THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

Future-Oriented Navigation for Autonomous Mobile Robots

Integration of Multimodel Motion Prediction and Model Predictive Control

Ze Zhang

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2025

Future-Oriented Navigation for Autonomous Mobile Robots

Integration of Multimodel Motion Prediction and Model Predictive Control

Ze Zhang

ISBN 978-91-8103-193-5

Acknowledgements, dedications, and similar personal statements in this thesis, reflect the author's own views.

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5651
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Printed by Chalmers Digital Printing
Gothenburg, Sweden, April 2025

Future-Oriented Navigation for Autonomous Mobile Robots

Integration of Multimodel Motion Prediction and Model Predictive Control

Ze Zhang
Department of Electrical Engineering
Chalmers University of Technology

Abstract

As industrial environments become more dynamic and collaborative, Autonomous Mobile Robots (AMRs) are being deployed to work alongside humans. While hybrid human-robot settings offer improved efficiency and adaptability, they also challenge safety due to human behavior's inherent unpredictability. This thesis is motivated by the goal of enabling AMRs to anticipate and react to dynamic environments by integrating learning-based prediction with optimization-based control. Specifically, it focuses on safer and more efficient future-oriented navigation, i.e., decision-making that incorporates predictive information about the near future.

The core problem addressed in the thesis is: How can AMRs safely and efficiently navigate human-populated, dynamic environments by integrating perception, motion prediction, and optimization-based control? Specifically, this involves: (1) developing motion prediction methods capturing the uncertainty of human behavior; (2) designing control strategies incorporating predictive information; (3) ensuring the real-time performance of the system.

The main contributions of this thesis are threefold: (i) a "Factory with Vision" framework integrating perception, prediction, planning, and control; (ii) enhanced multimodal prediction techniques for downstream motion planning and control, and (iii) integration of motion prediction with both model predictive control and on-manifold control barrier functions. The proposed methods were evaluated in simulated scenarios with static and dynamic obstacles, including multi-agent environments. Performance was assessed in terms of safety, efficiency, and adaptability. Results show that the framework outperforms previously proposed approaches, offering more accurate motion prediction, safer navigation, and better handling of complex, dynamic environments.

Keywords: Motion prediction, deep machine learning, mobile robot, predictive control, obstacle avoidance, navigation, automation.

To those who love and care

List of Publications

This thesis is based on the following publications and manuscripts:

[A] **Z. Zhang**, E. Dean, Y. Karayiannidis, and K. Åkesson, "Motion Prediction Based on Multiple Futures for Dynamic Obstacle Avoidance of Mobile Robots". *IEEE International Conference on Automation Science and Engineering (CASE)*, Lyon, France, 2021, pp. 475-481.

[B] **Z. Zhang**, H. Hajieghrary, E. Dean, and K. Åkesson, "Prescient Collision-Free Navigation of Mobile Robots with Iterative Multimodal Motion Prediction of Dynamic Obstacles". *IEEE Robotics and Automation Letters (RA-L)*, vol. 8, no. 9, 2023, pp. 5488-5495.

[C] K. Ceder*, **Z. Zhang***, A. Burman, I. Kuangaliyev, K. Mattsson, G. Nyman, A. Petersén, L. Wisell, and K. Åkesson, "Bird's-Eye-View Trajectory Planning of Multiple Robots using Continuous Deep Reinforcement Learning and Model Predictive Control". *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Abu Dhabi, UAE, 2024, pp. 8002-8008, *denotes equal contribution.

[D] **Z. Zhang**, G. Hess, J. Hu, E. Dean, L. Svensson, and K. Åkesson, "Future-Oriented Navigation: Dynamic Obstacle Avoidance with One-Shot Energy-Based Multimodal Motion Prediction". *Under Review.*

[E] Y. Xue*, **Z. Zhang***, K. Åkesson, and N. Figueroa, "Proactive Local-Minimum-Free Mobile Robot Navigation using MMP-MCBF Framework". *To be submitted for possible publication*, *denotes equal contribution.

Other publications by the author, not included in this thesis, are:

[F] J. Berlin, G. Hess, A. Karlsson, W. Ljungbergh, **Z. Zhang**, P. -L. Götvall, and K. Åkesson, "Trajectory Generation for Mobile Robots in a Dynamic Environment using Nonlinear Model Predictive Control". *IEEE International Conference on Automation Science and Engineering (CASE)*, Lyon, France, 2021, pp. 942-947.

[G] F. Bertilsson, M. Gordon, J. Hansson, D. Möller, D. Söderberg, **Z. Zhang**, and K. Åkesson, "Centralized versus Distributed Nonlinear Model Predictive

Control for Online Robot Fleet Trajectory Planning". *IEEE International Conference on Automation Science and Engineering (CASE)*, Mexico City, Mexico, 2022, pp. 701-706.

[H] **Z. Zhang**, E. Dean, Y. Karayiannidis, and K. Åkesson, "Multimodal Motion Prediction Based on Adaptive and Swarm Sampling Loss Functions for Reactive Mobile Robots". *IEEE International Conference on Automation Science and Engineering (CASE)*, Mexico City, Mexico, 2022, pp. 1110-1115.

[I] **Z. Zhang**, Y. Cai, K. Ceder, A. Enliden, O. Eriksson, S. Kylander, R. Sridhara, and K. Åkesson, "Collision-Free Trajectory Planning of Mobile Robots by Integrating Deep Reinforcement Learning and Model Predictive Control". *IEEE International Conference on Automation Science and Engineering (CASE)*, Auckland, New Zealand, 2023, pp. 1-7.

[J] **Z. Zhang**, Yifan Xue, Nadia Figueroa, and K. Åkesson, "Gradient Field-Based Dynamic Window Approach for Collision Avoidance in Complex Environments ". *Submitted to a conference*, 2025. [Online available] arXiv preprint arXiv:2504.03260.

Contents

Acknowledgments

The work of this report is initiated by the ViMCoR project, followed by the AIMCoR project, which is mainly supported by Volvo GTO, financially supported by CHAIR (Chalmers AI Research Centre). The continuity of this study is supported by the Vinnova project, AIHURO (Intelligent HUman-RObot collaboration). In the AIHURO project, AB Volvo and AstraZeneca offer industrial practices, while Chalmers, University of Skövde, and Chalmers Industriteknik develop algorithms, tools, and solutions for the research challenges. For the training of neural networks, the computations were enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Throughout this long and challenging journey, I have been fortunate to receive the support and encouragement of many individuals—both as friends and mentors. Without their warmth and guidance, this period would have felt much colder and more difficult. As I now express my heartfelt gratitude, filled with both excitement and emotion, I will do my utmost to acknowledge each and every one of them.

To start with, I would like to express my sincere gratitude to my supervisor, Prof. Knut Åkesson, for his guidance and support. I am also thankful to my former co-supervisors, Yiannis Karayiannidis and Torsten Sattler, for their support in the early phase of my study. I deeply appreciate the insightful and patient discussions with Emmanuel Dean and Lennart Svensson, whose help has been instrumental. I would also like to thank Hadi Hajieghrary for his collaboration and encouragement, which significantly positively influenced my academic direction. My thanks extend to all my co-authors for their collaborative efforts, and to Erik Agrell for his kind mental support during the final phase of my graduation. I am also grateful to Karinne Ramirez-Amaro for her encouragement and assistance throughout. Special thanks to Nadia Figueroa for hosting me during my visit to the University of Pennsylvania, and to my collaborator Yifan, as well as the other friendly and supportive colleagues in Nadia's lab. In addition, I want to express my appreciation to Per-Lage Götvall from Volvo GTO for his generous support of the project.

I want to thank all my colleagues for creating a friendly working environment, especially the ones from the Automation group: Sabino, Constantin, Endre, Martin, Julius, Mattias, Johan, Zahra, Ashfaq, Alvin, Nishant, Kris-

tian, Erik, Rikard, Lasse, Martina, Kristofer, and other newcomers. Also, the ones we work together, grab fika, or meet in the corridor from time to time: Carl-Johan, Ying, Maximilian, Rita, Francesco, Benedick, Ahmet, and Daniel. I enjoy all the fika time, climbing time, and afterworks. I hope everything goes well for you and always be happy.

I am deeply grateful for the support and joyful moments shared with my dear friends—Yicong, Naichen, Xiao, Mengqiao, Xixi, Huang, Yao, Fangting, Yibo, Yidong, Wenhao, Pinky, and many others—whether it was through playing sports, grabbing drinks, or simply spending time together. Finally, I extend my heartfelt thanks to my parents, whose love and care have always helped me mentally recharge during my visits home, and to my dear partner, Shuhan, whose unwavering belief in me and constant support have been an invaluable source of strength throughout this journey.

Acronyms

AGV:	Automated Guided Vehicle
AI:	Artificial Intelligence
AMR:	Autonomous Mobile Robot
BCE:	Binary Cross-Entropy
CBF:	Control Barrier Function
CGF:	Clustering and Gaussian Fitting
CNN:	Convolutional Neural Network
CVM:	Constant Velocity Model
DDPG:	Deep Deterministic Policy Gradient
DOA:	Dynamic Obstacle Avoidance
DQN:	Deep Q-Network
DRL:	Deep Reinforcement Learning
DSM:	Dynamical System Modulation
DWA:	Dynamic Window Approach
FwV:	Factory with Vision
GMM:	Gaussian Mixture Model
GPDF:	Gaussian Process Distance Field
KLD:	Kullback-Leibler Divergence
LQR:	Linear Quadratic Regulator
MDP:	Markov Decision Process
MPC:	Model Predictive Control

OGM:	Occupancy Grid Map
PANOC:	Proximal Averaged Newton-type method for Optimal Control
QP:	Quadratic Programming
RL:	Reinforcement Learning
ROS:	Robot Operating System
RRT:	Rapidly-exploring Random Tree
SFM:	Social Force Model
WTA:	Winner-Takes-All (loss)

Part I

Overview

CHAPTER 1

Introduction

The increasing demand for automation in various industries has led to the development of advanced indoor logistics systems that can efficiently handle the movement of goods and tools. In such systems, mobile robots play a crucial role in enhancing productivity and reducing operational costs. However, the integration of robots into these systems raises safety concerns, particularly when humans and robots work in close proximity. To address these challenges, it is essential to develop solutions that not only enhance the efficiency of logistics operations but also ensure the safety of human workers.

Autonomous Mobile Robots (AMRs) [1] have emerged as a promising solution for indoor logistics, as they are equipped with advanced perception capabilities that allow them to navigate complex and dynamic environments in real time. An AMR is a mobile robotic platform designed to navigate and perform tasks in dynamic, complex environments using a combination of local sensing and lightweight onboard control. While capable of basic functions such as obstacle avoidance and local motion execution, AMRs in this architecture rely heavily on external systems—particularly cloud-based services—for global planning, perception integration, and coordination. When operating as part of a fleet, AMRs receive task assignments and optimized navigation com-

mands from the cloud, enabling coordinated and adaptive behavior across the system. This technology is also known as cloud robotics [2]. Cloud robotics is a distributed robotic architecture in which the majority of computational intelligence—such as task allocation, path planning, semantic mapping, and multi-robot coordination—is offloaded from individual robots to centralized or distributed cloud infrastructure. This approach allows robots to operate with minimal onboard processing, drawing on shared resources such as global maps, live sensor fusion (e.g., from fixed ceiling cameras), and fleet-level control logic. Cloud robotics enables scalable, cost-efficient, and highly adaptive robot deployments, particularly in environments where collaborative or coordinated behavior is required. The primary task of AMRs can be concluded as follows: With the premise of no collisions, travel from the start configuration to the goal configuration, and at the same time satisfy certain constraints (such as reference speed and route) as well as certain objectives (such as low energy consumption or short travel time). To achieve this, AMRs must be able to perceive their surroundings, reason about potential interactions and conflicts, plan their paths, and execute their movements effectively.

For humans, it is ubiquitous and routine to evade another pedestrian or vehicle during locomotion. Although the process of decision-making and execution is immediate and voluntary, it involves several steps [3]. First, individuals perceive the approaching object and recognize it as a potential obstacle. Next, they estimate the object's movement to assess possible obstruction. Finally, they modify their actions to avert a potential collision if their paths are predicted to intersect. Without an accurate estimation of the motion of other objects, dynamic obstacle avoidance can become unresponsive and hazardous. This natural circuit in human beings inspires the implementation of motion prediction abilities in AMRs.

Most existing studies focus on either motion prediction or motion planning, but few have integrated both aspects into a cohesive framework. Meanwhile, dynamic obstacle avoidance remains a significant challenge due to the limitation of reactive control strategies and the simple motion models of dynamic obstacles. This research aims to bridge this gap by proposing a novel approach that combines learning-based motion prediction with optimization-based control for motion planning. By leveraging the strengths of both techniques, the proposed approach seeks to outperform traditional methods in dynamic obstacle avoidance.

This thesis discusses the challenge of dynamic obstacle avoidance for AMRs operating in regulated environments, such as industrial indoor environments, including warehouses and factories. A solution that integrates Model Predictive Control (MPC) with deep learning-based multimodal motion prediction is introduced. Multimodal motion prediction involves forecasting multiple plausible future positions of a target object, allowing AMRs to make informed decisions about their own movements in response to dynamic obstacles. The proposed approach aims to improve the safety and efficiency of AMRs in hybrid environments where they coexist with human workers.

In the following sections, the background, industrial context, objectives, and proposed solutions will be briefly outlined to provide a general understanding of the research. More detailed discussions on perception, planning, control, and implementation will follow in subsequent chapters.

1.1 General Background

The rapid advancement of automation technologies has revolutionized industries globally, leading to unprecedented levels of productivity and efficiency, where automation refers to the use of control systems and information technologies to reduce the need for human intervention in the production of goods and services [4]. As industries increasingly adopt new technologies such as robotics, Artificial Intelligence (AI), and the Internet of Things, logistics systems are significantly affected and challenged. Efficient logistics handling becomes essential to accommodate the increased production capabilities, necessitating the development of more efficient and responsive transportation systems.

Automatic robotic systems are increasingly integrated into logistics operations to streamline processes, reduce costs, and improve overall efficiency. This leads to inevitable human-robot interactions, which raise safety concerns. Ensuring the safety of human workers in environments where robots operate is paramount, as accidents can lead to severe injuries or fatalities. Consequently, the balance between maximizing productivity through automation and maintaining a safe working environment requires intelligent and reactive robots. In the field of indoor logistics, AMRs have emerged as a solution to address these challenges, providing a means to enhance flexibility and efficiency in logistics operations while prioritizing safety.

Figure 1.1: The Fetch mobile manipulator [5] (left) and the MiR100 mobile platform [6] (right).

1.2 Industrial Background

Historically, manufacturing processes have been characterized by high-volume, low-diversity production, where the focus was on maximizing output with minimal variation. However, the shift towards more flexible and customized manufacturing has transformed the landscape, as consumers increasingly demand personalized products and services. This evolution necessitates the adoption of advanced technologies that can accommodate varying production requirements and respond to changing market demands. As a key division in Volvo Group, Volvo Group Trucks Operations has recognized this trend and is actively working on finding solutions to achieve the shift.

In this context, the development of AMRs is particularly noteworthy due to their flexibility. The popular AMR architecture involves a movable platform equipped with various sensors so that it can navigate autonomously within a facility. Although this kind of AMRs can operate independently, the high autonomy level often comes with a high cost and complexity and cannot be utilized for applications other than the specific task they are designed for. To address these limitations, the *Factory with Vision* (FwV) solution is proposed as a more affordable and expandable alternative. This innovative approach allows for the integration of low-cost, modular AMRs with external vision systems, enabling them to adapt to various tasks and environments while maintaining a focus on safety and efficiency. In particular, as conceptualized in Figure. 1.2, the transportation part of the FwV solution relies on a vision system containing ceiling-mounted cameras that monitor the environment and

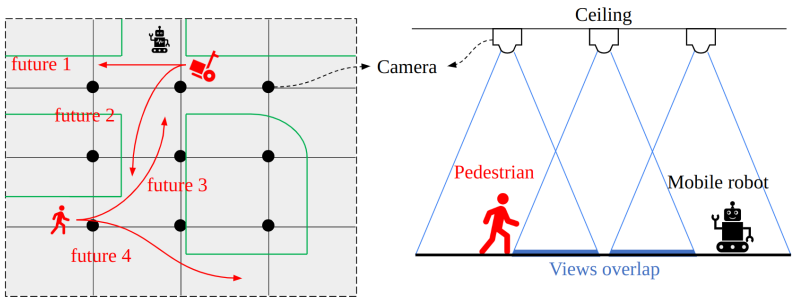

Figure 1.2: The concept of the Factory with Vision solution [7].

a fleet of AMRs equipped with safety sensors, where the vision system provides high coverage of the workspace, detecting obstacles and potential hazards in real-time, and guiding the AMRs accordingly. This combination of technologies not only enhances operational efficiency but also ensures a safer working environment for human workers.

1.3 Requirements and Objectives

As a summary of the project's background, the following requirements of the transportation system under the FwV framework are identified:

1. From safety to smoothness: Safety should always be the premise, which means that AMRs must stay a safe distance from humans and not cause any accidents that could lead to injuries. What's more, on top of safety, AMRs should also behave smoothly in their operations, ensuring that their movements are not abrupt and intrusive to humans.

2. From effectiveness to efficiency: Effectiveness refers to the ability of AMRs to complete their tasks successfully, while efficiency emphasizes the optimal use of resources, including time and energy. The transportation system should prioritize efficiency without compromising effectiveness, ensuring that AMRs can perform their tasks in a timely and resource-efficient manner.

3. From economy to expandability: The FwV solution is designed to be an economical alternative to traditional high-cost AMRs. Meanwhile,

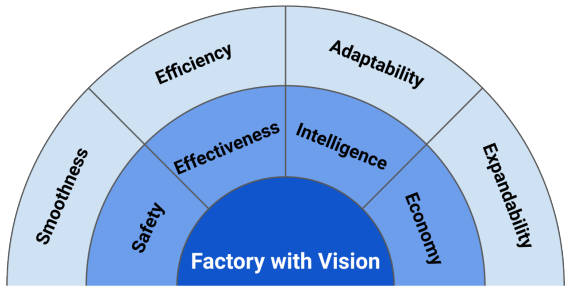

Figure 1.3: The requirements of the FwV solution.

the ceiling-mounted vision system allows for the easy addition of more functions, such as anomaly detection and guidance for robotic manipulators, making the system expandable to meet future needs. Although this is out of the scope of this thesis, possible future expansions will be brought up and briefly discussed.

4. From intelligence to adaptability: Despite various representations of intelligence, the focus here is on the adaptability and flexibility of AMRs in dynamic environments. The transportation system should enable AMRs to adapt to changing conditions, such as variations in the workspace layout or the presence of unexpected obstacles, ensuring that they can continue to operate effectively in diverse situations.

These requirements are reflected in the publications related to this project and will be discussed later.

Based on these requirements, considering the research field and the limited time and resources available for this project, the primary focus is to develop a pipeline from perception to action for AMRs in the FwV framework. Specifically, the objectives include:

1. Investigating different perception methods including the ceiling-mounted vision system.

2. Developing algorithms for handling the perception data and generating informative predictions of the environment for AMRs.

3. Implementing control strategies that allow AMRs to navigate safely and efficiently in dynamic environments.

Apart from these objectives, during the project, another two topics are touched upon: the combination of high-level scheduling with control and learning-based reference trajectory generation. These topics are not the main focus of this thesis but are relevant to the overall development of the transportation system.

1.4 Solution Overview

In this section, An overview of the proposed pipeline is provided, detailing the key components involved in the transportation system for AMRs within the FwV framework. The pipeline encompasses four main aspects: perception, reasoning, motion planning, and scheduling.

Perception

Perception is the process through which AMRs gather and interpret information about their environment. As humans rely on certain receptors to perceive their surroundings, AMRs utilize various sensors to achieve a similar understanding [1], [8]. Some initial and well-developed perception sensors include inductive sensors, optical sensors, and so on. These sensors cannot work alone but require corresponding infrastructure such as floor-embedded wire or markers to guide the direction. In contrast, more recent sensors can work independently, among which vision-based (e.g., RGB cameras) and range-based (e.g., LiDAR) sensors are the most prominent. There are also other auxiliary sensors that have limited detection ranges but can provide additional information and safety, such as ultrasonic and tactile sensors.

Infrastructure-dependent sensors are widely used in traditional industrial transportation robots, known as Automated Guided Vehicles (AGVs) [9], [10]. Strictly speaking, these sensors are not capable of perceiving the general surroundings including obstacles, except for the pre-laid guiding paths. Even though these sensors can provide reliable navigation, they cannot perceive the complete surroundings, thus lack the flexibility to adapt to dynamic environments. Newly developed AMRs leverage more advanced sensors with comprehensive perceptual capabilities, allowing them to deviate from fixed paths and navigate autonomously in dynamic environments. Light Detection and Ranging (LiDAR) sensors, for example, create detailed 3D maps

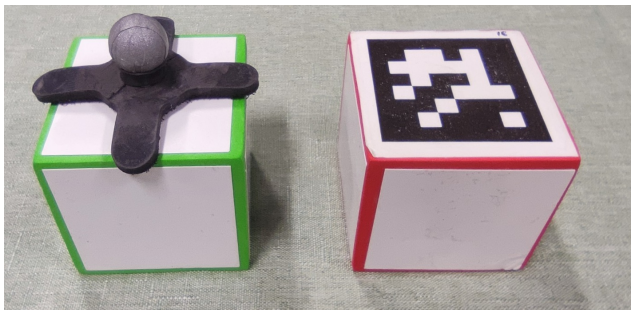

Figure 1.4: The examples of an optical marker (left) and a visual marker (right).

of the environment according to the distances to surrounding objects, which are widely used in autonomous vehicles because of their high work frequency and accuracy. While LiDAR sensors provide a range representation of space, cameras usually capture rich visual information that is in line with human instinct. Thanks to the development of computer vision and deep learning, visual information can be processed to identify, classify, and track objects in real time.

In this research, the main focus is on processing data from the most accessible and cost-effective visual sensor - RGB cameras. As shown in Figure. 1.2, AMRs utilize environmental information captured by ceiling-mounted RGB cameras to perceive the workspace and detect obstacles. The top-down view provides an omniscient perspective with less occlusion compared to cameras mounted on the robots themselves [7], [11], [12]. In addition to AMR guidance, the vision system can be used to monitor the overall workspace, providing valuable insights into the operational status and potential hazards. It can also be integrated with other automation systems, such as robotic manipulators. All these extensions are not available with sensors mounted on the robots.

After obtaining sensory data, it needs to be further digested and interpreted to facilitate informed decision-making. In the context of AMRs, the aim is to enable AMRs to determine their locations, identify surrounding obstacles, and track dynamic objects (humans or human-operated vehicles) in their workspace. Localization can be achieved through various techniques, such as GPS, SLAM (Simultaneous Localization and Mapping) [13], odome-

try, and visual or optical markers (as shown in Figure. 1.4). Different methods require different types of sensors and may have varying levels of accuracy and reliability. To benefit from the advantages of multiple sensors, sensor fusion is often employed to combine data from different sources, enhancing the overall localization accuracy. The perception of obstacles has different levels of complexity, ranging from detection to classification and tracking. Obstacle detection involves identifying the presence of an object, which is straightforward for range-based sensors like LiDAR or depth cameras. RGB cameras can provide rich visual information, allowing for more sophisticated classification but also requiring more complex computer vision algorithms for detection. For the purpose of navigation, the perception process should not only distinguish drivable areas from obstacles but also identify dynamic obstacles since they usually pose a higher risk of collision. When focusing on visual information from RGB cameras, semantic segmentation [14] is a powerful technique for drivable area detection, which classifies each pixel in an image to determine whether it belongs to a drivable area or an obstacle. Then, object detection and tracking algorithms can be applied to identify dynamic obstacles and record their trajectories for future inference.

Reasoning

Apart from the perception of the instantaneous environment, inferring human intentions and predicting the future states of dynamic obstacles are crucial in human-robot interactions [15]. Without accurate estimations of the motions of other objects, dynamic obstacle avoidance can be unresponsive and passive. Passive or reactive collision avoidance is generally sufficient for low-speed scenarios; for example, most social service robots are lightweight, operate at low speeds, and can easily stop or change direction to avoid close encounters with humans. However, in industrial environments, AMRs often operate at higher speeds and weights, especially when transporting heavy loads. In these cases, a proactive approach is necessary to ensure safety and efficiency and the motion prediction ability is an important premise for proactive decision-making. Traditional motion prediction methods often rely on simple motion models, which may not accurately capture the behavior of dynamic obstacles and their interactions with other objects in the environment. To address this limitation, this research utilizes learning-based motion prediction techniques that can forecast multiple plausible future positions of a target object. By

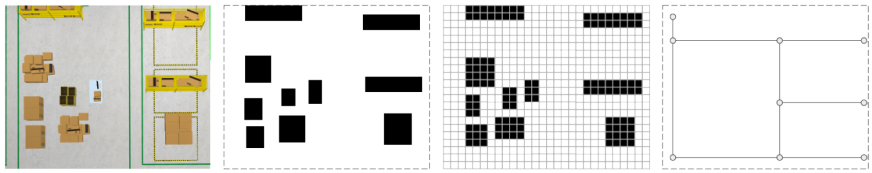

Figure 1.5: Different representations of a warehouse map, including (from left to right) the real map, the simplified geometric map, the occupancy grid map, and the topological map. The map is a part of a Gazebo world.

leveraging the historical data and image processing ability of Convolutional Neural Networks (CNNs), the inherent uncertainty in human behavior can be better captured, allowing AMRs to make informed decisions about their own movements in response to dynamic obstacles.

Planning and control

After reasoning about the perceived data, a 2D or 3D map of the environment can be constructed. Assuming the floor of the workspace is planar, this research focuses on 2D representations of the environment. There are usually two kinds of maps: Occupancy Grid Maps (OGMs) and geometric maps. OGMs are discrete representations of the environment, where each cell indicates whether it is occupied or free. Geometric maps, on the other hand, provide a continuous representation of the environment, recording exact distances, positions, and physical dimensions of elements. Besides the environmental representation, topological maps can represent the connectivity of the accessible areas for AMRs by constructing mathematical graphs, which are mostly used for high-level scheduling and routing, while the OGM and geometric maps contain more detailed and real-time information for low-level planning.

While different literature uses various terminologies, the planning process [1], [16], [17] is decomposed and concluded as follows (formal definitions are provided in later chapters):

1. Motion planning: As the umbrella term for other planning sub-processes, motion planning refers to the overall process of determining a feasible motion for an AMR from its current configuration to a desired goal configuration. In cases where the orientation of the robot is not important,

the configuration can be represented as a 2D point in the workspace.

2. Path planning: Path planning is a sub-process and normally the first step in motion planning, which focuses on finding a collision-free geometric path from the start to the goal point based on the environmental layout. The path can be represented as a series of waypoints through which the AMR should proceed.

3. Trajectory planning: Conceptually, trajectory planning is the next step after path planning, where the focus shifts from the geometric path to the timing of the motion and the kinematic constraints of the AMR. A trajectory is a time-parameterized sequence of states that the AMR should follow to execute the planned path. Unlike the reference path that is usually predefined, the reference trajectory, especially for the local reference trajectory, can be generated online based on the current state of the AMR and the surrounding environment.

4. Trajectory tracking: Strictly speaking, trajectory tracking is not a planning but a control task that ensures the AMR follows the generated trajectory accurately. It is listed here to compare different definitions and to provide a complete picture of the planning and control process. Even though some literature uses the term trajectory planning to refer to both trajectory planning and tracking, this research distinguishes them by emphasizing the difference between planning and control.

Since the refinement of the reference path, which is the global trajectory generation, is not the focus of this study, the term *reference trajectory* or *reference states* in the following text refers specifically to the local reference trajectory. When discussing the global reference trajectory, the modifier "global" will be explicitly included for clarity.

To achieve a navigation task, AMRs need to go through the motion planning and tracking process iteratively, but not necessarily in a sequential manner. For instance, trajectory planning and tracking can be completed together using optimization-based reference path tracking methods [18], while some learning-based controllers generate actions directly from the perception data without explicit reasoning, planning, and tracking [19]. The choice of planning and control methods depends on the specific requirements of the task, such as the complexity of the environment, the speed of the AMR, and the availability of computational resources.

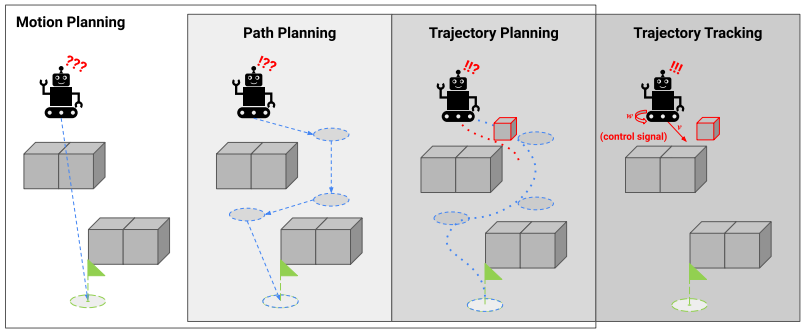

Figure 1.6: Motion planning and trajectory tracking in the mobile robot navigation problems. The robot should reach the goal from the current location. In path planning, the task is to find a feasible path (or path nodes defining a path), as the blue dashed line and circles show. In trajectory planning, both global trajectory and local trajectory are relevant. A global trajectory (blue dots) is a refined temporal state sequence, while a local trajectory (red dots) focuses on considering the kinodynamic constraints at the current time. Finally, the trajectory tracking problem is actually a control problem for generating control signals, such as velocity and angular velocity.

Path planning approaches [16], [20] can be categorized into map-based and graph-based methods. As mentioned earlier, map-based methods plan paths based on the concrete representation of the environment, in the form of OGMs or geometric maps. These methods focus on finding collision-free paths given the appearance of obstacles in the environment. On the other hand, graph-based methods require a pre-built topological map representing the connectivity of the environment, which already considers the static layout of the workspace. The graph-based methods are more suitable for high-level scheduling and routing, while the map-based methods are more suitable for low-level planning. In this research, a similar path planning sequence is adopted, where the global path is planned based on the topological map, and the local path is planned based on the concrete map representation.

A reference path consists of a series of sparse waypoints within the workspace, which an AMR can follow by adjusting its heading toward the next waypoint until reaching the final goal. However, this approach often results in jerky and inefficient movements, particularly when sharp turns or obstacle avoid-

ance are required. A reference trajectory (global or local) [17] represents a more refined version of the reference path, incorporating temporal information that accounts for the kinematic constraints of the AMR. Unlike reference paths, reference trajectories are more amenable to real-time updates, allowing for better adaptation to the AMR's current state and environmental changes.

After generating the motion plan, the AMR needs to execute the planned trajectory while considering the dynamic constraints as well as the uncertainties in the environment. Given the reference trajectory, a high-level control problem focuses on generating proper kinematic commands, such as velocity, angular velocity, and acceleration. A low-level control problem, on the other hand, focuses on generating proper dynamic commands, such as torque and force, to ensure the kinematic commands are executed accurately. Since the mapping from dynamic commands to kinematic commands depends on the specific AMR model and the floor conditions, it is fine-tuned through calibration and testing with simple and efficient control algorithms, such as PID control. In this research, the focus is on the high-level control problem, where the reference trajectory is tracked by adjusting the velocity and angular velocity of the AMR.

Scheduling and more

Task allocation and scheduling are essential components of an indoor logistics system, ensuring that resources are allocated efficiently and tasks are completed on time. A scheduler is responsible for assigning tasks to available AMRs based on their current status and the task requirements, at the same time optimizing the overall system performance in terms of certain objectives, such as minimizing the total travel time or energy consumption. A schedule is a time plan for AMRs including the start time, end time, and the path to follow for each task. The scheduling process is closely related to the planning and control process, as the planned paths and trajectories need to be coordinated with the task schedule to ensure that AMRs can complete their tasks without conflicts or collisions. In this research, the scheduling aspect is briefly discussed, referencing Sabino F. Roselli's work [21] on task allocation and scheduling for AMRs in the FwV framework.

Mobile robot fleet coordination [22] is another important aspect of the transportation system, when multiple AMRs need to work in the same workspace. The coordination can be centralized, decentralized, or distributed, each with

Feature	Centralized	Decentralized	Distributed
Strategy	Single controller for the entire system	Independent controllers for each AMR	Individual controllers with shared information
Communication	None (one controller)	None (independent)	Yes (locally/globally)
Performance	Global optimal	Suboptimal (local optima)	Suboptimal (near-global optimal)
Complexity	Highest	Lowest per controller	Intermediate per controller
Scalability	Poor	High	Intermediate and flexible
Interactions	Full interactions among AMRs	No interaction	Partial and (maybe) local interactions
Applications	Small to middle-size systems	Systems with relatively independent AMRs	Large systems with communication modules

Table 1.1: Comparison among different coordination and control strategies.

its own advantages and challenges [23]. Centralized coordination regards the planning and control of all AMRs as a single entity and requires a central computer to manage the entire fleet. Decentralized coordination treats each AMR as an independent agent that makes decisions based on local information and different agents do not communicate with each other. Distributed coordination is a compromise between centralized and decentralized coordination, where every AMR has its own planner and controller but can communicate with other AMRs to share information and coordinate their actions. In this research, a distributed implementation is adopted. A comparison with different coordination strategies is provided in Table 1.1.

Even though many aspects of the transportation system are discussed in this section, there are still more technical details and research topics uncovered, such as the design of the mechanical structure of AMRs, the implementation and calibration of sensors, the construction of the communication system and the latency handling, etc. These topics are not the focus of this thesis and are left for future research and development.

1.5 Related Work

In this section, a brief overview of the related work is provided, focusing on the research areas of motion prediction, control, and dynamic obstacle avoidance. The literature review aims to identify the current state-of-the-art methods and techniques used in these areas, highlighting the gaps and challenges that need to be addressed. By analyzing the existing research, this thesis aims to build upon the strengths of previous work and propose novel solutions that can outperform traditional methods in dynamic obstacle avoidance.

Motion prediction

Motion is a fundamental aspect of human behavior, and predicting human motion is essential for safe and efficient human-robot interactions. In the context of this work, as defined in [24], motion prediction refers to forecasting the future positions of dynamic obstacles, such as humans or human-operated vehicles. Traditional motion prediction methods rely on simplified motion models. The Constant Velocity Model (CVM) is one of the most common assumptions [25]–[27], where the object is assumed to move at a constant velocity. Despite its simplicity, it is favored for its computational efficiency and ease of implementation, and it is proven to be surprisingly effective [27]. Considering the predominance of straight-line movements in human trajectories, the CVM can provide a reasonable approximation of human motion in terms of overall accuracy. However, the limitations of the CVM become apparent when humans perform complex maneuvers, such as turning or deviating from their original paths, where the risk of collisions increases. In general, assuming all agents are rational—meaning they do not move randomly or collide intentionally—an agent's motion is influenced by both internal and external stimuli. The internal stimuli include the agent's destination, social etiquette [28], and physical constraints, while the external stimuli encompass the environment's layout and the presence of other agents. Numerous studies have explored pedestrian and crowd behavior by stressing the interactions among agents and between agents and the environment via mathematical models. One prominent model is the Social Force Model (SFM) [29], which describes the interactions among agents as repulsive forces, and the final motion is determined by the equilibrium of these forces. Velocity obstacle-based methods [30]–[32] offer another approach, considering the relative velocity of agents to

select an optimal, collision-free motion. While these interactive models are effective for crowd simulation and coordination, they are less suited for dynamic obstacle avoidance applications, as they typically do not incorporate environmental information or account for uncertainty in their predictions.

Advancements in deep learning have enabled the use of neural networks to model environmental contexts and agent interactions with greater precision. Neural networks also provide the ability to account for uncertainty, making them well-suited for dynamic environments. Social pooling [33], [34] is a technique that captures interactions among agents by aggregating their features to infer collective behavior, through which the network can learn motion patterns of agents with the consideration of their interactions. CNNs are utilized to extract spatial features from the environment [35]–[39], thereby enhancing the network's ability to perceive context. This is crucial for neural networks to make adaptive predictions when the environment is dynamic and the layout may change over time. Additionally, advanced methods like attention mechanisms [40]–[42] and inverse reinforcement learning [43], [44] further improve motion prediction by capturing intricate dependencies between agents and their environments. These approaches contribute to more accurate and adaptive predictions in dynamic, multi-agent settings. Since uncertainty is inherent in human behavior, it is also a challenge for deep learning models to include uncertainty estimation in their predictions. Mixture density models [45] can estimate the distribution of future motion in the form of a mixture of Gaussian distributions, known as the Gaussian Mixture Model (GMM). This method provides a parameterized probability directly from the neural network but is prone to mode collapse in high-dimensional spaces [35], [38], [46]. Variational inference [47] is another approach that models the distribution of future motion as a latent variable. By sampling from the latent variable, variational models can produce diverse trajectory outcomes [40], [48]. Apart from obtaining the parameterized distribution, there are other methods to estimate the uncertainty in motion prediction, such as sample generation and discrete probability map estimation. Multiple choice learning [46] is a technique that generates multiple plausible options for target outputs, allowing the model to capture different possible outcomes. The learning process is enabled by the use of a Winner-Takes-All (WTA) loss function, which updates the most likely output sample based on the ground truth. Some variations of the WTA loss, such as Evolving WTA [35] and Swarm WTA [38], have been

proposed to improve the stability of the model for motion prediction tasks. Discrete probability map estimation [36], [43] is an unparameterized method that estimates the probability distribution of predictions. The basic idea is to regard the ground truth as a discrete probability map on the image axis of the map, and the model is trained to match the probability map directly. This method is computationally efficient and can be easily implemented with existing neural network architectures, such as U-Net [36], [49]. For short-term prediction, contextual cues in the vicinity are usually sufficient to obtain accurate and reasonable results, while for long-term prediction, a common choice is to include the estimate of the possible intended goals of the agent as an intermediate condition [36], [41], [50]–[52]. With the development of generative deep learning, except for the variational autoencoder [40] and generative adversarial networks [34], more generative models are used for motion prediction tasks, such as flow-based generative models [53] and diffusion models [54].

By reviewing previous research on motion prediction, there are several achievements and challenges that need to be addressed [24], [55]. The aid of deep learning has significantly improved the accuracy and robustness of motion prediction models, and the integration of uncertainty analysis, especially multimodality, has enhanced the reliability of predictions. Most recent studies have focused on the following challenges:

- Enhancing the use of environmental context to make the model more adaptable to different scenarios.
- Capturing the interactions among agents and considering different behaviors in multi-agent settings.
- Integrating motion prediction with downstream planning and control algorithms in human-robot interaction applications.

Control of mobile robots

For mobile robots, the goal of the control task is to generate proper commands to drive the robot from its current state to the desired state while considering the constraints and objectives of the task. For low-level control, the control command can be the speed of wheels or the torque of motors, while for high-level control, which is the focus of this research, the control command is usually the kinematic action, such as velocity, angular velocity, and acceleration. The constraints of mobile robots can be kinematic constraints on velocity

and acceleration and environmental constraints on collision avoidance. The basic and necessary objectives of the control task include the safety of the navigation and the reachability of the desired goal, while more objectives can coexist depending on the specific task and application, such as minimal energy consumption [56], [57], short travel time [58], minimal deviation from the reference path, etc.

A simple yet practical control strategy is the classic wait-and-go, where the robot waits if the reference path is obstructed and continues when the path is clear. This is widely deployed in AGVs [9]. Pure pursuit [59] is another simple control algorithm that guides the robot to follow the reference path by adjusting its heading toward the next waypoint. While these algorithms are straightforward and fast, they do not consider constraints and objectives by design. One way to take constraints and objectives into account is to generate several candidates of control commands and evaluate them based on certain criteria. Dynamic Window Approach (DWA) [60], [61] is a sampling-based planning and control algorithm that generates a set of feasible control commands within a dynamic window and evaluates them based on the pre-defined constraints and objectives, such as the distance to obstacles and the goal. Essentially, DWA is a proof-by-exhaustion method, whose performance is highly dependent on the computational complexity of a single sample, sampling resolution, and the size of the dynamic window. Due to the limited sampling resolution, the proposed control action by DWA is usually sub-optimal.

On the contrary, optimization-based methods regard the control task as an optimization problem and solve it by finding the optimal control command that minimizes a certain cost function, while satisfying the constraints. Linear Quadratic Regulator (LQR) [62] is a classic optimization-based control algorithm that minimizes the quadratic cost function of the control commands and the state errors. LQR is widely used in linear systems and has a closed-form solution, but it requires a linearized model of the system and cannot handle constraints directly. Model Predictive Control (MPC) [63] is a more general optimization-based control algorithm that solves control tasks by predicting the future states of the system and optimizing control commands over a finite time horizon. MPC operates by minimizing an objective function while adhering to specified constraints. However, as it involves solving a constrained optimization problem at each time step, it can be computationally demanding. Thanks to the development of optimization solvers and the in-

creased computational power, MPC has obtained real-time performance in many applications. For example, PANOC (Proximal Averaged Newton-type method for Optimal Control) [64], a line-search method combining forward-backward iterations with fast Newton-type steps over a specifically designed merit function, is proposed to solve nonlinear optimization problems in real time. PANOC has been applied to solve the MPC problem for AMRs and self-driving vehicles and has shown promising results in terms of effectiveness and computational efficiency [18], [65], [66].

Apart from optimization-based methods, reactive control strategies are also widely used in obstacle avoidance for mobile robots. Reactive control means that the control command is generated based on the instantaneous state of the robot and the environment, without considering the future states. Interaction models, such as the SFM and velocity obstacle, can be used not only for motion prediction but also for control by regarding the robot as one of the agents and generating the control command based on the interaction forces or velocities [67]. Artificial Potential Field (APF) [68], [69] is another algorithm that generates control commands by constructing a potential field influenced by both obstacles and the target goal, which is similar to the principle of the SFM. APF is simple and fast but subject to local minima and oscillation around the obstacles. Dynamical System Modulation (DSM) [70] is a reactive control algorithm that reshapes the dynamical system according to the objectives and constraints so that global convergence can be guaranteed. Control Barrier Function (CBF) [71] is another reactive optimization method to ensure that certain system states remain within a safe set or desired region. It is often used in combination with Quadratic Programming (QP) to compute safe control inputs that avoid unsafe states [72].

Recently, Reinforcement Learning (RL) [73] has been used to solve control tasks by learning the optimal control policy through trial and error. Deep Reinforcement Learning (DRL) [74] combines deep learning with RL to handle high-dimensional state and action spaces so that it can be applied to complex control tasks and workspaces. Although DRL has shown high flexibility and learning ability, it has the risk of instability and requires a large amount of data for training. DRL also lacks interpretability, which makes it unsuitable for safety-critical industrial applications. New techniques have emerged to address these challenges, such as adding safe constraints to RL algorithms [75] or combining traditional model-based methods with RL [19].

Dynamic obstacle avoidance

Static obstacle avoidance has been well-studied and widely applied in mobile robots [17]. Since static obstacles do not move, once the map of the environment is constructed, the path planning and control tasks can be solved deterministically. For the case of unstructured environments, where the map is only locally explored, reactive control strategies can be used to avoid collisions effectively and efficiently [76], [77]. Compared to static obstacle avoidance, Dynamic Obstacle Avoidance (DOA) is more challenging due to the following reasons: (a) The states of dynamic obstacles change over time, which requires real-time adaptation of the control commands to avoid collisions; (b) The motion of dynamic obstacles is uncertain, and the controller must account for this uncertainty to ensure safe and efficient navigation; (c) The rising complexity, due to the appearance of multiple dynamic obstacles and consideration of motion prediction, requires more sophisticated control algorithms and challenges the computational efficiency of the system.

A basic idea to solve DOA is to treat dynamic obstacles as static at every time step and solve the obstacle avoidance problem in real time. This idea is effective in environments characterized by low-speed scenarios, cooperative agents, and simple motion patterns of dynamic obstacles. For proactive strategies, the consideration of the obstacles' future motion is necessary [55]. Some methods relied on model-based [78], [79] or velocity-based [25], [31], [80] predictions to estimate the future states of dynamic obstacles, and then generate control commands avoiding the predicted obstacle positions. More recent work focuses on two aspects [55]: integrating learning-based predictions into control and coupling prediction with control for more adaptive and robust navigation. Coupled prediction and control methods [81]–[84] have drawn increasing attention due to their ability to consider human-robot interactions in human motion prediction, giving a better chance to solve the "freezing robot problem" [85]. The coupled methods rely on cooperative behaviors of dynamic obstacles, which is reasonable for social robot navigation tasks in open spaces. However, in industrial environments, transporting robots often operate at higher speeds and weights, and human workers may not be able to cooperate with the robots due to obstruction or distraction. In these cases, the assumption of cooperative agents may not hold and coupled methods can lead to over-confident decisions and potential collisions. Since the proactive approach considers future information, decoupled methods (sepa-

rating prediction and control) can handle interactions between robots and humans explicitly or implicitly. The decoupled methods with explicit human-robot interaction modeling make motion predictions of humans considering the robot's presence, while the implicit methods adjust the prediction based on the current motion patterns of humans, which can be influenced by the robot's behavior in real time. While decoupled methods separate prediction and control, they are still *closed-loop* if the controller considers motion prediction results at each time step. One of the promising decoupled methods is to combine motion prediction with optimization-based methods, such as MPC [18], [86], [87]. The ability to handle predictive information in MPC makes the integration of motion prediction and control straightforward and natural.

1.6 Research Questions

The research questions are formulated based on the identified gaps and challenges in the related work, aiming to address the limitations of existing methods and propose innovative solutions for dynamic obstacle avoidance in industrial indoor environments. The research questions are as follows:

RQ1: What are the key design principles and architectural components required to develop an effective and scalable Factory with Vision framework for industrial applications?

RQ2: How can learning-based motion prediction techniques be integrated with optimization-based control methods to improve the safety and efficiency of AMRs in dynamic environments?

RQ3: Which motion prediction methods are suitable and efficient for capturing the uncertainty in human behavior and how can they provide actionable insights for downstream planning and control tasks?

RQ4: What planning and control strategies can be employed to ensure safe, stable, efficient, and flexible navigation of AMRs in hybrid environments?

1.7 Research Approach

The research approaches are outlined to provide a roadmap for addressing the research questions and achieving the objectives of the thesis. The research approaches include:

- Designing and implementing the Factory with Vision framework: The architecture of the FwV framework includes the vision system, AMR planner and controller, and other automation systems. Each module is implemented and tested in an experimental setup. The detailed validation is conducted in simulation environments.
- Integrating learning-based motion prediction with optimization-based control: The key is to develop a pipeline processing the perception data and generating informative predictions of dynamic obstacles. A proper interface is designed to connect the motion prediction module with the motion planning and control module, improving the safety and efficiency of AMRs in dynamic environments.
- Proposing novel motion prediction methods: Different learning-based motion prediction methods are explored and compared to identify the most suitable and efficient approach for capturing the uncertainty in human behavior. These methods are evaluated based on their prediction accuracy, computational efficiency, uncertainty expression, and compatibility with downstream planning and control algorithms.
- Developing planning and control strategies for hybrid environments: Various planning and control strategies are developed and tested to ensure safe, stable, efficient, flexible, and scalable navigation of AMRs in hybrid environments. Apart from optimization-based control methods, learning-based controllers are also explored to adapt to complex environments where dynamic and nonconvex constraints are present.

1.8 Outline

This thesis is comprised of two parts. Part I is an overview of the industrial project, the research background, related approaches, and summaries of several publications based on this project. Part II contains the selected publications that are most relevant to the research questions and objectives of

this thesis. In Part I, after the general introduction in Chapter 1, Chapter 2 provides a detailed explanation and discussion of perception and reasoning techniques, while Chapter 3 focuses on planning and control strategies, followed by a brief discussion on scheduling and fleet coordination as well as an extension on reactive control and control barrier functions. In Chapter 4, appended papers are summarized and reviewed from the perspective of the research questions and objectives. Finally, the conclusion and future work of this thesis are presented in Chapter 5.

CHAPTER 2

Perception and Reasoning

Similar to the recognition process of humans, robots need a two-step process to understand the environment and make decisions. The first step is perception, which involves sensing the environment and extracting relevant information. The second step is reasoning, which consists of interpreting the information and making instructive decisions based on it. Sensors play a crucial role in the perception process, providing robots with the necessary data to understand their surroundings. Based on the selection of sensors, the environment can be perceived in different representations. The raw data from the sensors are then processed and transformed into a world representation that can be used for reasoning and decision-making. This chapter briefly discusses sensors for AMRs and the perception process.

2.1 Sensor

A sensor is a device that detects and responds to some type of input from the physical environment [88]. They are widely used in various applications, such as robotics, automation, and smart homes. In the context of robotics, sensors are the eyes and ears of robots, playing a crucial role in perceiving the

environment and enabling robots to interact with it. They provide robots with information about their surroundings, allowing them to make decisions and take actions based on that information. There are different ways to classify sensors. For example, a common classification is based on the type of physical quantity they measure, such as temperature, pressure, light, sound, etc. In this thesis, sensors are categorized based on their purpose and detection range. From inner state estimation to outer environment perception, sensors used in robotics can be classified as follows:

- Sensors for robot state estimation: These sensors are used to estimate the internal state of the robot, such as position, orientation, velocity, and acceleration. They provide feedback to the robot's control system and kinematic information as a reference for the motion model.
- Sensors for close-range safety: These sensors are used to detect obstacles and hazards in the robot's immediate vicinity. They are directly connected to the safety layer and play a critical role in preventing serious accidents and injuries.
- Sensors for path guidance: These sensors are used to guide the robot along a predefined path, which relies on external infrastructure along the path. Most AGVs use this type of sensor for navigation.
- Sensors for environment perception: These sensors are used to perceive the general environment around the robot, such as objects, obstacles, and landmarks. They provide rich information about the surroundings and enable the robot to make informed decisions accordingly.

In practice, different sensors are often combined to provide a comprehensive perception, which is known as sensor fusion [89]. By fusing data from multiple sensors, robots can obtain a more accurate and reliable representation of the environment. In this thesis, the main focus is on the processing of visual perception of the environment.

Sensors for robot state estimation

The estimation of the robot's internal state is essential for reactive motion control and state feedback. The robot's state includes its kinematic features, such as position, orientation, and velocity, as well as other dynamic and physical

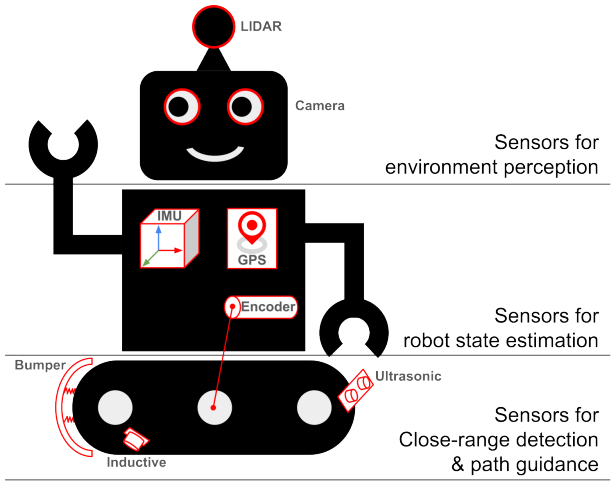

Figure 2.1: Conceptual examples of some sensors that can be used for AMRs.

properties. The scope of this research is focused on the sensors that provide kinematic information. Inertial Measurement Units (IMUs) [90] are commonly used for state estimation, usually consisting of accelerometers, gyroscopes, and sometimes magnetometers. IMUs measure the robot's linear acceleration and angular velocity to estimate its position and orientation. Global Positioning System (GPS) is another sensor used for localization, which provides the robot's global position and velocity. However, GPS has limitations in indoor environments due to signal blockage and low accuracy. Rotary encoders are used to measure the robot's wheel rotation and calculate its position and velocity. They are commonly used in mobile robots for odometry estimation.

These sensors measure the robot's kinematic state directly with high accuracy and low latency, which makes them suitable for real-time control and feedback. For example, they can provide ground truth data of the robot's state for calibration and validation purposes. They also serve as a reference in the motion model for other external sensors to correct their measurements.

Sensors for close-range Safety

Safety is paramount in human-robot collaboration, particularly in shared workspaces where robots and humans are not separated by physical barri-

ers. Close-range safety sensors are mounted on the robot to detect obstacles and hazards in its immediate vicinity, thereby preventing collisions and ensuring safe operation. Tactile sensors, which measure contact forces and pressure through mechanical deformation, displacement, or changes in electrical resistance and capacitance, play a crucial role in detecting physical interactions. In AMRs, tactile bumper sensors typically serve as the last line of defense, acting as emergency brakes when contact with an obstacle occurs. For near-range detection, ultrasonic sensors are employed to measure the distance to nearby obstacles by emitting high-frequency sound waves and calculating the time it takes for the waves to return. With a typical range of a few meters, ultrasonic sensors are well-suited for close-range safety applications. Infrared sensors have similar functionality to ultrasonic sensors but operate based on the reflection of infrared light.

These sensors provide real-time feedback on the robot's near surroundings, enabling it to react quickly to unexpected obstacles and alleviate or avoid serious accidents. They are often embedded in the robot's safety layer and are designed to trigger emergency stop mechanisms when necessary.

Sensors for path guidance

Path guidance sensors are used to guide mobile robots along predefined paths, typically supported by external infrastructure. For example, magnetic sensors detect magnetic fields generated by electrical wires embedded in the floor to localize and navigate mobile robots. Similarly, optical sensors, such as infrared sensors and cameras, identify visual markers or patterns on the ground to provide positional information. While path guidance sensors are well-suited for fixed routes and highly structured environments, their reliance on external infrastructure limits their flexibility in dynamic or unstructured settings. Despite the potential for layout changes in factories and warehouses, some permanent infrastructure remains available for use in calibrating the robot's position, such as fiducial markers.

Sensors for environment perception

To perceive the general environment around the robot, sensors should be able to establish a comprehensive representation of all objects in the vicinity. Humans can perceive the world through different senses, among which vision is

Figure 2.2: Depth image taken by an Intel RealSense depth camera. Each pixel on the depth image (right half) contains a value indicating the distance in the space w.r.t. the camera's position.

the most dominant. People can use visual information to recognize objects, infer spatial relationships, track moving objects, etc. Cameras are such sensors for visual perception, providing rich information about the environment. RGB cameras capture color images which is consistent with humans. The output of RGB cameras is a 3D matrix I of size $H \times W \times 3$, where H and W are the height and width of the image, respectively, and 3 represents the three color channels (red, green, and blue). By capturing a series of images from time t_1 to t_2, RGB cameras can represent the spatial and temporal information of the environment as $\{I_t\}_{t_1}^{t_2}$, where the cardinality of the set depends on the frame rate (also known as Frames Per Second, FPS) of the camera. Another type of camera widely used in robotics is depth cameras, which provide depth information in addition to color, as shown in Figure 2.2. The depth information can be obtained through various technologies, such as stereo vision, structured light, and Time-of-Flight (ToF). Stereo vision uses two cameras to capture the same scene from slightly different perspectives, allowing the system to calculate the depth by triangulation. Structured light projects a pattern of light onto the scene and measures the deformation of the pattern to infer depth. Time-of-flight cameras emit light pulses and measure the time it takes for the light to return, which is proportional to the distance to the object. Depth cameras output depth images besides color images, where each element represents the distance to the corresponding pixel in the color image.

Unlike most cameras receiving information passively, LiDAR sensors actively emit laser beams and measure the time it takes for the beams to return after reflecting off objects. LiDAR sensors provide accurate distance measure-

ments of the surrounding 3D space in the form of point clouds. The point cloud is a set of 3D points where the number of points and their spatial distribution depend on the sensor's field of view and resolution. The point cloud can be further processed to extract features such as planes, lines, and objects.

Placement of sensors

The common setup of AMR focuses on attaching various sensors to the robot to enhance its perception capabilities and autonomy, such as shown in Figure 2.1. Cameras and LiDAR sensors are often mounted on the robot's chassis or a raised platform to provide a broad field of view and reduce occlusions. The choice of sensor placement depends on the sensor's field of view, range, and purpose. The placement of sensors should be carefully designed to maximize coverage, minimize blind spots, and ensure robust perception in various scenarios. This setup allows the robot to perceive its surroundings from multiple perspectives and modalities, enabling it to make informed decisions and navigate complex environments autonomously. However, unlike public open spaces, industrial environments are scattered with obstacles obstructing the detection of sensors. To address this issue, another solution is to attach the sensors, especially visual sensors, to the ceiling or walls to provide a bird's-eye view of the environment, such as shown in Figure 1.2. This setup can improve the visibility of the sensors and reduce occlusions caused by obstacles on the ground. These sensors can also be used for other purposes, such as guiding other robots, gathering production data, and detecting anomalies and hazards. In this thesis, a visual system composed of ceiling-mounted cameras with overlapping fields of view is applied to provide a comprehensive perception of the environment. By using a camera grid, the workspace is covered from a bird's-eye view, and a real-time 2D map is generated to assist AMRs in navigation and obstacle avoidance.

2.2 Perception

Sensor data contains a wealth of raw environmental information that must be processed to extract meaningful features and insights. Perception is the process of interpreting sensor data to understand the environment and support informed decision-making. For instance, LiDAR point clouds can be analyzed

to detect objects based on their shapes and infer their spatial positions and relationships, while camera images provide color and texture information useful for object recognition and identification. In this work, the emphasis is on visual perception. Perception encompasses several critical tasks, including detection and classification, segmentation, and tracking. This section discusses these tasks in the context of visual perception.

The general form of a perception task involves a model that processes the input and generates a corresponding output. For a model G_θ parameterized by θ, the task is to generate an estimation $\hat{\boldsymbol{y}} = G(\boldsymbol{x})$ given an input $\boldsymbol{x}$ that the gap between the estimation and the ground truth is as small as possible. The "gap" is defined by a loss function $\mathcal{L}(\hat{\boldsymbol{y}}, \boldsymbol{y})$, for example, using Euclidean distance as a loss to evaluate the closeness of two 2D points. The form of the output and the design of the loss function vary for different perception tasks.

Detection and classification

Detection and classification are fundamental tasks in visual perception that involve identifying objects in the image and assigning them to predefined categories. Object detection aims to locate objects within an image or video frame and draw bounding boxes around them, while object classification focuses on assigning labels to detected objects based on their visual appearance. Detection and classification models are trained on annotated datasets to recognize specific object features, such as people, vehicles, and obstacles. The output $\boldsymbol{y}$ usually contains the coordinates of bounding boxes, the confidences of bounding boxes, and the confidences of classes (the class with the highest confidence is the identified class). There are many classic deep learning-based object detectors, such as Faster R-CNN [91], YOLO [92], [93], and SSD [94], which leverage Convolutional Neural Networks (CNNs) to extract features and estimate object locations and classes. Through detection and classification, robots can identify and localize objects of interest in their surroundings, enabling them to make informed decisions and take appropriate actions.

Segmentation

Segmentation is the process of partitioning an image into semantically meaningful regions or segments. Unlike object detection, which focuses on identifying objects as a whole, segmentation aims to assign a label to each pixel

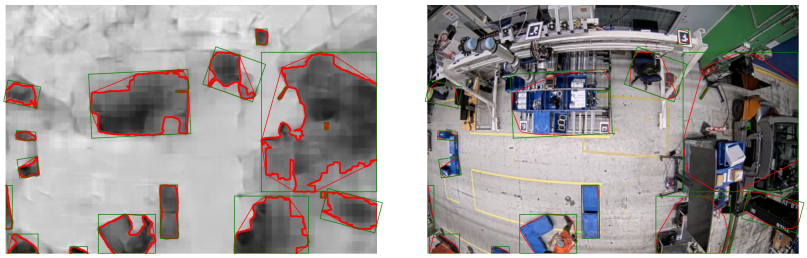

Figure 2.3: An example [18] of object segmentation and detection of an industrial laboratory from a BiSeNet neural network [95].

in the image, creating a pixel-wise mask that distinguishes different objects and background regions. Semantic segmentation assigns a single class label to each pixel, while instance segmentation differentiates between individual object instances of the same class. Deep learning models, such as U-Net [49] and other encoder-decoder architectures [96], have been widely used for semantic and instance segmentation tasks. By segmenting the environment into distinct regions, robots can better understand the spatial layout and relationships between objects, facilitating more accurate perception and interaction.

Tracking

Tracking involves following objects over time as they move through the environment. Object tracking is essential for maintaining consistent identities and trajectories of objects across frames, enabling robots to predict their future positions and avoid collisions. Tracking algorithms typically rely on motion models and appearance features to estimate object states and associate them with previous detections. Multi-object tracking extends this concept to handle multiple objects simultaneously, tracking their interactions and movements in complex scenes [97], [98]. By tracking objects in real time, robots can anticipate their behavior and plan to avoid potential collisions or conflicts, which is vital for dynamic obstacle avoidance.

2.3 Reasoning

Reasoning refers to the application of rational cognitive processes to extrapolate from existing knowledge to generate new insights. In the context of robotics, it is for robots to learn from past experiences and data to either infer the cause/intention/result of an event/action or predict the future state of the environment [15]. As humans, reasoning is a natural process and happens frequently, such as avoiding collisions with other pedestrians in advance by predicting their motion. An AMR, without extending the current information to predict the future state of the environment, cannot navigate proactively and smoothly.

Motion prediction

Motion prediction is a crucial component of reasoning in dynamic obstacle avoidance tasks. In this context, motion is defined as the change in an object's position in space over time, considering the object as a unified entity. Although motion can be interpreted in various ways, this definition focuses on the overall trajectory of the object. While motion is continuous in real life, for the facility of computation and compatibility with digital devices, the time domain is sampled according to sampling time t_s. To distinguish the discrete time domain from the continuous one, the subscript k is used to indicate a discrete time step. Formally, let $\boldsymbol{p}_k$ be the position of an object at time step k, and the motion of this object over a discrete time interval $[k, k+T]$ can be represented as a sequence $(\boldsymbol{p}_k, \boldsymbol{p}_{k+1}, \ldots, \boldsymbol{p}_{k+T})$, which is also known as a trajectory. Motion prediction aims to forecast the future positions of objects based on information available at the current time step. Let the current time step be 0; the task of motion prediction is to estimate the future positions $(\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_T)$ of an object over the next T time steps. Through object detection and tracking, the current and past positions of an object can be obtained, i.e., $(\boldsymbol{p}_{-h}, \boldsymbol{p}_{1-h}, \ldots, \boldsymbol{p}_0)$, where h is the history length. For a fixed workspace, the past trajectory of an object can be directly used to predict its future motion [7]. However, since object motion is restricted by the environment, it is important to consider environmental context, especially if the layout of the workspace is not fixed. As shown in Figure 2.3, segmentation techniques can be used to extract drivable areas, obstacles, and other relevant features from the environment, resulting in an occupancy grid or

semantic map. By combining object trajectories with environmental information, motion prediction models can anticipate the future positions of objects more accurately and adaptively, and they are more generalizable to unseen scenarios.

In summary, in a dynamic environment with N_d moving obstacles, given their past trajectories $\mathcal{H}_{-h:0}^{(n_d)}$ where $n_d = 1, 2, \ldots, N_d$ and the environmental information I^{env}, the motion prediction task can be formulated as estimating the future positions of the target object $\hat{\mathcal{H}}_{1:T}^{(*)}$ over the next T time steps:

$$\hat{\mathcal{H}}_{1:T}^{(*)} = f^{\text{MP}}(\mathcal{H}_{-h:0}^{(1)}, \mathcal{H}_{-h:0}^{(2)}, \ldots, \mathcal{H}_{-h:0}^{(N_d)}, I^{\text{env}}) \tag{2.1}$$

Note that the inputs to the motion prediction function are not absolute, which means that not only can more information be included, but the model may only rely on partial information based on the application scenario. However, the given representation of the motion prediction function covers the common case. As mentioned before, traditional model-based methods cannot handle the complexity of the motion prediction model f^{MP}, and deep learning-based methods have shown great ability in processing high-dimensional data, capturing complex features, and generating generalizable predictions. In this work, deep learning models are employed to learn the motion patterns of objects and predict their future positions accurately and efficiently.

Uncertainty in motion prediction

Uncertainty is an inherent aspect of human motion and can be conceptualized in two main forms: *aleatoric* and *epistemic* [99]. Aleatoric uncertainty refers to inherent, irreducible randomness within a system, which can be quantified given sufficient data. It is typically parameterized statistically as variances or ranges. In contrast, epistemic uncertainty arises from a lack of knowledge or understanding of the system, and it tends to be more pronounced when the randomness in motion is minimal. Epistemic uncertainty can potentially be reduced through the acquisition of more information or improved modeling. In motion prediction, as illustrated in Figure 2.4, aleatoric uncertainty refers to the inherent local randomness in an object's motion, such as jitter or noise, while epistemic uncertainty arises from the lack of complete information about the object's motion or the environment, or the model's inability to capture all relevant factors.

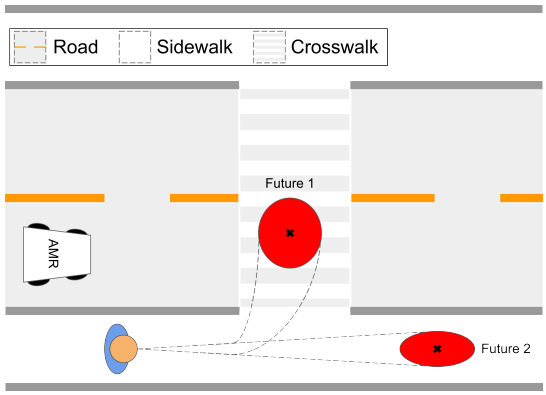

Figure 2.4: An example of different types of uncertainty for the motion prediction problem. Two possible futures describe the epistemic uncertainty while the increasing future motion area depicts the aleatoric uncertainty (local randomness).

To capture local uncertainty, instead of predicting a position point at each time step, a probability distribution over the future positions can be estimated. In practice, the probability distribution can be represented by a parametric distribution, such as a Gaussian distribution (as in Figure 2.4), or a discrete probability map. As for epistemic uncertainty, if the model has a limited ability to consider diversity due to inadequate model capacity or a lack of environmental information, it cannot accurately predict the change of motion in terms of speed or direction. A more challenging problem is to include Multimodal Motion Prediction (MMP), which is a comprehensive representation of epistemic uncertainty. The term "multimodal" refers to the existence of multiple modes (local peaks) in the probability distribution. Gaussian Mixture Models are common representations of multimodal distributions.

Considering a motion prediction task, where given an input $\boldsymbol{x}$, after a certain time steps, there are M ground-truth future positions $Y = \{{}^{m}\boldsymbol{y}\}_{m=1}^{M}$. Note that the multiple ground-truth positions are normally not available in practice. Define $\mathcal{M}$ as a mode set that contains all ground truth positions of the mode. For simplicity, mode sets cannot overlap with each other, i.e., a position can only belong to a single mode set. For any ${}^{i}\boldsymbol{y}, {}^{j}\boldsymbol{y} \in Y$, assuming their distance (such as Euclidean distance) is $d({}^{i}\boldsymbol{y}, {}^{j}\boldsymbol{y})$, they belong to the

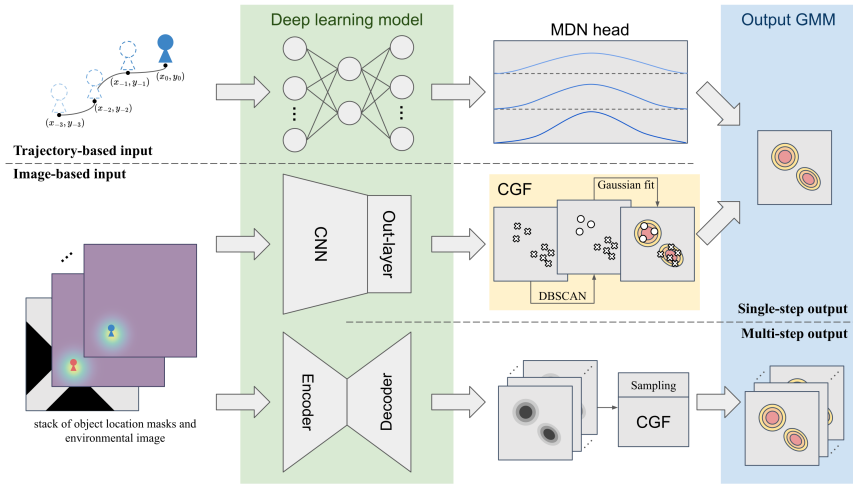

Figure 2.5: Three different motion prediction pipelines are involved in this work (each row is a pipeline). The first row is the MDN-based motion prediction method [7]. The second row is the multi-hypothesis-based method [18], [38]. The last one is energy-based motion predictors [100].

same mode if and only if their distance is no larger than a certain threshold ϵ_m [38], i.e.,

$$ {}^{i}\boldsymbol{y}, {}^{j}\boldsymbol{y} \in \mathcal{M} \iff d({}^{i}\boldsymbol{y}, {}^{j}\boldsymbol{y}) \leq \epsilon_m, \tag{2.2} $$

or there exists another $\boldsymbol{y} \in Y$ such that ${}^{i}\boldsymbol{y}$ and ${}^{j}\boldsymbol{y}$ can be proven to be in the same mode with $\boldsymbol{y}$, i.e.,

$$ {}^{i}\boldsymbol{y}, {}^{j}\boldsymbol{y} \in \mathcal{M} \iff \exists\, \boldsymbol{y} \in Y, ({}^{i}\boldsymbol{y}, \boldsymbol{y} \in \mathcal{M}) \wedge ({}^{j}\boldsymbol{y}, \boldsymbol{y} \in \mathcal{M}'), \tag{2.3} $$

where $\mathcal{M}'$ is another recognized mode set. The second rule has a chain effect that clusters data points in close vicinity. Traditional single-point motion prediction is insufficient for capturing multimodal characteristics in motion prediction. To achieve multimodal motion prediction, the model should be able to output either a probability distribution or multiple predictions at once. In this work, three different approaches are discussed for multimodal motion prediction: parameterized multimodal distribution estimation, multiple hypothesis generation, and discrete probability map prediction.

Learning the uncertainty

Gaussian Mixture Models (GMMs) are a common parameterized multimodal probability distribution representation. A GMM is a weighted sum of multiple Gaussian distributions,

$$Pr(\boldsymbol{y}) = \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m), \quad \sum_{m=1}^{M} \pi_m = 1,$$

where π_m is the weight of the m-th Gaussian component, $\boldsymbol{\mu}_m$ and $\boldsymbol{\Sigma}_m$ are the mean and covariance of the m-th Gaussian component, respectively. The number of components M is a hyperparameter that needs to be determined based on the scenario. Instead of estimating the future position $\boldsymbol{y}$ directly, the motion prediction model needs to estimate the parameters of the GMM. For a future time step $t \in \{1, 2, \ldots, T\}$,

$$\hat{\boldsymbol{P}}_t = f^{\text{MP-GMM}}(\boldsymbol{x}, t), \tag{2.4}$$

where $\hat{\boldsymbol{P}}_t$ is the output parameter vector containing the weights $\hat{\pi}$, means $\hat{\boldsymbol{\mu}}$, and covariances $\hat{\boldsymbol{\Sigma}}$ of the GMM. So, the probability distribution of the predicted future position $\hat{\boldsymbol{y}}_t$ is

$$Pr(\hat{\boldsymbol{y}}_t|\,\boldsymbol{x}) = \sum_{m=1}^{M} \hat{\pi}_m(\boldsymbol{x}, t) \mathcal{N}\left(\hat{\boldsymbol{y}}_t|\hat{\boldsymbol{\mu}}_m(\boldsymbol{x}, t), \hat{\boldsymbol{\Sigma}}_m(\boldsymbol{x}, t)\right). \tag{2.5}$$

This kind of deep-learning model is called the Mixture Density Network (MDN) [45]. To train an MDN, since it is a probabilistic model, the loss function can be designed as the negative log-likelihood of the ground truth future positions $\boldsymbol{y}$ given the input $\boldsymbol{x}$,

$$\mathcal{L}_{\text{NLL}} = -\log Pr(\boldsymbol{y}|\boldsymbol{x}) = -\log \sum_{m=1}^{M} \pi_m \mathcal{N}(\boldsymbol{y}|\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m). \tag{2.6}$$

MDNs have shown great potential in capturing multimodal distributions and generating diverse predictions, making them suitable for multimodal motion prediction tasks. However, it has been mentioned in the literature [35] that MDNs may struggle to capture complex multimodal distributions with high-dimensional data, and they have a tendency to generate mode collapse, which

is in line with our experimental observations [18]. To address these limitations, other approaches such as multiple hypothesis generation and discrete probability map prediction can be considered. A motion prediction pipeline based on MDNs is illustrated in Figure 2.5.

Multiple hypothesis generation is a straightforward approach to multimodal motion prediction, where the model generates multiple predictions for each time step. Instead of outputting a single prediction, the model generates Z hypotheses for the future position at each time step. The number of hypotheses Z is a hyperparameter that needs to be determined based on the scenario. The output of the model is a set of Z future positions $\{{}^z\hat{\boldsymbol{y}}_t\}_{z=1}^{Z}$, which is the z-th hypothesis of the future position at time step t. The difficulty of this approach lies in the design of the loss function, as the ground truth future positions are not unique. Multiple choice learning [46] is a learning paradigm that can be used to train models with multiple hypotheses. The basic idea is to treat each hypothesis separately and calculate the loss for each hypothesis given a single ground truth future position. Then, the hypothesis with the lowest loss is selected to be updated, in which way the model learns to generate diverse predictions covering the various future positions. This kind of loss function is also known as the Winner-Takes-All (WTA) loss. Formally, the WTA loss is defined as

$$\mathcal{L}_{\text{WTA}} = \sum_{z=1}^{Z} w_z \cdot l({}^z\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t), \quad w_z = \delta(z = \arg\min_i l({}^i\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t)), \tag{2.7}$$

where $l({}^z\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t)$ is the loss term between the z-th hypothesis and the ground truth future position and $\delta(\cdot)$ is the generalized Kronecker delta function which is 1 if the condition is satisfied and 0 otherwise. The WTA loss encourages the model to generate diverse predictions and learn to predict the correct future position among multiple hypotheses. This approach is simple and effective for tasks with multiple potential choices, as it allows the model to explore different possibilities and generate diverse predictions. There are two problems with the original WTA loss. First, if there are fewer hypotheses than targets (ground truth future positions) in the neighborhood, at least one hypothesis will be attracted by at least two targets and result in a local equilibrium. Second, each target only attracts one hypothesis, which may lead to the abandonment of some hypotheses. In the context of motion prediction, the first problem leads to wrong predictions, while the second problem leads to untrackable hypotheses. To address these problems, an Evolving WTA

(EWTA) loss was proposed [35], which alters the idea of winner-takes-all to top-winners-take-all, i.e., it updates the top z_{top} hypotheses. During training, z_{top} is a preset hyperparameter, and its value keeps decreasing as the training progresses until it reaches 1. In this way, more hypotheses are updated compared to the original WTA loss. However, it still leaves some equilibria between different targets. To further address this issue, we propose an extensive training strategy via two new loss functions: the Adaptive WTA (AWTA) loss and the Swarm WTA (SWTA) loss [38]. The AWTA loss is designed to attract hypotheses to the nearest target, depending on if the distance/loss between the hypothesis and the target is smaller than an adaptive range, defined as

$$r_{\text{adp}} = \min_{z \in \mathcal{Z}}({}^{z}l) + \alpha_{\text{adp}} \left(\max_{z \in \mathcal{Z}}({}^{z}l) - \min_{z \in \mathcal{Z}}({}^{z}l) \right), \tag{2.8}$$

where $\mathcal{Z} = \{1, 2, \ldots, Z\}$, ${}^{z}l$ is the loss between the z-th hypothesis and the target, and α_{adp} is a hyperparameter that controls the range. Then, the AWTA loss is

$$\mathcal{L}_{\text{AWTA}} = \sum_{z=1}^{Z} w'_z l({}^{z}\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t), \quad w'_z = \delta({}^{z}l \leq r_{\text{adp}}). \tag{2.9}$$

The AWTA loss has a clustering effect that attracts all hypotheses to their nearest targets, so hypotheses gather around the mean positions of components/modes from the ground truth distribution. Although the clustering effect shows the mean positions of the modes, it loses the representation of the local uncertainty. To address this issue, the SWTA loss is proposed to recover the local uncertainty after the clustering effect. The SWTA loss is defined as

$$\mathcal{L}_{\text{SWTA}} = \sum_{z=1}^{Z} w'_z \min_{i \in \mathcal{Z}} l({}^{i}\hat{\boldsymbol{y}}_t, \boldsymbol{y}_t), \quad w'_z = \delta({}^{z}l \leq r_{\text{adp}}). \tag{2.10}$$

The SWTA loss can counteract the clustering effect of the AWTA loss by giving greater updates to hypotheses within each cluster that are closer to the ground truth positions. The combination of the AWTA and SWTA losses provides a comprehensive training strategy for multiple hypothesis generation. The overall training process requires three steps: the evolving step (EWTA), the clustering step (AWTA), and the recovery step (SWTA). The trained model can generate multiple hypotheses for a single time step, as shown in

Figure 2.5. To predict a sequence of future positions, the model can be applied iteratively to generate multiple hypotheses at each time step, which is time-consuming. To address this issue, a more efficient approach is to generate multi-channel probability maps for all future time steps at once.

Given a ground-truth position $\boldsymbol{y}_t$, instead of the one-hot encoding, a probability distribution, such as the Gaussian distribution, can be assumed to represent the uncertainty of the ground-truth position. By converting the real-world position to the pixel position and covering it using a probability distribution, a ground-truth probability mask $\boldsymbol{A}$ can be generated. Then, the new task is to learn the probability mask representing the ground-truth position. As validated in the literature, the U-Net [49] architecture is suitable for this task, which is a fully convolutional network with an encoder-decoder structure. For the learning strategy, the Binary Cross-Entropy (BCE) loss is commonly used to train the model, which is defined as

$$\mathcal{L}_{\text{BCE}} = -\sum_{w',h'} [\boldsymbol{A}_{w',h'} \ln \sigma(\boldsymbol{E}_{w',h'})] - \sum_{w',h'} [(1-\boldsymbol{A}_{w',h'}) \ln(1-\sigma(\boldsymbol{E}_{w',h'}))], \tag{2.11}$$

where $\boldsymbol{E}$ is the output of the model, $\sigma(\cdot)$ is the sigmoid function, and (w', h') is the pixel position. Assuming the size of the probability mask is $W \times H$, we have $w', h' \in \{1, 2, \ldots, W\} \times \{1, 2, \ldots, H\}$. The BCE loss encourages the model to predict a probability mask that best matches the ground truth mask pixel to pixel, and may lead to overestimation. Similarly but from a different perspective, the Kullback-Leibler Divergence (KLD) can be used to match the probability mask by minimizing the difference between the two probability distributions. The KLD loss is defined as

$$\mathcal{L}_{\text{KLD}} = \sum_{w',h'} \boldsymbol{A}_{w',h'} \ln \frac{\boldsymbol{A}_{w',h'}}{\boldsymbol{P}_{w',h'}}, \tag{2.12}$$

where $\boldsymbol{P}$ is the processed output of the model in the form of a probability map. Both BCE and KLD can achieve multimodal motion prediction with notable uncertainty representation and give high coverage of the ground truth positions. This is particularly useful in open spaces when the movement of objects is highly random [36]. However, for downstream tasks, the overestimation of the probability mask may lead to a high false positive rate and the planner struggles to find a feasible path. To address this issue, we propose to use energy-based learning for more accurate and concentrated prediction. Energy-based models are widely studied for their generative ability [101], but they are

also used for regression tasks [102]. For an energy model $G_\theta(\cdot)$ with parameters θ, given input $\boldsymbol{x}$ and output $\boldsymbol{y}$, the model should assign lower energy to the ground truth position $\boldsymbol{y}$ than to other positions $\boldsymbol{y}'$, i.e., $G_\theta(\boldsymbol{x}, \boldsymbol{y}) < G_\theta(\boldsymbol{x}, \boldsymbol{y}')$. The output of the model is the energy map $\boldsymbol{E} = G_\theta(\boldsymbol{x})$, where $\boldsymbol{E}_{w',h'}$ is the energy at pixel position (w', h'). The predicted future position $\hat{\boldsymbol{y}}$ is the position with the lowest energy in the energy map, i.e., $\hat{\boldsymbol{y}} = \arg\min_{w',h'} \boldsymbol{E}_{w',h'}$. The desired energy map should enforce that $||\boldsymbol{y} - \hat{\boldsymbol{y}}||$ is minimized. Multiple predictions can be drawn by Monte Carlo sampling from the energy map, then the energy map is regarded as an unnormalized probability distribution. According to the Gibbs-Boltzmann distribution,

$$\boldsymbol{P}^{(c)} := \Pr(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{e^{-G_\theta(\boldsymbol{x},\boldsymbol{y})}}{\int e^{-G_\theta(\boldsymbol{x},\boldsymbol{y}')} d\boldsymbol{y}'}. \tag{2.13}$$

Since the integral in the denominator is normally intractable, its approximation is used in practice, e.g., via important sampling. Another option is to discretize the energy space and substitute the integral with a summation. For motion prediction tasks, if the discretized grid size is equal to the resolution of the environmental image, the approximation of the integral is sufficiently accurate. For a discrete energy map $\boldsymbol{E}$ with size $W \times H$, the probability map $\boldsymbol{P}$ is

$$\boldsymbol{P}_{w,h} := \Pr(\boldsymbol{y}|\boldsymbol{x}; \boldsymbol{\theta}) = \frac{e^{-\boldsymbol{E}_{w,h}}}{\sum_{w'=1}^{W} \sum_{h'=1}^{H} e^{-\boldsymbol{E}_{w',h'}}}, \tag{2.14}$$

where $\boldsymbol{y} = [w, h]^\top$ indicating the row index w and column index h of the pixel position. Training based on the probability representation above is straightforward but unstable, and the result tends to be noisy when the size of the energy map is large. The learned model is sensitive to noise and likely to generate too high or low energy values. To solve this problem, the weighted soft loss and the Positive Exponential Linear Unit (PELU) are proposed.

The weighted soft loss is similar to the probability mask $\boldsymbol{A}$, but instead of regarding it as a distribution, it is normalized as a weight mask for the energy space to refine the probability:

$$\bar{\boldsymbol{P}}_{w,h} = \frac{e^{-\bar{\boldsymbol{A}}_{w,h}\boldsymbol{E}_{w,h}}}{\sum_{w'=1}^{W} \sum_{h'=1}^{H} e^{-\boldsymbol{E}_{w',h'}}}, \tag{2.15}$$

where $\bar{\boldsymbol{A}}$ is the normalized weight mask ($\boldsymbol{A}$ divided by its maximum value).

The PELU activation function is defined as

$$f_{\mathrm{PELU}}(x) = \max(0, x) + \min(0, \beta_{\mathrm{ELU}}(e^x - 1)) + 1 + \epsilon, \tag{2.16}$$

where β_{ELU} is a positive hyperparameter that controls the slope of the negative part [103], and ϵ is a small positive constant to avoid numerical instability. Using the PELU layer to substitute the exponential function in the energy model can stabilize the training process [100]. With $\boldsymbol{E}' = f_{\mathrm{PELU}}(-\boldsymbol{E})$, the probability map is

$$\boldsymbol{P}'_{w,h} = \frac{\boldsymbol{E}'_{w,h}}{\sum_{w'=1}^{W} \sum_{h'=1}^{H} \boldsymbol{E}'_{w',h'}}. \tag{2.17}$$

Combining the new probability representation with the NLL loss, we get the Energy-oriented NLL (ENLL) loss,

$$\mathcal{L}_{\mathrm{ENLL}} = -\ln \sum_{w',h'} \bar{\boldsymbol{A}}_{w',h'} \boldsymbol{E}'_{w',h'} + \ln \sum_{w',h'} \boldsymbol{E}'_{w',h'}. \tag{2.18}$$

From the experimental results [100], the energy-based learning strategy is shown to be more suitable for downstream motion planning tasks compared to the other aforementioned methods in terms of safety and efficiency.

Implicit reasoning in motion prediction

Implicit reasoning refers to the underlying cognitive processes that guide decision-making without explicit awareness or representation of the reasoning results. In the context of obstacle avoidance, especially for dynamic obstacles, implicit reasoning means that the robot can avoid collisions as it considers the future states of the obstacles without explicitly predicting them. As an observation while training Deep Reinforcement Learning (DRL) agents for obstacle avoidance, the agents can learn to avoid collisions with dynamic obstacles in advance without any explicit motion prediction [19], [104]. This phenomenon suggests that the agents implicitly reason about the future states of the obstacles based on their current and past positions. The agents learn to anticipate the future positions of the obstacles and adjust their trajectories accordingly to avoid collisions. This implicit reasoning is a powerful capability that enables robots to navigate complex environments and interact with dynamic obstacles effectively. Since this is not the focus of this work, the detailed mechanism of implicit reasoning in obstacle avoidance is not discussed here. However, it is an interesting research direction that warrants further investigation.

CHAPTER 3

Planning and Control

After the perception and reasoning of the surroundings, a mobile robot needs to plan its path according to the scheduled task. Then, the robot needs to control its motion to follow the planned path. The planning and control of the robot are crucial for the robot to navigate safely and efficiently in the environment. The planning usually has two levels: global (long-term) planning and local (short-term) planning. Global planning is to generate a reference path from the start to the goal by considering the static layout of the environment. Local planning is to generate a local reference trajectory to follow the reference path while avoiding unexpected obstacles during global planning, such as dynamic obstacles. Finally, the controller generates the control signals to follow the local reference trajectory. While there are various planning and control algorithms, the choice of the algorithm depends on specific requirements, such as the environment, robot, and task.

3.1 Planning in the environment

According to the application, there are two common types of environments: *regulated* workspaces and *open* freespaces. Factories and warehouses are exam-

ples of regulated workspaces, where the high-level layout of the environment is known and fixed. There are regulations and standards to follow in these environments, such as the division of human and vehicle areas. AGVs [9] are widely applied in these environments simply to follow the predefined paths. On the other hand, open freespaces are environments where the layout may frequently change, such as some public areas where people are moving around or some indoor environments where, for example, furniture may be moved frequently. Navigation in these environments is often referred to as the social navigation problem [55], where the robot needs to be more flexible to interact with the environment and people. Planning in different environments requires different approaches. Besides, planning *globally* and *locally* in the environment is also different. Global planning focuses on generating a reference path from the start to the goal according to the high-level layout of the environment. The reference path is usually fixed after the global planning, unless the environment is significantly changed and the original plan is no longer valid. Global planning in open spaces requires the robot to find valid path nodes to reach the goal while avoiding obstacles, which makes sampling-based algorithms popular choices, such as Rapidly Exploring Random Trees (RRTs), probabilistic roadmaps, and their variants. As for regulated workspaces, there are usually predefined drive lanes and robots are not allowed to find their own paths, which makes graph-based algorithms, such as A* and Dijkstra, more suitable.

After obtaining the reference path, it is common to generate a local reference trajectory during runtime. Although it is not necessary to have local planning, for instance, reactive controllers can directly follow the reference path and avoid obstacles reactively, local planning can help the robot avoid unexpected obstacles more smoothly and efficiently. The local planning is based on the reference path, the current state of the robot, and the current situation of the environment. For example, Timed Elastic Band (TEB) [105] and Dynamic Window Approach (DWA) [60], [106] are two popular local planning algorithms. TEB is a trajectory optimization algorithm that considers the robot's kinematic constraints and the environment's dynamic constraints. DWA is a sampling-based algorithm that generates a set of feasible trajectories and selects the best according to the cost function. Other algorithms, such as RRT-based algorithms [107]–[109] and artificial potential field methods [69], are also used for local planning. Local planning is often closely related to the

control of the robot, and many local planning algorithms can be extended to control the robot directly or directly generate the control signals. For example, DWA generates the trajectory by repeating the sampled control signals, then the robot can follow the trajectory by executing the control signals.

Apart from the classic planning algorithms, there are also learning-based planning algorithms, such as Reinforcement Learning (RL) and neural network-based methods [73]. RL is a type of machine learning that enables an agent to learn how to interact with the environment to achieve a goal. The agent learns by trial and error, and the learning process is based on the reward signal from the environment. Neural networks are introduced to process more complex data and learn more complex policies. Learning-based planning algorithms have shown promising results in some specific tasks, such as playing games [110] and controlling robots [111]. However, learning-based planning algorithms are often data-hungry and require a large amount of data to train the model. Besides, the learning process is often time-consuming and the model may not be interpretable. Therefore, learning-based planning algorithms are not widely used in the industry yet, but they have the potential to be applied in the future.

In this work, we focus on the planning and control of mobile robots in regulated workspaces, such as factories and warehouses. The robots are required to follow predefined paths and avoid unexpected obstacles. Global planning uses either a visibility graph with A* [112] or a predefined path graph [18]. For a given graph, if the traffic load is low, the A* algorithm is sufficient to provide the reference path; Otherwise, there should be a high-level scheduler to manage the traffic [113]. In the following sections, we will assume the reference path is given for each robot and focus on the local planning and control of the robots.

3.2 Obstacle avoidance

Obstacle avoidance is a critical part of the local planning and control of mobile robots. The robot needs to avoid unexpected obstacles while following the reference path. There are two types of obstacles: static obstacles and dynamic obstacles. Static obstacles are obstacles that do not move, such as pallets, parked vehicles, and containers in the factory. Dynamic obstacles are obstacles that move, such as other mobile robots, humans, and forklifts. To avoid an

obstacle, we need to consider the sizes of both the robot and the obstacle. A simple way to avoid an obstacle is to inflate the obstacle by the size of the robot so that the robot can be regarded as a point [112].

Obstacles can be represented as geometric shapes (such as polygons and ellipses) or occupancy grids. In this work, static obstacles are modeled as polygons represented by sets of inequalities. Let $\mathcal{O} = \cup_n O^{(n)}$ be the set of all static obstacles, where each obstacle $O^{(n)}$ with $N_e^{(n)}$ edges is a closed-set representing the occupied area of the obstacle and can be defined by a closed intersection of half-spaces, i.e., $O^{(n)} = \{\boldsymbol{p} \in \mathbb{R}^2 | \boldsymbol{b}_i^{(n)} - (\boldsymbol{a}_i^{(n)})^\top \boldsymbol{p} > 0, \forall i \in \mathbb{N}_{[1,N_e^{(n)}]}\}$ with $\boldsymbol{a}$ and $\boldsymbol{b}$ being the coefficients of the half-spaces, and $\mathbb{N}_{[1,N_e^{(n)}]} := \{1, 2, \cdots, N_e^{(n)}\}$. Assuming convex polygons, a robot is inside a polygonal obstacle if it is inside all half-spaces defining the obstacle. A penalty term for the robot being inside an obstacle can be defined as

$$J_{\mathcal{O}}(\boldsymbol{p}) = Q_{\mathcal{O}} \sum_{n=1}^{N_o} \prod_{i=1}^{N_e^{(n)}} \max\left(0, \boldsymbol{b}_i^{(n)} - (\boldsymbol{a}_i^{(n)})^\top \boldsymbol{p}\right), \tag{3.1}$$

where $Q_{\mathcal{O}}$ is the penalty weight for collision with static obstacles. The penalty term is zero if the robot is outside the obstacle, and it increases as the robot gets closer to the center of the obstacle. For dynamic obstacles, to consider the uncertainty, they are modeled by two-dimensional Gaussian distributions in the geometric form of ellipses. For an ellipse centered at $\boldsymbol{\mu} = [\mu_x, \mu_y]^\top$ with axes $\boldsymbol{\sigma} = [\sigma_x, \sigma_y]^\top$ (the correlation between the axes is neglected for simplicity), an indicator can be defined to judge if a point $\boldsymbol{p} = [p_x, p_y]^\top$ is inside the ellipse as

$$\iota(\boldsymbol{p}|\boldsymbol{\mu}, \boldsymbol{\sigma}) = \max\left\{0, 1 - \frac{(p_x - \mu_x)^2}{\sigma_x^2} - \frac{(p_y - \mu_y)^2}{\sigma_y^2}\right\}. \tag{3.2}$$

Then, the area $D^{(n)}$ occupied by the n-th dynamic obstacle can be defined as

$$D^{(n)} = \left\{\boldsymbol{p} \in \mathbb{R}^2 \,\middle|\, \iota(\boldsymbol{p}|\boldsymbol{\mu}^{(n)}, \boldsymbol{\sigma}^{(n)}) \geq 0\right\}, \tag{3.3}$$

and the corresponding penalty term is

$$J_{\mathcal{D}}(\boldsymbol{p}) = Q_{\mathcal{D}} \sum_{n=1}^{N_d} \iota(\boldsymbol{p}|\boldsymbol{\mu}^{(n)}, \boldsymbol{\sigma}^{(n)}), \tag{3.4}$$

where N_d is the number of dynamic obstacles and $Q_{\mathcal{D}}$ is the penalty weight for collision with dynamic obstacles. The penalty term for dynamic obstacles is usually time-varying, and it is updated at each time step according to the current estimations of the dynamic obstacles. These obstacle-avoidance constraints and penalty terms can be used in optimization-based planning and control algorithms. To obtain a smooth and efficient trajectory, instead of considering just the current and next time steps, the trajectory should be optimized over a horizon of multiple time steps. By moving the predictive horizon over time, we can obtain an MPC formulation for the optimization problem. The MPC problem can be solved online at each time step to generate the control signals for the robot to follow the reference path while avoiding obstacles.

Given a mobile robot with the discrete motion model $\boldsymbol{c}_{k+1} = f(\boldsymbol{c}_k, \boldsymbol{u}_k)$, where $\boldsymbol{c}_k$ and $\boldsymbol{u}_k$ are the state and control input at time step k, the obstacle avoidance problem can be formulated as follows (assuming the current time step is 0):

$$
\begin{aligned}
\min_{\boldsymbol{u}_{0:N-1}} \; & J_N + \sum_{k=0}^{N-1} \|\boldsymbol{c}_k - \tilde{\boldsymbol{c}}_k\|_{Q_c} + \|\boldsymbol{u}_k - \tilde{\boldsymbol{u}}_k\|_{Q_u}, \\
\text{s.t.} \quad & \boldsymbol{c}_{k+1} = f(\boldsymbol{c}_k, \boldsymbol{u}_k), \forall k \in \mathbb{N}_{[0,N-1]}, \\
& \boldsymbol{c}_0 = \bar{\boldsymbol{c}}_0, \\
& \boldsymbol{p}_k \notin \mathcal{O} \cup \mathcal{D}_k, \forall k \in \mathbb{N}_{[0,N-1]},
\end{aligned}
\tag{3.5}
$$

where J_N is the terminal cost, $\bar{\boldsymbol{c}}_0$ is the known initial state, $\boldsymbol{p}_k$ is the position of the robot, $\tilde{\boldsymbol{c}}_k$ and $\tilde{\boldsymbol{u}}_k$ are the reference state and control input at time k, Q_c and Q_u are the weights for the state and control input, and N is the prediction horizon. The reference states, which is the local reference trajectory in this case, can be drawn from the reference path without considering the obstacles. The MPC controller can handle obstacle avoidance even though the reference states are inside the obstacles. To successfully avoid the obstacles, the optimization problem should not have local minima, which means the obstacles should be convex and not too large. However, a better solution is to find a better reference trajectory that avoids the obstacles as much as possible.

3.3 Reference trajectory generation

While the reference path is fixed after the global planning, the local reference trajectory is generated during runtime according to the current state of the robot and the environment. The simplest way to generate the reference trajectory is to sample the reference path according to a reference speed. While most common convex obstacles can be handled by the MPC controller, non-convex obstacles or large obstacles may either cost a lot of computation time or lead to local minima. Therefore, it is better to generate a reference trajectory that avoids the obstacles as much as possible. Traditional approaches normally use trajectory optimization, such as TEB [105] and Stochastic Trajectory Optimization for Motion Planning (STOMP) [114], or sampling methods, such as DWA and RRT. In this work, a learning-based approach is proposed to generate the reference trajectory.

Deep Reinforcement Learning (DRL) [73] is a type of machine learning that combines deep learning and reinforcement learning. The basic idea of DRL is to learn a policy that maps the state of the environment to the action of the agent. While DRL has been used for navigation and control of mobile robots, it has limited usage in the real world due to the lack of interpretability and stability. To take the benefits of DRL's ability to learn complex environments as well as its constant and efficient computation speed, instead of directly using DRL to control the robot, we propose to use its action as a tendency of the robot's motion and generate the reference trajectory accordingly. To introduce the general idea, we use Deep Q-Network (DQN) [115] as an example; in a later paper, the Deep Deterministic Policy Gradient (DDPG) [116] algorithm is implemented to substitute DQN for better performance [104]. The first step is the regular DRL agent training for action generation. The basis of RL is the Markov Decision Process (MDP), which is a discrete-time stochastic process with the Markov property. The MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{P}$ is the transition probability, $\mathcal{R}$ is the reward function, and $\gamma \in [0, 1]$ is the discount factor. The agent interacts with the environment by taking actions and receiving rewards. The goal of the agent is to learn a policy π that maximizes the discounted cumulative reward $G_k = \sum_{i=0}^{\infty} \gamma^i R_{k+i+1}$ at time step k, where R_k is the reward at time k. The policy is a mapping from the state space to the action space, i.e., $\pi : \mathcal{S} \to \mathcal{A}$. The optimal policy in a state is the one that

maximizes the expected cumulative reward,

$$\pi^*(\boldsymbol{s}) = \arg\max_{\pi} \underbrace{\mathbb{E}_\pi \left[G_k | S_k = \boldsymbol{s}, \pi\right]}_{:=V_\pi(\boldsymbol{s})}, \tag{3.6}$$

where the expectation is called value V of state $\boldsymbol{s}$ following policy π. Similarly, the state-action value function, also known as the Q-value, is defined as

$$q_\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi \left[G_k | S_k = \boldsymbol{s}, A_k = \boldsymbol{a}\right]. \tag{3.7}$$

According to the Bellman expectation equation for MDPs, the Q-value function can be expressed recursively as

$$q_\pi(\boldsymbol{s}, \boldsymbol{a}) = \mathbb{E}_\pi \left[R_{k+1} + \gamma q_\pi(S_{k+1}, A_{k+1}) | S_k = \boldsymbol{s}, A_k = \boldsymbol{a}\right]. \tag{3.8}$$

Temporal Difference (TD) learning is a type of RL algorithm that learns by updating the estimate of the value function based on the difference between the expected and real rewards at each time step. As a famous variant of TD learning, Q-learning maximizes the Q-value iteratively and stores the value in a look-up table called Q-table according to the following update rule:

$$Q(\boldsymbol{s}, \boldsymbol{a}) \leftarrow Q(\boldsymbol{s}, \boldsymbol{a}) + \alpha \left[r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q(\boldsymbol{s}', \boldsymbol{a}') - Q(\boldsymbol{s}, \boldsymbol{a})\right], \tag{3.9}$$

where $\boldsymbol{s}'$ and $\boldsymbol{a}'$ are the next state and action, r is the reward at ($\boldsymbol{s}'$, and α is the learning rate. The DQN algorithm is a deep learning-based Q-learning algorithm that uses a neural network to approximate the Q-value function rather than storing it in a look-up table with limited capacity. The neural network is trained to minimize the loss function,

$$\mathcal{L}(\theta) = \left[r + \gamma \max_{\boldsymbol{a}' \in \mathcal{A}} Q(\boldsymbol{s}', \boldsymbol{a}'; \theta^{\text{target}}) - Q(\boldsymbol{s}, \boldsymbol{a}; \theta)\right]^2, \tag{3.10}$$

where θ and θ^{target} are the parameters of the current and target networks, respectively. The target network is a copy of the current network that is updated less frequently to stabilize the training process. Except for the target network, the DQN algorithm can also use experience replay to stabilize the training process and improve the convergence speed. The experience replay stores the agent's experience in a replay buffer and samples a batch of experiences randomly to train the network.

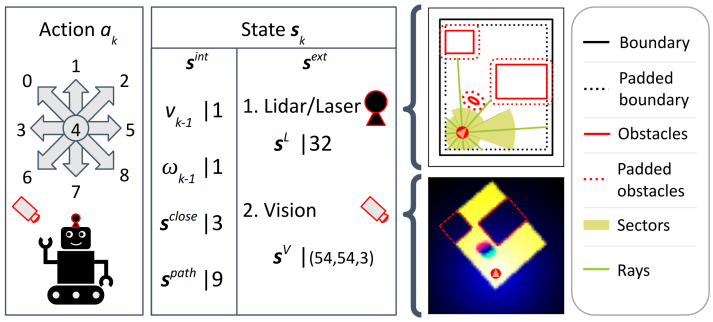

Figure 3.1: The action and state spaces of the RL agent [19].

For the motion planning and control task, the action space is the velocity and angular velocity of the robot. The state space can be divided into two parts: the state of the robot and the state of the environment. Figure 3.1 visualized the action and state spaces of the RL agent. For DQN, since it can only handle discrete action space, the continuous action command needs to be discretized. To have more velocity options, the action space is defined as the combination of the linear acceleration $\{-1, 0, 1\}$ where 1 means full acceleration, 0 means no acceleration, and -1 means full deceleration, and angular acceleration $\{-1, 0, 1\}$ where 1 means turning right with maximum angular acceleration, 0 means no turning, and -1 means turning left with maximum angular acceleration. Overall, there are nine possible actions for the robot to choose from. The state space is divided into internal state $\boldsymbol{s}^{\text{int}}$ and external state $\boldsymbol{s}^{\text{ext}}$. The internal observation $\boldsymbol{s}^{\text{int}}$ contains the previous speed and angular velocity of the robot, the description of the closest point on the reference path, and the description of the upcoming nodes of the reference path. The external observation $\boldsymbol{s}^{\text{ext}}$ for the environment is obtained from the sensor data in practice. LiDAR and cameras are commonly used sensors for mobile robots. The LiDAR data is simulated by a ray-and-sector model, where several rays are clustered as a sector whose range is determined by the closest distance of rays in the sector. The camera data is simulated by a multi-channel image from a bird's eye view, which includes a distance map channel indicating each pixel's distance to the agent, and two occupancy map channels indicating the current and one past occupancy of each pixel. After training the DQN agent, it outputs Q-values of all action options, and the

action with the highest Q-value is chosen as the action of the robot during inference. To generate a local reference trajectory, the action is used as a tendency of the robot's motion. Specifically, the agent is simulated to run N steps. In the first step, the agent takes the original action, while for the rest steps, to improve stability, an extra constant reference speed is used instead, and the angular velocity decays over time. Then, a local reference trajectory is generated with N steps.

3.4 Proactive control

By perceiving others' motion intentions, humans can easily avoid each other in advance. Unlike traditional obstacle avoidance methods that only consider the current state of the obstacles, proactive control allows the robot to take evasive action in advance by introducing motion prediction of dynamic obstacles in the loop. The MPC controller can be naturally combined with motion prediction due to the predictive horizon. Assuming accurate motion prediction, a sufficient condition for an AMR to avoid a dynamic obstacle within the predictive horizon is that the robot avoids collisions with all potential obstacles at the corresponding time steps, i.e.,

$$\forall t_k \in \mathbb{N}_{[k,k+N]},\, m \in \mathbb{N}_{[1,M]}, \quad \boldsymbol{p}_{t_k} \notin^m \hat{\mathcal{D}}_{t_k}, \tag{3.11}$$

where ${}^m\hat{\mathcal{D}}_{t_k}$ is the predicted area occupied by the m-th potential mode of the obstacle at time step t_k and M is the number of potential modes. This is an over-conservative condition, especially when considering multimodal motion prediction. If too much uncertainty is considered, potential obstacles may occupy the entire drivable area and the robot can not find any feasible trajectory. Two strategies are proposed to balance conservatism and feasibility: using soft constraints rather than hard constraints to predict potential obstacle avoidance and using grouped obstacle predictions at each step rather than individual predictions.

The grouped prediction is achieved through unsupervised Clustering and Gaussian Fitting (CGF) [7], [18], [100], as shown in Figure 2.5, which uses DBSCAN [117] to find clusters of predicted position points of all dynamic obstacles based on their spatial proximity, then fit each cluster into a Gaussian distribution to obtain a GMM. As discussed in [100], CGF can mitigate the computation burden of the optimization solver and also alleviate the freezing

robot problem [85].

The overall optimization problem [100] can be formulated as (assuming the current time step is 0, omitting robot index i if there is no ambiguity)

$$\min_{\boldsymbol{u}_{0:N-1}} \sum_{k=0}^{N-1} \Bigg[J_R(k) + J_{\mathcal{O}}(\boldsymbol{p}_k) + J_{\mathcal{D}}(\boldsymbol{p}_k) + \sum_{j=1, j\neq i}^{n_r} J_F(\boldsymbol{p}_k^i, \boldsymbol{p}_k^j) \Bigg], \tag{3.12}$$

$$\text{s.t.} \quad \boldsymbol{c}_{k+1} = f(\boldsymbol{c}_k, \boldsymbol{u}_k),\ \forall k \in \mathbb{N}_{[0,N-1]}, \tag{3.13}$$

$$\boldsymbol{u}_k \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}],\ \forall k \in \mathbb{N}_{[0,N-1]}, \tag{3.14}$$

$$\dot{\boldsymbol{u}}_k \in [\dot{\boldsymbol{u}}_{\min}, \dot{\boldsymbol{u}}_{\max}],\ \forall k \in \mathbb{N}_{[0,N-1]}, \tag{3.15}$$

$$\boldsymbol{p}_k \notin \mathcal{O},\ \forall k \in \mathbb{N}_{[0,N-1]}, \tag{3.16}$$

$$\boldsymbol{p}_k \notin \mathcal{D}_k,\ \forall k \in \mathbb{N}_{[k,k+N_{\text{crit}}]}. \tag{3.17}$$

where

$$J_R(k) = ||\boldsymbol{c}_k - \tilde{\boldsymbol{c}}_k||^2_{\boldsymbol{Q}_c} + ||\boldsymbol{u}_k - \tilde{\boldsymbol{u}}_k||^2_{\boldsymbol{Q}_u} + ||\boldsymbol{u}_k - \boldsymbol{u}_{k-1}||^2_{\boldsymbol{Q}_a} \tag{3.18}$$

is the state and action constraint term with corresponding penalty weights ($\boldsymbol{Q}_c$, $\boldsymbol{Q}_u$, and $\boldsymbol{Q}_a$), and (let d_{fleet} is the safe distance between two robots)

$$J_F(\boldsymbol{p}_k^{(i)}, \boldsymbol{p}_k^{(j)}) = \max \left[0, \boldsymbol{Q}_f \cdot \left(d_{\text{fleet}} - || \boldsymbol{p}_k^{(i)} - \boldsymbol{p}_k^{(j)} || \right)^2 \right] \tag{3.19}$$

is the fleet collision avoidance term that will be further discussed in the next section.

3.5 Fleet control and coordination

While fleet coordination is not the main focus of this research, basic fleet collision avoidance methods are implemented and tested [100], [104], [118], this problem will be briefly discussed in this section.

For collision avoidance in multi-robot systems, the MPC problem can be formulated in three different ways, as outlined in Table 1.1. To achieve a balance between optimality and scalability, distributed MPC is identified as the most suitable approach for our application, particularly when a large and dynamically changing fleet of robots operates simultaneously with opportunities

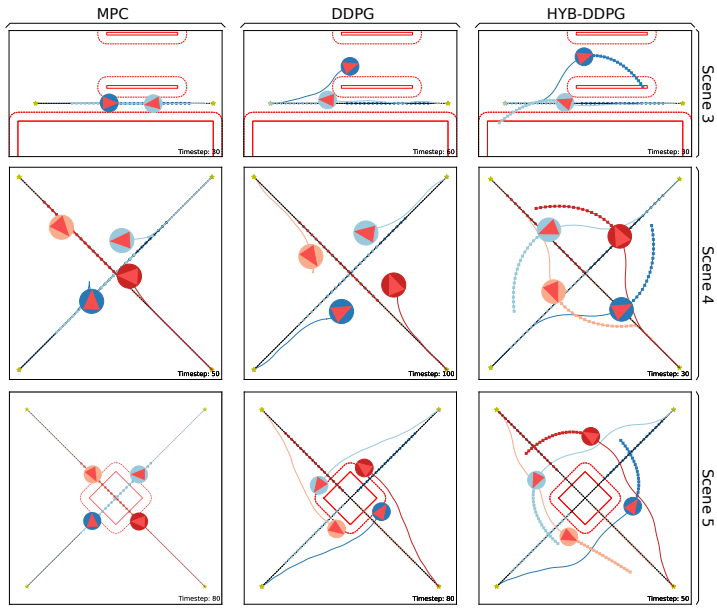

Figure 3.2: Comparison among MPC, DRL, and DRL-MPC hybrid methods in multi-agent cases [104].

for information exchange [118]. As shown in Eq. (3.19), the distributed MPC approach requires each robot to query the locations and predicted states of other robots within the planning horizon. While direct point-to-point communication is feasible, it may lead to excessive communication overhead as the number of robots increases. To mitigate this, the proposed pipeline incorporates a "robot manager" as an intermediary. This manager collects and stores the current and predicted states of all robots, allowing any robot to query information efficiently through the manager. This approach is also implemented in a ROS-based simulation [100], facilitating streamlined communication and coordination. While distributed MPC effectively resolves some fleet-level conflicts [118], it remains inherently constrained by the limitations of MPC itself. As illustrated in Figure 3.2, if multiple robots and static obstacles create nonconvex configurations, MPC struggles to find feasible solutions due to its convex optimization constraints. To address this limitation,

a DRL-MPC hybrid approach is introduced [19], [104]. This method follows a distributed-training-distributed-execution framework [119], where agents are trained in single-agent settings, treating other robots as dynamic obstacles during execution. During runtime, deep reinforcement learning (DRL) generates a reference trajectory based on inference, which is subsequently utilized by MPC for motion planning. This hybrid approach enhances collision avoidance capabilities, enabling more effective navigation in complex and highly dynamic environments.

Beyond local coordination, a high-level scheduler can further optimize fleet management by coordinating the execution of robot motion plans, thereby minimizing the risk of fleet-wide collisions [113]. While this aspect falls outside the scope of this research, integrating local coordination with a high-level scheduling system presents a promising direction for future work.

3.6 Reactive control in complex environment

While MPC is effective in handling predictive information and optimizing control commands under constraints, it faces challenges in real-time computation and handling highly complex obstacles. Conversely, CBFs [71] provide formal safety guarantees and computational efficiency, but traditional CBF methods struggle with nonconvex and highly concave obstacles. As part of a collaborative project[1], this research builds on the on-Manifold CBF (MCBF) framework [120], extending its applicability to complex dynamic environments by integrating motion prediction for proactive navigation. Compared to traditional CBF, which operates in Euclidean space and can be limited in handling complex obstacle geometries, MCBF incorporates an on-manifold approach [121] with geodesic approximation to provide effective, efficient, and feasible navigation around obstacles. This enables the controller to accurately manage highly intricate shapes, such as concave obstacles, which are challenging for conventional methods.

The prior MCBF framework [120] demonstrated strong performance in navigating complex scenarios, such as maneuvering around a C-shaped obstacle that encloses a mobile robot. However, one major challenge remained: the critical parameters for geodesic approximation required manual tuning for

[1] This section is based on the collaboration with Yifan Xue and Asst. Prof. Nadia Figueroa from the University of Pennsylvania.

different obstacle geometries, which could limit its adaptability in dynamic environments. To address this, the collaborative project introduces two major extensions:

- Integrating multimodal motion prediction into the MCBF pipeline to enable proactive obstacle avoidance rather than purely reactive behavior.
- Automating adaptive parameter tuning for geodesic approximation, ensuring that MCBF remains effective across diverse and evolving environments without requiring manual adjustments.

These enhancements significantly improve the robustness, adaptability, and practicality of MCBF, making it a more viable solution for collision-free navigation in complex, dynamic settings.

Gaussian Process Distance Field. As an unparameterized probabilistic approach to modeling the environment, Gaussian Process Distance Fields (GPDF) [122], [123] provide distance and gradient information from all detected obstacles in a continuous and differentiable form. For a set of points $\mathcal{P} = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_M\}$ representing obstacles, each point can be modeled as a Gaussian distribution. Define a latent field $o(\boldsymbol{p})$ and the inverse function f_{inv} mapping to the distance field $f_{\text{dist}}(\boldsymbol{p})$,

$$f_{\text{inv}}\left(k_o(\boldsymbol{p}, \mathcal{P})\right) \coloneqq ||\boldsymbol{p} - \mathcal{P}||, \tag{3.20}$$

$$f_{\text{dist}}(\boldsymbol{p}) = f_{\text{inv}}(o(\boldsymbol{p})), \tag{3.21}$$

$$o(\boldsymbol{p}) \sim \mathcal{G}(0, k_o\left(\boldsymbol{p}, \mathcal{P}\right)), \tag{3.22}$$

where $\mathcal{G}$ represents Gaussian process and k_o is the covariance kernel function. The latent field can be regarded as a smooth occupancy field. According to Gaussian process regression,

$$\bar{o}(\boldsymbol{p}) = k_o(\boldsymbol{p}, \mathcal{P}) \underbrace{\left(K_o(\mathcal{P}, \mathcal{P}) + \sigma_o^2 \mathbf{I}\right)^{-1} \cdot \mathbf{1}}_{\alpha(\mathcal{P}, \sigma_o)}, \tag{3.23}$$

$$\begin{aligned} \mathbf{cov}(o(\boldsymbol{p})) = & \, k_o(\boldsymbol{p}, \boldsymbol{p}) \\ & - k_o(\boldsymbol{p}, \mathcal{P}) \left(K_o(\mathcal{P}, \mathcal{P}) + \sigma_o^2 \mathbf{I}\right)^{-1} k_o(\mathcal{P}, \boldsymbol{p}), \end{aligned} \tag{3.24}$$

where $K_o(\mathcal{P}, \mathcal{P})$ is the covariance kernel of the given points, σ_o is the noise covariance, $\mathbf{1}$ is a vector of ones (let $k_o(\mathcal{P}, \mathcal{P}) = \mathbf{1}$), and $\alpha(\cdot)$ is the Gaussian process model. The gradient of the distance field can be derived as

$$\begin{aligned} f_{\text{grad}}(\boldsymbol{p}) &= \nabla_{\boldsymbol{p}} f_{\text{dist}}(\boldsymbol{p}) \\ &= \frac{\partial f_{\text{inv}}}{\partial o} \cdot \left[\nabla_{\boldsymbol{p}} k_o(\boldsymbol{p}, \mathcal{P}) \cdot \alpha(\mathcal{P}, \sigma_o)\right]. \end{aligned} \tag{3.25}$$

In GPDFs, the distance field provides information about how far a point is from the surrounding obstacles, while the gradient field encodes both the direction and rate of change of the distance to obstacles with respect to the robot's position. These properties of GPDFs make them suitable to integrate with the control algorithm.

On-Manifold Control Barrier Function. To define safe control, Control Barrier Functions (CBFs) [124] are proposed to introduce the definition of safe control from the perspective of keeping the dynamic system inside a safe set, such as a configuration set without any collisions with obstacles. In general, A CBF defines a safe set via a continuously differentiable function $h(\boldsymbol{x})$, given $\boldsymbol{x}$ as the state vector, where

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^n | h(\boldsymbol{x}) \leq 0\}. \tag{3.26}$$

The set $\mathcal{C}$ is referred to as the safe set. To keep the system in $\mathcal{C}$, a constraint on the derivative can be enforced,

$$\dot{h}(\boldsymbol{x}, \boldsymbol{u}) \leq \alpha h(\boldsymbol{x}), \tag{3.27}$$

where $\boldsymbol{u}$ is the control input for the system. Note that $\dot{h}$ contains $\dot{\boldsymbol{x}}$, which is related to the control input $\boldsymbol{u}$ as well. For a nonlinear control-affine system, this can be transferred as a linear inequality in the control input [124]. Given a nominal control input, the safe control problem can be solved through Quadratic Programming (QP), and the whole method is called CBF-QP. While CBF-QP is designed for safe control, it cannot handle local minima or saddle points [120]. To address this issue, a modulated version utilizing the on-manifold method [125] is introduced in [120]. The basic idea of this On-Manifold CBF (MCBF) method is to modulate the dynamics and project it onto the tangent planes of the obstacle barrier functions.

One problem is to determine the correct direction of projection. For example, in 2D cases, the dynamics can be projected to either the left or the right direction, which means the robot can choose to turn left or right in front of an obstacle. In [120], the direction is selected based on a geodesic approximation strategy, which is to iteratively move along the obstacle surface or the isoline according to the relative position of the robot with respect to the obstacle, and then choose the direction leading to the desired goal position (for details, see [120]). While this method is effective, the distance to move along the obstacle surface or isoline requires manual selection through tuning a step size parameter.

In this work, we proposed an adaptive MCBF in cooperation with multimodal motion prediction for proactive and local-minima-free obstacle avoidance, named the MMP-MCBF framework. First of all, instead of hand-tuning the parameter, an automatic parameter selection algorithm (see *Paper E*) is proposed so that MCBF can work in environments where the shapes of unsafe regions may change over time due to the movement of dynamic obstacles. The MMP-MCBF approach follows the same structure as the integration of motion prediction and MPC, which is that a motion predictor generates motion predictions of dynamic obstacles, and a controller then takes the motion prediction results to generate proper actions. However, there is a major difference. Unlike MPC, CBF considers the unsafe region as a whole set, so motion predictions over multiple predictive time steps are summed to form a complete unsafe region. The consequence is that the overall unsafe region is normally nonconvex, especially with multimodal motion prediction. To generate a smooth distance field that can be converted into barrier functions for CBF, unsafe regions are modeled by GPDFs by sampling their boundary points. Note that MMP-MCBF is able to handle non-convex obstacles but also create more non-convex obstacles since it considers predictions over all time steps as a total unsafe region.

In the simulated scenarios and real-world experiments, this MMP-MCBF framework is shown to be safer and more efficient compared to other methods, including MPC with motion prediction, CBF with motion prediction, and MCBF without motion prediction, especially with the presence of nonconvex obstacles.

CHAPTER 4

Summary of included papers

This chapter provides a summary of the included papers. These papers have been chosen to represent the research contributions while ensuring minimal overlap in theoretical foundations and content.

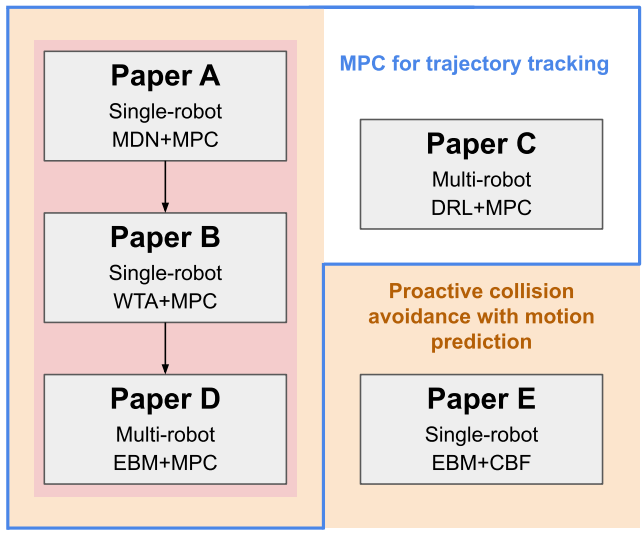

Figure 4.1: The roadmap of this thesis is structured around the attached papers. The red block (Papers A, B, and C) represents the primary research trajectory, focusing on the exploration of multimodal motion prediction and its integration with Model Predictive Control (MPC) for proactive dynamic obstacle avoidance. The blue-bordered block extends the application of MPC-based trajectory tracking by incorporating a Deep Reinforcement Learning (DRL) heuristic for reference trajectory generation. The orange block builds upon the integration of learning-based motion prediction with optimization-based trajectory tracking, replacing MPC with a Control Barrier Function (CBF)-based method to handle obstacles with more complex geometries. Motion prediction models are Mixture Density Network (MDN), Winner-Takes-All (WTA) loss-based model, and Energy-Based Model (EBM).

	Paper A	Paper B	Paper C	Paper D	Paper E
RQ1	✓	✓		✓	
RQ2	✓	✓		✓	✓
RQ3	✓	✓		✓	
RQ4	✓	✓	✓	✓	✓

Table 4.1: The coverage of the research questions is mapped to the attached papers, demonstrating how each study contributes to addressing the key research objectives.

4.1 Paper A

Z. Zhang, E. Dean, Y. Karayiannidis, and K. Åkesson
Motion Prediction Based on Multiple Futures for Dynamic Obstacle Avoidance of Mobile Robots
IEEE International Conference on Automation Science and Engineering (CASE), Lyon, France, 2021, pp. 475-481

To explore the possibility of combining deep learning methods with traditional optimization-based methods and solutions to dynamic obstacle avoidance problems in industrial environments, this paper introduces a collision-free mobile robot navigation pipeline and an industrial perception system consisting of a grid of ceiling-mounted top-view cameras. The vision system gathers environmental information, such as locations and identifications of static and dynamic obstacles. Trajectories of dynamic obstacles are sent to a Mixture Density Network (MDN), which outputs motion predictions of the obstacles in the form of Gaussian Mixture Models (GMMs). Geometrically, GMMs are ellipses and can be formulated as nonlinear constraints in the Model Predictive Control problem. The receding horizon feature of MPC makes it natural to handle motion predictions of dynamic obstacles, so mobile robots can take proactive actions to better avoid dynamic obstacles. The use cases in this paper display that the proposed approach provides valid predictions of motion

in a dynamic environment and has the capability to avoid dynamic obstacles in advance.

My contribution to this paper includes conceptualization, methodology development, programming, result validation, writing, editing, visualization, etc. The motion prediction part was implemented by me. The MPC controller was based on another work [112] that I supervised. I designed the experiments and all the figures. The manuscript was initially drafted by me, and after incorporating feedback from co-authors and reviewers, I revised and finalized the paper.

The individual contributions of each author are outlined as follows.

- **Z. Zhang** (Main) worked on the ideas, method development, programming, result evaluation, writing and editing, visualization, etc.
- E. Dean (Conceptualization | Review) was involved in discussing the research idea and reviewing the manuscript.
- Y. Karayiannidis (Conceptualization | Review) was involved in discussing the research idea and reviewing the manuscript.
- K. Åkesson (Conceptualization | Review | Supervision | Project administration | Funding acquisition) provided supervision on both the research and the project funding the research activities.

4.2 Paper B

Z. Zhang, H. Hajieghrary, E. Dean, and K. Åkesson
Prescient Collision-Free Navigation of Mobile Robots with Iterative Multimodal Motion Prediction of Dynamic Obstacles
IEEE Robotics and Automation Letters (RA-L), vol. 8, no. 9, 2023, pp. 5488-5495

This paper introduces an integrated approach for collision-free navigation of autonomous mobile robots in dynamic and uncertain environments. The

method combines multimodal motion prediction, utilizing enhanced Winner-Takes-All (WTA) loss functions to predict dynamic obstacles, with Model Predictive Control (MPC) to incorporate these predictions into real-time motion planning. Following motion prediction by the neural network, an unsupervised technique—clustering and Gaussian fitting—is employed to generate geometric representations of obstacles, which are then used to formulate avoidance constraints within the MPC framework. The proposed approach demonstrates superior performance compared to navigation strategies that either disregard motion prediction or rely on non-learning-based motion prediction methods.

My contribution to this paper includes conceptualization, methodology development, programming, result validation, writing, editing, visualization, etc. The motion prediction part was implemented by me. The MPC controller was based on another work [112] that I supervised. I designed the Python experiments and all the figures except for the ones related to the Gazebo simulation. The manuscript was mostly written by me except for the ROS implementation, and after collecting opinions from co-authors and reviews, I revised and edited the final version of the paper.

The individual contributions of each author are outlined as follows.

- **Z. Zhang** (Main) worked on the ideas, method development, programming (algorithm and Python), result evaluation, writing and editing, visualization, etc.

- H. Hajieghrary (Programming | Visualization | Review) worked on the programming (ROS, Gazebo simulation) and visualization of the Gazebo simulation, and was involved in discussing the research idea and reviewing the manuscript.

- E. Dean (Conceptualization | Review) was involved in discussing the research idea and reviewing the manuscript.

- K. Åkesson (Conceptualization | Review | Supervision | Project administration | Funding acquisition) provided supervision on both the research and the project funding the research activities.

4.3 Paper C

K. Ceder*, **Z. Zhang***, A. Burman, I. Kuangaliyev, K. Mattsson, G. Nyman, A. Petersén, L. Wisell, and K. Åkesson
Bird's-Eye-View Trajectory Planning of Multiple Robots using Continuous Deep Reinforcement Learning and Model Predictive Control
IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Abu Dhabi, UAE, 2024, pp. 8002-8008, *denotes equal contribution,

This paper introduces a novel method for efficient motion planning and control of multiple mobile robots, which uses a learning-based algorithm for reference generation and Model Predictive Control (MPC) for trajectory tracking. Deep Reinforcement Learning (DRL) agents trained via the Deep Deterministic Policy Gradient (DDPG) algorithm are utilized to generate reference trajectories for mobile robots. The learning ability and fast inference of the DRL make it suitable for real-time navigation tasks in complex environments, while MPC produces stable and optimized action based on the given reference. By combining DRL and MPC, the overall computation time is shortened and robots are able to handle more complex environments (such as nonconvex obstacles) compared to pure MPC, and the stability is enhanced compared to pure learning-based method.

This work extends the original single-robot setting to multi-robot applications by leveraging DDPG agent training only on single-agent environments with distributed MPC. As a Decentralized Training with Decentralized execution (DTDE) approach, the training process maintains the same training process as single-agent learning, ensuring efficiency and scalability. This multi-agent extension exhibits emergent cooperative behaviors among mobile robots and demonstrates superior performance in handling complex scenarios compared to standalone DRL or MPC approaches.

My contribution to this paper includes conceptualization, methodology

development, programming, result validation, writing, editing, visualization, etc. I initiated this project and proposed the research idea [19]. The MPC implementation and multi-agent part were implemented by me. I designed the experiments for the multi-agent part and the other experiments are based on my previous paper [19]. All the figures were designed and generated by me. The manuscript was partially written by me (MPC and multi-agent parts), and after collecting opinions from co-authors and reviews, I revised and edited the final version of the paper.

The individual contributions of each author are outlined as follows (*K. Ceder and Z. Zhang have each contribution).

- K. Ceder* (Main) worked on the ideas, method development, programming, result evaluation, writing and editing, visualization, etc. The major work includes single-agent training, implementation, and evaluation. The majority of the paper, except for the MPC and multi-agent parts, is written by K. Ceder.

- **Z. Zhang*** (Main) worked on the ideas, method development, programming, result evaluation, writing and editing, visualization, etc. The major work includes multi-agent implementation and evaluation, as well as DRL-MPC integration. The MPC and multi-agent parts of the paper are written by Z. Zhang. Z. Zhang initiated this project.

- A. Burman, I. Kuangaliyev, K. Mattsson, G. Nyman, A. Petersén, and L. Wisell (Programming | Visualization | Writing) worked on the programming (training of DDPG agents), and were involved in discussing the research process of DDPG training and implementation. They wrote a report on the DDPG training and implementation, which was used as a reference for the final publication.

- K. Åkesson (Review | Supervision | Project administration) provided supervision on the research project.

4.4 Paper D

Z. Zhang, G. Hess, J. Hu, E. Dean, L. Svensson, and K. Åkesson
Future-Oriented Navigation: Dynamic Obstacle Avoidance with One-

Shot Energy-Based Multimodal Motion Prediction
Under Review.

The paper presents an integrated approach for collision-free navigation of multiple autonomous mobile robots in dynamic and uncertain environments, such as warehouses. The approach combines:

- Multimodal Motion Prediction using an Energy-Based Learning (EBL) model to predict dynamic obstacles,
- Model Predictive Control (MPC) to incorporate these predictions into real-time motion planning.

The method groups predicted obstacles based on proximity to improve efficiency and mitigate the freezing robot problem, where a robot halts unnecessarily due to uncertainty. The proposed system is validated across industrial-like warehouse scenarios, showing superior performance over existing obstacle avoidance techniques.

The main contributions include proposing an energy-based negative log-likelihood loss function, integrating the one-shot energy-based multimodal motion prediction with MPC controllers, and extensively validating the proposed method in a warehouse environment.

My contribution to this paper includes conceptualization, methodology development, programming, result validation, writing, editing, visualization, etc. The motion prediction part was implemented by me. The code pipeline was based on my preceding work [18]. I designed the Python and ROS2 experiments and all the figures. The manuscript was written by me, and after collecting opinions from co-authors and reviews, I revised and edited the final version of the paper.

The individual contributions of each author are outlined as follows.

- **Z. Zhang** (Main) worked on the ideas, method development, programming, result evaluation, writing and editing, visualization, etc.
- G. Hess (Programming | Review) worked on the programming (improving the deep learning training process).
- J. Hu (Programming) worked on the programming (regulated pure pursuit control algorithm and part of the motion prediction function).

- E. Dean (Review) was involved in reviewing the manuscript.
- L. Svensson (Conceptualization | Supervision) co-supervised this paper and provided guidance for the deep learning part.
- K. Åkesson (Conceptualization | Review | Supervision | Project administration | Funding acquisition) provided supervision on both the research and the project funding the research activities.

4.5 Paper E

Y. Xue*, **Z. Zhang***, K. Åkesson, and N. Figueroa
Proactive Local-Minimum-Free Mobile Robot Navigation using MMP-MCBF Framework
To be submitted for possible publication, *denotes equal contribution.

In this work, we proposed a safe and efficient mobile robot navigation pipeline, named MMP-MCBF, based on the integration of motion predictions of dynamic obstacles and an improved on-manifold control barrier function method, which enables mobile robots to avoid dynamic obstacles proactively. To consider motion prediction in the control barrier function framework, we further improved the on-manifold CBF algorithm for adaptively and automatically parameter searching. Apart from the traditional CVM-based prediction, a learning-based EBM motion predictor is investigated in this pipeline to consider multimodal motion prediction for more comprehensive and flexible prediction. From both simulated scenarios and real-world experiments, the proposed MMP-MCBF approach outperforms other popular obstacle avoidance approaches, shown to be safe and efficient in complex and dynamic environments, especially with the presence of non-convex static or dynamic obstacles.

The individual contributions of each author are outlined as follows (*Z. Zhang and Y. Xue have equal contributions).

- **Z. Zhang*** (Main) worked on the ideas, method development, programming (improving MCBF, adding motion prediction into the pipeline), investigation (simulation data collection and neutral network training), result evaluation, writing and editing, visualization, etc. Z. Zhang initiated this project.

- Y. Xue* (Main) worked on the ideas, method development, programming (developing the MCBF code and improving the integration), investigation (leading the real-world experiments), result evaluation, writing and editing, visualization, etc.
- K. Åkesson (Review | Supervision | Funding acquisition) provided supervision and helped to acquire funding for this collaboration.
- N. Figueroa (Conceptualization | Review | Supervision | Resources) provided supervision and relevant experimental conditions. N. Figueroa hosted this collaboration.

4.6 Contributions

The contributions of my research are directly connected to the aforementioned research questions. A coverage mapping from selected papers to research questions is shown in Table 4.1.

Research Question 1: *What are the key design principles and architectural components required to develop an effective and scalable Factory with Vision framework for industrial applications?*
Contribution 1: As discussed in Papers A, B, and D, the Factory with Vision framework is designed based on three key principles: effective modular components, coherent software interfaces, and practical input/output design tailored to real-world conditions. The primary hardware components include cameras, mobile robots, and various communication and computing devices, while the software components encompass image processing, environmental perception and reasoning, motion planning, and motion control. Modularization is essential for scalability and maintainability, ensuring that individual components can be upgraded or replaced without disrupting the overall system. Software inputs should be directly accessible from physical sensors or derived from sensor outputs, while software outputs must be executable by the corresponding hardware and actuators. Smooth and coherent interfaces between different components are crucial for system integration. For instance, motion prediction outcomes must seamlessly integrate with the motion planning and control module. To facilitate this, the clustering and Gaussian fitting method was developed to transform motion prediction results into a form suit-

able for downstream planning and control processes.

Research Question 2: *How can learning-based motion prediction techniques be integrated with optimization-based control methods to improve the safety and efficiency of AMRs in dynamic environments?*
Contribution 2: According to Papers A, B, D, and E, integrating motion prediction with control requires consideration of two key aspects: (1) whether the motion prediction outcomes can be effectively utilized by the control method, and (2) whether the chosen control approach has the capability to incorporate predictive information. In Papers A, B, and D, MPC is selected due to its receding horizon feature, which allows it to naturally incorporate predictive information. With a predictive horizon, MPC can utilize motion prediction at each corresponding time step within the horizon, enabling proactive obstacle avoidance. In contrast, Paper E employs a CBF-based method instead of MPC. Since CBFs do not inherently support predictive information, motion predictions over all time steps are aggregated into a single obstacle representation. This approach ensures that control constraints remain feasible while incorporating prediction. Regardless of the control method, motion prediction outcomes must be transformed into geometric representations that can be processed by optimization algorithms. To achieve this, the CGF method is introduced to convert neural network-based predictions into structured geometric constraints suitable for control. Furthermore, as highlighted in Papers B and D, the selection of motion prediction techniques should be informed by their compatibility with downstream planning and control tasks. Overly conservative predictions that encompass excessively large areas may lead to overly cautious behavior, reducing efficiency. Thus, motion prediction should be tailored to strike a balance between safety and feasibility, ensuring that predictions provide actionable and practical information for navigation and control.

Research Question 3: *Which motion prediction methods are suitable and efficient for capturing the uncertainty in human behavior and how can they provide actionable insights for downstream planning and control tasks?*
Contribution 3: In Papers A, B, and D, various motion prediction methods are proposed and evaluated to determine their suitability for capturing the uncertainty in human behavior while providing actionable insights for down-

stream planning and control tasks. Paper A uses Mixture Density Models (MDNs) to generate GMMs directly. However, MDNs are subject to mode collapse with high-dimension data space and cannot handle image inputs. Paper B introduces an improved Winner-Takes-All (WTA) loss function to enhance the accuracy and diversity of multimodal motion prediction, ensuring that predicted motions better reflect real-world human motion variations. Because WTA works with convolutional neural networks, it can be used with image inputs and the model can encode environmental information. Paper D further advances the motion prediction framework by incorporating energy-based learning, which refines predictive accuracy while maintaining computational efficiency for real-time applications. To ensure compatibility with control algorithms, motion prediction outcomes are transformed into structured geometric representations using the CGF method. Additionally, Papers B and D discuss the importance of balancing prediction accuracy and conservativeness. While capturing uncertainty is essential for robust motion planning, excessively broad predictions may lead to over-conservative navigation behavior, reducing efficiency. The proposed methods strive to optimize this trade-off, ensuring that predictions are both informative and feasible for real-world deployment in autonomous navigation.

Research Question 4: *What planning and control strategies can be employed to ensure safe, stable, efficient, and flexible navigation of AMRs in hybrid environments?*

Contribution 4: This research question is addressed across all selected papers. Papers A to D employ Model Predictive Control (MPC) for collision-free navigation, leveraging MPC's capability to incorporate predictive information and enforce constraints. Additionally, Paper C and its preceding work propose a Deep Reinforcement Learning (DRL)-MPC hybrid approach, where DRL is used to generate a reference trajectory that MPC subsequently follows. This combination integrates the strengths of both methods: DRL excels in navigating complex obstacles through learning, while MPC provides stability and optimized behavior by ensuring constraint satisfaction and smooth trajectory execution. Paper E explores an alternative to MPC by testing the proactive collision avoidance pipeline using Control Barrier Functions (CBFs). Specifically, it employs an improved variant known as on-Manifold CBF (MCBF), which enables smooth handling of concave obstacles. Unlike MPC, CBF-based

methods do not inherently incorporate predictive information; however, they guarantee non-penetration of obstacles and generally offer faster computation compared to MPC. While the combination of motion prediction and MCBF results in more conservative behavior—due to the CBF's inability to explicitly handle predictions—the integrated approach demonstrates strong performance in navigating nonconvex dynamic obstacles. This highlights the trade-off between computational efficiency, predictive adaptability, and constraint satisfaction when selecting control strategies for proactive obstacle avoidance in dynamic environments.

CHAPTER 5

Concluding Remarks and Future Work

This thesis investigates the development of multimodal motion prediction techniques for dynamic obstacles, which are designed to enhance mobile robot navigation algorithms for proactive collision avoidance. Additionally, it explores motion planning and control strategies tailored for mobile robots operating in complex environments. The core objective is to integrate learning-based approaches with optimization-based controllers to enable safe, efficient, stable, and intelligent navigation, particularly in industrial settings. A portion of this research has been conducted in collaboration with Volvo GTO, with a specific focus on implementing the Factory with Vision framework to improve robotic perception and decision-making in manufacturing environments.

The primary research trajectory of this thesis focuses on integrating deep learning-based multimodal motion prediction with MPC-based mobile robot controllers to enable proactive dynamic obstacle avoidance. As demonstrated in *Papers A*, *B*, and *D*, the proposed approach outperforms conventional obstacle-avoidance strategies in terms of both safety and efficiency. In *Paper B* and its preceding work, an improved winner-takes-all loss function is proposed to enhance the multimodality of motion prediction. In *Paper D*, a novel energy-based learning strategy is proposed to refine motion prediction.

Paper C addresses real-time performance and the capability of the motion planner and controller to handle complex, nonconvex obstacles. By employing deep reinforcement learning to generate reference trajectories for MPC, the proposed hybrid approach shows improved real-time performance and a higher success rate in avoiding complex obstacles. This hybrid approach is also extended to multi-agent cases in *Paper C* through distributed training and execution fashion. *Paper E* explores alternatives to MPC for managing more complex navigation scenarios, such as avoiding highly concave obstacles. As part of a collaborative project, this work builds upon the on-manifold Control Barrier Function (CBF) method, introducing two key modifications: adaptive parameter tuning for dynamic environments and motion prediction for dynamic obstacles. The proposed approach is demonstrated to be adaptive, flexible, and robust, enabling mobile robots to navigate complex dynamic environments with concave obstacles effectively.

In summary, this research investigates proactive dynamic obstacle avoidance for mobile robots by exploring the integration of learning-based methods for environmental perception and reasoning with traditional optimization-based control techniques. The objective is to achieve a balance between adaptability, computational efficiency, and theoretical guarantees in mobile robot navigation. The contributions of this thesis present a comprehensive framework that seamlessly integrates motion prediction, planning, and control, with promising applications in industrial automation, smart manufacturing, and human-robot collaboration.

Future work includes further refinement of learning-based motion prediction models, development of adaptive control strategies for dynamic environments, and real-world deployment and validation in large-scale industrial applications. Particularly,

- Enhancing Motion Prediction Stability: From *Papers A* and *B* to *Paper D*, the motion prediction methodology has evolved in terms of visual scene understanding, faster and more stable inference, and providing probabilistic information. However, experimental results, particularly those involving human motion noise, indicate that motion predictions are not consistently stable. Due to the inherent uncertainty in human motion, sudden changes in predicted constraints can lead to infeasibility issues in control algorithms. While such uncertainty and motion noise are unavoidable, implementing

a memory mechanism in motion prediction could enable predictions to be made correlatively over time, thereby improving stability.

- Integrating a Unified Navigation Framework: A more cohesive pipeline that integrates the approaches from *Papers C* and *D* can be developed to achieve more flexible and intelligent navigation within a fleet of mobile robots.

- Expanding Real-World Deployment: While *Paper E* includes some real-world experiments, further implementation and validation of the proposed approach in industrial settings are beneficial. Additional experiments will help assess its practical feasibility and contribute to increasing automation in factory environments.

- Addressing the Trade-Off Between Safety and Efficiency: Observations from *Papers D* and *E* highlight the significant impact of dynamic obstacle uncertainty on control strategies. Striking a balance between safe yet overly conservative navigation and efficient but potentially less stable navigation remains a critical challenge. Future research should explore both technical advancements and philosophical perspectives to address this trade-off effectively.

These potential research directions influence both academic advancements and industrial applications in the field. While the primary motivation of this research has been to bridge the gap between motion prediction and downstream planning and control, this work represents one step toward more autonomous and intelligent industrial transportation solutions, with the potential for applications in broader everyday scenarios. Although the complete realization of the *Factory with Vision* framework remains a long-term goal, it is my sincere hope that this thesis contributes to and inspires further research in this evolving field.

References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots (2nd Edition)*. England: MIT Press, 2011.

[2] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg, "A survey of research on cloud robotics and automation," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 2, pp. 398–409, 2015.

[3] A. Milner and M. Goodale, "Two visual systems re-viewed," *Neuropsychologia*, vol. 46, no. 3, pp. 774–785, 2008, ISSN: 0028-3932.

[4] M. P. Groover, *Automation, Production Systems, and Computer-integrated Manufacturing*. USA: Pearson, 2007.

[5] M. Wise, M. Ferguson, D. King, E. Diehr, and D. Dymesich, "Fetch and freight: Standard platforms for service robot applications," in *Workshop on autonomous mobile service robots*, 2016, pp. 1–6.

[6] R. Hercik, R. Byrtus, R. Jaros, and J. Koziorek, "Implementation of autonomous mobile robot in smartfactory," *Applied Sciences*, vol. 12, no. 17, p. 8912, 2022.

[7] Z. Zhang, E. Dean, Y. Karayiannidis, and K. Åkesson, "Motion prediction based on multiple futures for dynamic obstacle avoidance of mobile robots," in *International Conference on Automation Science and Engineering*, 2021, pp. 475–481.

[8] N. Correll, B. Hayes, C. Heckman, and A. Roncone, *Introduction to autonomous robots: mechanisms, sensors, actuators, and algorithms*. Mit Press, 2022.

[9] S. A. M. Saleh, S. A. Suandi, H. Ibrahim, Q. S. Hamad, and I. A. Amoudi, "AGVs and AMRs robots: A brief overview of the differences and navigation principles," in *International Conference on Robotics, Vision, Signal Processing and Power Applications*, Springer, 2021.

[10] J. Zhang, X. Yang, W. Wang, J. Guan, L. Ding, and V. C. Lee, "Automated guided vehicles and autonomous mobile robots for recognition and tracking in civil engineering," *Automation in Construction*, vol. 146, p. 104 699, 2023.

[11] E. Kruse and F. M. Wahl, "Camera-based monitoring system for mobile robot guidance," in *International Conference on Intelligent Robots and Systems*, IEEE, vol. 2, 1998, pp. 1248–1253.

[12] M. Scholz, X. Zhang, and J. Franke, "Implementation of an intralogistics routing-service basing on decentralized workspace digitization," *Applied Mechanics and Materials*, vol. 882, pp. 90–95, 2018.

[13] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. M. Montiel, and J. D. Tardós, "ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[14] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 1013–1020.

[15] G. Hoffman, T. Bhattacharjee, and S. Nikolaidis, "Inferring human intent and predicting human action in human-robot collaboration," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2024.

[16] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robotics and Autonomous Systems*, vol. 100, pp. 171–185, 2018.

[17] A. Gasparetto, P. Boscariol, A. Lanzutti, and R. Vidoni, "Path planning and trajectory planning algorithms: A general overview," *Motion and operation planning of robotic systems: Background and practical approaches*, pp. 3–27, 2015.

[18] Z. Zhang, H. Hajieghrary, E. Dean, and K. Åkesson, "Prescient collision-free navigation of mobile robots with iterative multimodal motion prediction of dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 8, no. 9, pp. 5488–5495, 2023.

[19] Z. Zhang, Y. Cai, K. Ceder, *et al.*, "Collision-free trajectory planning of mobile robots by integrating deep reinforcement learning and model predictive control," in *International Conference on Automation Science and Engineering (CASE)*, 2023, pp. 1–7.

[20] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *Journal of Intelligent & Robotic Systems*, vol. 84, pp. 829–858, 2016.

[21] S. F. Roselli, "Conflict-free routing of mobile robots," Ph.D. dissertation, Chalmers Tekniska Hogskola (Sweden), 2022.

[22] P. Forte, A. Mannucci, H. Andreasson, and F. Pecora, "Online task assignment and coordination in multi-robot fleets," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4584–4591, 2021.

[23] F. Bertilsson, M. Gordon, J. Hansson, *et al.*, "Centralized versus distributed nonlinear model predictive control for online robot fleet trajectory planning," in *International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 701–706.

[24] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: A survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, 2020.

[25] C. Cao, P. Trautman, and S. Iba, "Dynamic channel: A planning framework for crowd navigation," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 5551–5557.

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

[27] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.

[28] A. Robicquet, A. Sadeghian, A. Alahi, and S. Savarese, "Learning social etiquette: Human trajectory understanding in crowded scenes," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14*, Springer, 2016, pp. 549–565.

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

[30] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," in *International Conference on Robotics and Automation*, IEEE, 2008, pp. 1928–1935.

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

[32] S. Kim, S. J. Guy, W. Liu, *et al.*, "BRVO: Predicting pedestrian trajectories using velocity-space reasoning," *The International Journal of Robotics Research*, vol. 34, no. 2, pp. 201–217, 2015.

[33] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[34] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[35] O. Makansi, E. Ilg, O. Cicek, and T. Brox, "Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[36] K. Mangalam, Y. An, H. Girase, and J. Malik, "From goals, waypoints & paths to long term human trajectory forecasting," in *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 15 213–15 222.

[37] D. Ridel, N. Deo, D. Wolf, and M. Trivedi, "Scene compliant trajectory forecast with agent-centric spatio-temporal grids," *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2816–2823, 2020.

[38] Z. Zhang, E. Dean, Y. Karayiannidis, and K. Åkesson, "Multimodal motion prediction based on adaptive and swarm sampling loss functions for reactive mobile robots," in *International Conference on Automation Science and Engineering (CASE)*, 2022, pp. 1110–1115.

[39] L. Fang, Q. Jiang, J. Shi, and B. Zhou, "TPNet: Trajectory proposal network for motion prediction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 6797–6806.

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

[41] L. F. Chiara, P. Coscia, S. Das, S. Calderara, R. Cucchiara, and L. Ballan, "Goal-driven self-attentive recurrent networks for trajectory prediction," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 2518–2527.

[42] C. Yang, H. Pan, W. Sun, and H. Gao, "Social self-attention generative adversarial networks for human trajectory prediction," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 4, pp. 1805–1815, 2024.

[43] K. Guo, W. Liu, and J. Pan, "End-to-end trajectory distribution prediction based on occupancy grid maps," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 2232–2241.

[44] M. Geng, Z. Cai, Y. Zhu, X. Chen, and D.-H. Lee, "Multimodal vehicular trajectory prediction with inverse reinforcement learning and risk aversion at urban unsignalized intersections," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 11, pp. 12 227–12 240, 2023.

[45] C. M. Bishop, "Mixture density networks," Neural Computing Research Group, Aston University, Tech. Rep. NCRG/94/004, 1994.

[46] C. Rupprecht, I. Laina, R. DiPietro, *et al.*, "Learning in an uncertain world: Representing ambiguity through multiple hypotheses," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[47] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[48] B. Ivanovic, K. Leung, E. Schmerling, and M. Pavone, "Multimodal deep generative models for trajectory prediction: A conditional variational autoencoder approach," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 295–302, 2021.

[49] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI*, Springer, 2015, pp. 234–241.

[50] E. Rehder and H. Kloeden, "Goal-directed pedestrian prediction," in *International Conference on Computer Vision Workshops*, 2015, pp. 50–58.

[51] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 2543–2549.

[52] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" In *CVPR*, 2011, pp. 1345–1352.

[53] B. Zhang, T. Wang, C. Zhou, N. Conci, and H. Liu, "Human trajectory forecasting using a flow-based generative model," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105 236, 2022.

[54] I. Bae, Y.-J. Park, and H.-G. Jeon, "Singulartrajectory: Universal trajectory predictor using diffusion model," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 17 890–17 901.

[55] C. Mavrogiannis, F. Baldini, A. Wang, *et al.*, "Core challenges of social robot navigation: A survey," *ACM Transactions on Human-Robot Interaction*, vol. 12, no. 3, pp. 1–39, 2023.

[56] L. Hua, H. Dourra, and G. G. Zhu, "Route and speed co-optimization for improving energy consumption of connected vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 29, no. 4, pp. 2830–2838, 2024.

[57] S. A. Mohamed, M.-H. Haghbayan, A. Miele, O. Mutlu, and J. Plosila, "Energy-efficient mobile robot control via run-time monitoring of environmental complexity and computing workload," in *International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 7587–7593.

[58] J. Z. Ben-Asher and E. D. Rimon, "Time optimal trajectories for a car-like mobile robot," *IEEE Transactions on Robotics*, vol. 38, no. 1, pp. 421–432, 2021.

[59] S. Macenski, S. Singh, F. Martín, and J. Ginés, "Regulated pure pursuit for robot path tracking," *Autonomous Robots*, vol. 47, pp. 685–694, 2023.

[60] M. Seder and I. Petrovic, "Dynamic window based approach to mobile robot motion control in the presence of moving obstacles," in *International Conference on Robotics and Automation*, IEEE, 2007, pp. 1986–1991.

[61] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 8620–8626.

[62] W. J. Rugh, *Linear system theory*. Prentice-Hall, Inc., 1996.

[63] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl, *Model Predictive Control: Theory, Computation, and Design (2nd Edition)*. Nob Hill Publishing, LLC, 2020.

[64] L. Stella, A. Themelis, P. Sopasakis, and P. Patrinos, "A simple and efficient algorithm for nonlinear model predictive control," in *IEEE Conference on Decision and Control (CDC)*, 2017, pp. 1939–1944.

[65] A. Sathya, P. Sopasakis, R. Van Parys, A. Themelis, G. Pipeleers, and P. Patrinos, "Embedded nonlinear model predictive control for obstacle avoidance using PANOC," in *European Control Conference (ECC)*, IEEE, 2018, pp. 1523–1528.

[66] A. Katriniok, P. Sopasakis, M. Schuurmans, and P. Patrinos, "Nonlinear model predictive control for distributed motion planning in road intersections using panoc," in *Conference on Decision and Control (CDC)*, 2019, pp. 5272–5278.

[67] J. A. Douthwaite, S. Zhao, and L. S. Mihaylova, "Velocity obstacle approaches for multi-agent collision avoidance," *Unmanned Systems*, vol. 7, no. 01, pp. 55–64, 2019.

[68] C. W. Warren, "Global path planning using artificial potential fields," in *IEEE International Conference on Robotics and Automation*, IEEE, 1989, pp. 316–317.

[69] N. Malone, H.-T. Chiang, K. Lesser, M. Oishi, and L. Tapia, "Hybrid dynamic moving obstacle avoidance using a stochastic reachable set-based potential field," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1124–1138, 2017.

[70] A. Billard, S. Mirrazavi, and N. Figueroa, *Learning for adaptive and reactive robot control: a dynamical systems approach*. Mit Press, 2022.

[71] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[72] I. Jang and H. J. Kim, "Safe control for navigation in cluttered space using multiple lyapunov-based control barrier functions," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2056–2063, 2024.

[73] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[74] X. Wang, S. Wang, X. Liang, *et al.*, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 4, pp. 5064–5078, 2024.

[75] L. Brunke, M. Greeff, A. W. Hall, *et al.*, "Safe learning in robotics: From learning-based control to safe reinforcement learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, no. 1, pp. 411–444, 2022.

[76] C. Wang, L. Meng, S. She, *et al.*, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 109–116.

[77] L. Huber, J.-J. Slotine, and A. Billard, "Avoiding dense and dynamic obstacles in enclosed spaces: Application to moving in crowds," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 3113–3132, 2022.

[78] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2011.

[79] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic Foundations of Robotics X*, Springer, 2013, pp. 475–491.

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

[81] S. Schaefer, K. Leung, B. Ivanovic, and M. Pavone, "Leveraging neural network gradients within trajectory optimization for proactive human-robot interactions," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 9673–9679.

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

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

[84] D. Fridovich-Keil, E. Ratner, L. Peters, A. D. Dragan, and C. J. Tomlin, "Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games," in *International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1475–1481.

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

[86] L. Heuer, L. Palmieri, A. Rudenko, A. Mannucci, M. Magnusson, and K. O. Arras, "Proactive model predictive control with multi-modal human motion prediction in cluttered dynamic environments," in *International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2023, pp. 229–236.

[87] A. Wang, C. Mavrogiannis, and A. Steinfeld, "Group-based motion prediction for navigation in crowded environments," in *Conference on Robot Learning*, PMLR, 2022, pp. 871–882.

[88] J. S. Wilson, *Sensor technology handbook*. Elsevier, 2004.

[89] F. Gustafsson, *Statistical sensor fusion*. Studentlitteratur, 2010.

[90] J. B. Bancroft and G. Lachapelle, "Data fusion algorithms for multiple inertial measurement units," *Sensors*, vol. 11, no. 7, pp. 6771–6798, 2011.

[91] R. Girshick, "Fast R-CNN," in *IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015.

[92] J. Redmon, "You only look once: Unified, real-time object detection," in *IEEE conference on computer vision and pattern recognition (CVPR)*, 2016.

[93] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "YOLO-world: Real-time open-vocabulary object detection," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 16 901–16 911.

[94] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single shot multibox detector," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, Springer, 2016, pp. 21–37.

[95] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *European conference on computer vision (ECCV)*, 2018, pp. 325–341.

[96] A. Kirillov, E. Mintun, N. Ravi, *et al.*, "Segment anything," in *IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[97] B.-N. Vo and W.-K. Ma, "The gaussian mixture probability hypothesis density filter," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4091–4104, 2006.

[98] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[99] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? does it matter?" *Structural safety*, vol. 31, no. 2, pp. 105–112, 2009.

[100] Z. Zhang, G. Hess, J. Hu, E. Dean, L. Svensson, and K. Åkesson, *Future-oriented navigation: Dynamic obstacle avoidance with one-shot energy-based multimodal motion prediction*, Paper E in this thesis, under review, 2025.

[101] Y. Song and D. P. Kingma, "How to train your energy-based models," *arXiv preprint arXiv:2101.03288*, 2021.

[102] F. K. Gustafsson, M. Danelljan, G. Bhat, and T. B. Schön, "Energy-based models for deep probabilistic regression," in *European Conference on Computer Vision*, Springer, 2020, pp. 325–343.

[103] S. H. Djork-Arné Clevert Thomas Unterthiner, "Fast and accurate deep network learning by exponential linear units (ELUs)," in *International Conference on Learning Representations (ICLR)*, 2016.

[104] K. Ceder, Z. Zhang, A. Burman, *et al.*, "Bird's-eye-view trajectory planning of multiple robots using continuous deep reinforcement learning and model predictive control," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 8002–8008.

[105] C. Rösmann, F. Hoffmann, and T. Bertram, "Timed-elastic-bands for time-optimal point-to-point nonlinear model predictive control," in *European control conference (ECC)*, IEEE, 2015, pp. 3352–3357.

[106] M. Dobrevski and D. Skočaj, "Adaptive dynamic window approach for local navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 6930–6936.

[107] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE international conference on intelligent robots and systems*, IEEE, 2014, pp. 2997–3004.

[108] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, "Neural RRT*: Learning-based optimal path planning," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1748–1758, 2020.

[109] B. Liao, F. Wan, Y. Hua, R. Ma, S. Zhu, and X. Qing, "F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate," *Expert Systems with Applications*, vol. 184, p. 115 457, 2021.

[110] V. Mnih, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[111] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 31–36.

[112] J. Berlin, G. Hess, A. Karlsson, *et al.*, "Trajectory generation for mobile robots in a dynamic environment using nonlinear model predictive control," in *IEEE International Conference on Automation Science and Engineering (CASE)*, 2021, pp. 942–947.

[113] S. F. Roselli, P.-L. Götvall, M. Fabian, and K. Åkesson, "A compositional algorithm for the conflict-free electric vehicle routing problem," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1405–1421, 2022.

[114] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *IEEE International conference on robotics and automation*, IEEE, 2011, pp. 4569–4574.

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

[116] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016.

[117] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *KDD, AAAI*, vol. 96, 1996, pp. 226–231.

[118] F. Bertilsson, M. Gordon, J. Hansson, *et al.*, "Centralized versus distributed nonlinear model predictive control for online robot fleet trajectory planning," in *International Conference on Automation Science and Engineering (CASE)*, IEEE, 2022, pp. 701–706.

[119] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artificial Intelligence Review*, vol. 55, pp. 895–943, 2022.

[120] Y. Xue and N. Figueroa, *No minima, no collisions: Combining modulation and control barrier function strategies for feasible dynamical collision avoidance*, 2025.

[121] C. K. Fourie, N. Figueroa, and J. A. Shah, "On-manifold strategies for reactive dynamical system modulation with non-convex obstacles," *IEEE Transactions on Robotics*, 2024.

[122] H. J. Choi and N. Figueroa, "Towards feasible dynamic grasping: Leveraging gaussian process distance field, SE (3) equivariance, and riemannian mixture models," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 6455–6461.

[123] C. Le Gentil, O.-L. Ouabi, L. Wu, C. Pradalier, and T. Vidal-Calleja, "Accurate gaussian-process-based distance fields with applications to echolocation and mapping," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1365–1372, 2023.

[124] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European control conference (ECC)*, IEEE, 2019, pp. 3420–3431.

[125] C. K. Fourie, N. Figueroa, and J. A. Shah, "On-manifold strategies for reactive dynamical system modulation with nonconvex obstacles," *IEEE Transactions on Robotics*, vol. 40, pp. 2390–2409, 2024.