scenarios to guide decision-making. However, the MCTS algorithm frequently could only explore one simulation prior to making a decision due to the real-time limitations.

It's interesting to note that this single-simulation exploration was essentially a direct use of the heuristic, which already offered useful anticipatory behavior. Building on this finding, Heuristic-Based Social Navigation (HBSN) was proposed as a simplified version of MBSN that only uses this effective heuristic. Despite this simplification, HBSN demonstrates that the heuristic captures the essential anticipatory elements of decision-making by achieving reliable and socially compliant navigation in real time.

Finally, as the most practically applicable solution, we concentrated on the heuristic-based variant, HBSN. The high computational cost and complexity of MBSNv1 and MBSNv2 make them unsuitable for real-time deployment in larger or dynamic environments, despite the fact that all three approaches—MBSNv1, MBSNv2, and HBSN—have shown strong social behavior and dependable anticipation capabilities. HBSN, on the other hand, accomplishes anticipatory and socially compliant navigation in real time. In contrast with continuous-action approaches, the discretization of the environment into polygonal grids decreases the accuracy of robot control, as it takes time to compute a middle path between the high-level planner and the low-level control system. The shift to a learning-based approach, which maintains the anticipatory behaviors and adaptability demonstrated by HBSN while allowing for fine-grained, continuous control with almost instantaneous decision-making, was motivated by these limitations.

Our Solution Based on Deep Reinforcement Learning: MDHC

In Chapter 4, we present our Deep Reinforcement Learning (DRL) solution capable of navigating in any environment, with different topologies and numbers of humans. As a novelty, we used a sub-goal calculated using our heuristic HBSN and we structured the neural network into several branches, each responsible for a specific perception aspect of the problem: applying a human-aware control.

Like some existing solutions, we introduced a Spatio-Temporal Graph (ST-Graph) as a branch into the network modeling Human-Human Interaction (HHI), aside of a branch for robot intrinsic state and another for obstacles scan. In addition, we introduced a Social Angular Field (SAF) branch specifically addressing human-robot interactions.

These novelties contribute to the agent's convergence speed during learning. As another contribution, we implemented a Curriculum Learning (CL) approach, allowing the agent to learn to navigate with increasing difficulty depending on the topology and the number of humans present in the environment. Unlike many works on DRL addressing the HAN problem, we considered the full range of problems that the robot faces in reality: indoor spaces with multiple rooms involving many narrow passages, confrontation with crowds and the mix of both.

Evaluation through simulations and real-world experiments

In Chapter 5, we positioned our work within current approaches and evaluated our contributions. The first series of experiments was carried out through extensive simulations using RobotSNAP, providing multiple indicators across diverse scenarios. The second series focused on a restricted set of real-world experiments, aiming to identify the remaining discrepancies between simulated and real robotics.

As a result, we demonstrate that the MBSN method outperformed HATEB in terms of socially responsible navigation and displayed good performance in sparse scenarios. In various settings, including sparse and cluttered, HBSN outperformed state-of-the-art conventional methods on average. MDHC displayed less consistent and more variable performance than HBSN, despite encouraging simulation results. The performance of our DRL method did not meet expectations in sparse scenarios, and we hypothesize that this limitation originates from the CL strategy. The latter stages of the curriculum (on complex scenarios) seems to override the knowledge acquired during the initial phases. In conclusion, although our heuristic approach HBSN does not achieve the best results in every scenarios, it is the only method that consistently maintains stable and satisfactory performance across all configurations. In addition, it has the benefits of being simple to model, interpretable, and requiring no training.

Similar trends were confirmed by early real-world experiments, which showed that both MBSN and HBSN displayed the same socially acceptable behaviors than in simulation. These results demonstrate the effectiveness of our methods, but they also show that more extensive testing in the real-world is needed to properly assess their scalability and robustness.

Future Works

The contributions made in this thesis could be expanded in a number of ways to further the development of Human-Aware Robot Navigation.

Progressive Multi-Modal Training Progressive multimodal learning is a complementary approach. Convergence and robustness can be improved during learning by gradually integrating various sensory modalities and social cues. Early experiments (Appendix B.2) have indeed shown that it is difficult to train a network integrating multi-layered cross-modal interactions in multiple scenarios. With progressive modal learning, agents could learn to navigate more efficiently in complex human environments by gradually adding social features and spatio-temporal representations after starting with simpler inputs, such as laser scans. Thus, the agent could more easily learn to handle different modalities and cross-modal attention.

Richer Structured Observations A promising direction for future work is to provide agents with richer and more structured observations of the environment and social context.

At the environmental level, this could involve the automatic identification of key locations and critical areas within the topology. At the social level, integrating contextual knowledge—such as the robot's task, human intentions, or the broader situation—could dynamically adjust the weight of social signals and sub-goals, allowing the robot to adapt its behavior to the current scenario and overall objectives.

These ideas naturally extend the approaches developed in this thesis. For MBSN and HBSN, new social rules (task/place) could be seamlessly integrated into the heuristics modifying weights. For MDHC, context could be treated as an additional modality, enabling the network to differentiate strategies depending on the robot's task or the type of environment.

Such improvements would significantly enhance the agent's ability to plan and navigate in very different environments, while maintaining socially appropriate and adaptive behavior.

Augmented sim2real by Incorporating Noise into Learning The gap between simulation and reality is one of the main obstacles to implementing learned navigation policies. Due to dynamic variations, sensor noise, and environmental variability, policies trained in simulation often fail to generalize to the real world. To address this issue, future research could explore adding noise to the learning process, both in sensor readings and in the robot's dynamics. Intentionally adding this type of variability would improve the agent's resilience to real-world situations by teaching it to adapt to the sim2real gap during simulation.

Real-World Benchmark A natural progression of our work would be to extend RobotSNAP to a real-world benchmark. Since any environment (map) can be imported into the simulator, it becomes possible to reproduce real-world settings in simulation for training purposes. Establishing such a benchmark would allow researchers to directly assess the sim2real transfer, robustness and scalability of various navigation methods, thus providing complementary information to purely simulation-based evaluations. Furthermore, it would provide a structured basis for developing navigation strategies that are both safer and more generalizable to different populated environments.

Bibliography

[Abir et al. 2022]. Bellarbi Abir et al. (Sept. 2022). "A new approach for social navigation and interaction using a dynamic proxemia modeling". In: *Evolutionary Intelligence* 15. DOI: 10.1007/s12065-021-00633-7 (cit. on p. 12).

[Alahi et al. 2016]. Alexandre Alahi et al. (2016). "Social lstm: Human trajectory prediction in crowded spaces". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 961–971 (cit. on p. 22).

[Bacchin et al. 2024]. Alberto Bacchin et al. (2024). "Preference-Based People-Aware Navigation for Telepresence Robots". In: *International Journal of Social Robotics*, pp. 1–21 (cit. on p. 12).

[Barchard et al. 2018]. Kimberly A Barchard et al. (2018). "Perceived Social Intelligence (PSI) Scales Test Manual (August, 2018)". In: (cit. on p. 29).

[Bartneck et al. 2009]. Christoph Bartneck et al. (2009). "Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots". In: *International journal of social robotics* 1, pp. 71–81 (cit. on p. 29).

[Bellman 1957]. Richard Bellman (1957). *Dynamic Programming*. 1st ed. Princeton, NJ, USA: Princeton University Press (cit. on pp. 54, 55, 56, 57).

[Berg et al. 2011]. Jur van den Berg et al. (2011). "Reciprocal n-Body Collision Avoidance". In: *Robotics Research*. Ed. by Cédric Pradalier, Roland Siegwart, and Gerhard Hirzinger. Red. by Bruno Siciliano, Oussama Khatib, and Frans Groen. Berlin, Heidelberg: Springer Berlin Heidelberg, 3–19 vol. 70. (Visited on 01/10/2023) (cit. on p. 116).

[Bilen et al. 2024]. Baris Bilen et al. (2024). "Social robot navigation with adaptive proxemics based on emotions". In: *arXiv preprint arXiv:2401.17663* (cit. on p. 12).

[Biswas et al. 2022]. Abhijat Biswas et al. (2022). "Socnavbench: A grounded simulation testing framework for evaluating social navigation". In: *ACM Transactions on Human-Robot Interaction (THRI)* 11.3, pp. 1–24 (cit. on p. 28).

[Borenstein, Koren, et al. 1991]. Johann Borenstein, Yoram Koren, et al. (1991). "The vector field histogram-fast obstacle avoidance for mobile robots". In: *IEEE transactions on robotics and automation* 7.3, pp. 278–288 (cit. on p. 98).

[Brito et al. 2019]. Bruno Brito et al. (Oct. 2019). "Model Predictive Contouring Control for Collision Avoidance in Unstructured Dynamic Environments". In: *IEEE Robotics and Automation Letters* 4.4, pp. 4459–4466. ISSN: 2377-3766. DOI: 10.1109/LRA.2019.2929976. URL: https://ieeexplore.ieee.org/document/8768044/ (visited on 04/17/2025) (cit. on p. 13).

[Brody, Alon, and Yahav 2021]. Shaked Brody, Uri Alon, and Eran Yahav (2021). "How attentive are graph attention networks?" In: *arXiv preprint arXiv:2105.14491* (cit. on p. 98).

[Burlet, Aycard, and Fraichard 2004]. J. Burlet, O. Aycard, and T. Fraichard (2004). "Robust motion planning using Markov decision processes and quadtree decomposition". In: *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004.* Vol. 3, 2820–2825 Vol.3. DOI: 10.1109/ROBOT.2004.1307488 (cit. on p. 58).

[Carpinella et al. 2017]. Colleen M Carpinella et al. (2017). "The robotic social attributes scale (RoSAS) development and validation". In: *Proceedings of the 2017 ACM/IEEE International Conference on human-robot interaction*, pp. 254–262 (cit. on p. 29).

[Charalampous and Gasteratos 2023]. Ioannis Kostavelis Charalampous and Antonios Gasteratos (2023). *Robot navigation in large-scale social maps: An action recognition approach | Elsevier Enhanced Reader*. DOI: 10.1016/j.eswa.2016.09.026. URL: https://reader.elsevier.com/reader/sd/pii/S0957417416305103?token=8BC132C403F6909D356122328A9C718
originRegion=eu-west-1&originCreation=20230111130117 (visited on 01/11/2023) (cit. on p. 12).

[Chen, Hu, et al. 2020]. Changan Chen, Sha Hu, et al. (2020). "Relational graph learning for crowd navigation". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10007–10013 (cit. on p. 15).

[Chen, Liu, Kreiss, et al. 2019]. Changan Chen, Yuejiang Liu, Sven Kreiss, et al. (2019). "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning". In: *2019 international conference on robotics and automation (ICRA)*. IEEE, pp. 6015–6022 (cit. on pp. 15, 27).

[Chen, Wang, et al. 2024]. Lin Chen, Yaonan Wang, et al. (2024). "Decentralized Multi-Robot Navigation Coupled with Spatial-Temporal RetNet Based on Deep Reinforcement Learning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 10289–10296 (cit. on p. 89).

[Chen, Everett, et al. 2017]. Yu Fan Chen, Michael Everett, et al. (2017). "Socially aware motion planning with deep reinforcement learning". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 1343–1350 (cit. on p. 15).

[Chen, Liu, Everett, et al. 2017]. Yu Fan Chen, Miao Liu, Michael Everett, et al. (2017). "Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning". In: *2017 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 285–292 (cit. on p. 15).

[Cheng et al. 2018]. Jiyu Cheng et al. (Dec. 2018). "Autonomous Navigation by Mobile Robots in Human Environments: A Survey". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2018 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 1981–1986. DOI: 10.1109/ROBIO.2018.8665075 (cit. on pp. 10, 17).

[Cormons et al. 2020]. Lucie Cormons et al. (2020). "Testing social robot acceptance: What if you could be assessed for dementia by a robot? A pilot study". In: *2020 6th international conference on mechatronics and robotics engineering (ICMRE)*. IEEE, pp. 92–98 (cit. on p. 29).

[Cybulski, Wegierska, and Granosik 2019]. B. Cybulski, A. Wegierska, and G. Granosik (2019). "Accuracy comparison of navigation local planners on ROS-based mobile robot". In: *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, pp. 104–111. DOI: 10.1109/RoMoCo.2019.8787346 (cit. on p. 20).

[Dijkstra 1959]. Edsger W Dijkstra (1959). "A note on two problems in connexion with graphs". In: *Numerische mathematik* 1.1, pp. 269–271 (cit. on p. 7).

[Dugas et al. 2022]. Daniel Dugas et al. (2022). "Navdreams: Towards camera-only rl navigation among humans". In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2504–2511 (cit. on p. 89).

[Duszak and Siemiątkowska 2020]. Piotr Duszak and Barbara Siemiątkowska (2020). "The Application of Hexagonal Grids in Mobile Robot Navigation". In: *Mechatronics 2019: Recent Advances Towards Industry 4.0*. Ed. by Roman Szewczyk et al. Cham: Springer International Publishing, pp. 198–205. ISBN: 978-3-030-29993-4 (cit. on p. 68).

[Duszak, Siemiątkowska, and Więckowski 2021]. Piotr Duszak, Barbara Siemiątkowska, and Rafał Więckowski (2021). "Hexagonal grid-based framework for mobile robot navigation". In: *Remote Sensing* 13.21, p. 4216 (cit. on p. 68).

[Eirale, Leonetti, and Chiaberge 2024]. Andrea Eirale, Matteo Leonetti, and Marcello Chiaberge (2024). "Learning Social Cost Functions for Human-Aware Path Planning". In: *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 5364–5371 (cit. on p. 14).

[Elnoor et al. 2025]. Mohamed Elnoor et al. (2025). "Vi-LAD: Vision-Language Attention Distillation for Socially-Aware Robot Navigation in Dynamic Environments". In: *arXiv preprint arXiv:2503.09820* (cit. on p. 16).

[Everett, Chen, and How 2018]. Michael Everett, Yu Fan Chen, and Jonathan P How (2018). "Motion planning among dynamic, decision-making agents with deep reinforcement learning". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 3052–3059 (cit. on pp. 15, 42, 89, 98).

[Fahmy, Shehata, and Maged 2024]. Tareq A Fahmy, Omar M Shehata, and Shady A Maged (2024). "Trajectory Aware Deep Reinforcement Learning Navigation Using Multichannel Cost Maps". In: *Robotics* 13.11, p. 166 (cit. on p. 19).

[Fiorini and Shiller 1998]. Paolo Fiorini and Zvi Shiller (1998). "Motion Planning in Dynamic Environments Using Velocity Obstacles". In: *The International Journal of Robotics Research* 17.7, pp. 760–772 (cit. on p. 7).

[Fox, Burgard, and Thrun 1997]. Dieter Fox, Wolfram Burgard, and Sebastian Thrun (1997). "The Dynamic Window Approach to Collision Avoidance". In: *Robotics & Automation Magazine, IEEE*, 23–33 vol. 4. DOI: 10.1109/100.580977 (cit. on pp. 7, 42, 64).

[Francis et al. 2025]. Anthony Francis et al. (2025). "Principles and guidelines for evaluating social robot navigation algorithms". In: *ACM Transactions on Human-Robot Interaction* 14.2, pp. 1–65 (cit. on pp. 1, 20, 26).

[Galati et al. 2022]. Giada Galati et al. (2022). "Game theoretical trajectory planning enhances social acceptability of robots by humans". In: *Scientific reports* 12.1, p. 21976 (cit. on p. 14).

[Gao and Huang 2022]. Yuxiang Gao and Chien-Ming Huang (2022). "Evaluation of socially-aware robot navigation". In: *Frontiers in Robotics and AI* 8, p. 420 (cit. on pp. 20, 22).

[Gil, Garrell, and Sanfeliu 2021]. Oscar Gil, Anaıs Garrell, and Alberto Sanfeliu (2021). "Social robot navigation tasks: Combining machine learning techniques and social force model". In: *Sensors* 21.21, p. 7087 (cit. on p. 16).

[Gines Clavero et al. 2021]. Jonatan Gines Clavero et al. (2021). "Defining adaptive proxemic zones for activity-aware navigation". In: *Advances in Physical Agents II: Proceedings of the 21st International Workshop of Physical Agents (WAF 2020), November 19-20, 2020, Alcalá de Henares, Madrid, Spain*. Springer, pp. 3–17 (cit. on p. 12).

[Groshev et al. 2018]. Edward Groshev et al. (2018). "Learning generalized reactive policies using deep neural networks". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 28, pp. 408–416 (cit. on p. 14).

[Guillen-Ruiz et al. 2023]. Silvia Guillen-Ruiz et al. (2023). "Evolution of socially-aware robot navigation". In: *Electronics* 12.7, p. 1570 (cit. on p. 18).

[Guzzi et al. 2013]. Jérôme Guzzi et al. (2013). "Human-friendly robot navigation in dynamic environments". In: *2013 IEEE international conference on robotics and automation*. IEEE, pp. 423–430 (cit. on p. 20).

[Haarnoja et al. 2018]. Tuomas Haarnoja et al. (2018). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. Pmlr, pp. 1861–1870 (cit. on p. 88).

[Hall 1969]. E.T. Hall (1969). *The Hidden Dimension: Man's Use of Space in Public and Private*. Bodley Head. ISBN: 9780370013084 (cit. on pp. 1, 11, 22, 95).

[Han et al. 2025]. James R Han et al. (2025). "DR-MPC: Deep Residual Model Predictive Control for Real-world Social Navigation". In: *IEEE Robotics and Automation Letters* (cit. on p. 16).

[Hart, Nilsson, and Raphael 1968]. Peter E Hart, Nils J Nilsson, and Bertram Raphael (1968). "A formal basis for the heuristic determination of minimum cost paths". In: *IEEE transactions on Systems Science and Cybernetics* 4.2, pp. 100–107 (cit. on p. 7).

[Helbing and Molnár 1995]. Dirk Helbing and Péter Molnár (May 1995). "Social force model for pedestrian dynamics". In: *Phys. Rev. E* (5), 4282–4286 vol. 51 (cit. on pp. 36, 116).

[Howard 1960]. R. A. Howard (1960). *Dynamic Programming and Markov Processes*. Cambridge, MA: MIT Press (cit. on pp. 56, 57).

[Jeanpierre et al. 2017]. Laurent Jeanpierre et al. (2017). "COACHES: An assistance Multi-Robot System in public areas". In: *2017 European Conference on Mobile Robots (ECMR)*. IEEE, pp. 1–6 (cit. on p. 58).

[Karwowski, Szynkiewicz, and Niewiadomska-Szynkiewicz 2024]. Jarosław Karwowski, Wojciech Szynkiewicz, and Ewa Niewiadomska-Szynkiewicz (2024). "Bridging requirements, planning, and evaluation: A review of social robot navigation". In: *Sensors* 24.9, p. 2794 (cit. on p. 20).

[Kathuria et al. 2025]. Tribhi Kathuria et al. (2025). "Learning Implicit Social Navigation Behavior using Deep Inverse Reinforcement Learning". In: *arXiv preprint arXiv:2501.06946* (cit. on p. 15).

[Kavraki et al. 1996]. Lydia E Kavraki et al. (1996). "Probabilistic roadmaps for path planning in high-dimensional configuration spaces". In: *IEEE transactions on Robotics and Automation* 12.4, pp. 566–580 (cit. on pp. 7, 54, 59).

[Kendon 1990]. Adam Kendon (1990). *Conducting interaction: Patterns of behavior in focused encounters*. Vol. 7. CUP Archive (cit. on p. 11).

[Khambhaita and Alami 2019]. Harmish Khambhaita and Rachid Alami (2019). "Viewing robot navigation in human environment as a cooperative activity". In: *Robotics Research: The 18th International Symposium ISRR*. Springer, pp. 285–300 (cit. on p. 13).

[Kim and Pineau 2016]. Beomjoon Kim and Joelle Pineau (2016). "Socially adaptive path planning in human environments using inverse reinforcement learning". In: *International Journal of Social Robotics* 8, pp. 51–66 (cit. on p. 15).

[Kim, Lee, et al. 2018]. Minkyu Kim, Jaemin Lee, et al. (2018). "Social Navigation Planning Based on People's Awareness of Robots". In: *CoRR* abs/1809.08780. arXiv: 1809.08780. URL: http://arxiv.org/abs/1809.08780 (cit. on p. 58).

[Kocsis and Szepesvári 2006]. Levente Kocsis and Csaba Szepesvári (2006). "Bandit based monte-carlo planning". In: *European conference on machine learning*. Springer, pp. 282–293 (cit. on pp. 54, 66, 71).

[Kollmitz, Hsiao, et al. 2015]. Marina Kollmitz, Kaijen Hsiao, et al. (2015). "Time Dependent Planning on a Layered Social Cost Map for Human-Aware Robot Navigation". In: *Proc. of the IEEE Eur. Conf. on Mobile Robotics (ECMR)*. DOI: 10.1109/ECMR.2015.7324184. URL: http://ais.informatik.uni-freiburg.de/publications/papers/kollmitz15ecmr.pdf (cit. on pp. 12, 42).

[Kollmitz, Koller, et al. 2020]. Marina Kollmitz, Torsten Koller, et al. (Oct. 2020). "Learning Human-Aware Robot Navigation from Physical Interaction via Inverse Reinforcement Learning". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866, pp. 11025–11031. DOI: 10.1109/IROS45743.2020.9340865 (cit. on p. 15).

[Kruse et al. 2013]. Thibault Kruse et al. (Dec. 2013). "Human-Aware Robot Navigation: A Survey". In: *Robotics and Autonomous Systems*, 1726–1743 vol. 61. URL: https://hal.science/hal-01684295 (cit. on pp. 17, 20).

[LaValle 1998]. Steven LaValle (1998). "Rapidly-exploring random trees: A new tool for path planning". In: *Research Report 9811* (cit. on p. 7).

[Le et al. 2023]. Viet-Anh Le et al. (Oct. 2023). *Multi-Robot Cooperative Navigation in Crowds: A Game-Theoretic Learning-Based Model Predictive Control Approach*. arXiv:2310.06964 [cs]. DOI: 10.48550/arXiv.2310.06964. URL: http://arxiv.org/abs/2310.06964 (visited on 04/17/2025) (cit. on p. 14).

[Li, Xia, et al. 2021]. Chengshu Li, Fei Xia, et al. (2021). "igibson 2.0: Object-centric simulation for robot learning of everyday household tasks". In: *arXiv preprint arXiv:2108.03272* (cit. on p. 28).

[Li, Xu, et al. 2019]. Keyu Li, Yangxin Xu, et al. (Dec. 2019). "Deep Reinforcement Learning based Human-Aware Navigation for Mobile Robot in Indoor Environments". In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019 IEEE International Conference on Robotics and Biomimetics (ROBIO), pp. 688–694. DOI: 10.1109/ROBIO49542.2019.8961764 (cit. on pp. 1, 15, 42, 89, 98).

[Ling et al. 2024]. Bo Ling et al. (2024). "SocialGAIL: Faithful Crowd Simulation for Social Robot Navigation". In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 16873–16880 (cit. on p. 28).

[Liu, Chang, et al. 2021]. Shuijing Liu, Peixin Chang, et al. (2021). "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning". In: *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 3517–3524 (cit. on pp. 89, 90).

[Liu, Xia, et al. 2024]. Shuijing Liu, Haochen Xia, et al. (2024). "Height: Heterogeneous interaction graph transformer for robot navigation in crowded and constrained environments". In: *arXiv preprint arXiv:2411.12150* (cit. on pp. 88, 89, 90, 96).

[Lozenguez et al. 2013]. Guillaume Lozenguez et al. (2013). "Interleaving Planning and Control of Mobiles Robots in Urban Environments Using Road-Map". In: *Intelligent Autonomous Systems 12: Volume 1 Proceedings of the 12th International Conference IAS-12, held June 26-29, 2012, Jeju Island, Korea*. Ed. by Sukhan Lee et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 683–691. DOI: 10.1007/978-3-642-33926-4_65. URL: https://doi.org/10.1007/978-3-642-33926-4_65 (cit. on p. 58).

[Martinez-Baselga et al. 2024]. Diego Martinez-Baselga et al. (2024). "SHINE: Social homology identification for navigation in crowded environments". In: *arXiv preprint arXiv:2404.16705* (cit. on p. 14).

[Mavrogiannis, Alves-Oliveira, et al. 2022]. Christoforos Mavrogiannis, Patrıcia Alves-Oliveira, et al. (2022). "Social momentum: Design and evaluation of a framework for socially competent robot navigation". In: *ACM Transactions on Human-Robot Interaction (THRI)* 11.2, pp. 1–37 (cit. on p. 14).

[Mavrogiannis, Balasubramanian, et al. 2022]. Christoforos Mavrogiannis, Krishna Balasubramanian, et al. (2022). "Winding through: Crowd navigation via topological invariance". In: *IEEE Robotics and Automation Letters* 8.1, pp. 121–128 (cit. on p. 14).

[Mavrogiannis, Baldini, et al. 2023]. Christoforos Mavrogiannis, Francesca Baldini, et al. (Feb. 2023a). "Core Challenges of Social Robot Navigation: A Survey". In: *J. Hum.-Robot Interact.* DOI: 10.1145/3583741. URL: https://doi.org/10.1145/3583741 (cit. on p. 1).

[Mavrogiannis, Baldini, et al. 2023]— . (2023b). "Core challenges of social robot navigation: A survey". In: *ACM Transactions on Human-Robot Interaction* 12.3, pp. 1–39 (cit. on p. 17).

[Mavrogiannis, Hutchinson, et al. 2019]. Christoforos Mavrogiannis, Alena M Hutchinson, et al. (2019). "Effects of distinct robot navigation strategies on human behavior in a crowded environment". In: *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, pp. 421–430 (cit. on p. 20).

[Mavrogiannis and Knepper 2019]. Christoforos Mavrogiannis and Ross Knepper (Mar. 2019). "Multi-Agent Path Topology in Support of Socially Competent Navigation Planning". In: *The International Journal of Robotics Research* 38, pp. 338–356. DOI: 10.1177/0278364918781016 (cit. on p. 12).

[Mehta, Ferrer, and Olson 2016]. Dhanvin Mehta, Gonzalo Ferrer, and Edwin Olson (Oct. 2016). "Autonomous navigation in dynamic social environments using Multi-Policy Decision Making". In: *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Daejeon, South Korea: IEEE, pp. 1190–1197. ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759200. URL: http://ieeexplore.ieee.org/document/7759200/ (visited on 01/13/2023) (cit. on p. 13).

[Mehta, Ferrer, and Olson 2017]. Dhanvin Mehta, Gonzalo Ferrer, and Edwin Olson (May 2017). "Fast discovery of influential outcomes for risk-aware MPDM". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. 2017 IEEE International Conference on Robotics and Automation (ICRA), pp. 6210–6216. DOI: 10.1109/ICRA.2017.7989736 (cit. on p. 13).

[Melo and Moreno 2022]. Francisco Melo and Plinio Moreno (Apr. 2022). "Socially Reactive Navigation Models for Mobile Robots". In: *2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*. 2022 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), pp. 91–97. DOI: 10.1109/ICARSC55462.2022.9784789 (cit. on p. 12).

[Mirsky et al. 2024]. Reuth Mirsky et al. (2024). "Conflict avoidance in social navigation—a survey". In: *ACM Transactions on Human-Robot Interaction* 13.1, pp. 1–36 (cit. on p. 17).

[Mnih, Badia, et al. 2016]. Volodymyr Mnih, Adria Puigdomenech Badia, et al. (2016). "Asynchronous methods for deep reinforcement learning". In: *International conference on machine learning*. PmLR, pp. 1928–1937 (cit. on p. 88).

[Mnih, Kavukcuoglu, et al. 2013]. Volodymyr Mnih, Koray Kavukcuoglu, et al. (2013). "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (cit. on p. 88).

[Mohamed et al. 2020]. Abduallah Mohamed et al. (2020). "Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 14424–14432 (cit. on p. 98).

[Möller et al. 2021]. Ronja Möller et al. (2021). "A survey on human-aware robot navigation". In: *Robotics and Autonomous Systems* 145, p. 103837. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2021.103837. URL: https://www.sciencedirect.com/science/article/pii/S0921889021001226 (cit. on pp. 10, 17).

[Muchen Sun et al. 2024]. Max Muchen Sun et al. (2024). "Mixed strategy Nash equilibrium for crowd navigation". In: *The International Journal of Robotics Research*, p. 02783649241302342 (cit. on pp. 14, 43).

[Nair et al. 2022]. Anirudh Nair et al. (2022). "Dynabarn: Benchmarking metric ground navigation in dynamic environments". In: *2022 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, pp. 347–352 (cit. on p. 28).

[Naotunna and Wongratanaphisan 2020]. Isira Naotunna and Theeraphong Wongratanaphisan (2020). "Comparison of ROS Local Planners with Differential Drive Heavy Robotic System". In: *2020 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pp. 1–6. DOI: 10.1109/ICAMechS49982.2020.9310123 (cit. on p. 20).

[Pellegrini et al. 2009]. Stefano Pellegrini et al. (2009). "You'll never walk alone: Modeling social behavior for multi-target tracking". In: *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268 (cit. on p. 22).

[Pérez-D'Arpino et al. 2021]. Claudia Pérez-D'Arpino et al. (2021). "Robot navigation in constrained pedestrian environments using reinforcement learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1140–1146 (cit. on pp. 89, 90, 98).

[Pérez-Higueras, Caballero, and Merino 2018]. Noé Pérez-Higueras, Fernando Caballero, and Luis Merino (2018a). "Learning human-aware path planning with fully convolutional networks". In: *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 5897–5902 (cit. on p. 14).

[Pérez-Higueras, Caballero, and Merino 2018]— . (2018b). "Teaching robot navigation behaviors to optimal RRT planners". In: *International Journal of Social Robotics* 10, pp. 235–249 (cit. on p. 15).

[Pérez-Higueras, Otero, et al. 2023]. Noé Pérez-Higueras, Roberto Otero, et al. (2023). "Hunavsim: A ros 2 human navigation simulator for benchmarking human-aware robot navigation". In: *IEEE robotics and automation letters* 8.11, pp. 7130–7137 (cit. on p. 28).

[Pfeiffer et al. 2017]. Mark Pfeiffer et al. (May 2017). "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)*, pp. 1527–1533. DOI: 10.1109/ICRA.2017.7989182. arXiv: 1609.07910[cs]. URL: http://arxiv.org/abs/1609.07910 (visited on 01/13/2023) (cit. on p. 14).

[Pintos Gómez de las Heras, Martınez-Tomás, and Cuadra Troncoso 2023]. Borja Pintos Gómez de las Heras, Rafael Martınez-Tomás, and José Manuel Cuadra Troncoso (2023). "Self-learning robot autonomous navigation with deep reinforcement learning techniques". In: *Applied Sciences* 14.1, p. 366 (cit. on p. 18).

[Pirk et al. 2022]. Sören Pirk et al. (2022). "A protocol for validating social navigation policies". In: *arXiv preprint arXiv:2204.05443* (cit. on p. 30).

[Pokle et al. 2019]. Ashwini Pokle et al. (2019). "Deep local trajectory replanning and control for robot navigation". In: *2019 international conference on robotics and automation (ICRA)*. IEEE, pp. 5815–5822 (cit. on p. 16).

[Puig et al. 2023]. Xavi Puig et al. (2023). *Habitat 3.0: A Co-Habitat for Humans, Avatars and Robots* (cit. on p. 28).

[Raffin et al. 2021]. Antonin Raffin et al. (2021). "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22.268, pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html (cit. on p. 99).

[Rios-Martinez, Renzaglia, et al. 2012]. Jorge Rios-Martinez, Alessandro Renzaglia, et al. (2012). "Navigating between people: A stochastic optimization approach". In: *2012 IEEE International Conference on Robotics and Automation*. IEEE, pp. 2880–2885 (cit. on p. 12).

[Rios-Martinez, Spalanzani, and Laugier 2011]. Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier (2011). "Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 2014–2019 (cit. on p. 12).

[Rios-Martinez, Spalanzani, and Laugier 2015]— . (2015). "From proxemics theory to socially-aware navigation: A survey". In: *International Journal of Social Robotics* 7.2, pp. 137–153 (cit. on p. 11).

[Roesmann et al. 2012]. Christoph Roesmann et al. (2012). "Trajectory modification considering dynamic constraints of autonomous robots". In: *ROBOTIK 2012; 7th German Conference on Robotics*, pp. 1–6 (cit. on pp. 8, 42).

[Rosmann, Makarow, and Bertram 2021]. Christoph Rosmann, Artemi Makarow, and Torsten Bertram (June 2021). "Online Motion Planning based on Nonlinear Model Predictive Control with Non-Euclidean Rotation Groups". In: *2021 European Control Conference (ECC)*. IEEE. DOI: 10.23919/ecc54610.2021.9654872. URL: https://doi.org/10.23919%2Fecc54610.2021.9654872 (cit. on p. 8).

[Rösmann et al. 2013]. Christoph Rösmann et al. (2013). "Efficient trajectory optimization using a sparse model". In: *2013 European Conference on Mobile Robots*, pp. 138–143. DOI: 10.1109/ECMR.2013.6698833 (cit. on pp. 8, 42).

[Rudenko et al. 2018]. Andrey Rudenko et al. (Oct. 2018). "Human Motion Prediction Under Social Grouping Constraints". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). ISSN: 2153-0866, pp. 3358–3364. DOI: 10.1109/IROS.2018.8594258 (cit. on p. 12).

[Samadi and Othman 2013]. Masoud Samadi and Mohd Fauzi Othman (2013). "Global path planning for autonomous mobile robot using genetic algorithm". In: *2013 International Conference on Signal-Image Technology & Internet-Based Systems*. IEEE, pp. 726–730 (cit. on p. 68).

[Samavi et al. 2025]. Sepehr Samavi et al. (2025). "SICNav: Safe and Interactive Crowd Navigation using Model Predictive Control and Bilevel Optimization". In: *IEEE Trans. Robot.* 41. arXiv:2310.10982 [cs], pp. 801–818. ISSN: 1552-3098, 1941-0468. DOI: 10.1109/TRO.2024.3484634. URL: http://arxiv.org/abs/2310.10982 (visited on 04/17/2025) (cit. on p. 13).

[Schulman et al. 2017]. John Schulman et al. (2017). "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (cit. on pp. 88, 99).

[Sebastian, Banisetty, and Feil-Seifer 2017]. Meera Sebastian, Santosh Balajee Banisetty, and David Feil-Seifer (Aug. 2017). "Socially-aware navigation planner using models of human-human interaction". In: *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN). ISSN: 1944-9437, pp. 405–410. DOI: 10.1109/ROMAN.2017.8172334 (cit. on p. 12).

[Selim et al. 2022]. Mahmoud Selim et al. (2022). "Safe reinforcement learning using data-driven predictive control". In: *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*. IEEE, pp. 1–6 (cit. on p. 17).

[Shao et al. 2025]. Xuyang Shao et al. (2025). "A Personalized Comfort Space With Variable Shape Based on Environmental Information for Robot Navigation in Homes". In: *IEEE Robotics and Automation Letters* (cit. on p. 12).

[Shcherbyna et al. 2024]. Volodymyr Shcherbyna et al. (2024). "Arena 4.0: A Comprehensive ROS2 Development and Benchmarking Platform for Human-centric Navigation Using Generative-Model-based Environment Generation". In: *arXiv preprint arXiv:2409.12471* (cit. on pp. 28, 41).

[Singamaneni and Alami 2020]. Phani Teja Singamaneni and Rachid Alami (2020). "HATEB-2: Reactive Planning and Decision making in Human-Robot Co-navigation". In: *International Conference on Robot & Human Interactive Communication*. DOI: 10.1109/RO-MAN47096.2020.9223463 (cit. on pp. 1, 13, 58).

[Singamaneni, Bachiller-Burgos, et al. 2024]. Phani Teja Singamaneni, Pilar Bachiller-Burgos, et al. (2024). "A survey on socially aware robot navigation: Taxonomy and future challenges". In: *The International Journal of Robotics Research* 43.10, pp. 1533–1572 (cit. on p. 10).

[Singamaneni, Favier, and Alami 2021]. Phani Teja Singamaneni, Anthony Favier, and Rachid Alami (2021). "Human-Aware Navigation Planner for Diverse Human-Robot Ineraction Contexts". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (cit. on pp. 42, 54, 64).

[Singamaneni, Favier, and Alami 2023]— . (2023). "Towards benchmarking human-aware social robot navigation: A new perspective and metrics". In: *2023 32nd IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, pp. 914–921 (cit. on p. 22).

[Smith et al. 2023]. Trevor Smith et al. (2023). "Socially aware robot obstacle avoidance considering human intention and preferences". In: *International journal of social robotics* 15.4, pp. 661–678 (cit. on p. 58).

[Sousa et al. 2022]. Raphaell Maciel de Sousa et al. (June 18, 2022). "A New Approach for Including Social Conventions into Social Robots Navigation by Using Polygonal Triangulation and Group Asymmetric Gaussian Functions". In: *Sensors* 22.12. ISSN: 1424-

8220. DOI: 10.3390/s22124602. URL: https://www.mdpi.com/1424-8220/22/12/4602 (cit. on p. 12).

[Sprague et al. 2023]. Zayne Sprague et al. (2023). "Socialgym 2.0: Simulator for multi-agent social robot navigation in shared human spaces". In: *arXiv preprint arXiv:2303.05584* (cit. on p. 28).

[Tingguang et al. 2019]. Li Tingguang et al. (2019). "HouseExpo: A Large-scale 2D Indoor Layout Dataset for Learning-based Algorithms on Mobile Robots". In: *arXiv preprint arXiv:1903.09845* (cit. on pp. 40, 99).

[Towers et al. 2024]. Mark Towers et al. (2024). "Gymnasium: A standard interface for reinforcement learning environments". In: *arXiv preprint arXiv:2407.17032* (cit. on p. 99).

[Trautman and Krause 2010]. Peter Trautman and Andreas Krause (Oct. 2010). "Unfreezing the robot: Navigation in dense, interacting crowds". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010). Taipei: IEEE, pp. 797–803. ISBN: 978-1-4244-6674-0. DOI: 10.1109/IROS.2010.5654369. (Visited on 01/11/2023) (cit. on p. 12).

[Tsoi et al. 2022]. Nathan Tsoi et al. (2022). "SEAN 2.0: Formalizing and Generating Social Situations for Robot Navigation". In: *IEEE Robotics and Automation Letters*, pp. 1–8. DOI: 10.1109/LRA.2022.3196783 (cit. on p. 28).

[Uher et al. 2019]. Vojtěch Uher et al. (2019). "Hierarchical hexagonal clustering and indexing". In: *Symmetry* 11.6, p. 731 (cit. on p. 68).

[Ulrich and Borenstein 1998]. Iwan Ulrich and Johann Borenstein (1998). "VFH+: Reliable obstacle avoidance for fast mobile robots". In: *Proceedings. 1998 IEEE international conference on robotics and automation (Cat. No. 98CH36146)*. Vol. 2. IEEE, pp. 1572–1577 (cit. on p. 98).

[Vanhée, Jeanpierre, and Mouaddib 2022]. Loıs Vanhée, Laurent Jeanpierre, and Abdel-Illah Mouaddib (2022). "Anxiety-sensitive planning: from formal foundations to algorithms and applications". In: *Proceedings of the International Conference on Automated Planning and Scheduling*. Vol. 32, pp. 730–740 (cit. on p. 58).

[Wang, Chan, et al. 2022]. Junxian Wang, Wesley P Chan, et al. (2022). "Metrics for evaluating social conformity of crowd navigation algorithms". In: *2022 IEEE International Conference on Advanced Robotics and Its Social Impacts (ARSO)*. IEEE, pp. 1–6 (cit. on p. 20).

[Wang, Obi, and Min 2025]. Weizheng Wang, Ike Obi, and Byung-Cheol Min (2025). "Multi-Agent LLM Actor-Critic Framework for Social Robot Navigation". In: *arXiv preprint arXiv:2503.09758* (cit. on p. 16).

[Wang, Wang, Mao, et al. 2023]. Weizheng Wang, Ruiqi Wang, Le Mao, et al. (2023). "Navistar: Socially aware robot navigation with hybrid spatio-temporal graph transformer and

preference learning". In: *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 11348–11355 (cit. on pp. 96, 98).

[Wang, Wang, Xie, et al. 2025]. Zhongrui Wang, Shuting Wang, Yuanlong Xie, et al. (2025). "Autonomous Navigation of Mobile Robots: A Hierarchical Planning–Control Framework with Integrated DWA and MPC". In: *Sensors (Basel, Switzerland)* 25.7, p. 2014 (cit. on p. 13).

[Watkins and Dayan 1992]. Christopher J. C. H. Watkins and Peter Dayan (May 1992). "Q-learning". In: *Machine Learning* 8.3, pp. 279–292 (cit. on p. 88).

[Williams 1992]. Ronald J Williams (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3, pp. 229–256 (cit. on p. 88).

[Wu et al. 2020]. Chenyan Wu et al. (2020). "MEBOW: Monocular Estimation of Body Orientation In the Wild". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3451–3461 (cit. on p. 113).

[Wulfmeier et al. 2017]. Markus Wulfmeier et al. (2017). "Large-scale cost function learning for path planning using deep inverse reinforcement learning". In: *The International Journal of Robotics Research* 36.10, pp. 1073–1087 (cit. on p. 15).

[Xiao et al. 2022]. Xuesu Xiao et al. (2022). "Motion planning and control for mobile robot navigation using machine learning: a survey". In: *Autonomous Robots* 46.5, pp. 569–597 (cit. on pp. 10, 17).

[Xie and Dames 2023]. Zhanteng Xie and Philip Dames (2023). "Drl-vo: Learning to navigate through crowded dynamic scenes using velocity obstacles". In: *IEEE Transactions on Robotics* 39.4, pp. 2700–2719 (cit. on pp. 1, 15, 42, 88, 89, 90, 92, 93, 94, 96, 98).

[Yang and Peters 2019]. Fangkai Yang and Christopher Peters (2019). "Social-aware navigation in crowds with static and dynamic groups". In: *2019 11th International Conference on Virtual Worlds and Games for Serious Applications (VS-Games)*, pp. 1–4. DOI: 10.1109/VS-Games.2019.8864512 (cit. on p. 12).

[Zhang et al. 2025]. Ze Zhang et al. (2025). "Future-Oriented Navigation: Dynamic Obstacle Avoidance With One-Shot Energy-Based Multimodal Motion Prediction". In: *IEEE Robotics and Automation Letters* 10.8, pp. 8043–8050. DOI: 10.1109/LRA.2025.3575969 (cit. on p. 43).

[Zhou et al. 2024]. Xinyu Zhou et al. (2024). "Her-drl: Heterogeneous relational deep reinforcement learning for decentralized multi-robot crowd navigation". In: *arXiv preprint arXiv:2403.10083* (cit. on p. 96).

Appendix A

Visual Results of Simulations

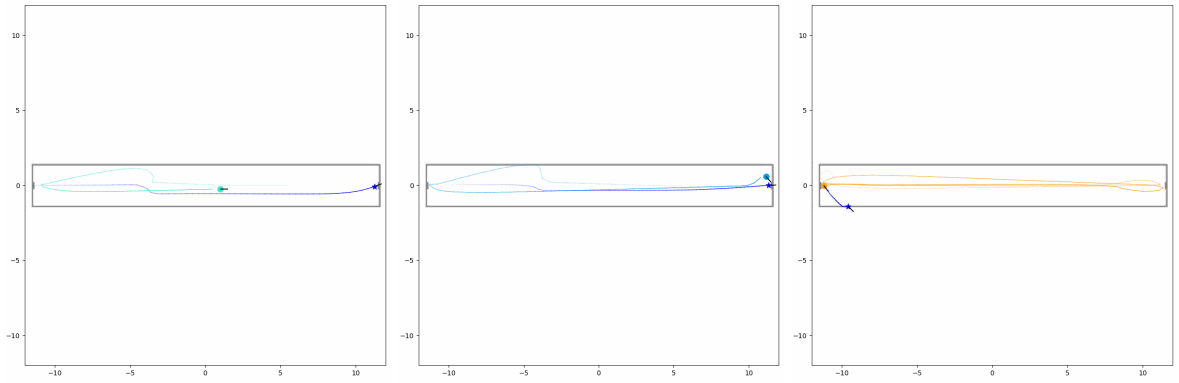
In addition of the quantitative analyses presented in Chapters 2 and 5, this appendix provides additional visual results from our simulations. The robots' behavior in two sets of experiments—the assessment of state-of-the-art methods and the results of our proposed HBSN and MDHC approaches—is illustrated in the figures. Representative scenarios that highlight navigation performance, interactions with human, and social norm adherence are presented in each section. Here the list of these scenarios:

- Frontal Approach,
- Narrow Passage,
- Corner,
- Intersection,
- Circle Crowd,
- and Perpendicular Traffic.

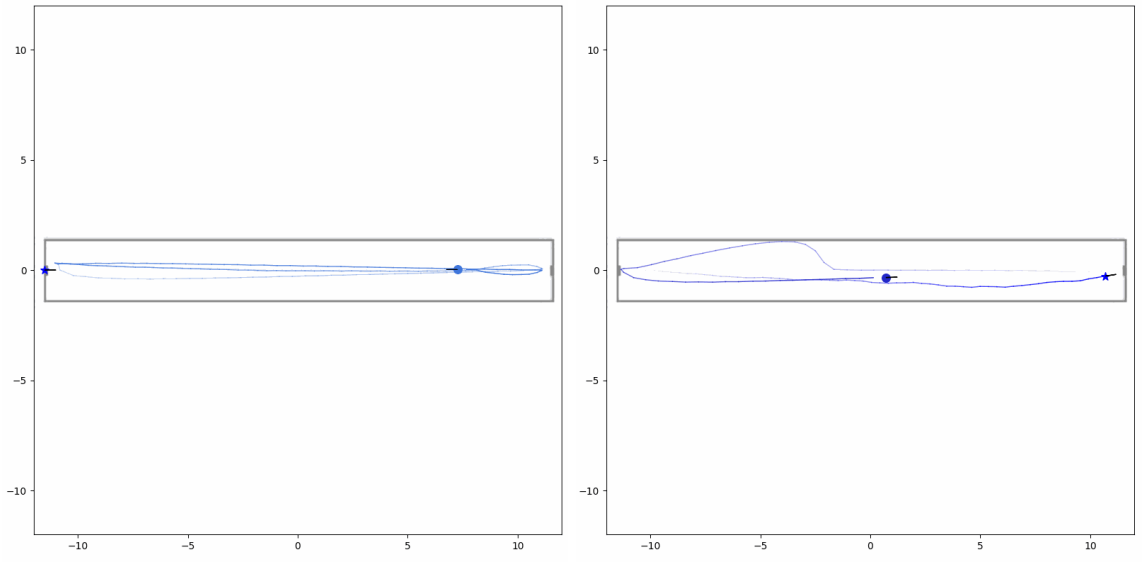
A qualitative evaluation of the methods' advantages and disadvantages in various scenarios is made possible by these visualizations.

A.1 State-of-the-art Benchmark Results

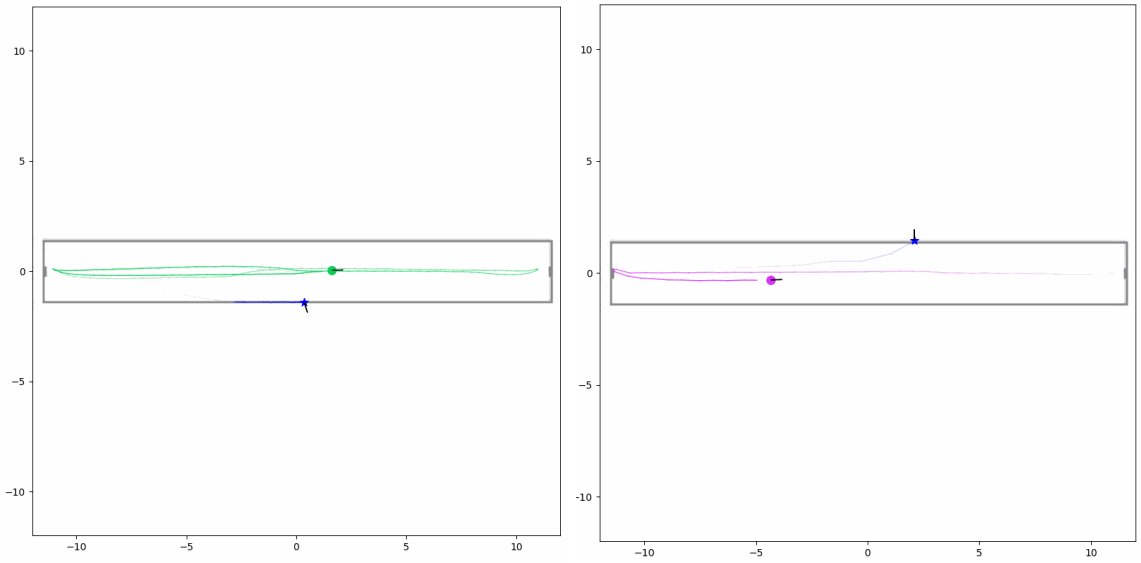
This section displays the visual results from the RobotSNAP platform evaluation of state-of-the-art social navigation methods: DWA, HATEB, CADRL, SARL*, DRL-VO, BRNE and MPC-EBM. Each figure shows the behavior of the robot and nearby humans in a particular scenario. Humans are shown as colored circles, and the robot is represented by a blue star. Over time, orientations and previous trajectories are indicated by arrows and faded traces. A qualitative comparison of the approaches in various settings and crowd densities is made possible by these visualizations.



(a) DWB (b) HATEB (c) CADRL

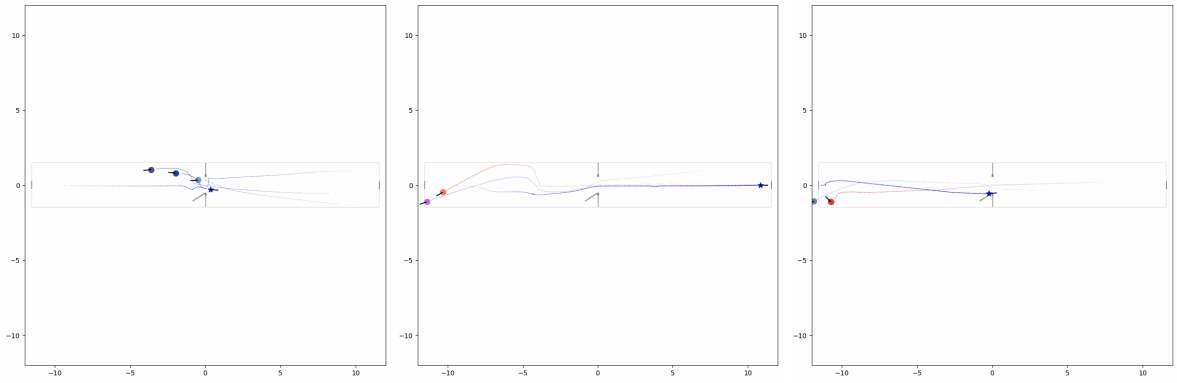


(d) SARL* (e) DRL_VO

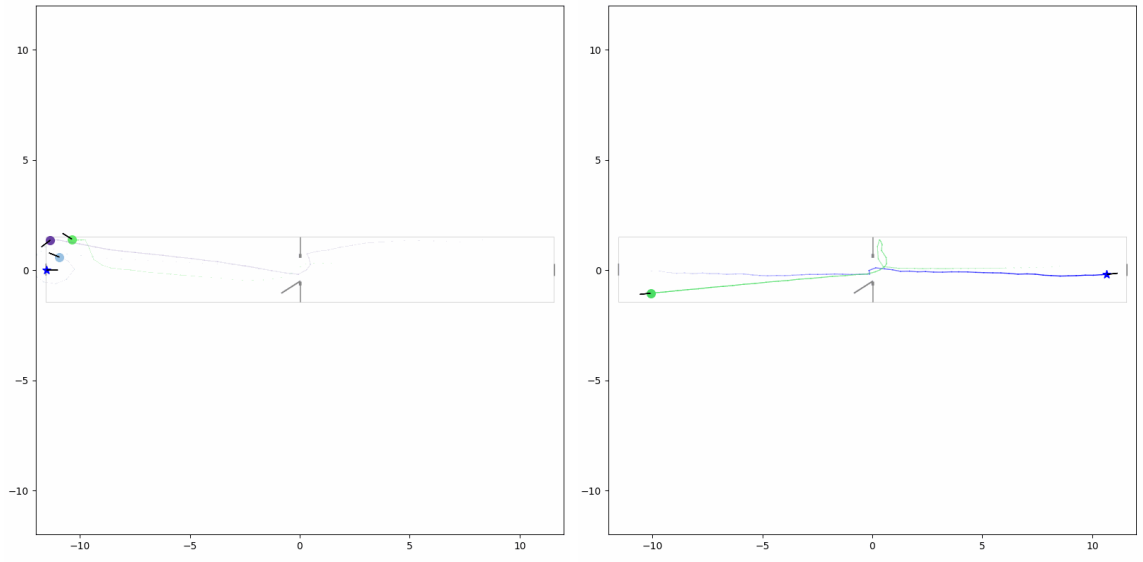


(f) BRNE (g) MPC-EBM

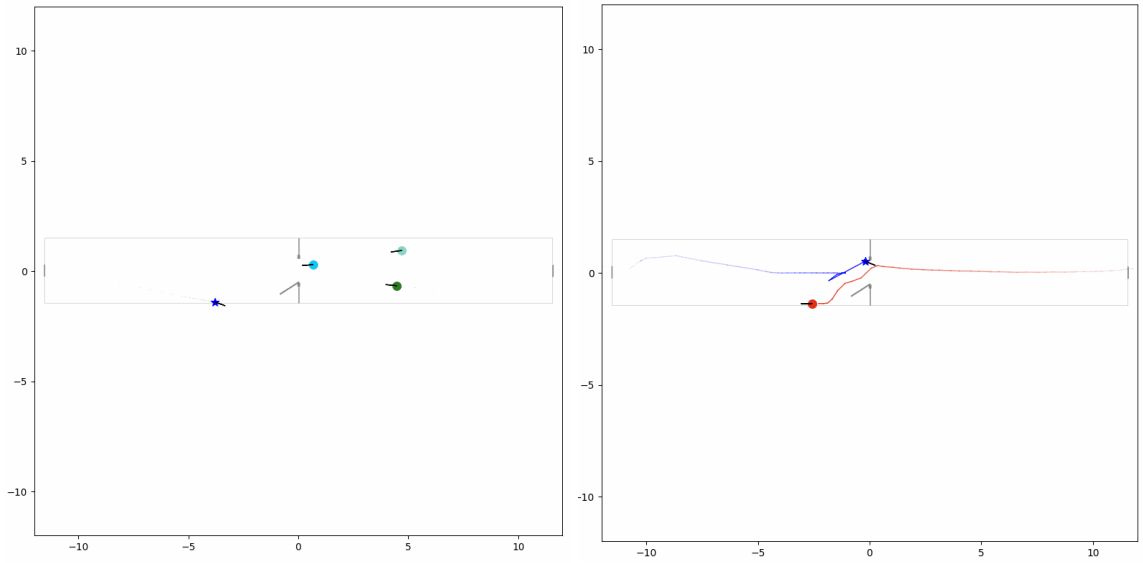
Figure A.1 – Frontal Approach - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) DWB (b) HATEB (c) CADRL

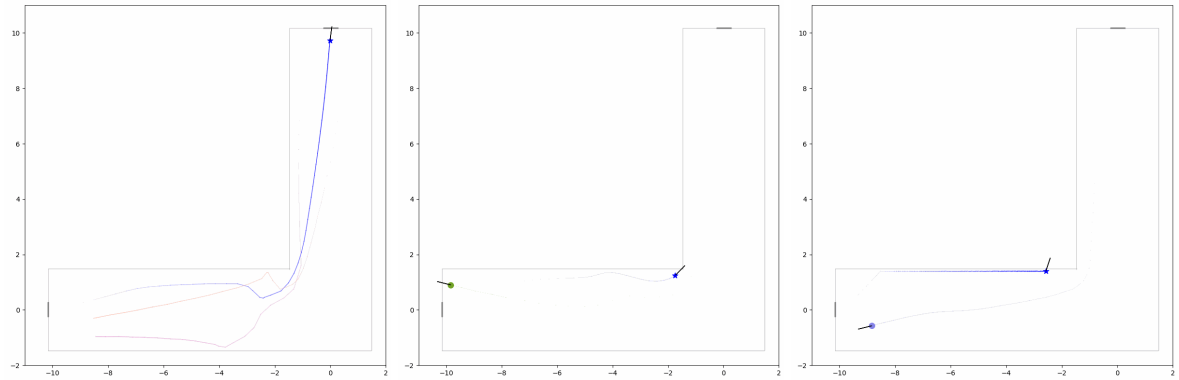


(d) SARL* (e) DRL_VO

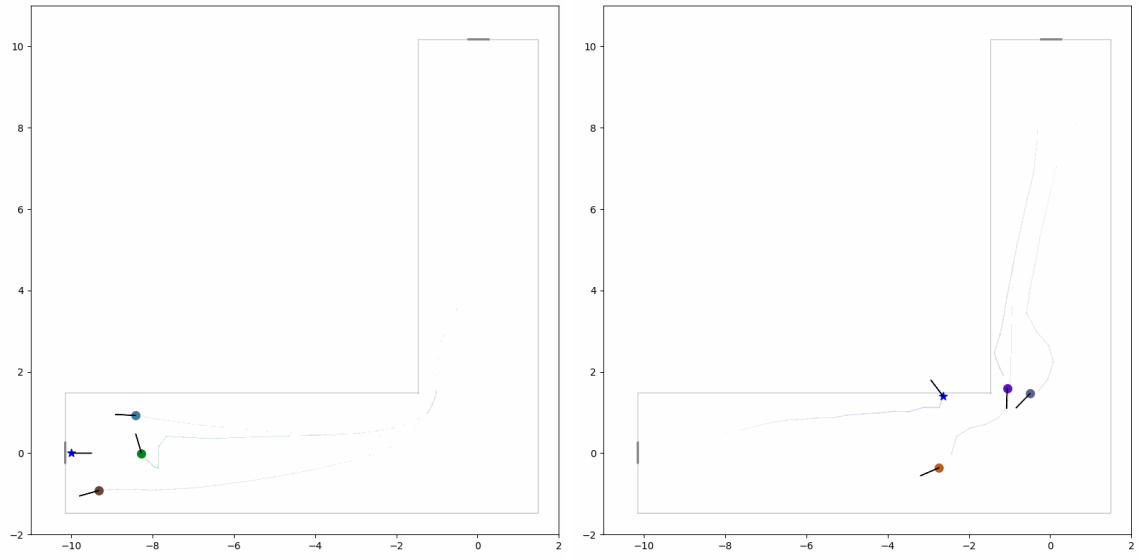


(f) BRNE (g) MPC-EBM

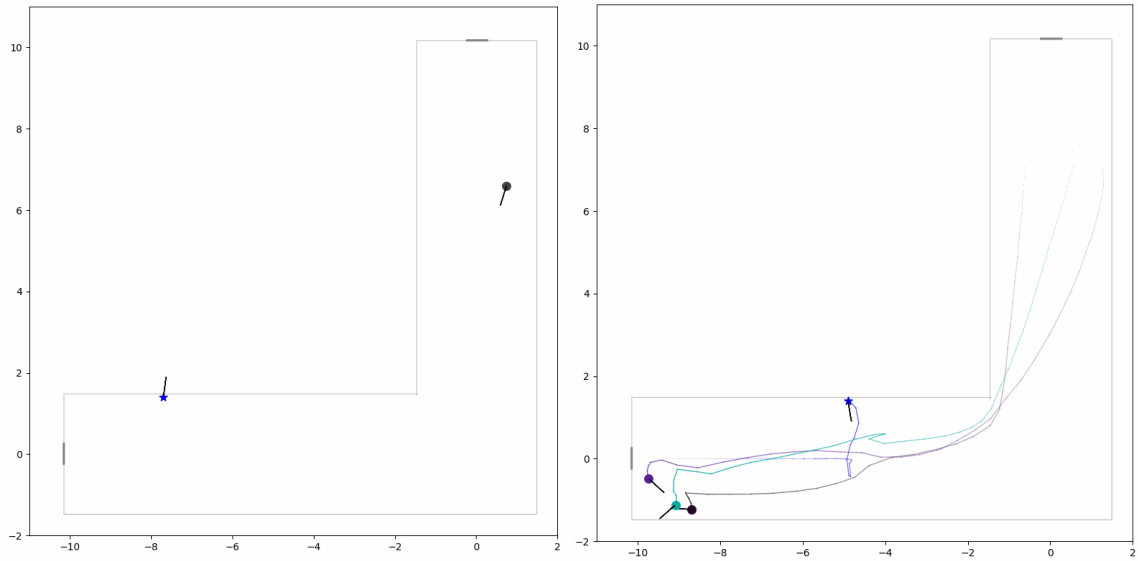
Figure A.2 – Narrow Passage - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) DWB (b) HATEB (c) CADRL

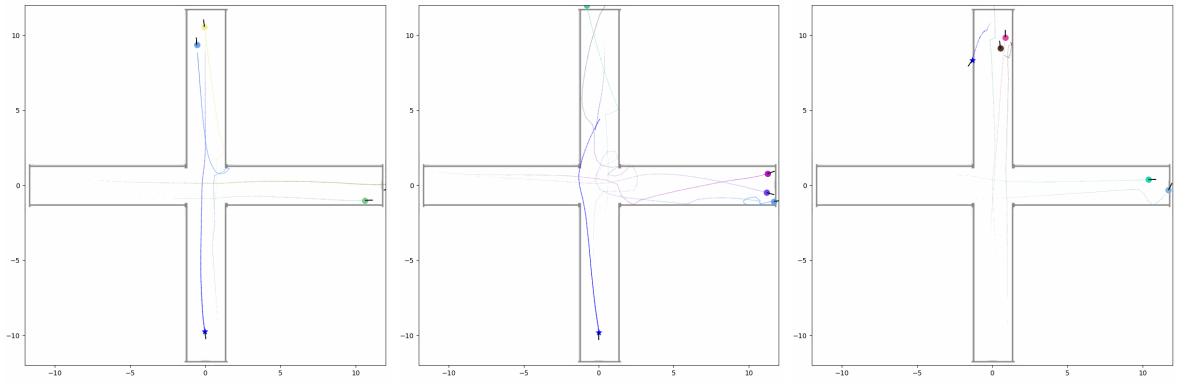


(d) SARL* (e) DRL_VO

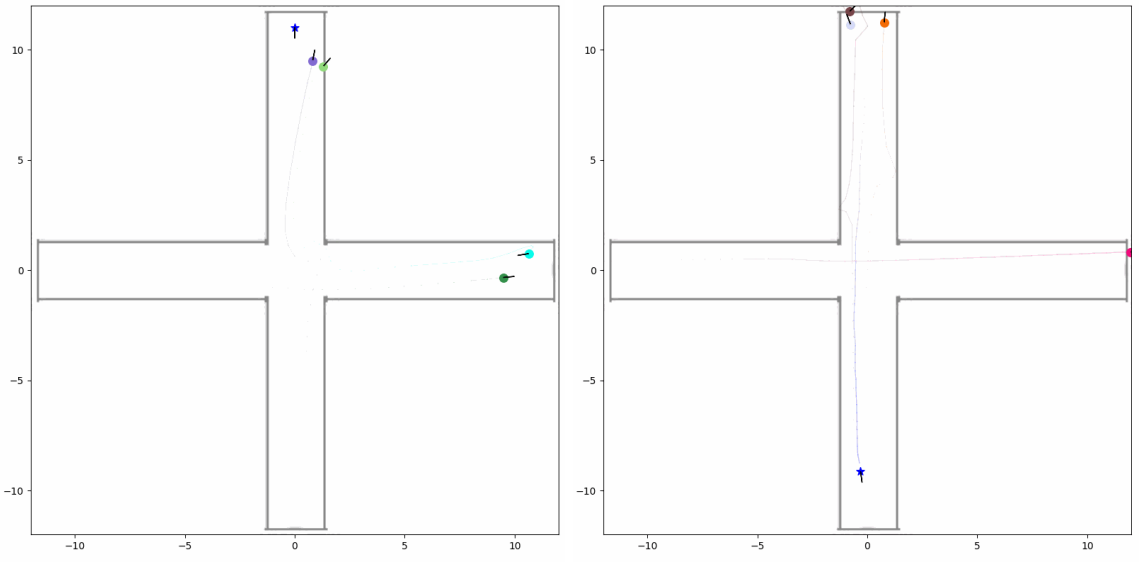


(f) BRNE (g) MPC-EBM

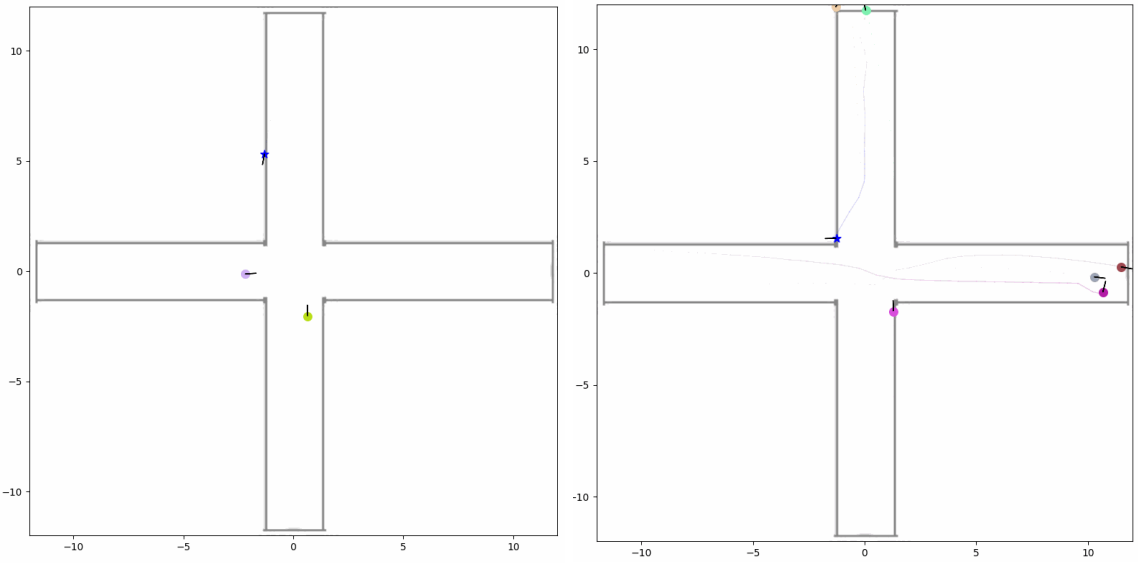
Figure A.3 – Corner - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) DWB (b) HATEB (c) CADRL

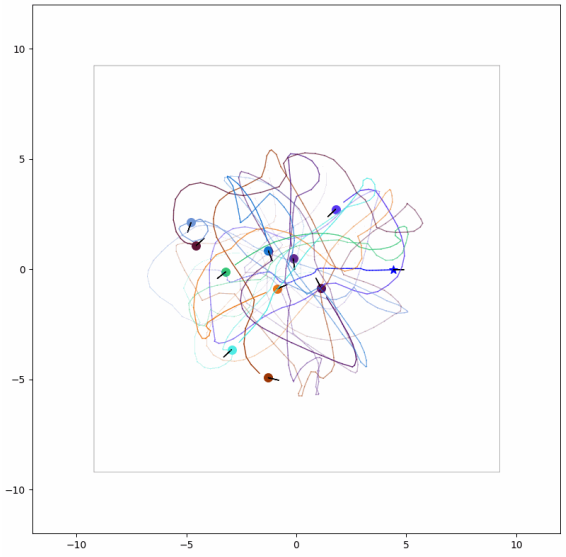


(d) SARL* (e) DRL_VO

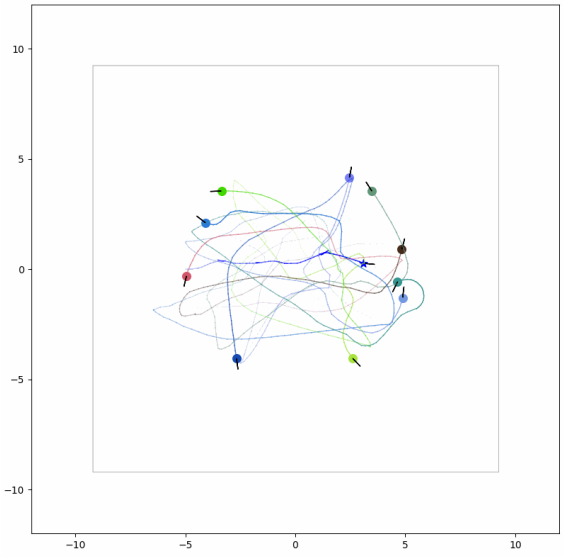


(f) BRNE (g) MPC-EBM

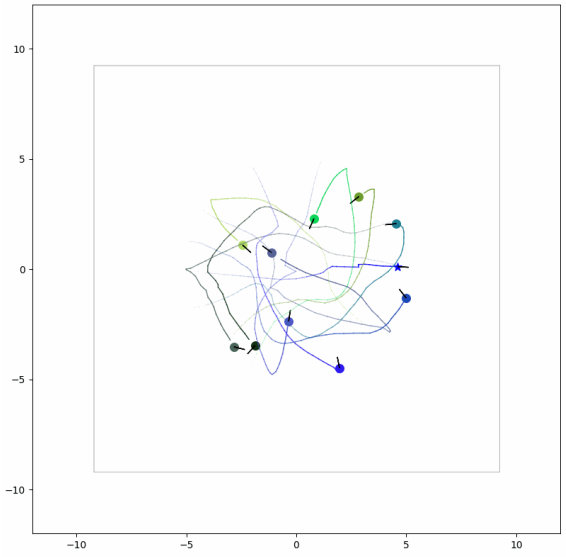
Figure A.4 – Intersection - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



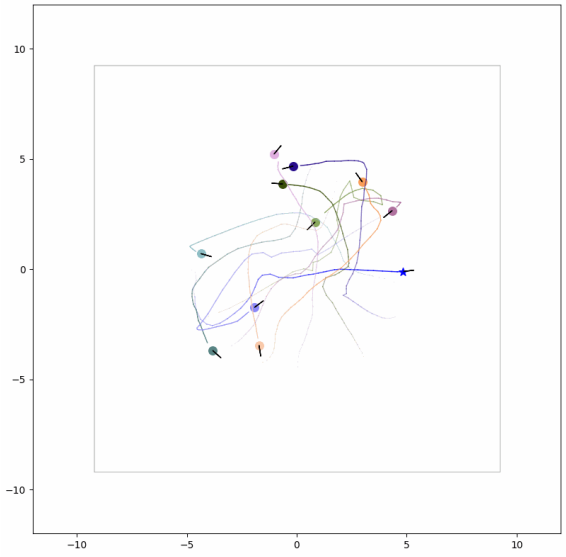
(a) DWB



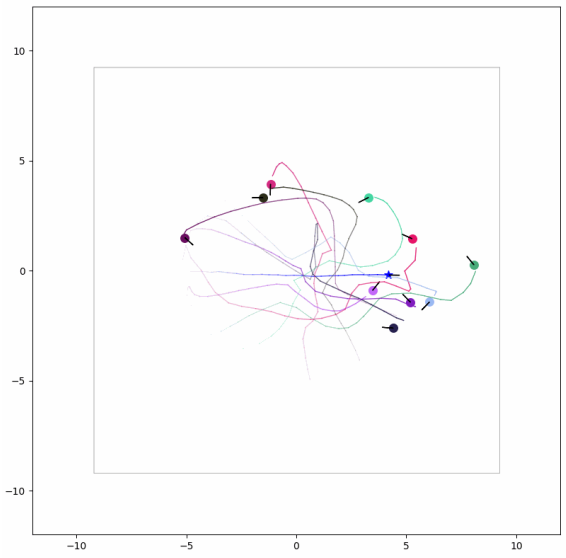
(b) HATEB



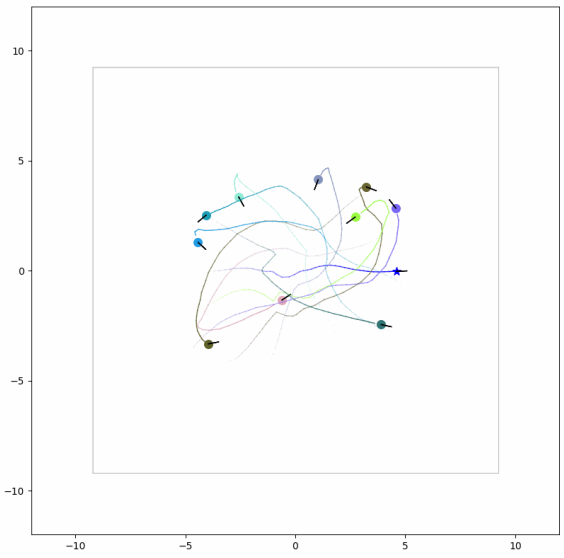
(c) CADRL



(d) SARL*

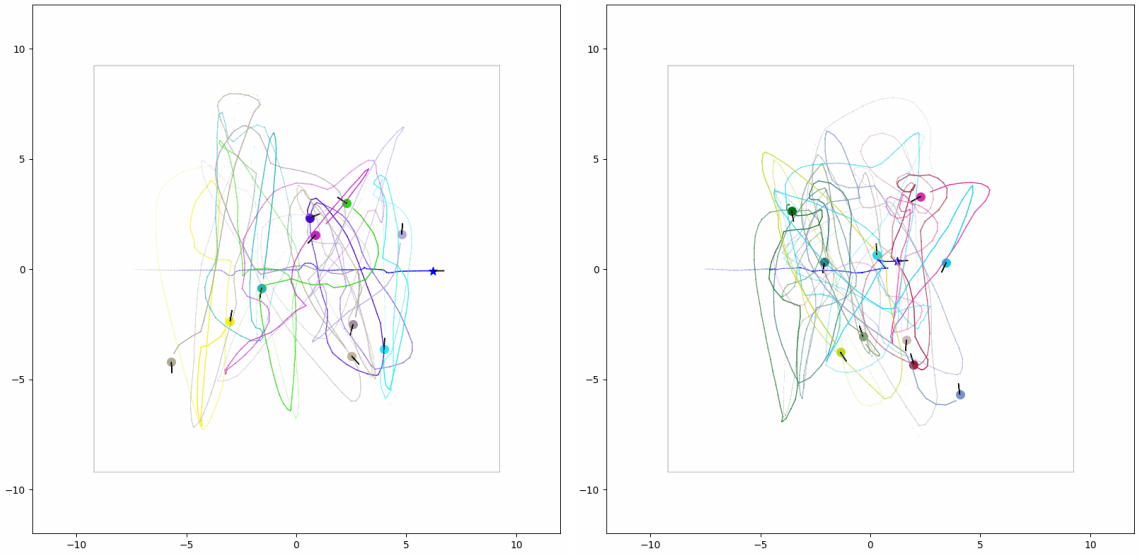


(e) DRL_VO

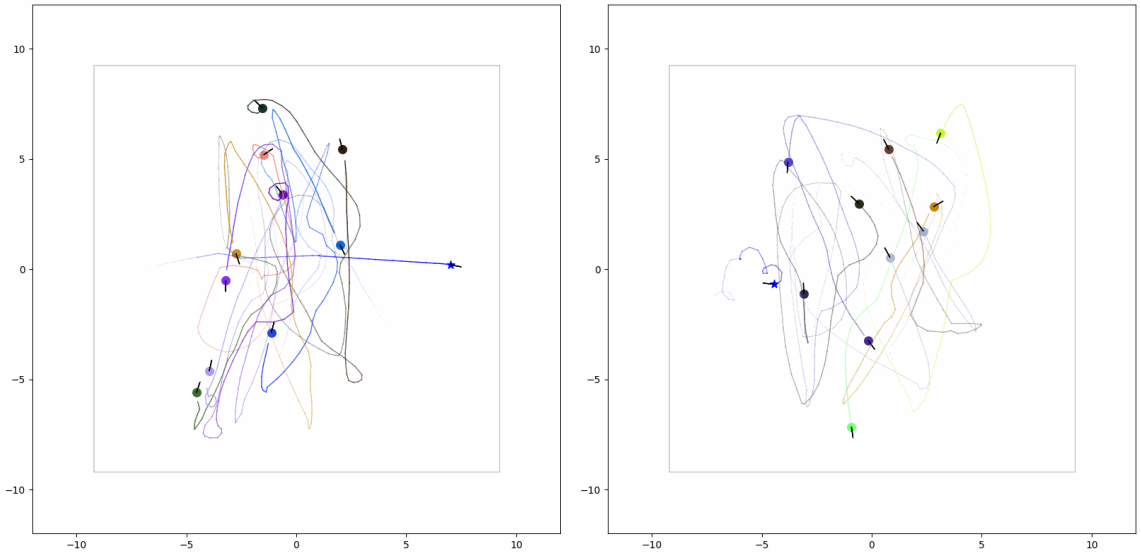


(f) BRNE

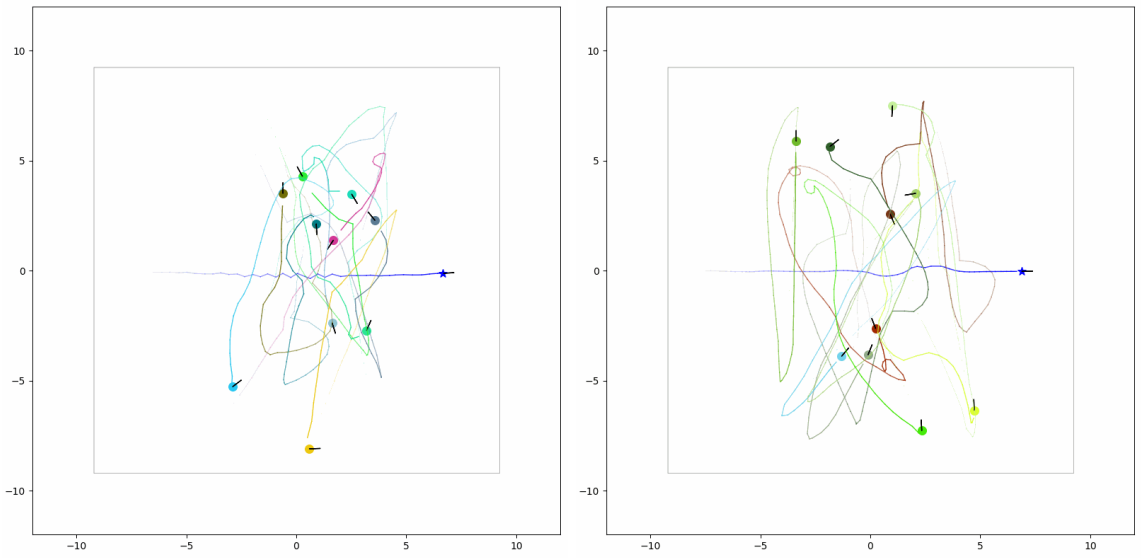
Figure A.5 – Circle Crowd - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) DWB (b) HATEB



(c) CADRL (d) SARL*

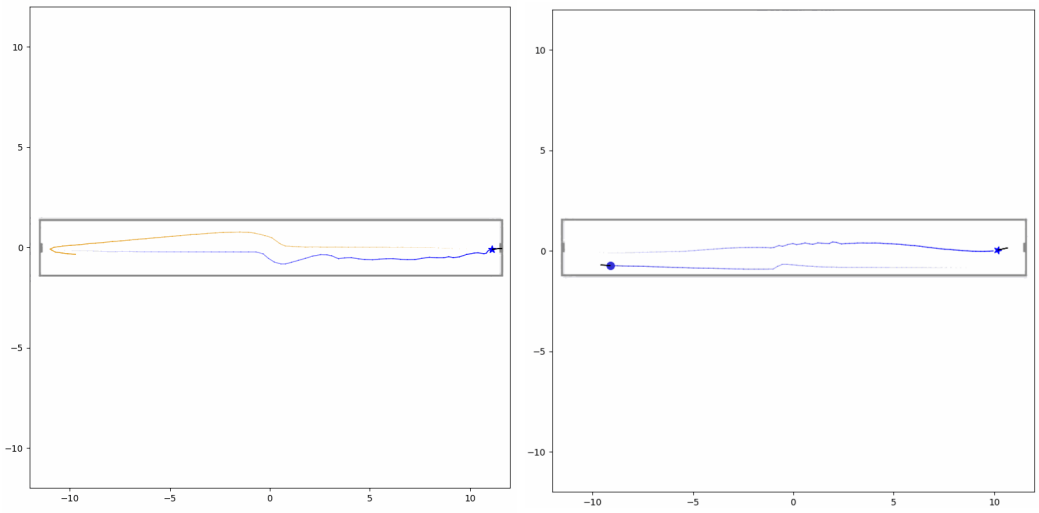


(e) DRL_VO (f) BRNE

Figure A.6 – Perpendicular Traffic - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

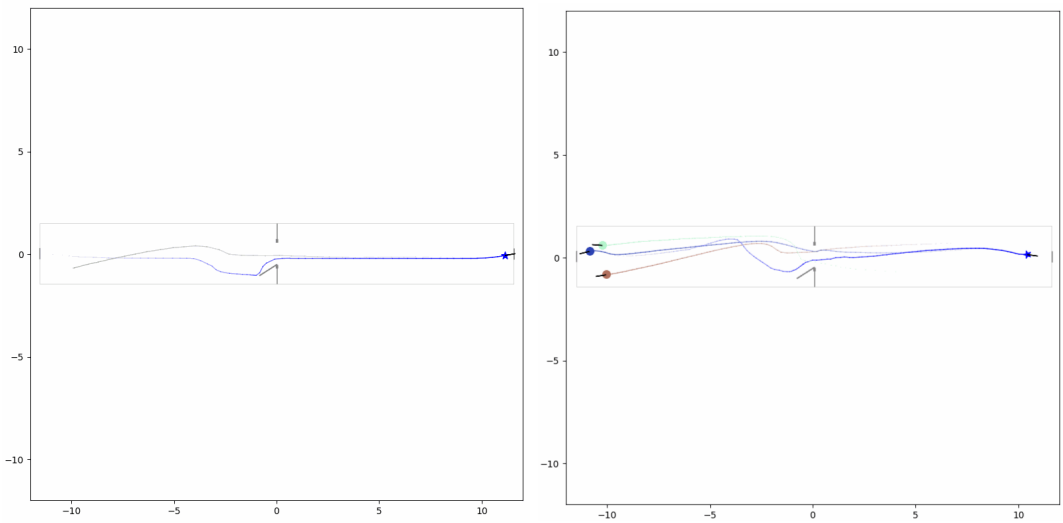
A.2 HBSN and MDHC Results

Here we present visual results for our proposed HBSN and MDHC methods.



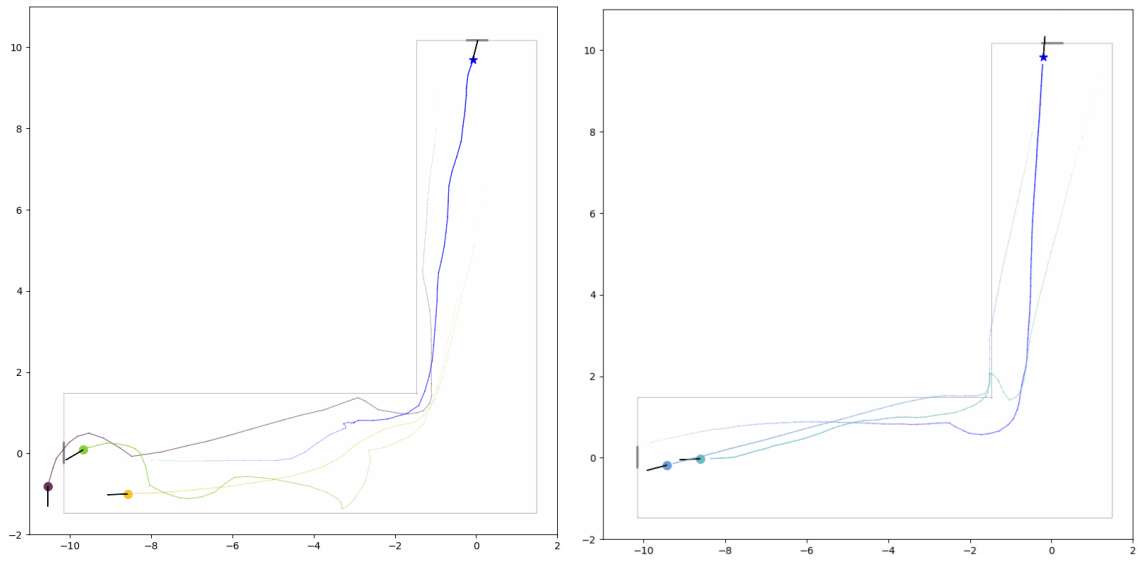
(a) HBSN (b) MDHC

Figure A.7 – Frontal Approach - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



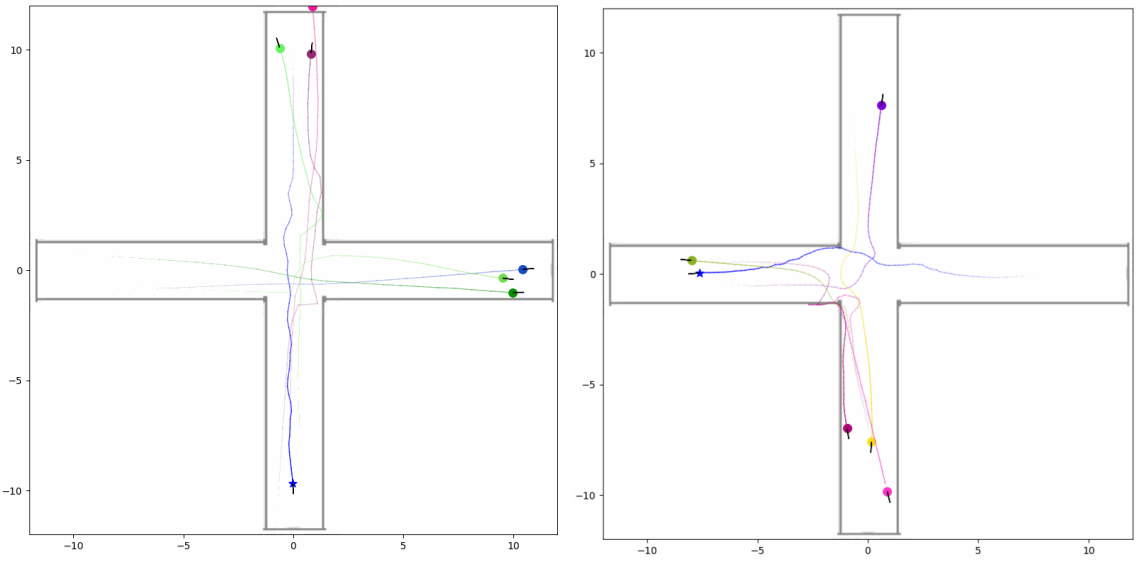
(a) HBSN (b) MDHC

Figure A.8 – Narrow Passage - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



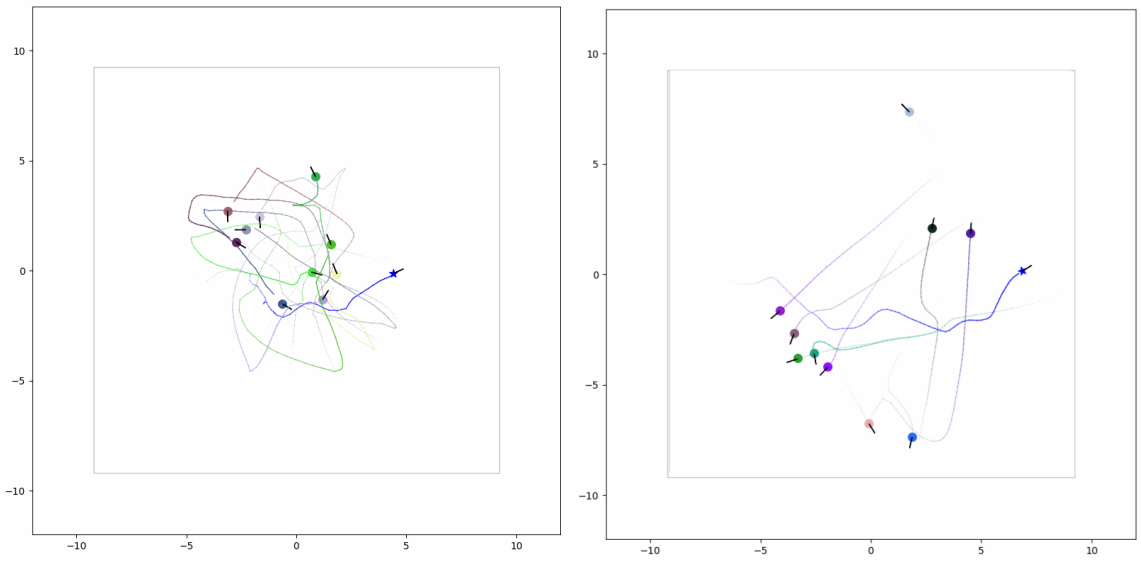
(a) HBSN (b) MDHC

Figure A.9 – Corner - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



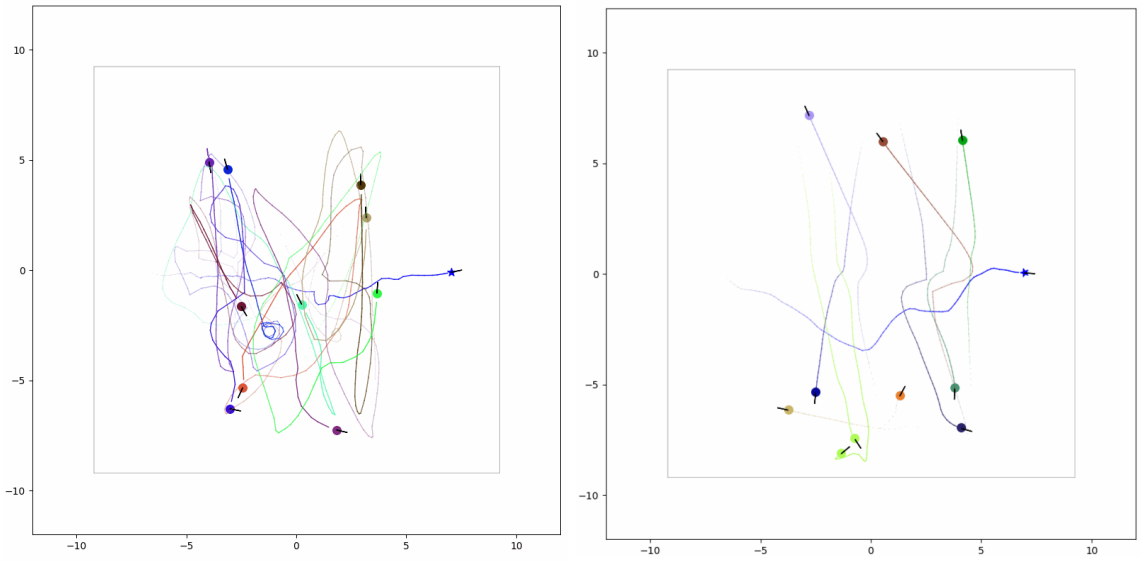
(a) HBSN (b) MDHC

Figure A.10 – Intersection - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) HBSN (b) MDHC

Figure A.11 – Circle Crowd - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.



(a) HBSN (b) MDHC

Figure A.12 – Perpendicular Traffic - The robot is represented by a blue star, while humans are depicted as circles with random colors. Arrows and faded traces indicate their orientations and past trajectories over time.

Appendix B

Complementary Information for MDHC

In addition to the descriptions in Chapter 4, this appendix offers more information on the Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC) approach. We describe the network architectures, training parameters, curriculum learning schedules, and mathematical formulations that were used to create and assess the approach.

B.1 Mathematical Details

The mathematical foundations of MDHC are described in detail in this section. This covers input encoding and social information representation.

B.1.1 Social Angular Field (SAF)

Human positions and orientations in relation to the robot are encoded in a structured angular representation by the Social Angular Field (SAF). The discretization of angles, weighting of contributions, and creation of the SAF feature vector that serves as the network's input are all covered in this subsection.

Angular discretization and weighting For each human i with relative position (d_x^i, d_y^i) and velocity (v_x^i, v_y^i), we compute its distance and angle:

$$d_i = \sqrt{(d_x^i)^2 + (d_y^i)^2}, \quad \theta_i = \text{atan2}(d_y^i, d_x^i). \tag{B.1}$$

The influence of this human on each bin k is given by a Gaussian kernel:

$$w_{ik} = \exp\left(- \frac{(\theta_i - \theta_k)^2}{2\sigma^2} \right), \tag{B.2}$$

where σ controls the spread of influence across neighboring bins (small $\sigma \rightarrow$ local, precise contribution; large $\sigma \rightarrow$ smoother, wider influence).

Occupancy The occupancy channel measures the relative presence of humans in each bin:

$$\text{occ}_k = \frac{\sum_i w_{ik}}{N} \in [0, 1], \tag{B.3}$$

where N is the total number of humans.

Distance We emphasize closer humans by computing a soft-min over distances:

$$\text{dist}_k = \frac{\sum_i w_{ik}\, d_i\, e^{-\alpha d_i}}{\sum_i w_{ik} e^{-\alpha d_i}}, \tag{B.4}$$

where $\alpha > 0$ controls the soft-min sharpness.

Time-to-collision The radial velocity of each human along the line connecting it to the robot is:

$$v_{\text{radial},i} = \frac{d_x^i v_x^i + d_y^i v_y^i}{d_i}. \tag{B.5}$$

The TTC is then:

$$\tau_i = \begin{cases} \frac{d_i}{-v_{\text{radial},i}}, & \text{if } v_{\text{radial},i} < 0, \\ \infty, & \text{otherwise.} \end{cases} \tag{B.6}$$

We aggregate into each bin using a soft-min similar to distance:

$$\text{TTC}_k = \frac{\sum_i w_{ik}\, \tau_i\, e^{-\alpha \tau_i}}{\sum_i w_{ik} e^{-\alpha \tau_i}}. \tag{B.7}$$

Final embedding Stacking the three channels (occupancy, distance, TTC) yields a tensor of shape $(3, K)$. A lightweight CNN encoder then produces the SAF embedding h_{saf} for decision making. This representation preserves angular geometry, captures both density and interaction risk, and remains compact for real-time navigation.

B.1.2 Spatio-Temporal Graph (ST-Graph)

Let N be the number of humans and H the history length (number of previous timesteps considered). For each human i, we define its temporal feature sequence over the last H timesteps as

$$x_i^{0:H} = [\, x_i^{t-H+1}, x_i^{t-H+2}, \ldots, x_i^t \,] \in \mathbb{R}^{H \times F_h}, \tag{B.8}$$

where x_i^t contains the relative position and velocity at timestep t, and F_h is the feature dimension.

Temporal encoding A small MLP maps each human's feature vector at each timestep to a hidden dimension, followed by a 1D temporal convolution across the history:

$$h_i = \text{Conv1D}\Big(\text{MLP}(x_i^{0:H})\Big), \quad i = 1, \ldots, N, \tag{B.9}$$

producing a compact embedding h_i that encodes short-term motion patterns.

Spatial graph construction Humans at the current timestep are connected in a fully connected spatial graph, where edges encode the Euclidean distance between humans:

$$e_{ij} = \|\mathbf{p}_i - \mathbf{p}_j\|_2, \quad i \neq j, \tag{B.10}$$

with $\mathbf{p}_i$ the relative position of human i to the robot.

Graph neural network propagation We apply two layers of a GATv2 (graph attention) network, incorporating the edge distances:

$$h_i' = \text{GATv2Layer}(h_i, \{h_j, e_{ij}\}_{j \in \mathcal{N}(i)}), \tag{B.11}$$

where $\mathcal{N}(i)$ is the set of neighboring nodes (all other humans). This allows each node to refine its embedding based on both its own motion and interactions with nearby humans.

Attention pooling over humans To aggregate node embeddings into a single vector for the network, we compute a weighted sum using a learned attention mechanism:

$$h_{st} = \sum_{i=1}^{N} \alpha_i h_i', \quad \alpha_i = \sigma(\text{MLP}([h_i', d_i])), \tag{B.12}$$

where d_i is the distance of human i to the robot and σ is a sigmoid activation. This pooling emphasizes humans that are close or potentially more relevant for collision risk.

B.2 Multi-Layer Cross-Modal

The embeddings produced by each branch, namely h_r, h_l, h_{saf}, and h_{st}, are first projected into a common embedding space of dimension d:

$$\tilde{h}_m = W_m h_m, \quad m \in \{r, l, saf, st\}. \tag{B.13}$$

where W_m is a learned projection matrix. This ensures that all modalities can be compared and combined within the same latent space.

We then construct a set of tokens $\mathcal{T} = \{\tilde{h}_l, \tilde{h}_{saf}, \tilde{h}_{st}\}$ representing the environment and social context. While the robot state $\tilde{h}_r$ is used as a query q, reflecting the fact that the robot must actively seek relevant information from the other modalities to guide its decision-making.

The fusion is performed using a stack of *L Cross-Modal Layers*, each combining cross-attention, self-attention, and feed-forward updates. For each layer, the cross-attention mechanism between robot query and the modality tokens can be expressed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\frac{QK^\top}{\sqrt{d}}V \tag{B.14}$$

where $Q = qW_Q$, $K = \mathcal{T}W_K$, and $V = \mathcal{T}W_V$. The query Q represents the modality that is seeking relevant information, while keys K and values V encode potential contributions from modalities.

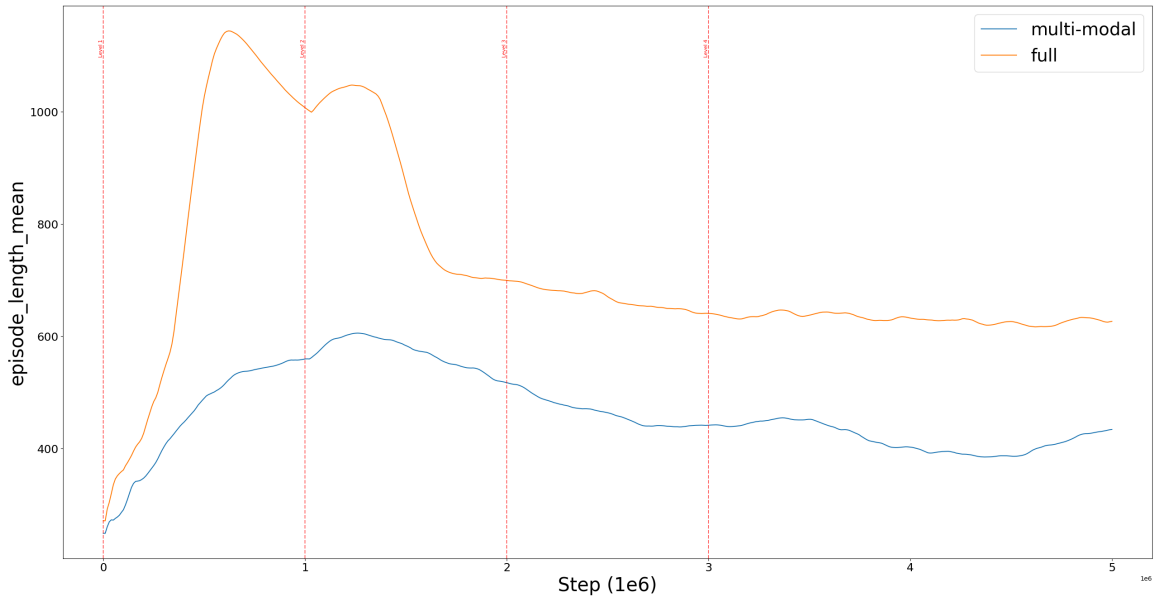
Stacking multiple layers of such cross-modal attention allows the network to reason about higher-order interactions, e.g., how combinations of human proximity, human motion trends, and environmental context should influence the robot's decisions.

After the final layer, the robot query and the aggregated modality tokens are concatenated:

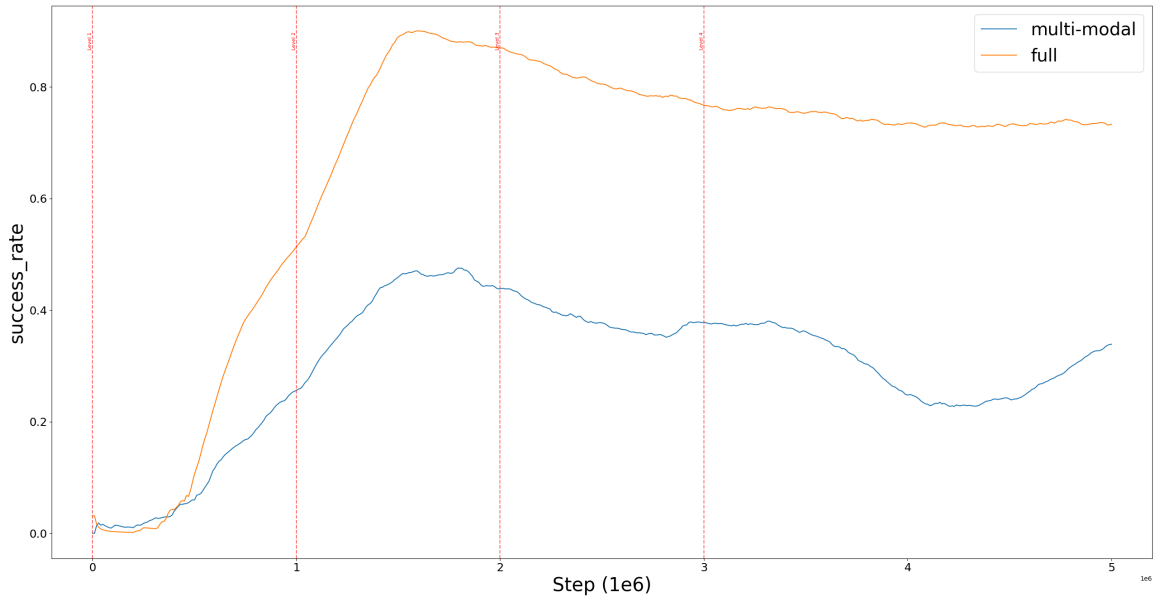
$$h_{\text{fused}} = [q^*; \text{mean}(\mathcal{T}^*)], \tag{B.15}$$

where q^* denotes the updated embedding of the robot state after the cross-modal layers, and $\mathcal{T}^*$ correspond to the set of tokens encoding LiDAR, SAF and ST-Graph. By concatenating q^* and mean($\mathcal{T}^*$), the model preserves a robot-centered representation and a global contextual representation, which are used to guide the policy network.

Despite these motivations, our experiments did not show improvements compared to simpler fusion strategies. We tried to use the multi-layer cross-modal in same context than network variant comparison in 4.5.1, the results is shown in Figure B.1. This is probably due to the low number of steps for learning, so more experiments would be needed to get better results. We nevertheless report this formulation as a potential direction for future exploration.



(a)



(b)

Figure B.1 – (a) Mean episode length (episode_length_mean) and (b) success rate (success_rate) during training for different network variants. Vertical dashed lines indicate CL levels.

B.3 Map Selection and Difficulty Scoring for HouseExpo

To implement curriculum learning (CL) with diverse environments, we introduce a preprocessing step to categorize maps from the HouseExpo dataset into different difficulty levels. HouseExpo provides a large collection of procedurally generated floorplans, but without explicit information about task difficulty. We therefore designed an automatic scoring and filtering mechanism to construct balanced sets of maps for training.

B.3.1 Difficulty Scoring

The geometry and room layout of each map are described in a JSON file. We combine a number of geometric and structural factors to calculate a *difficulty score*:

- **Number of vertices:** Maps with more walls and corners have more structural complexity.
- **Number of rooms:** In general, a greater number of rooms makes navigation more challenging.
- **Density:** ratio of free area to vertices, estimating the degree of clutter.
- **Overlap penalty:** penalty term applied when rooms overlap geometrically, usually resulting in narrow passageways.

The definition of the difficulty score is:

$$\text{score} = 0.4 \cdot \log(\text{vertices} + 1) + 0.3 \cdot \text{room\_num} + 0.2 \cdot \text{density} + 0.1 \cdot \text{overlap\_penalty}.$$

B.3.2 Categorization into Levels

We calculate the mean and standard deviation of all scores across HouseExpo in order to divide maps into different difficulty levels. We then define thresholds as:

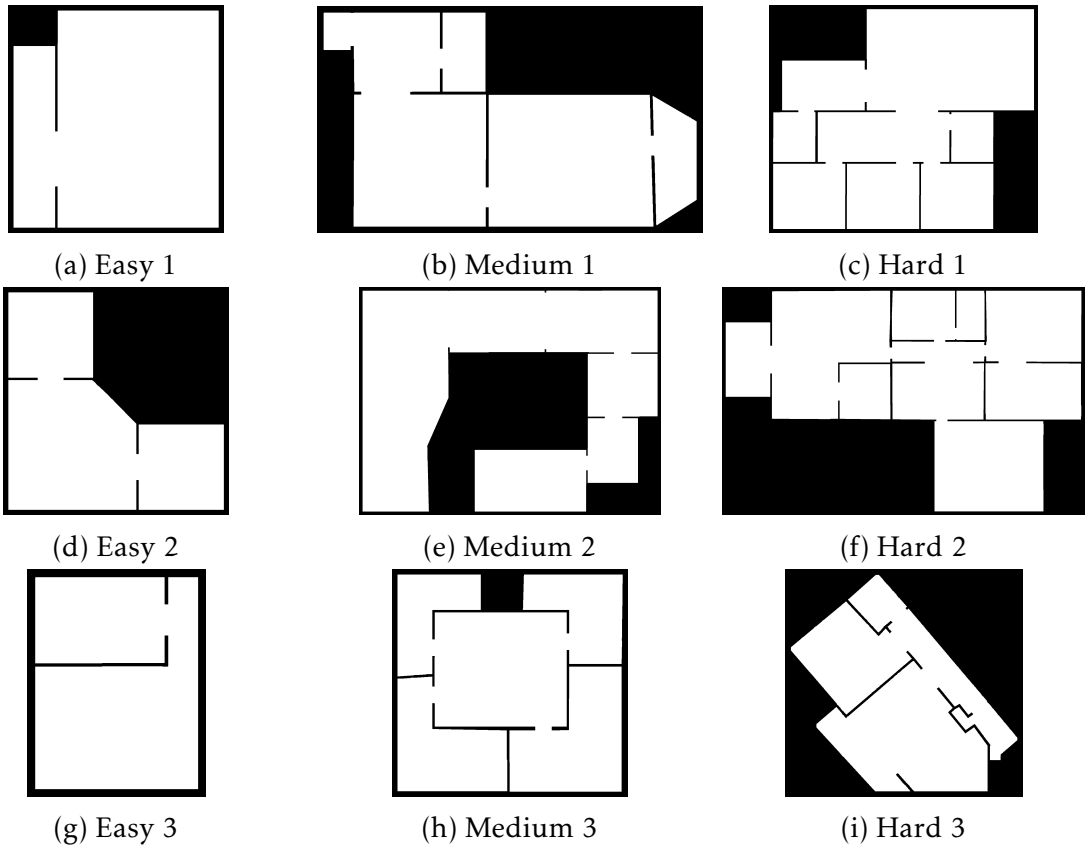
$$\text{low} = \mu - 0.5\sigma, \quad \text{high} = \mu + 0.5\sigma,$$

and assign maps into three difficulty categories:

- *easy*: score $\leq$ low,
- *medium*: low $<$ score $\leq$ high,
- *hard*: score $>$ high.

Finally, to maintain diversity while ensuring balance across categories, we randomly sample up to 100 maps per level. Both JSON files (map geometry) and PNG renderings are copied into level-specific folders.

B.3.3 Occupancy Map Example



(a) Easy 1 (b) Medium 1 (c) Hard 1

(d) Easy 2 (e) Medium 2 (f) Hard 2

(g) Easy 3 (h) Medium 3 (i) Hard 3

Figure B.2 – Examples of selected maps at different difficulty levels from HouseExpo. The first column corresponds to easy maps, the second to medium, and the third to hard. Each row shows different instances sampled within the same difficulty level.

B.4 MDHC Implementation Details

The Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC) policy's training configuration is covered in detail in this section. We provide an overview of the curriculum learning schedule and training hyperparameters that are utilized to gradually raise scenario difficulty and guarantee consistent learning in a variety of settings.

B.4.1 Parameters

The MDHC policy was trained using the PPO algorithm with the following hyperparameters. These parameters were chosen to strike a balance between computational efficiency, convergence speed, and learning stability.

Hyperparameter	**Value**
Algorithm	PPO
Number of environments	10
Learning rate	5×10^{-5}
Steps per rollout (n_steps)	1024
Batch size	256
Number of epochs (n_epochs)	10
Clip range	0.2
Value function coefficient (v_f)	0.5
Entropy coefficient (c_ent)	0.005
Discount factor (γ)	0.99
Policy feature dimension	512
Policy layers (π)	[512, 256]
Value function layers (V)	[256, 128]
Activation function	Tanh

Table B.1 – Hyperparameters used for training the MDHC policy.

B.4.2 Curriculum Learning Schedule

A Curriculum Learning (CL) approach was used to enable consistent training and gradual adaptation to complex scenarios. As indicated in Table B.2, the schedule gradually raises the number of people in the environment and the level of scenario difficulty from level 1 (very simple) to level 16 (most challenging).

Level	Dataset	Min Humans	Max Humans
1	basic	0	0
2	basic	0	1
3	basic	0	3
4	basic	0	5
5	easy	0	0
6	easy	0	1
7	easy	0	2
8	easy	1	2
9	medium	0	0
10	medium	0	2
11	medium	0	4
12	medium	2	5
13	hard	0	5
14	hard	2	5
15	hard	5	9
16	hard	10	15

Table B.2 – Curriculum learning schedule configuration.

Contents

Human-Aware Robot Navigation: From Comprehensive and Realistic Benchmarks to Versatile and Efficient Solutions

Abstract

A major challenge for mobile robotics is Human-Aware Robot Navigation (HAN); requiring robots to navigate safely and effectively in human-shared environments while adhering to social norms and adapting to dynamic contexts. Despite extensive research, key challenges remain due to the topic's inherent complexity. On the one hand, evaluation protocols frequently use limited sets of scenarios, usually concentrating on either dense crowds or sparse environments, rarely covering both, and ignoring mixed or intermediate scenarios. State-of-the-art approaches are still struggling to present solutions that are robust in a wide range of environments, socially acceptable, efficient in navigation, and real-time.

Three complementary contributions are made in this thesis to address these issues. First, we present Robot Social Navigation Assessment Platform (RobotSNAP), a benchmarking platform that allows for systematic comparison of current approaches by offering realistic simulation, a variety of scenarios, and extensive social and performance metrics. Second, we present model-based social navigation techniques: MDP-Based Social Navigation (MBSN), which is a Markov Decision Process (MDP) over a navigation graph, and its computationally efficient variant Heuristic-Based Social Navigation (HBSN), which employs a heuristic to provide real-time socially compliant navigation. Third, to manage humans and the environment, we create a robust Multi-branch Deep Reinforcement-learning for Human-aware Control (MDHC) method. It leverages a novel Curriculum Learning (CL) strategy and a multi-branch architecture integrating a new model of Human-Robot Interaction (HRI) to enable stable learning across a variety of environments and scenarios.

Extensive simulation using RobotSNAP and real-world experiments have confirmed that each of these solutions has distinctive trade-offs: MBSN offers interpretability and strong results but suffers from high computational cost. Meanwhile, MDHC offers real-time execution and adaptability but requires extensive training, is less interpretable, and struggles with robustness. Although it is simple, HBSN strikes the best balance by combining real-time performance, robustness, and interpretability.

Overall, this thesis prepares the way for future research on implementing such solutions in increasingly complex and realistic environments while building the foundation for scalable, interpretable, and reliable human-aware navigation.

Keywords: robotics, artificiel intelligente, social robotics navigation

---

Centre d'Enseignement de Recherche et d'Innovation Systèmes Numériques
764 Bd Lahure, 59500 Douai, France

Navigation Robotique Consciente de l'Humain : Des Benchmarks Complets et Réalistes aux Solutions Polyvalentes et Efficaces

Résumé

Un défi majeur pour la robotique mobile est la navigation robotique consciente de l'humain (HAN) ; nécessitant que les robots naviguent en toute sécurité et efficacement dans des environnements partagés avec des humains tout en respectant les normes sociales et en s'adaptant aux contextes dynamiques. Malgré des recherches approfondies, des défis majeurs subsistent en raison de la complexité inhérente du sujet. D'une part, les protocoles d'évaluation utilisent fréquemment des ensembles limités de scénarios, se concentrant généralement soit sur des foules denses, soit sur des environnements clairsemés, couvrant rarement les deux, et ignorant les scénarios mixtes ou intermédiaires. Les approches de pointe ont encore du mal à présenter des solutions robustes dans une large gamme d'environnements, socialement acceptables, efficaces en navigation et en temps réel.

Trois contributions complémentaires sont apportées dans cette thèse pour aborder ces problèmes. Tout d'abord, nous présentons la Plateforme d'Évaluation de la Navigation Sociale des Robots (RobotSNAP), une plateforme de référence qui permet une comparaison systématique des approches actuelles en offrant une simulation réaliste, une variété de scénarios et des métriques sociales et de performance étendues. Deuxièmement, nous présentons des techniques de navigation sociale basées sur des modèles : Navigation sociale basée sur les MDP (MBSN), qui est un processus de décision de Markov (MDP) sur un graphe de navigation, et sa variante computationnellement efficace Navigation Sociale Basée sur l'Heuristique (HBSN), qui utilise une heuristique pour fournir une navigation conforme aux normes sociales en temps réel. Troisièmement, pour gérer les humains et l'environnement, nous créons une méthode robuste d'apprentissage par renforcement profond multi-branches pour le contrôle conscient des humains (MDHC). Il exploite une nouvelle stratégie d'apprentissage par programme (CL) et une architecture multi-branches intégrant un nouveau modèle d'interaction homme-robot (HRI) pour permettre un apprentissage stable dans une variété d'environnements et de scénarios.

Une simulation extensive utilisant RobotSNAP et des expériences réelles ont confirmé que chacune de ces solutions présente des compromis distincts : MBSN offre une interprétabilité et des résultats solides, mais souffre d'un coût de calcul élevé. Pendant ce temps, MDHC offre une exécution en temps réel et une adaptabilité, mais nécessite une formation approfondie, est moins interprétable et a des difficultés avec la robustesse. Bien qu'il soit simple, HBSN trouve le meilleur équilibre en combinant performance en temps réel, robustesse et interprétabilité. Dans l'ensemble, cette thèse prépare le terrain pour des recherches futures sur la mise en œuvre de telles solutions dans des environnements de plus en plus complexes et réalistes, tout en jetant les bases d'une navigation consciente de l'humain évolutive, interprétable et fiable.

Mots clés : robotique, intelligence artificielle, navigation robotique sociale