Safe and Efficient Hierarchical Planning and Control of Mobile Robots: Moving-Horizon Optimization with Local Sensing

Dissertation
zur Erlangung des akademischen Grades
Doktoringenieur
(Dr.-Ing.)

von
Mohamed Abd Alla Mohamed Soliman
geboren am 09.02.1987 in Fayoum, Egypt

genehmigt durch die Fakultät für Elektrotechnik und Informationstechnik
der Otto-von-Guericke-Universität Magdeburg

Gutachter: Prof. Dr.-Ing. Rolf Findeisen
Prof. Dr.-Ing. Achim Kienle

Promotionskolloquium am 20.June 2025

“Science is the great antidote to the poison of enthusiasm and superstition.”

Adam Smith, Scottish philosopher ($\star$1723 †1790) [184]

Contents

Abstract

Mobile robots are increasingly being used in various fields such as delivery logistics, healthcare, elder care, and industrial automation. Many of these applications require high reliability and safe operation, especially when robots operate in dynamic environments shared with humans. A key challenge in this context is the planning of an optimal path to reach a desired goal –commonly referred to as path planning. Under ideal conditions, path planning relies on complete environmental information, including all obstacles, dynamic and static objects, and uncertainties regarding the available data. However, in real-world applications, mobile robots often have only limited information, necessitating path planning based on locally available sensor data and the ability to respond to unforeseen changes, such as sudden obstacles.

Technological advancements have helped to address this challenge. Modern mobile robots can be equipped with various sensors such as LiDAR, ultrasound, and camera systems, enabling them to capture environmental data in real-time. These data serve as the foundation for the planning and control layers, which guide the robot safely from an initial state to a desired location. However, this introduces the challenge of ensuring that path planning under limited information remains efficient and safe in real time.

Generating a collision-free path for mobile robots in a partially or fully unknown environment presents a significant challenge. The planned trajectory must avoid obstacles and adhere to the robot's dynamics, while being computed in real-time. Typically, this problem is solved hierarchically, where planning and control tasks are executed step by step based on available information. However, this approach often leads to overly cautious behavior and suboptimal paths, as the robot must continuously avoid potential collisions. Adaptive navigation in dynamic environments offers a potential solution, as it provides additional information and reduces uncertainties. Taking a slight detour using onboard sensors can improve environmental perception and lead to a more efficient movement strategy.

This work focuses on the interaction between planning and control for mobile robots, aiming to achieve more efficient and faster navigation through adaptive navigation strategies, even in partially unknown environments. To accomplish this, the planner and controller must generate a safe path within the sensor's field of view while simultaneously gathering new information during movement, without compromising safety. By formulating the planning problem as a mathematical optimization problem, this thesis addresses key challenges and develops corresponding solutions.

First, a hierarchical control framework over a moving horizon is introduced, enabling real-time path planning and control under limited sensor information. This concept includes a local planner, which solves a mixed-integer optimization problem using only local sensor data to generate a safe path within the sensor's field of view. The robot's dynamics are approximated using a linear model, while constraints such as obstacles and maximum velocities are enforced through linear inequalities with mixed-integer components. A low-level controller, based on a nonlinear model predictive control approach, ensures that the robot accurately follows the planned trajectories while accounting for potential nonlinearities. Simulation results confirm the real-time feasibility of the approach while ensuring safe obstacle avoidance.

Building on this, a two-stage planning and control strategy over a rolling horizon is introduced, enabling adaptive navigation to enhance environmental perception within the robot's sensor range. The local planner provides the low-level controller with a safe region in which it can perform navigation adjustments. New sensor data is integrated into the controller through an additional term in the cost function, allowing the robot to refine its navigation strategy while ensuring safe and efficient movement within its environment. Additionally, a fallback strategy is developed to limit the additional computational effort required for navigation adaptation. This strategy terminates adjustments when they become too costly and ensures that the robot can return to a preplanned safe path if necessary. Simulation results demonstrate that the approach is computationally efficient in real time, leading to shorter, safer, and more efficient paths.

In summary, this work makes a significant contribution to the development of more efficient, adaptive, and safer mobile robots, capable of navigating reliably in complex environments. Particularly in dynamic scenarios where robots interact with humans, such as in care facilities, hospitals, or public spaces, safe and predictive movement is essential. The methods presented in this work enhance robotic assistance in daily life, enabling mobile robots to operate safely and efficiently, detect and avoid obstacles in real time, and dynamically adjust their navigation in changing environments, ultimately improving their usability and effectiveness in real-world applications.

Deutsche Kurzfassung

Mobile Roboter finden zunehmend Einsatz in verschiedenen Bereichen wie der Lieferlogistik, dem Gesundheitswesen, der Pflege und der industriellen Automatisierung. Viele dieser Anwendungen erfordern eine hohe Zuverlässigkeit und einen sicheren Betrieb, insbesondere wenn Roboter in dynamischen Umgebungen mit Menschen agieren. Eine zentrale Aufgabe dabei ist die Planung eines optimalen Weges, um ein gewünschtes Ziel zu erreichen – die sogenannte Pfadplanung. Unter idealen Bedingungen basiert die Pfadplanung auf vollständigen Umgebungsinformationen, einschließlich aller Hindernisse, dynamischer und statischer Objekte sowie möglicher Unsicherheiten bezüglich der verfügbaren Daten. In der Realität jedoch stehen mobilen Robotern oft nur begrenzte Informationen zur Verfügung, was eine Pfadplanung auf Grundlage lokal verfügbarer Sensordaten sowie die Reaktion auf unvorhergesehene Veränderungen, wie plötzlich auftretende Hindernisse, erforderlich macht.

Diese Herausforderung wird durch den technologischen Fortschritt erleichtert. Moderne mobile Roboter können mit verschiedenen Sensoren wie LIDAR- und Kamerasystemen ausgestattet werden, die es ermöglichen, Umgebungsdaten in Echtzeit zu erfassen. Diese Daten dienen als Grundlage für die Planungs- und Regelungsschicht, die den Roboter sicher von einem Ausgangszustand zu einem gewünschten Ziel führt. Dies bringt jedoch die Herausforderung mit sich, dass die Pfadplanung unter begrenzten Informationen effizient und sicher in Echtzeit erfolgen muss.

Die Entwicklung eines kollisionsfreien Pfades für mobile Roboter in einer teilweise oder vollständig unbekannten Umgebung stellt eine erhebliche Herausforderung dar. Die geplante Trajektorie muss nicht nur Hindernisse vermeiden, sondern auch der Dynamik des Roboters entsprechen, während sie in Echtzeit berechnet wird. Typischerweise wird diese Problematik hierarchisch gelöst, indem Planungs- und Steuerungsaufgaben auf Grundlage der verfügbaren Informationen schrittweise ausgeführt werden. Diese Vorgehensweise führt jedoch häufig zu vorsichtigem Verhalten des Roboters und suboptimalen Pfaden, da jederzeit eine Kollision vermieden werden soll. Eine adaptive Navigation in dynamischen Umgebungen kann eine Lösung bieten, da sie die Umgebungswahrnehmung verbessert und Unsicherheiten bei der Bewegung reduziert. Ein „Umwegünter Nutzung der Bord-Sensoren kann die Umgebungswahrnehmung erweitern und zu einer effizienteren Bewegung führen.

Diese Arbeit konzentriert sich auf die Interaktion von Planung und Regelung für mobile Roboter mit dem Ziel, durch adaptive Navigation effizientere und sicherere Pfade zu finden, selbst in dynamischen Umgebungen mit begrenzten Sensorinformationen. Dazu muss der Planer/Regler einen sicheren Pfad innerhalb des Sichtfeldes der

Bord-Sensoren berechnen und gleichzeitig sicherstellen, dass während der Bewegung des Roboters neue Informationen gewonnen werden, ohne die Sicherheit zu gefährden. Basierend auf der Grundidee, Planungs- und Regelungsprobleme als mathematische Optimierungsprobleme zu formulieren, stehen die folgenden spezifischen Fragestellungen im Mittelpunkt dieser Arbeit, und es werden geeignete Lösungen entwickelt.

Zunächst wird ein hierarchisches Regelungskonzept über einen bewegten Zeithorizont vorgestellt, das eine Echtzeit-Pfadplanung und -regelung unter begrenzten Sensorinformationen ermöglicht. Dieses Konzept umfasst einen lokalen Planer, der ein gemischt-ganzzahliges Optimierungsproblem löst. Dabei werden ausschließlich lokale Sensordaten genutzt, um einen sicheren Pfad innerhalb des Sichtfelds der Sensoren zu planen. Die Dynamik des mobilen Roboters wird durch eine lineare Beschreibung approximiert, während die Einhaltung von Beschränkungen, wie Hindernissen oder maximalen Geschwindigkeiten, durch geeignete lineare Ungleichheitsbedingungen mit gemischt-ganzzahligen Komponenten sichergestellt wird. Ein unterlagerter Regler, der auf einem nichtlinearen modellprädiktiven Regelungsverfahren zur Verfolgung der geplanten Trajektorien basiert, garantiert die Einhaltung der Dynamik des Roboters, unter Berücksichtigung möglicher Nichtlinearitäten. Simulationsergebnisse belegen die Echtzeitfähigkeit sowie die garantierte Vermeidung von Hindernissen.

Darauf aufbauend wird eine zweistufige Planungs- und Regelungsstrategie über einen beweglichen Horizont vorgestellt, die eine adaptive Navigation zur verbesserten Wahrnehmung und präziseren Umgebungserfassung im Sensorumfeld des Roboters ermöglicht.

Der lokale Planer stellt dem unterlagerten Regler den sicheren Bereich in der Umgebung zur Verfügung, in dem dieser seine Navigation anpassen kann. Die Gewinnung neuer Informationen wird im Regler durch einen zusätzlichen Term in der Kostenfunktion berücksichtigt, wodurch neben der möglichst exakten Verfolgung des geplanten Pfades auch eine verbesserte Umgebungserfassung ermöglicht wird. Zusätzlich wird eine adaptive Anpassungs- und Rückfallstrategie vorgestellt, um den Rechenaufwand zu begrenzen. Die Strategie sorgt dafür, dass der Roboter seine Navigation effizient anpasst und im Bedarfsfall zu einem vorab geplanten sicheren Pfad zurückkehren kann. Simulationsergebnisse belegen, dass das Verfahren in Echtzeit umsetzbar ist und zu kürzeren, sichereren und effizienteren Pfaden führt.

Zusammenfassend leistet diese Arbeit einen wichtigen Beitrag zur Entwicklung effizienter, adaptiver und sicherer mobiler Roboter, die in der Lage sind, in komplexen Umgebungen zuverlässig zu navigieren. Besonders in dynamischen Szenarien, in denen Roboter mit Menschen interagieren, wie beispielsweise in Pflegeeinrichtungen, Krankenhäusern oder öffentlichen Räumen, ist eine sichere und vorausschauende Bewegung essenziell. Durch die hier vorgestellten Methoden kann Unterstützung im Alltag verbessert werden, indem mobile Roboter sicher und effizient arbeiten, Hindernisse autonom erkennen und vermeiden sowie ihre Navigation dynamisch anpassen, um ihren Einsatzbereich, ihre Effektivität und ihre Sicherheit weiter zu erhöhen.

1 Introduction and Motivation

The more original a discovery, the more obvious it seems afterward.

Arthur Koestler

As technology advances, there is a growing focus on the development of intelligent and autonomous robots to support humans in performing complex or physically demanding tasks. These systems contribute to various applications, including firefighting, delivery services [1], disaster response and emergency assistance [3], and home cleaning and elder care [5].

A key challenge in deploying autonomous robots is ensuring safe and reliable navigation while effectively completing their assigned tasks. Robots often operate in environments with obstacles that can affect their movement and task execution. Additionally, these environments may be partially or entirely unknown, requiring planning and control strategies that account for uncertainty and real-time adaptability to maintain operational safety and efficiency.

The problem of motion planning and controlling a mobile robot is often hierarchically decomposed; cf. Figure 1.1, where the planner provides a path to track to the controller. Frequently global/offline path planners are used, which are based on "offline" information about the working environment, considering environmental information like obstacles' position, geometry, and the goal to reach. The controller uses the planned path to steer the vehicle until it reaches the destination safely, cf. Figure 1.2a. However, in real applications, autonomous robots are typically deployed in only partially known environments, resulting in the need to perform a sensor-based/online path planning problem. Taking advantage of the rapid advancements in sensor technology, autonomous robots are equipped with sensors, including LiDAR and camera systems. These sensors provide the controller and the planner with environmental information within the vehicle's proximal surroundings; cf. Figure 1.2b. These sensors are characterized by a limited range or field of view, and these limitations culminate in a scarcity of information accessible for path planning. Consequently, the controller must anticipate the likelihood that the vehicle will encounter unforeseen obstacles owing to its limited sensing abilities, for example, an obstacle that obstructs the autonomous robot's path around a corner, posing a potential collision hazard. To ensure vehicle safety during operation, the planner is tasked with devising a safe path based on currently available sensor information. As the vehicle progresses on its path, new information becomes available, prompting updates to the planned path accordingly.

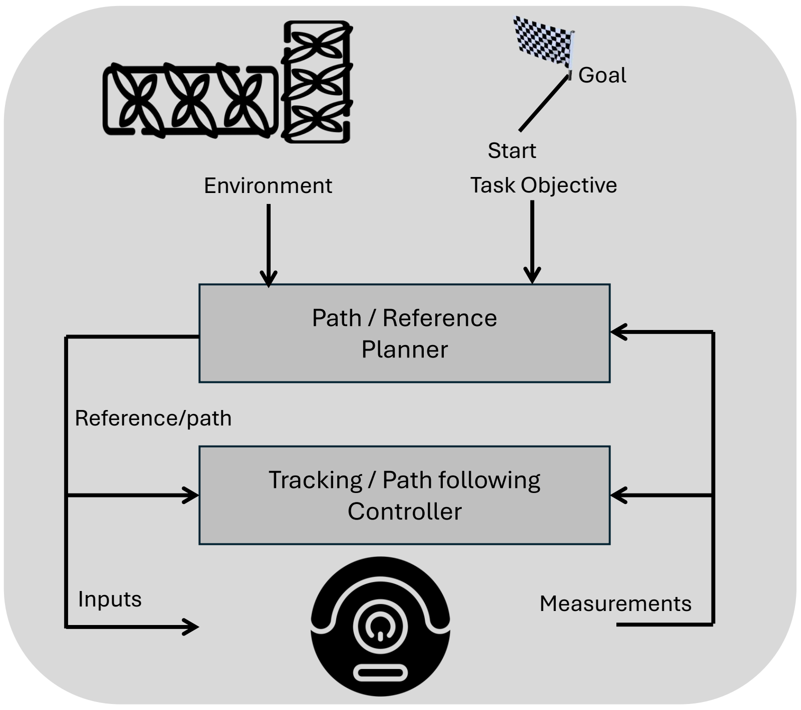


Figure 1.1: Interconnection between motion planner and control for mobile robots.

This iterative process constitutes a strategy known as passive navigation; cf. Figure 1.3a. However, the passive navigation approach allows the autonomous robot to safely move from an initial state to the destination, but the control strategy remains cautious. Improved performance can be achieved by enabling the robot to adapt its navigation based on real-time sensor data, highlighting the need for perception-aware and information-driven planning and control.

In perception-aware planning and control, a more efficient and adaptive path can be determined when the autonomous robot adjusts its trajectory based on sensor inputs. The robot may temporarily modify its planned route to improve environmental awareness by refining its perception of detected obstacles through onboard sensors. As a result, the overall navigation strategy can be optimized; for example, the robot may reach its destination more efficiently.

To achieve this, the control objective is enhanced with a term that prioritizes real-time obstacle awareness, allowing the robot to proactively adjust its trajectory. Therefore, the planner must balance sensor-informed navigation adjustments with overall task efficiency; cf. Figure.1.3b. Notably, the planner and controller should ensure that the vehicle can follow the planned path. To achieve this, they must consider the vehicle's dynamics and inherent constraints, such as maximum speed, steering angle, acceleration, and turning rates. To guarantee safe operation, the planning phase must account for partially detected obstacles within the current field of view. Moreover, the entire planning problem must be solvable in real-time on the embedded hardware of the mobile robot.

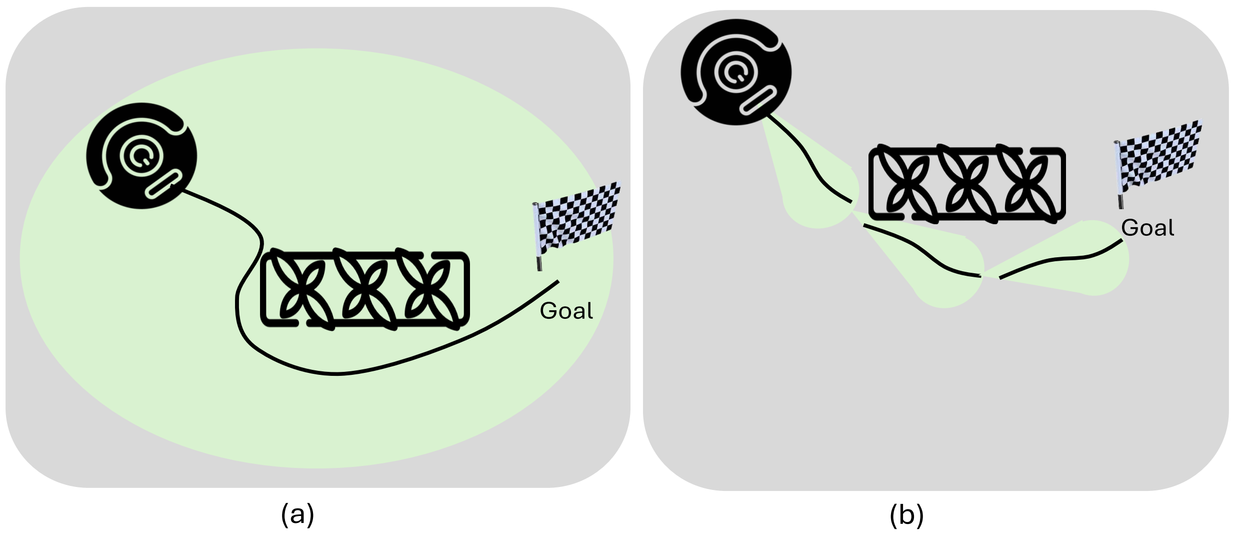


Figure 1.2: ((a) Offline path planning, used when all required information is available. (b) Online path planning, which leverages local sensor data in real-time.

Many approaches exist for adaptive navigation and planning. In [48], the authors proposed a perception-aware path planning framework that uses geometric and photometric information to improve localization. Planning a path with enriched environmental information allows for faster and more precise navigation. The authors in [51] consider planning a path that maximizes coverage while accounting for obstacles, localization, and uncertainty detection. In [171], a perception-aware planner was proposed, where perception uncertainty is pose-dependent and represented by ellipsoids, which shrink as the robot moves closer to known points. Adaptive navigation can also be achieved using reward-based functions. In [92], an entropy-based information measure is used, while the Fisher information matrix is exploited in [132].

Typically, path planning is formulated as an optimization problem, often solved using search-based approaches that involve discretizing the search and action space [13, 84, 103, 103, 169]. However, many search-based methods neglect vehicle dynamics, leading to trajectories that may not be feasible for execution. To address this, a common strategy is to integrate robot dynamics into the optimization process, employing a rolling horizon approach, commonly referred to as moving-horizon control and planning. This involves solving an optimization problem that accounts for obstacles, task objectives, and robot dynamics. The iterative nature of this optimization process, known as Model Predictive Control (MPC) or Moving Horizon Control, has been widely adopted in control systems.

We propose a hierarchical planning and control strategy for local navigation, based on mathematical programming formulations while ensuring real-time feasibility. The

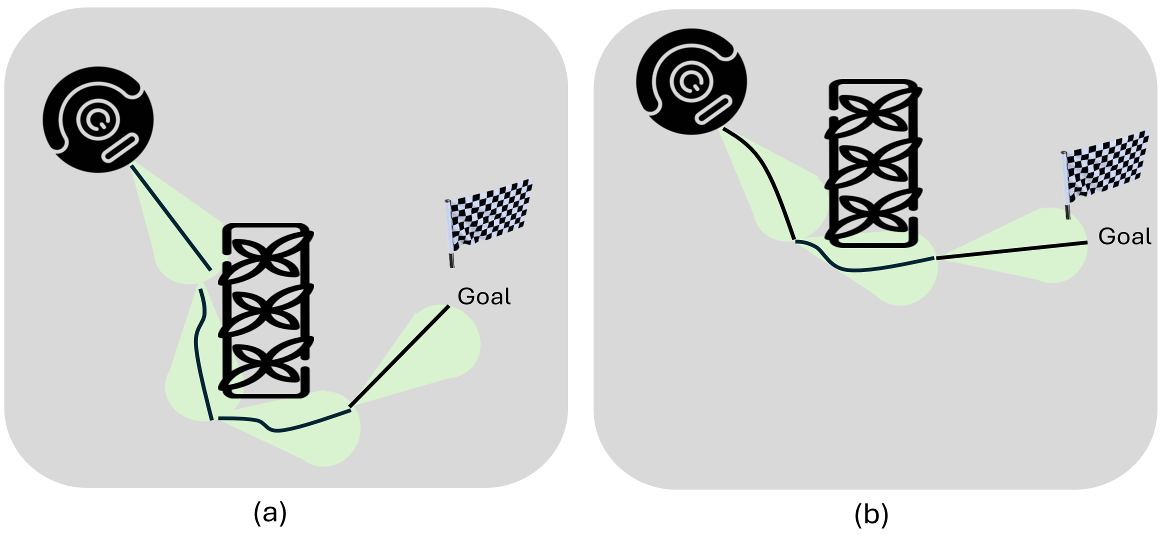


Figure 1.3: Comparison of adaptive and passive navigation for planning under limited environmental information: Without real-time adaptation (a), planning and control rely solely on previously acquired information. With sensor-informed adjustments (b), the robot dynamically refines its path based on real-time data.

.

approach employs model predictive control (MPC) at both the local planning and control layers, enabling the robot to navigate within the safe space provided by sensor data. The principle of MPC/Moving Horizon Control [69, 180] is particularly suited for this task, as it allows the explicit consideration of obstacle constraints while maintaining formulation flexibility.

MPC is an advanced control strategy that accounts for nonlinear systems, multiple inputs and outputs, time delays, and constraints. It has been widely applied across various civilian domains, including the process industry [121] and autonomous systems [117, 200]. When formulated correctly, MPC ensures constraint satisfaction, provided that a feasible solution exists. At each time step, MPC solves an optimization problem based on a user-defined performance metric, such as minimizing travel distance or reducing energy consumption. The optimization problem is subject to system dynamics (e.g., robot motion), constraints (e.g., obstacles), and stability and feasibility conditions [69, 180]. In its standard form, the solution to the optimization problem provides an open-loop input trajectory. Feedback is introduced by applying the first control input, then re-measuring/estimating system states and re-solving the optimization problem at each time step.

Planning and control for autonomous mobile robots is an adaptive research field, particularly in computer science and control engineering. While hierarchical MPC frameworks have been used, their formulation as a mathematical programming problem has not been as widespread [107, 162, 221, 236]. To address real-time feasibility, we introduce a Hierarchical Receding Horizon Planning and Control strategy, which generates a safe path and ensures trajectory tracking within the sensor field of view, where obstacles are detected (see Figure 1.4).

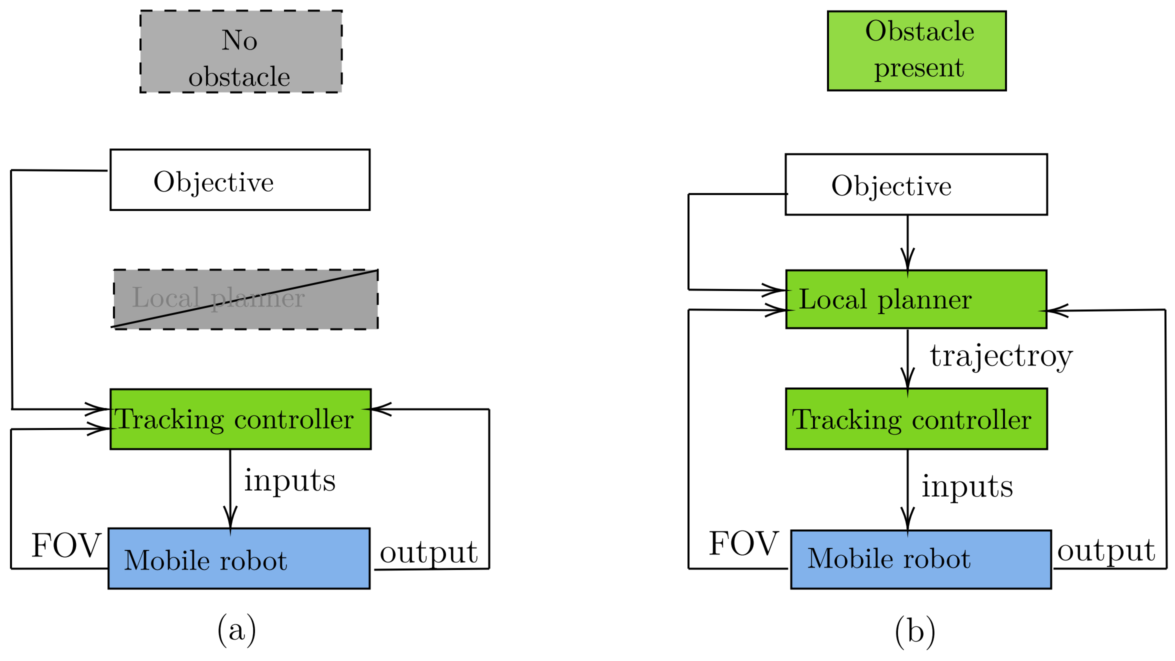


Figure 1.4: Proposed hierarchical scheme, which switches between operational modes depending on the presence of obstacles. When no obstacles are detected, the planning layer is bypassed to reduce computational effort. When obstacles are present, the planner generates a safe path, which the low-level controller follows to ensure obstacle avoidance. FOV refers to the field of view of the sensor, which can only provide sensory information in a certain area.

The proposed scheme operates in two distinct modes. When no obstacles are detected within the sensor field of view, the planning layer is bypassed, reducing computational effort. When obstacles are present, the high-level planner computes a safe trajectory, which is then executed by the low-level controller, ensuring constraint satisfaction. This results in a safe trajectory forwarded to the low-level controller. Subsequently, the low-level controller utilizes a model predictive control path following formulation to follow the path, accommodating the possibly nonlinear robot dynamics and constraints such as maximum turning rate and acceleration. The first control input candidate is applied to the system, altering the state of the robot and providing new information for subsequent hierarchical iterations, as shown in Figure 1.4 b. This iterative process continues throughout the robot's operation, with planning and control problems being solved based on newly captured information at each time instance. Notably, the path is not adaptively adjusted to obtain further environmental information, that is, passive navigation is used. The question arises as to how control behavior can be improved through the design of an adaptive navigation strategy.

From a control perspective, the adaptive navigation path planning problem is closely related to the so called dual control problem, in which the goal is to obtain further state or system information while controlling a system. Feldbaum [65] first recognized that the control inputs to an uncertain system should have a probing effect to adaptively learn the system uncertainty and system dynamics, thus improving the control performance. However, it is necessary to balance the improved system information with the overall control objective. In general, solving the original dual control problem is computationally expensive. Approximations have been proposed, leading to the so-called implicit dual control problem and the explicit dual control problem [152, 153].

To avoid the complexity associated with solving a dual control problem while accounting for nonlinear robot dynamics and obstacles, a hierarchical dual rolling horizon control and planning strategy is proposed. In the proposed strategy, the high-level planner addresses the planning problem in a traditional way, purely based on available and potentially limited information. The information transmitted from the planner layer to the low-level controller includes a safe trajectory within the current field of view, a safe navigation set, and the available environmental information. The low-level controller is extended by a dual control capability. The control inputs derived from solving the dual optimization problem keep the system states in the safe set provided by the planner and contain probing capabilities to decrease obstacle uncertainty and enhance accessible environmental information.

Although adaptive navigation can improve control actions and progress toward the overall objective, it may also lead to additional movement that increases the overall effort . Furthermore, obstacles encountered during adaptive navigation may require an adaptation of the approach, see Figure 1.5. A fallback strategy is proposed to address these challenges. The strategy ends the adaptive navigation if the effort required becomes too high. It also ensures that the mobile robot remains on a feasible path by guiding it toward the preplanned safe path leading to the goal. The proposed approaches are validated through simulations using a mobile robot equipped with onboard sensors such as LiDAR or camera systems. The simulation results demonstrate that adaptive navigation leads to an improved performance while avoiding obstacles.

1.1 Main contributions

This thesis introduces a new hierarchical receding horizon framework for real-time motion planning and control of mobile robots in dynamic, partially unknown environments based on mathematical optimization. The proposed techniques ensure safe, efficient, and human-aware navigation by integrating local sensor-based planning with optimization-based decision-making and model predictive control strategies. These methods lay the foundation for future advancements in reliable, adaptable, and intelligent robotic systems. The main contributions of this work are:

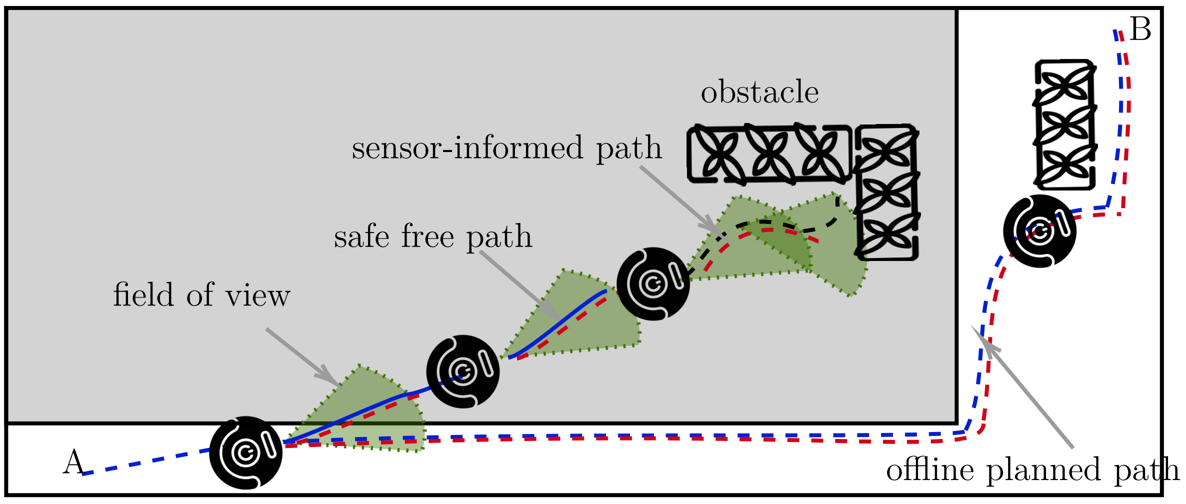


Figure 1.5: A mobile robot follows a safe preplanned path from point A to point B (dashed blue line). Upon reaching an unknown region, detected by local sensors, an online path planner is activated that steers the robot. Obstacle detection activates the sensor-driven navigation adjustment scheme (dashed black line). When the mobile robot is obstructed by other obstacles, the fallback controller is activated to guide the robot along the safe and the preplanned path to the destination (dashed red line).

Real-Time Feasibility in Local Planning and Control: A hierarchical receding horizon approach enables safe and dynamically feasible path planning within the sensor field of view. A computationally efficient mixed-integer programming formulation ensures real-time path updates based on live sensor data, allowing the robot to operate smoothly and predictably in human-populated environments while avoiding obstacles in a timely manner.

Hierarchical Planning and Control for Reliable Navigation: The proposed framework integrates local path planning with MPC-based control, enabling robots to adapt their motion predictably and safely in changing environments. The local planner computes a safe corridor, while the low-level controller accurately tracks the planned path, significantly reducing the computational complexity of global planning while enhancing responsiveness and safety.

Safe and adaptive navigation for Enhanced Perception: A novel adaptive navigation strategy enables robots to dynamically adjust their movement based on real-time sensor data while ensuring safe operation within predefined limits. By balancing efficiency and environmental awareness, the robot can optimize its navigation, leading to smoother and more natural movement, particularly in human-centered spaces such as hospitals, warehouses, and service environments.

Safety Fallback Strategy for Human-Friendly and Reliable Navigation: To ensure controlled and predictable behavior, a fallback mechanism prevents unnecessary deviations during adaptive navigation. If adaptive navigation becomes computationally demanding or impractical, the system seamlessly transitions to a precomputed safe path, enhancing reliability while ensuring smooth, consistent, and human-aware operation.

Laying the Foundation for Future Safe and Intelligent Robotics: The methods introduced in this dissertation mark a significant step toward safe, efficient, and adaptive mobile robot navigation. By integrating real-time optimization with perception-aware control, this work lays the groundwork for future research in human-robot interaction, assistive robotics, and urban autonomous mobility, ensuring that mobile robots can operate seamlessly and safely in dynamic, shared spaces.

1.2 Outline

The thesis is structured as follows:

In Chapter 2, the general problem of path planning in the context of mobile robotics is introduced. An overview of the current state of the art is provided, reviewing existing concepts such as search algorithms and the formulation of path planning as a mathematical optimization problem. These approaches aim to improve the efficiency and reliability of autonomous navigation in applications such as logistics, healthcare, and assistive robotics.

Chapter 3 introduces Model Predictive Control in the context of planning and control, using illustrative examples to outline key formulations employed in this thesis, including set-point stabilization, path following, and trajectory tracking. The chapter also discusses stability and repeated feasibility conditions to ensure consistent and safe operation of mobile robots. Additionally, it outlines how Model Predictive Control can be applied to path planning on moving horizons, formulating it as a mathematical programming problem that includes mixed-integer variables to guarantee obstacle avoidance. The concepts are demonstrated through illustrative examples to highlight their practical applicability.

Chapter 4 presents the first major contribution of this work, formulating local path planning as a Hierarchical Receding Horizon Control and Planning problem. The term local refers to the fact that only sensor data from the immediate surroundings, which may be limited in range and precision, is used for planning. Consequently, only a local path is generated, constrained by the field of view of the sensors. The resulting local planner solves a mixed-integer optimization problem, incorporating linear robot dynamics and approximated robot constraints. Obstacle avoidance is achieved by reformulating obstacle constraints as mixed-integer constraints using the Big-M method. The low-level controller ensures accurate execution of the planned trajectory using a trajectory-tracking formulation that accounts for nonlinear robot dynamics. This hierarchical approach significantly reduces computational complexity while ensuring safe and efficient robot movement. The effectiveness and real-time feasibility of the approach are demonstrated through simulations.

Building on this, Chapter 5 introduces methods for enhancing performance through adaptive environmental aware navigation. The chapter provides an overview of dual control and its approximations, which allow robots to adapt their navigation strategies based on newly available sensor data, with a focus on mathematical programming-based formulations. A hierarchical dual-rolling horizon approach is introduced, where the local planner generates a safe trajectory within the sensor field of view, along with a time-varying safe navigation set and environmental information. The low-level controller extends its objective to include adaptive navigation, leading to an explicit dual control formulation that allows safe navigation within the sensor's field of view when beneficial. A fallback controller is presented to guide the robot back to a predetermined safe path if the adaptive navigation does not yield a feasible path forward. This strategy enhances real-time adaptability, efficiency, and safety in applications such as warehouse automation, indoor service robots, and autonomous delivery. The approach is validated through simulations, demonstrating that adaptive navigation improves efficiency while remaining computationally feasible and ensuring safe robot movement.

Finally, Chapter 6 summarizes the proposed approaches and results, providing insights into potential future research directions to further improve the adaptability, efficiency, and safety of mobile robots in real-world environments. The findings of this work contribute to the development of more intelligent, responsive, and human-centered robotic systems, capable of seamlessly operating in dynamic and shared spaces. Future advancements could focus on enhancing robot perception, increasing interaction capabilities with humans, and improving decision-making in complex, uncertain environments. These improvements would enable mobile robots to better assist in everyday tasks, healthcare support, smart logistics, and public service applications, ultimately contributing to more accessible, efficient, and sustainable solutions that enhance human well-being and quality of life.

The following articles cover parts of the work presented in this thesis:

1. **M. Soliman**, R. Findeisen. Moving Horizon Planning and Control for Autonomous Vehicles with Active Exploration and Fallback Strategies. In Proceedings of 21^{th} International Conference on Informatics in Control, Automation and Robotics, Porto, Portugal, 2024.

2. **M. Soliman**, B.Morabito, R. Findeisen. Towards Safe Exploration for Autonomous Vehicles using Dual Model Predictive Control. In The 9^{th} IFAC Symposium on Mechatronic Systems, 2022.

3. J. Matschek, J. Bethge, **M. Soliman**, B. Elsayed, R. Findeisen. Constrained reference learning for continuous-time model predictive tracking control of autonomous systems. IFAC Conference on Nonlinear Model Predictive Control, pages 329-334, Bratislava, Slovakia, 2021.

2 Optimization Based Motion Planning for Mobile Robots

Any man who afflicts the human race with ideas must be prepared to see them misunderstood.

HL Mencken

Navigating from one location to another may seem straightforward for humans, but for autonomous mobile robots, it is a complex challenge, especially in dynamic environments such as warehouses with other robots and workers, hospitals, or elderly care facilities. Ensuring safe and efficient navigation is essential for applications where robots assist humans in logistics, healthcare, or daily activities. The efficiency of the planned path directly impacts the overall system performance, influencing aspects such as energy consumption, task completion time, and operational safety.

2.1 Mobile Robots and Navigation

Robots are commonly classified into two broad categories-mobile robots and industrial robots-based on their structural and functional characteristics (see Figure 2.1). Mobile robots are distinguished by their ability to autonomously navigate and operate in various environments. They are used in a wide range of civilian applications, from household assistance to automated delivery systems in hospitals, warehouses, and logistics [196]. In contrast, industrial robots typically consist of articulated robotic arms equipped with grippers or other end-effectors, which are stationary and designed for repetitive tasks in manufacturing and automation settings [24].

Mobile robots can be classified based on their operating environment into three categories: land-based (ground or indoor robots), commonly used in service and logistics applications; aerial robots, often deployed for infrastructure inspection and environmental monitoring; and underwater robots, utilized in marine research and exploration. The integration of advanced sensors, such as cameras, LiDAR, and ultrasound, along with enhanced computational capabilities, has expanded the capabilities of autonomous robots to perform complex, high-precision tasks in collaborative and assistive environments. These include automated material handling in warehouses, patient assistance in healthcare, and mobility support in public spaces. The role of advanced sensors is critical, as they provide real-time environmental data, enabling decision-making systems to ensure safe and reliable robot operation [45, 158, 195].

(a) Agricultural mobile robot [2].

(b) Mobile robot in hospital [6].

(c) Food delivery robot [147].

(d) Industrial mobile robot [4].

Figure 2.1: Mobile robots used in various applications, spanning from agriculture, to hospital and elderly care, food delivery and logistics, up to industrial production.

One of the primary challenges in designing navigation systems for autonomous mobile robots is ensuring safe, collision-free movement while adapting to changing environments. To formalize this process, we introduce the following key concepts:

Definition 1. *The configuration space, or $\mathcal{C}$-space, of a mobile robot is the set of all possible positions and orientations (configurations) that the robot can achieve within its environment.*

Definition 2. *An obstacle/unsafe workspace is a subset of the configuration space that is permanently or temporarily occupied. In the configuration space, this region is referred to as the obstacle space, denoted by $\mathcal{C}_O$ or in short $\mathcal{O}$.*

Definition 3. *A safe path is a collision-free trajectory within the configuration space along which the mobile robot can move.*

Definition 4. *A trajectory is a continuous curve that connects two points in the configuration space.*

Definition 5. *Path planning refers to the task of determining a safe path from an initial configuration $\mathcal{C}_O$ or in short A to a final desired configuration $\mathcal{C}_f$ or a final configuration B while avoiding unsafe workspace.*

Generally, solving the path planning problem involves a series of subtasks [120]:

- *Localization*: This refers to the task of determining the position of the robot within the environment. Typically, a variety of methods are used for localization, such as camera-based systems or GPS for outdoor applications. Localization may also be based on known landmarks in the environment.

- *Mapping*: The planner requires environmental information and a map of the environment to plan a safe and optimal path. This map can be obtained from offline data, discovered by the robot through sensor information, or a combination of both.

- *Motion planning*: This refers to the task of deciding, *What is the best way to reach a certain final position or achieve a goal?*. Based on a user-defined metric, such as the shortest distance, the planner calculates an obstacle-free path between a starting point and a given final point.

The complete framework resulting from these subtasks is generally referred to as *the navigation system*, cf. Figure 2.2.

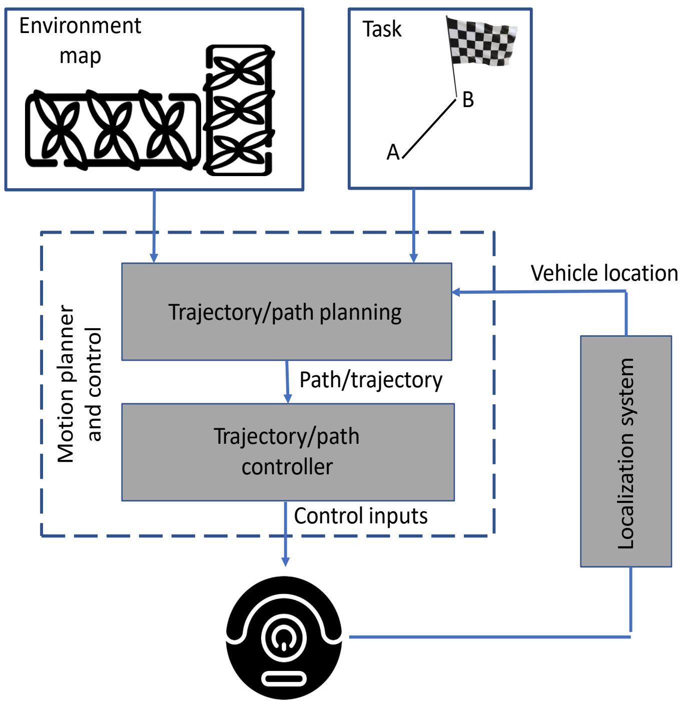


Figure 2.2: Hierarchical structure of navigational systems for autonomous mobile robot.

2.2 The General Path Planning Problem

In this section, we focus on path planning and review existing methodologies used to address it. The general motion or route planning problem can be stated as follows

[129]:

Problem 1. *Given a mobile robot V moving in an environment E that contains static or dynamic obstacles $\mathcal{O}$ (see Figure 2.3), find a (safe) path $P \subset E$ starting from an initial position $A \in \mathbf{R}^n$, such that the robot reaches a final position $B \in \mathbf{R}^n$. Here, P represents the state of V, ensuring that the robot does not intersect the obstacle set $\mathcal{C}_O$.*

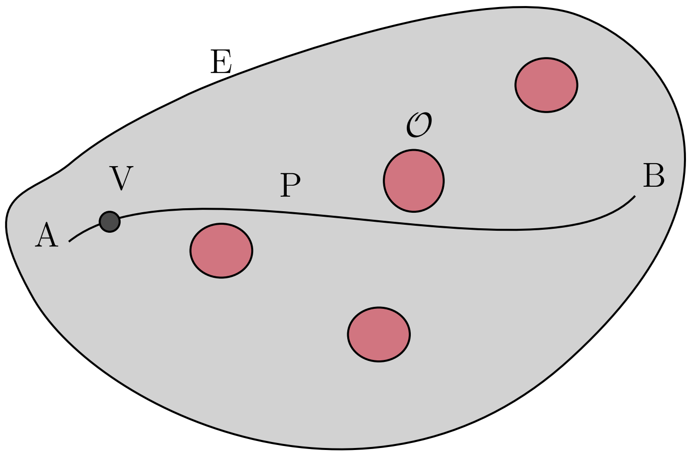


Figure 2.3: General path planning problem avoiding unsafe workspace $\mathcal{C}_O$ moving from A to B, while staying in the set E.

The motion/path planning problem for autonomous mobile robots is challenging due to the various factors that must be considered during the planning process. Specifically, the problem requires accounting for the mobile robotâ€™s equations of motion, its velocity and acceleration restrictions, uncertainties in the mobile robot's state, and limited environmental awareness due to restricted sensing capabilities and ambiguities in sensor data. Numerous algorithms have been proposed to obtain or approximate solutions to problem 1. These algorithms vary in terms of soundness, completeness, optimality, precision, and computational complexity, which can be categorized as follows [84]:

Definition 6. ***Completeness:*** *A path planning algorithm is considered complete if, under all conditions, it either finds a safe and obstacle-free path connecting the initial point to the goal point or correctly states that no such path exists.* Resolution completeness *applies to algorithms that discretize the solution space: as the resolution of the discretization increases, the algorithm should converge toward an exact solution.* Probabilistic completeness *means that the probability of finding a solution approaches 1 as the number of samples or the computation time increases.*

Definition 7. ***Optimality:*** *A path planning algorithm is considered optimal if it can find the best possible path with respect to a specified criterion, such as the shortest distance, minimum time, or lowest energy consumption.*

Definition 8. ***Soundness subject to uncertainty:*** *A sound planner ensures that the autonomous mobile robot reaches the goal state without colliding with an unsafe workspace, even in the presence of uncertainties in state estimation, sensor measurements, and control actions.*

Definition 9. ***Polynomial complexity:*** *An algorithm exhibits polynomial complexity if it can find a solution within a time frame that scales polynomially with the size of the input. Although the general path planning problem is known to be NP-hard, approximate solutions can often be found using approximation algorithms [116].*

Definition 10. ***Polynomial-time planners (P):*** *These are planners capable of solving the path planning problem in polynomial time, with time complexity functions such as logarithmic, linear, or quadratic [103].*

Definition 11. ***Nondeterministic polynomial-time (NP) planners:*** *These planners can verify a solution to the path planning problem in polynomial time.*

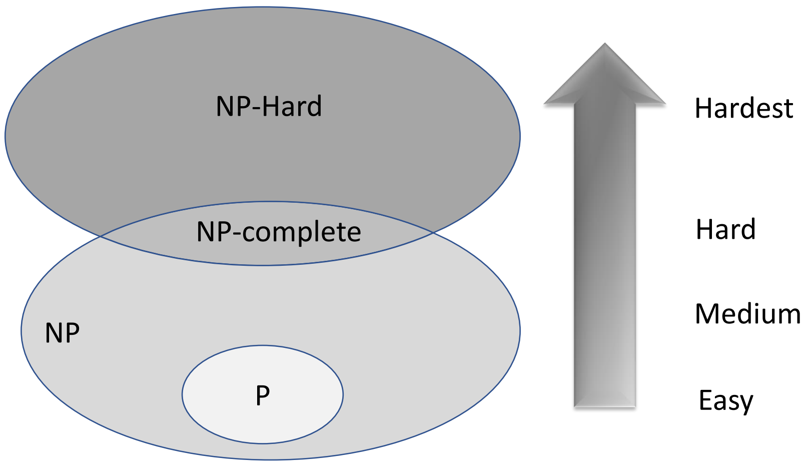


Figure 2.4: Computational complexity of algorithms.

2.3 Classifications of Planning Problems

The general path planning problem (1) can be categorized into two main classes: *static* and *dynamic* path planning problems. In a static path planning problem, extensive information about the environment may be available, such as workspace dimensions, obstacle locations, and geometry. However, mobile robot dynamics is either not considered or only indirectly accounted for. In contrast, dynamic path planning problems directly incorporate the dynamic properties of the mobile robot into the planning process.

In addition, path planning problems can be further classified according to whether the environment is *time-varying* or *time-invariant*. Time-varying path planning problems involve finding a safe, obstacle-free path for autonomous mobile robots operating in environments with moving obstacles or changing environmental conditions [43, 101]. In contrast, time-invariant path planning problems focus on determining paths in environments where obstacles remain stationary.

Another classification for path-/motion-planning problems is based on the presence of differential constraints. In a *differentially constrained* path planning problem, the equation of motion of the mobile robot is incorporated as a constraint on the planned path. This means that the planner must respect the kinematic and dynamic properties of the mobile robot. In contrast, in *differentially unconstrained* problems, mobile robot movement is not restricted by its dynamics, allowing the planner to focus solely on the geometric aspects of the route.

In addition, path planning problems often account for the shape of the mobile robot. In many simplified models, the mobile robot is treated as a point mass, resulting in what is known as a *mass point* motion planning problem. In these cases, the mobile robot is approximated as a point in the configuration space, while unsafe workspace is expanded by the radius of the mobile robot's bounding ball to ensure safety, as shown in Figure 2.5 [88]. This expansion ensures that the simplified model avoids collisions with an unsafe workspace despite the approximation. In more detailed planners, the mobile robot's shape and dynamics are fully considered by incorporating differential constraints, allowing the mobile robot to follow the planned path while minimizing a given criterion [130].

The so-called *piano mover's problem* represents a specific class of path planning problems in which the mobile robot is treated as a rigid body without dynamic constraints. In this scenario, the planner's goal is to find a safe and barrier-free path for the mobile robot to travel through an environment populated by static obstacles [84, 103, 222]. The solution is defined as a sequence of positions and orientations in the free configuration space (Figure 2.5a). Although algorithms exist to solve this problem, its complexity increases with the size of the configuration space, that is, the state space of the autonomous robot [104].

Another important class of problems involves *autonomous mobile robots with differential constraints*, which includes both kinematic and dynamic constraints. *Kinematic constraints* govern the mobile robot's motion without accounting for the forces applied, leading to what is commonly called a *nonholonomic path planning problem. Dynamic constraints*, on the other hand, impose second-order or higher constraints, taking into account factors such as acceleration and force. Solving path-planning problems under these full sets of differential constraints is particularly complex when an unsafe workspace is introduced into the environment.

Finally, path planning problems can also be categorized based on the number of autonomous robots operating in the same environment. This distinction leads to two primary cases: single-robot path planning, where one robot navigates independently, and multi-robot path planning, where multiple robots operate within a shared space.

In multi-robot path planning, robots must coordinate their movements to ensure safe and efficient navigation in environments such as hospitals, warehouses, or public spaces. The planning approach must account for robot interactions, ensuring that paths remain collision-free and that tasks such as cooperative transport, delivery, or facility maintenance are executed smoothly [76, 194].

The complexity of multi-robot navigation arises from the need to dynamically adapt routes based on real-time information while avoiding congestion and maintaining efficiency. Planners must consider how the collective movement of multiple robots affects navigation, ensuring that coordination strategies enable smooth operation in shared environments [106].

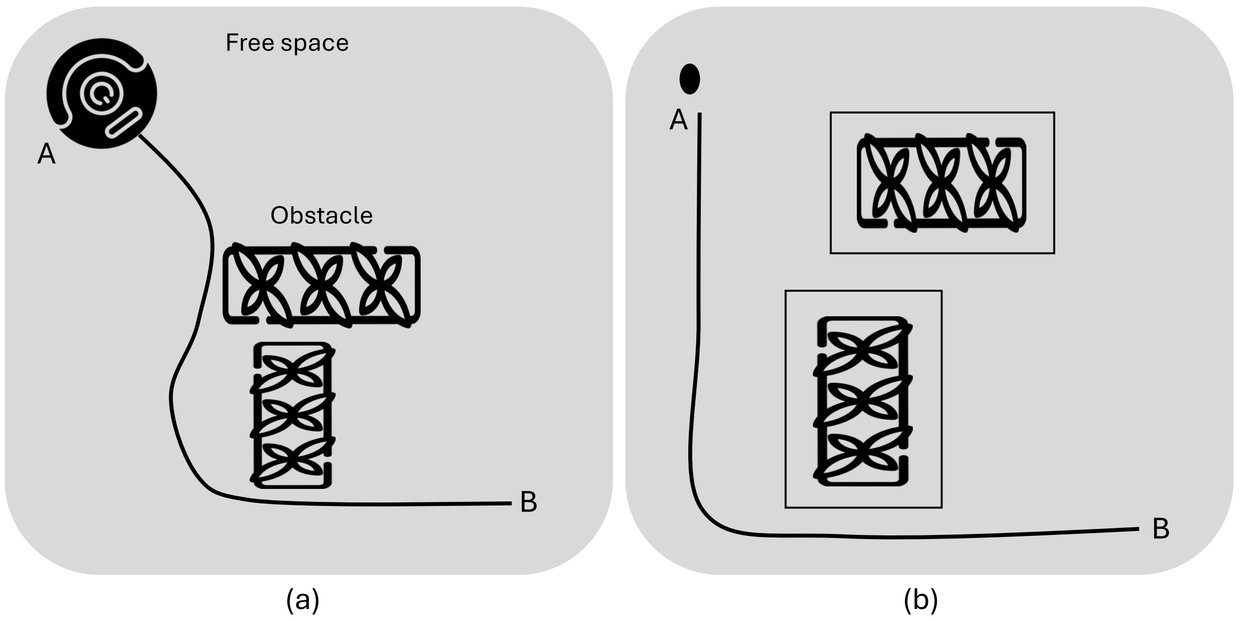


Figure 2.5: Safety in path planning can be ensured by expanding unsafe workspace boundaries while approximating the mobile robot as a mass point.

2.4 Existing Motion Planning Algorithms

Algorithms for motion planning for mobile robots must take into account the mobile robot's dynamics and constraints. The algorithm identifies configurations within the state space of the mobile robot, linking the initial position to the desired destination. The planned trajectory or path must navigate the environment while avoiding both static and dynamic obstacles.

Definition 12. *A* path *refers to a planned route or geometric curve that an autonomous mobile robot follows to reach a destination. It is typically defined in terms of the mobile robot's position in a two- or three-dimensional space (e.g., latitude, longitude, and altitude or coordinates X, Y, and Z). The path represents the desired spatial route, but does not include information on the mobile robot's speed or timing. Essentially, the path is the spatial plan that the mobile robot should adhere to, without considering the dynamics of motion, such as speed or acceleration.*

Definition 13. *A* trajectory *is a plan that specifies not only the path of the mobile robot (the spatial route) but also the timing and dynamic aspects of its movement along this path. It includes information on the mobile robot's position, velocity, and acceleration at each point in time. A trajectory defines where the mobile robot should be at every moment to ensure safe and efficient travel from the starting point to the destination.*

Numerous classifications of path planning algorithms have been described in the literature [84, 103]. A common categorization is based on the level of environmental awareness, distinguishing between *global* and *local* path planners. Global path planners operate with full knowledge of the environment, whereas local path planners make decisions based on immediate surroundings. Another classification is based on the differential constraints imposed, segmenting the motion planning algorithms into *differentially unconstrained* and *differentially constrained* path planning algorithms [84].

2.4.1 Global Path Planner

Global path planners are particularly useful in structured environments where detailed maps are available, such as hospitals, warehouses, manufacturing facilities, or public transportation hubs. They enable efficient long-term navigation by leveraging preexisting knowledge of the environment, including information on workspace dimensions, static obstacles, and infrastructure layouts [37, 169]. These planners are well-suited for environments with predictable conditions, where navigation can be planned in advance to optimize efficiency, energy consumption, and operational safety.

2.4.2 Local Path Planner

Local path planners, also known as sensor-based planners, rely on real-time environmental perception to determine safe trajectories at each time step. These planners are particularly valuable in dynamic environments, where changes such as rearranged objects in a warehouse, moving hospital beds in a healthcare facility, or pedestrian movement in public spaces require continuous adjustments.

Unlike global path planners, local planners do not rely solely on preexisting maps. Instead, they process sensor data from technologies such as LiDAR, cameras, or ultrasound to detect obstacles and adapt routes accordingly. This capability is crucial for applications in assistive robotics, autonomous service robots, and indoor mobility solutions, where unexpected environmental changes must be handled safely and efficiently.

In practice, local planners are often used in combination with global planners, allowing robots to follow a general precomputed route while dynamically adjusting their movements based on immediate surroundings. This hybrid approach ensures that robots can operate reliably and safely in both structured and semi-structured environments [46].

The distinction between global and local path planning can be illustrated using the example of an autonomous service robot navigating in an indoor environment, such as a hospital, shopping mall, or warehouse. A global planner computes a collision-free and efficient trajectory from the starting location to the final destination, utilizing preexisting environmental information, such as floor layouts, static obstacles, and designated pathways. This allows the robot to follow an optimized route while considering factors such as travel efficiency and energy consumption.

However, real-world environments are dynamic and unpredictable. Changes such as rearranged furniture in a hospital, a temporarily blocked aisle in a warehouse, or increased foot traffic in a shopping center may disrupt the planned path. Since a global planner alone cannot adapt to such real-time changes, a local planner is used to process sensor input and adjust the trajectory accordingly. The local planner ensures that the robot can safely navigate around unexpected obstacles, modifying its path in real-time to maintain smooth and efficient movement.

Often, global and local planners work in combination. The global planner establishes a structured route, while the local planner continuously refines it based on newly acquired sensor data. This hierarchical approach is beneficial because global planners optimize long-term efficiency but may require substantial computation time, whereas local planners prioritize real-time reactivity and ensure that the robot remains responsive to changes in its immediate surroundings.

By integrating both planning strategies, service and assistive robots can operate safely and effectively in dynamic environments, ensuring uninterrupted navigation in hospitals, public spaces, and industrial settings [99].

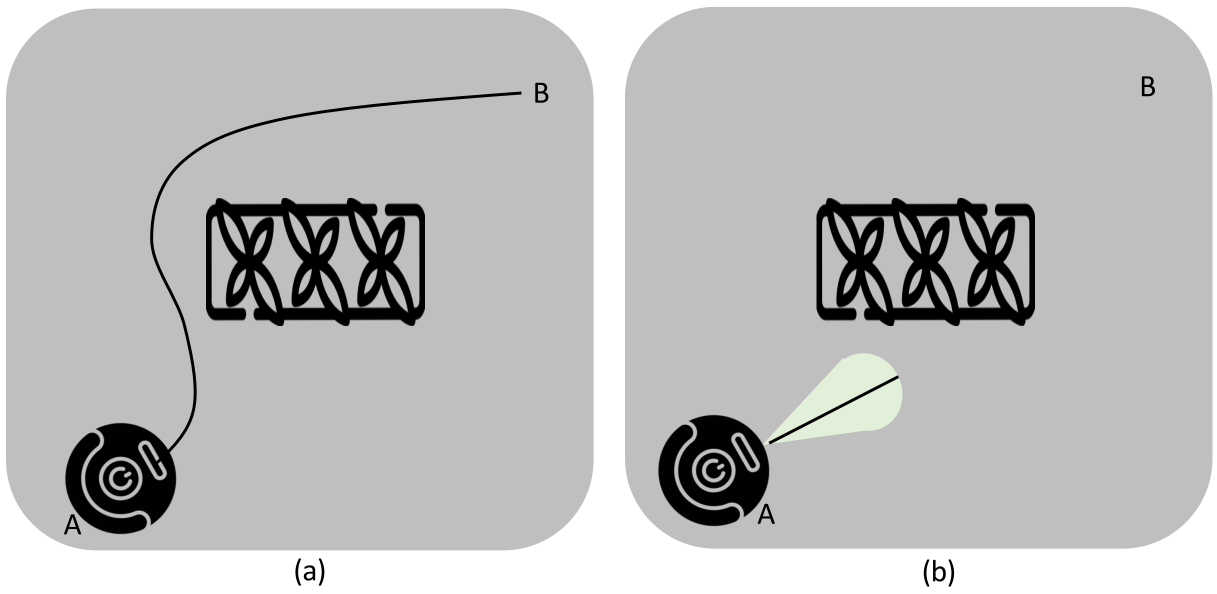


Figure 2.6: The global path planner (a) considers all offline-available information about the environment to plan a safe path connecting the starting point (A) and the goal point (B). A local path planner (b) often considers only local environmental information in the sensor range (shown by the green cone) to plan a safe path. Global path planners are useful for structured environments where detailed maps are available, such as hospitals, warehouses, or public transportation hubs. They provide efficient long-term navigation but may require additional real-time local adjustments in dynamic environments.

2.4.3 Differentially Unconstrained Path Planners

The path planning problem for autonomous mobile robots is widely recognized as an *NP-Hard* problem [107, 181, 192, 224]. Several approaches have been proposed to simplify the problem of path planning for autonomous mobile robots. An effective simplification involves treating the autonomous mobile robot as a mass point, which is handled by differentially unconstrained algorithms. These algorithms disregard mobile robot dynamics and solve the planning problem using geometric and topological methods. Solutions to differentially unconstrained planning problems can be found using search algorithms based on representations of the unobstructed space, such as *Roadmap methods*, *Cell decomposition*, *Potential field methods*, and *Probabilistic approaches* [84, 103].

In so-called *Roadmap* approaches, the unobstructed space is explored through a set of mobile robot movements, which are represented as a network of connected lines. The motion planning problem is then converted to a graph search, where algorithms such as A^* are used to find the optimal path according to a specific metric, such as the shortest path [61, 136]. The prominent roadmap methodologies include *Visibility*

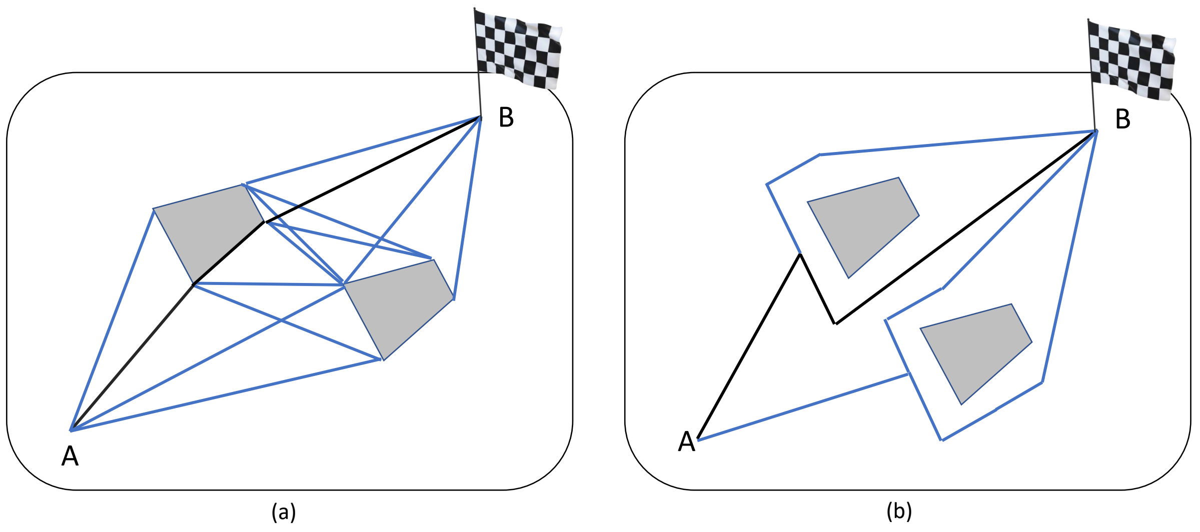


Figure 2.7: (a) Visibility Graph, where the free space is represented by lines connecting the mobile robot's possible movements, with the shortest path (in black) intersecting the obstacle vertices. (b) Voronoi Roadmap, which increases the distance between the obstacle and the path to ensure safe navigation.

Graph method and *Voronoi Roadmap*. In the visibility graph method, a series of lines connect all feasible mobile robot positions in the unobstructed space. However, the shortest path in this graph may intersect obstacle vertices, posing a safety hazard. To address this, obstacles are expanded by a safety margin, such as a sphere with a radius equal to the mobile robot's maximum dimension [84].

In contrast, *Voronoi Roadmap* ensures that the mobile robot maintains a safe distance from the unsafe workspace. The algorithm constructs a roadmap with a built-in safety margin, allowing the mobile robot to navigate efficiently while avoiding close proximity to obstacles, as shown in Figure 2.7 [13, 103].

The concept of *Cell Decomposition* in path planning involves partitioning the free configuration space into smaller, interconnected cells that are then used to form a graph. A graph search algorithm is applied to solve the path planning problem by navigating through these cells [84, 103, 169]. The process of finding an obstacle-free path requires identifying the cells that contain both the starting and the goal configurations of the autonomous mobile robot and then determining a sequence of connected cells that link them.

A specific method for cell decomposition is *trapezoidal* or *vertical cell decomposition*, which is frequently used in path planning for autonomous systems [201]. In this approach, the free space is vertically divided, with the vertical lines aligned to the obstacle. Then a roadmap is constructed connecting the midpoints of these cells, and a graph search algorithm, such as A^*, is used to calculate the shortest path (see Figure 2.8) [130].

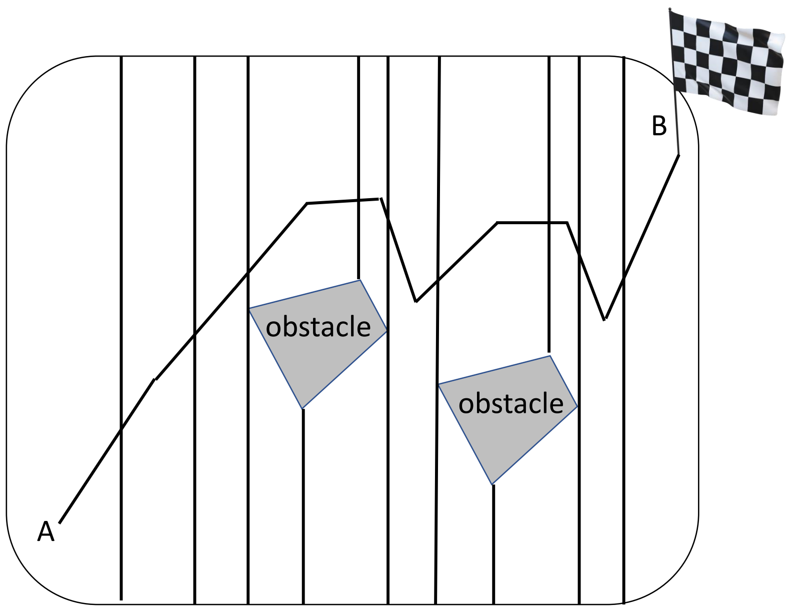


Figure 2.8: In vertical cell decomposition, the free configuration space is divided using vertical sections, and the obstacle-free path is represented by connections between the midpoints of each cell.

In the *Potential Field* method [80, 105, 118, 119], a cost function -such as the time to reach the destination - is minimized while penalizing proximity to the obstacle. As the mobile robot approaches an obstacle, the cost increases, creating a potential field that guides the mobile robot away from collisions (see Figure 2.9). Potential field methods are computationally efficient to implement; however, they are susceptible to getting trapped in local minima, where the attractive and repulsive forces balance out, preventing the mobile robot from progressing.

In *Probabilistic Approaches* to path planning, algorithms utilize probabilistic methods to find feasible and efficient paths for autonomous mobile robots or robots in environments that may be uncertain, dynamic, or partially known. These methods account for uncertainties in the environment, the robot state, or sensor data, using probabilistic models to incorporate this uncertainty into the planning process. Although the solutions generated may not always be nominally optimal, the probability of successfully solving the path planning problem approaches one as the computation time tends toward infinity [85, 206]. Examples of such methods include *randomized path planners* and *Rapidly-Exploring Random Trees* (RRT) [206].

A randomized path planner, often based on potential fields, addresses the issue of local minima in a probabilistic manner. In this context, the autonomous mobile robot follows a planned path that connects the initial configuration (starting point) to the

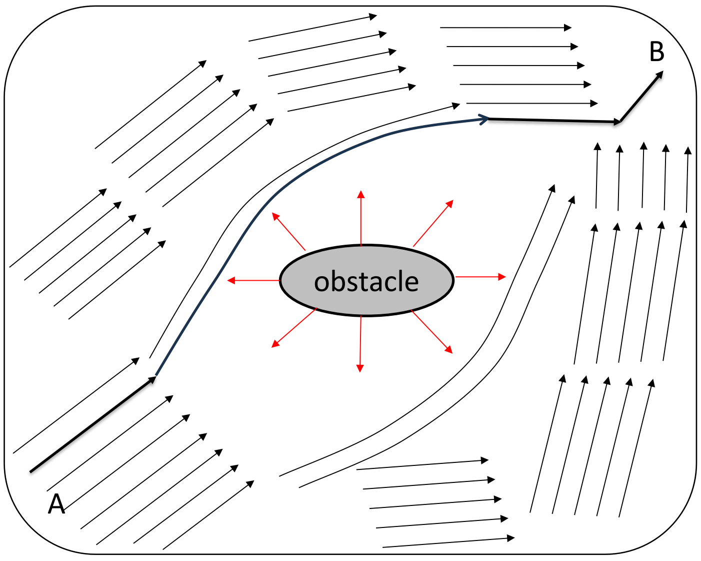


Figure 2.9: The potential field is generated using a cost function based on the distance to the goal (attractive force) and repulsive forces around the unsafe workspace to ensure safety.

desired destination. If the mobile robot becomes trapped in a local minimum-where the attractive forces guiding it to the goal and the repulsive forces from the obstacle balance each other-the planner introduces random movements to escape this state. These random movements may involve temporarily substituting the global goal with a virtual local goal or introducing a virtual obstacle to bypass the local minima [44, 238]. This iterative process continues until a lower-cost or lower-potential solution is found. As the size of the configuration space increases, the complexity of the problem increases significantly, resulting in longer computation times to find a solution [18].

Rapidly Exploring Random Tree (RRT) algorithms are primarily designed to build an expanding tree starting from the initial state, with the aim of exploring the available configuration space $\mathcal{C}_{free}$ until the desired destination is reached [115, 124]. During each iteration, a new point or vertex is randomly selected and collisions are checked. Should the point be situated within the free space, it is attached to the tree by connecting it to the nearest existing vertex. This step-by-step method continues until the tree reaches the final configuration, as depicted in Figure 2.10 a. It is crucial to understand that RRT algorithms do not emphasize optimization of the route between the initial state and other nodes, making them efficient for finding paths but often not crafting them optimally. To overcome this issue, RRT* refines the standard RRT approach by introducing two major improvements [115, 163]. Initially, RRT* tracks the cost, such as the travel distance, from the initial node to every other sampled node. Once the closest node is found, a set of nearby vertices is examined within a predetermined range of the new node. If one of these vertices offers a more efficient path to the starting node, it replaces the nearest node in the path. Furthermore, RRT* evaluates neighboring vertices to see if connecting them to the newly incorporated vertex can reduce their costs. If such a reduction is possible, the neighbor is reconnected to the new vertex, as illustrated in Figure 2.10b. Related research indicates that bidirectional artificial potential fields have been integrated with RRT* to enhance obstacle avoidance in tight spaces [225]. Furthermore, an adapted version of RRT* has been designed to minimize memory usage to store nodes [9]. In this variant, the RRT* algorithm runs until the tree grows to a set number of nodes. Should a viable path from start to goal not be identified, nodes deemed less useful are removed whenever a superior node, such as one showing a reduced total path cost, is introduced. This strategy has been expanded to address motion planning challenges in dynamic settings [10].

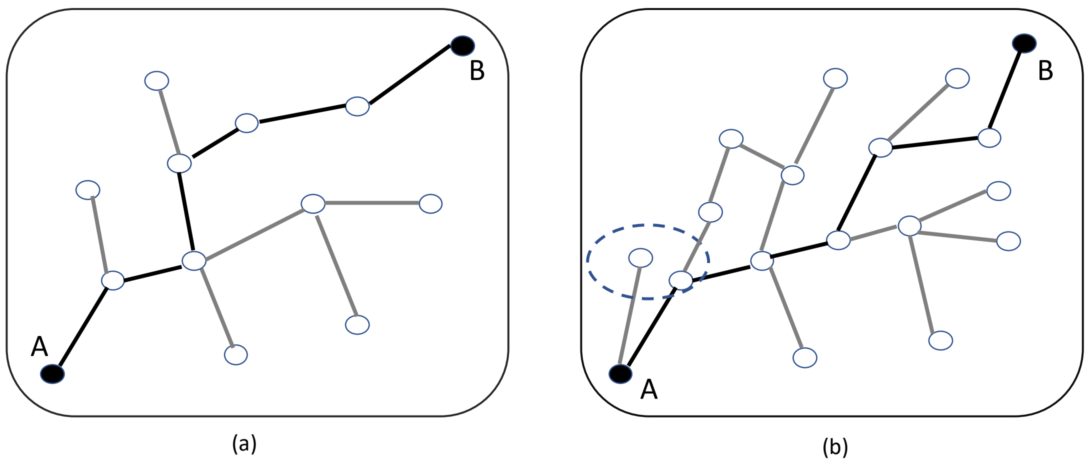


Figure 2.10: (a) RRT algorithm: the path displayed in black links the start point A to the goal point B; (b) The RRT* algorithm incorporates nearby vertices within a set radius (illustrated by a circle with dotted lines), resulting in fan-like formations.

2.4.4 Differentially Constrained Path Planners

In real-world mobile robot applications, addressing differential constraints when dealing with the path planning problem is important. Neglecting these constraints can yield overly cautious solutions, leading to suboptimal solutions with respect to energy efficiency or travel time. To account for these constraints, the concept of *Nonholo-*

nomic/Kinodynamic Planning, specifically for planning for wheeled mobile robots has emerged [130]. With respect to deferentially constrained planners, the following notions and definitions are important:

Nonholonomic Constraints are differential constraints that cannot be integrated or formulated to produce a finite set of algebraic equations. Non-holonomic constraints typically manifest as restrictions on the system velocity in the configuration space $\mathcal{C}$. A common example is the inability of wheeled mobile robots to move sideways instantaneously. As a result of these constraints, the system state space becomes restricted, rendering certain configurations inaccessible through direct paths. This limitation characterizes what is known as a non-holonomic system. Such systems are often classified as underactuated, meaning that the number of independently controllable degrees of freedom exceeds the number of control inputs. The presence of non-holonomic constraints significantly impacts path planning and control strategies for mobile robots, necessitating specialized algorithms to navigate the constrained state space effectively [130].

An *Underactuated System* refers to a mechanical system where the number of independent control inputs or actuators is strictly less than the total degrees of freedom of the system.

Kinodynamic planning addresses motion planning problems that incorporate both kinematic and dynamic constraints. Originally, it focused on satisfying second-order constraints in the configuration space $\mathcal{C}$, specifically dealing with limitations on velocity and acceleration. However, the concept has evolved to encompass a wider range of dynamic considerations. Modern kinodynamic planning extends beyond simple bounds on velocity and acceleration, integrating more complex dynamic models that may include forces, torques, and other physical properties affecting the system's motion. This expansion allows for more realistic and physically accurate motion plans, particularly crucial in advanced robotic systems and dynamic environments [56, 130].

To illustrate nonholonomic constraints, consider a car-like robot (Figure 2.11). This robot cannot move directly sideways because its rear wheels are fixed, allowing only rolling motion. Consequently, certain positions are unreachable through direct paths, necessitating a series of forward and backward movements. This constraint makes tasks such as parallel parking challenging, as the robot must execute specific movement sequences rather than simple lateral shifts. Such limitations significantly impact path planning and navigation strategies for wheeled mobile robots. Indeed, problems involving kinodynamic constraints are inherently more challenging to solve due to the temporal dependence in the differential states, especially in the presence of an obstacle. As a result, algorithms that incorporate approximations are often necessary to find optimal or suboptimal solutions, while also addressing hardware limitations effectively.

The *Grid-based State Space Search* method provides a deterministic approach to solving path planning problems under differential constraints [56]. This method dis-

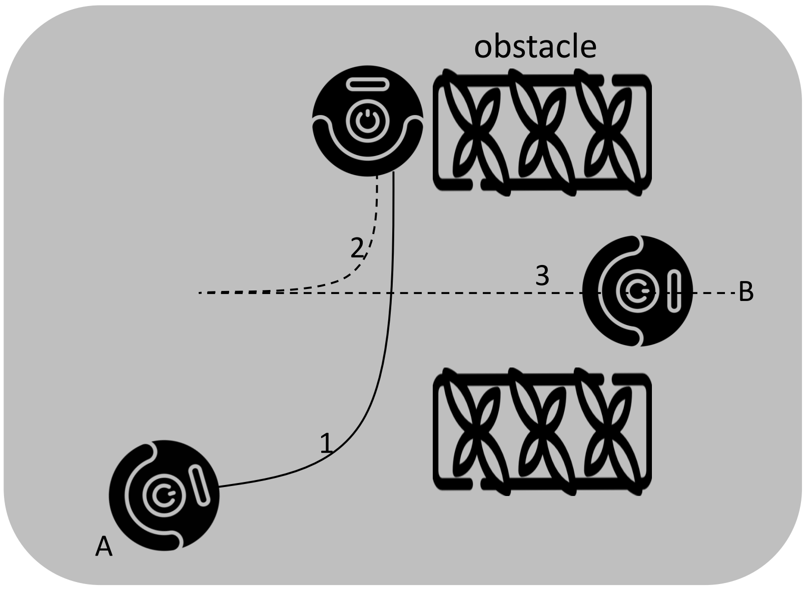


Figure 2.11: Nonholonomic constraints force the autonomous mobile robot to follow a complex path (solid and dashed lines) from point A to B, involving steering, forward, and backward movements, rather than a direct route.

cretizes the configuration space of the autonomous mobile robot into a lattice, transforming the continuous-time motion planning problem into a discretized hyperdimensional grid. A graph search is then conducted to identify a dynamically feasible path that connects the initial and final states, based on a specified performance metric such as the shortest path.

There are two primary categories of lattice generation techniques: *control-sampling* and *state-lattice* sampling [22]. In the control-sampling method, the control space is sampled to ensure that the resulting state space adheres to the mobile robot's differential constraints [31, 60]. However, designing control-sampling primitives can be challenging due to the often nonlinear relationship between the mobile robotâ€™s control inputs and its state evolution under kinematic and dynamic constraints [173].

In contrast, the *state-lattice* approach first discretizes the state space, and then a boundary value problem (BVP) is solved to connect states within the discretized space. For example, in [172], the problem of path planning for wheeled ground mobile robots was addressed by constructing a state lattice using an inverse path generator, which established feasible paths between lattice nodes/configurations characterized by polynomials of arbitrary curvature. Based on this lattice configuration, a heuristic search algorithm was used to identify the feasible optimal path connecting the initial and final lattice nodes.

In another example, [137] introduced a state-space lattice for a quadrotor navigating an obstacle-laden environment by generating a set of motion primitives. The discretized configuration space was then explored to identify dynamically feasible, smooth, minimum-time trajectories. The authors of [22] further enhanced this approach by introducing a motion primitive generation framework that automatically optimizes boundary constraints within the boundary value problem, along with the corresponding motions. This framework also optimizes the maneuverability at the end-point of the resulting path, which can be determined using fast optimization techniques.

The concept of *Dynamic Programming* (DP) can be employed to solve the path planning problem for mobile robots under differential constraints [21]. In this approach, the configuration space $\mathcal{C}$ is discretized into nodes, with each node representing a distinct configuration of the autonomous mobile robot. The optimal path, defined according to a specified performance metric, is constructed as a sequence of these nodes connecting the initial configuration to the final configuration. An optimization criterion is formulated to identify this sequence of nodes while ensuring adherence to the differential constraints, which is governed by the following equation:

$$L(S, G) = min \sum_{i=1}^{N} L(x_i, x_{i+1}).$$

where $L(x_i, x_{i+1})$ denotes the cost of transitioning from node x_i to node x_{i+1}, and N represents the number of nodes (cf. Figure 2.12).

However, the use of dynamic programming for path planning in mobile robots presents certain drawbacks. One notable limitation is that the resultant path can be more expensive (longer) than the theoretically optimal path [187]. This discrepancy arises because the DP algorithm requires the planned path to visit specific grid nodes, which may not align with the optimal path in a continuous space. Furthermore, the computational demands of the DP algorithm can be significant, particularly in large-scale environments [67].

To mitigate these limitations, a variation of dynamic programming augmented with discretization of the randomized grid is proposed in [187]. In this approach, the configuration space is initially discretized into a regular grid of nodes, with obstacles enlarged by a safety margin. Then each node is randomly displaced before the path search is executed. This randomized grid enables the exploration of a wider variety of paths, potentially discovering paths that are shorter or more efficient than those produced by a standard grid-based DP approach.

In a different application, [112] applied dynamic programming to plan the paths of a small autonomous aerial robot operating in a 2D environment with wind effects. The configuration space was discretized into nodes that represented the headings of the aircraft. DP was then used to calculate suboptimal paths comprising heading sequences, with the point of departure defined by distances d_i and angle of departure θ_i, where i ranges from 0 to n, denoting the number of points of departure. The problem was solved for all discrete initial and final headings, resulting in a database of turn progressions.

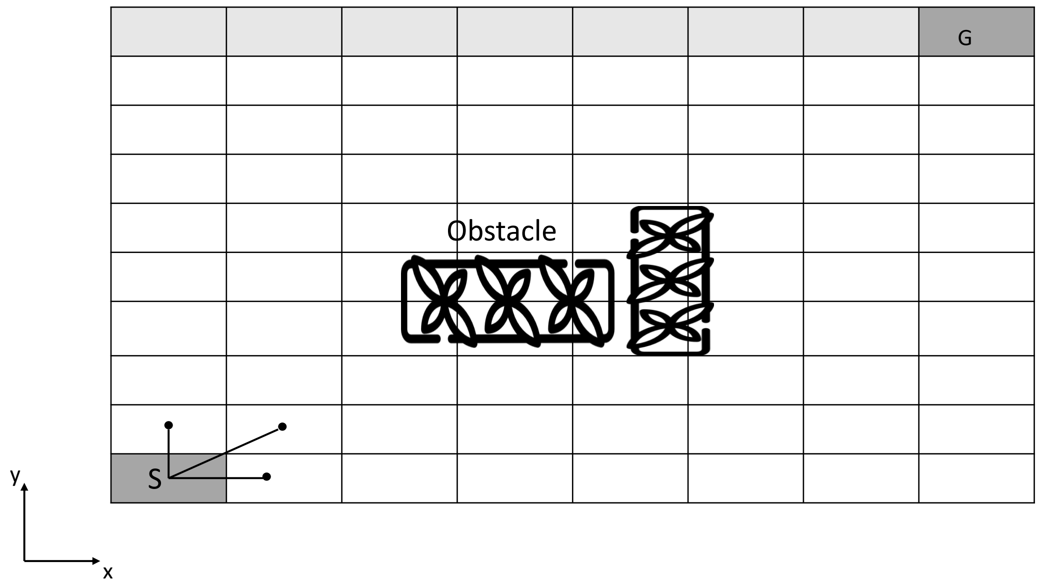


Figure 2.12: The configuration space is discretized into cells with resolution N, and the initial configuration is S, while the goal configuration is G. Potential paths are represented by solid black lines starting from S.

In [231], iterative dynamic programming was used for motion planning in autonomous mobile robots. This approach incorporated a path improvement strategy in which the configuration space was discretized using a Voronoi graph, and a search algorithm was used to find the shortest path. The planned path consisted of waypoints connecting the starting and destination locations. To improve the smoothness of the path, the authors applied Hermite interpolation between the waypoints. Dynamic programming was then used to modify these interpolated waypoints, identifying a time-optimal trajectory that accounted for both kinematic and dynamic constraints. This iterative process refined the trajectory step by step, ensuring that the motion plan adhered to system constraints while optimizing for time efficiency.

Rapidly-Exploring Random Tree (RRT) can also be used to address the differential constraints of autonomous mobile robots by building a tree structure by random sampling of the configuration space (see Figure 2.13) [131]. In [77], an RRT algorithm is proposed that considers the dynamics of the system and incorporates knowledge of the motions of obstacles. The algorithm expands the tree by identifying nodes or states that minimize a cost function while adhering to the constraints imposed by system dynamics.

A closed-loop RRT algorithm (CL-RRT) is introduced for autonomous urban driving in [125]. Unlike traditional RRT, this algorithm samples inputs to the stable closed-loop system of the controller and mobile robot rather than sampling the configuration space directly, generating dynamically feasible trajectories. Furthermore, Kinodynamic RRT* is introduced in [220], extending the capabilities of RRT* by solving a fixed final-state-free-final-time optimal control problem. This problem optimally connects pairs of states while minimizing a cost function, subject to the constraints imposed by the mobile robot linear dynamics.

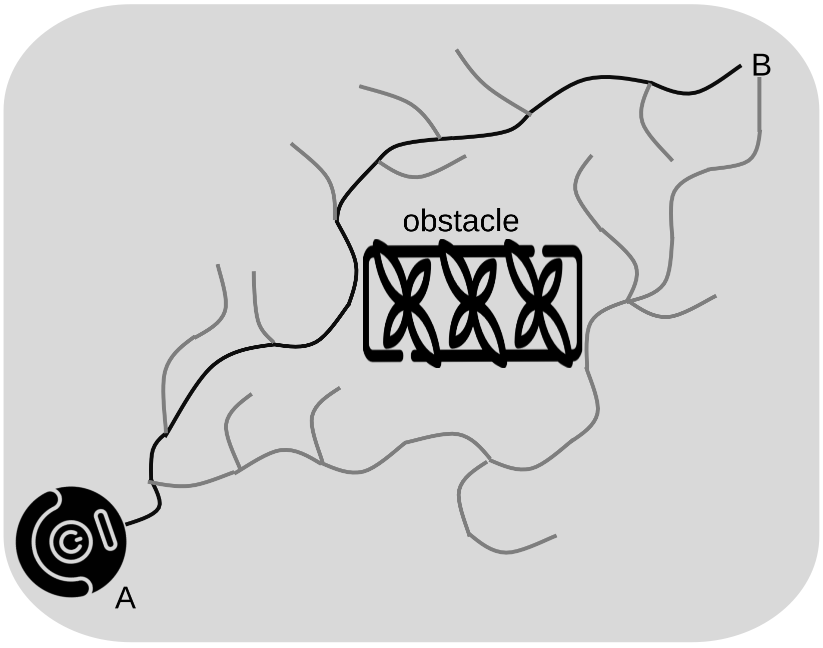


Figure 2.13: The configuration space tree is explored and expanded according to mobile robot dynamics, such as the steering angle, ensuring the near-optimal path, depicted in solid black, is dynamically feasible while avoiding unsafe workspace.

Another approach to solving the problem of autonomous mobile robot path planning while considering differential constraints involves searching for a discrete configuration space interpolated with polynomial arcs [84]. In this method, the planner-generated waypoints are fitted with a spline constructed from polynomial arcs, ensuring that the planned path is dynamically feasible and adheres to mobile robot constraints, such as the turning radius. For example, in [58], B-splines were used to smooth the planned path, ensuring $\mathcal{C}^2$ continuity and bounded maximum curvature for a carlike robot. Similarly, the work in [81] introduced a polar polynomial curve and a straight line to smoothly connect two configurations for differential-drive nonholonomic robots while satisfying dynamic constraints. Moreover, [217] combined path optimization with spline interpolation to generate an optimized (minimum length) trajectory for a carlike model, demonstrating the versatility of spline-based interpolation techniques for solving path planning problems with differential constraints.

The *Dynamic Window Approach* (DWA) restricts the input space to values that satisfy the dynamic constraints of the mobile robot, limiting the velocity space to values achievable by the robot while ensuring safety (cf. Figure 2.14) [74]. This method consists of two primary steps: first, the velocity space is pruned and then an optimal solution is selected from the remaining feasible velocities, restricting the search to safe circular trajectories that can be achieved within a short time interval [74]. The optimization objective is to select a pair (v, w) of translational and rotational velocities that guide the robot to the destination while maximizing clearance from the unsafe workspace.

The DWA is often integrated with other planning techniques to ensure feasible paths for autonomous mobile robots. For example, in [219], the RRT approach is combined with DWA, where RRT is used to plan a global path in the configuration space, and DWA computes translational and rotational velocity commands along the path. Similarly, in [135], the Dijkstra algorithm is used to plan a global path, while DWA calculates feasible velocities for a smart car. Furthermore, [233] integrates DWA with a global path planner, where the global planner generates a reference trajectory, and an evaluation function within DWA is used. This function considers factors such as the distance of the robot from the global path, the distance to the local final point, and the proximity to the unsafe workspace. To ensure smooth motion, the evaluation function also takes into account path directivity, smoothness of motion, and speed of reaching the final point.

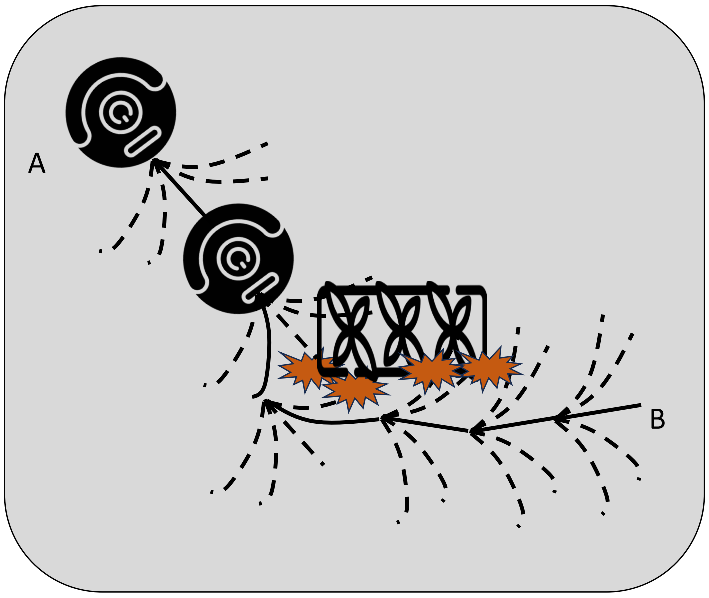


Figure 2.14: DWA is applied to steer an autonomous mobile robot from a starting point A to the destination point B. Feasible safe trajectories are shown in solid black.

Motion primitives can also be used in the context of path generation considering system dynamic constraints. These primitives consist of precalculated motions that the mobile robot can execute by applying control input from a set of permissible controls. They encompass various positions and states that enable the autonomous mobile robot to smoothly transition from a specific initial state to different states. In [27], the concept of motion primitives is employed to smooth a sequence of 3D waypoints, ensuring that the resulting trajectory is feasible from a helicopter dynamic perspective. This is achieved by selecting an appropriate sequence from a pre-computed library of motion primitives. Similarly, the work in [52] adopts a motion primitives-based exploration path planner to rapidly plan a path for aerial robots. In [47], a motion primitives-based graph is utilized for a 7 degree-of-freedom mobile manipulation platform, where transitions between states are restricted to a predefined set of feasible motion primitives. Furthermore, the authors in [34] presented the application of motion primitives and state lattices for ground robots operating in partially GPS-denied environments. The motion primitive-based controller facilitates navigation by discarding states leading to GPS-denied regions and enabling motion primitives based on lane or wall following.

In [168], the authors demonstrated 3D path planning for autonomous aerial robots in dense obstacle environments using motion primitives. Two families of motion primitives were introduced: one comprising 3D circular paths between points in space, along with the necessary control input, and the other consisting of aggressive turnaround maneuvers to effectively avoid unsafe workspace.

The path planning problem for autonomous mobile robots can also be tackled using mathematical programming techniques, which formulate the problem as a numerical optimization task. The goal is to determine the optimal route according to a predefined performance metric, such as minimizing the distance or travel time between the initial and final destinations, while adhering to constraints related to mobile robot dynamics, mechanical safety, and human safety. This approach can solve differentially unconstrained problems with linear equality or inequality constraints, resulting in a *linear programming* optimization problem [40]. Two widely adopted methods for addressing these problems in the presence of differential constraints are *Mixed Integer Programming* (MIP) and *Nonlinear Programming* (NLP). These techniques involve formulating and solving an optimization problem to identify the most suitable path for the autonomous mobile robot, considering its dynamic capabilities and environmental factors [87].

Mixed Integer Programming problems can involve linear, quadratic, or nonlinear objective functions, with constraints that may consist of combinations of linear or nonlinear equalities or inequalities. The subsets of variables may be integers or real numbers that represent the state of the autonomous mobile robot[109, 223].

The classification of MIP problems depends on the form of the objective function and the constraints. A problem with a linear objective function is referred to as a Mixed-Integer Linear Program (MILP), while a quadratic objective function results in a Mixed-Integer Quadratic Program (MIQP). If any constraints involve quadratic terms, the problem is classified as a Mixed Integer Quadratically Constrained Program (MIQCP).

Various optimization-based approaches for MIP have been proposed in different planning scenarios, ranging from minimizing travel time to considering mobile robot turning rates. The computational complexity of these problems can be influenced by the number of binary variables, which depends on factors such as the number of constraints, obstacle characteristics, and the operational environment [49].

In [107], a MILP formulation was introduced to alleviate the computational demands of solving path-planning problems for autonomous aerial robots. Similarly, [190] tackled a fuel-optimal path planning problem for multiple vehicles using mixed-integer linear programming. The work in [182] employed MILP with a linear aircraft model to find optimal trajectories for multiple aircraft to prevent collisions. This MILP formulation can be extended for multi-waypoint path planning, as shown in [108] and [229].

Moreover, [55] introduced a computationally efficient MILP formulation for the optimal path planning of robotic manipulators, while [207] used MIQP to plan trajectories for quadrotors with cable-suspended payloads in obstacle-filled environments. In [151], MIQP was used for teams of quadrotors operating in 3D environments with obstacles, incorporating collision avoidance through integer constraints. Furthermore, [174] used the MIQP formulation to solve the optimal trajectory planning problem for autonomous driving, considering robot dynamics, obstacle avoidance, multiple maneuver choices, overtaking, and lane change decisions.

Alternatively, the path planning problem for autonomous mobile robots can be formulated as a nonlinear program, where the objective function and/or some of the constraints are nonlinear [20]. In contrast, if both the objective function and the constraints are linear, the problem becomes a linear program [142].

Model Predictive Control (MPC), also known as receding-horizon control, can address the motion planning problem for autonomous mobile robots by solving an optimization problem over a finite time horizon using non-linear or linear programming techniques [36, 179]. The primary goal of an MPC-based motion planner is to determine a feasible reference trajectory or path for the control system while minimizing an objective function designed, such as minimizing time or path length, subject to a set of linear or non-linear constraints. These limitations reflect the operating characteristics of the system and ensure safety, such as avoiding unsafe workspace.

MPC generates a sequence of states and control input, starting from the current state of the mobile robot and ending at the end of the prediction period. An advantage of MPC-based/receding horizon motion planning is its ability to account for physical constraints while generating optimal paths or waypoints. However, model predictive control can be computationally demanding, leading to research efforts aimed at reducing their computational requirements.

For example, [235] employed an MPC-based planning algorithm to maneuver an underwater robot manipulator to a predefined pose while maintaining tight control over robot roll and pitch. In [102], a receding horizon-based planner was used to minimize energy consumption for an autonomous underwater robot under varying ocean disturbances and model uncertainty. In [35], a sample-based MPC approach was introduced to plan the path of an autonomous underwater robot by sampling the control space that satisfies input constraints and evaluating the generated nodes using a A^*-like algorithm.

For autonomous mobile robots, an MPC-based planner in [133] generated safe trajectories for low-level trajectory tracking controllers, considering moving obstacles and selecting appropriate mobile robot maneuvers. Similarly, [38] validated an MPC planner for autonomous mobile robots, capable of generating smooth trajectories that follow reference paths while avoiding unsafe workspace. In [143], a distributed MPC approach was proposed for real-time trajectory planning, incorporating obstacle avoidance through soft constraints and penalty terms in the objective function.

However, in many real-world scenarios, the location of mobile robots can be uncertain,

2.4.5 Completeness and Planning

Completeness is a desirable property in motion planning problems, as it ensures that a planning algorithm returns a solution whenever one exists. Motion planners can be categorized based on their completeness properties. A complete planner guarantees that a solution will always be found if one exists. In contrast, an incomplete planner does not provide such a guarantee, meaning it may fail to find a solution even when one is available. Another category is resolution-complete planners, which ensure the discovery of a solution at a specific level of resolution or discretization of the configuration space, provided one exists. Finally, a planner can be probabilistically complete, meaning that as the number of samples approaches infinity, the probability of finding a solution-if one exists-converges to one [46].

2.5 Uncertainty- and Perception-aware Planning

In path planning, it is often assumed that precise knowledge of the mobile robot's position, as well as the shape and location of the unsafe workspace is available. Moreover, it is typically assumed that the planned path will be executed with precision. However, these assumptions are often impractical in real-world scenarios. Consequently, path planners must account for uncertainties such as localization errors, imprecise motion execution, and inaccuracies in sensor measurements.

Thus, the objective of path planning goes beyond simply reaching the destination. It also involves navigating through uncertain or unknown environments to gather additional information, thereby increasing the likelihood of achieving the main goal. This expanded problem, which accounts for uncertainties, is referred to as *perception-aware path planning*.

A perception-aware path planner seeks to mitigate the challenges posed by uncertainty by generating paths that not only avoid obstacles but also maximize information gathering. This additional information helps reduce uncertainty in mobile robot location or enhances exploration to improve overall performance. Consequently, the planning problem can be categorized into two main types, depending on the nature of the uncertainties to be addressed [177].

2.5.1 Uncertainty-aware Planning

In uncertainty-aware planning, also known as belief-space planning, the planner aims to select trajectories that minimize uncertainty in the mobile robot's location. This problem is often tackled using techniques such as Partially Observable Markov Decision Processes or graph search in belief space. However, the computational complexity of these methods grows exponentially with the number of possible actions and observations [48].

To overcome this challenge and enable rapid trajectory computation, the Rapidly Exploring Random Belief Tree (RRBT) approach has been proposed [32]. This algorithm leverages the Rapidly Exploring Random Tree (RRT) method to plan paths within the uncertainty space.

When perceptual information is incorporated as a planning metric, the problem is termed active perception or, more broadly, active simultaneous localization and mapping (active SLAM). One of the objectives of active perception is to reduce uncertainty in the location of autonomous mobile robots by using sensor information, a concept known as active localization [138].

In [48], a perception-aware path planning framework was proposed, which uses both geometric and photometric information to improve location. This framework facilitates faster and more precise movements by planning paths using enriched environmental data.

The authors in [171] introduced a perception-aware planner operating in environments with uncertainty. They represented this uncertainty using ellipsoids that decrease in size as the robot approaches known landmarks.

In [167], the authors proposed an online algorithm that samples a finite-iteration random tree. Each branch in this tree is evaluated on the basis of the amount of area to be explored, and the best branch is selected to minimize expected robot localization and mapping uncertainty.

Furthermore, [183] developed a visual navigation algorithm based on artificial potential fields, which assigns attractive and neutral potential energy to areas rich in high and low features, respectively, to improve the visual location of mobile robots. Finally, in [234], the Fisher Information Matrix (FIM) was introduced as a perception metric to minimize the uncertainty of localization.

2.5.2 Adaptive environmental perception

In adaptive environmental perception, the primary objective of the planner is to generate motions that allow the autonomous mobile robot to gather more information about its environment. This approach finds applications in tasks such as searching, building inspections, and leak detection [177].

Various methods have been developed to address the adaptive navigation task, which can be broadly categorized into two main types: frontier-based methods and information-gain-based methods [39]. The *frontier-based method*, initially proposed by Yamauchi [228], identifies the boundaries between the known and unknown areas, known as the frontiers. The robot then navigates to the nearest accessible and unvisited frontier. Upon reaching a new frontier, the robot explores new regions and updates its map with newly acquired information. The coordination of a team of mobile robots for adaptive environmental perception in an occupancy grid environment is addressed in [33]. A comparative analysis of frontier-based strategies is presented in [97], while an improved frontier-based algorithm is introduced in [79]. A combination of frontier-based and sampling-based approaches is applied in [50].

Information-gain-based mapping strategies optimize an information-theoretic measure for exploration. In [29], the expected Shannon information obtained from an occupancy grid map is maximized, while minimizing uncertainty in both the robot pose and the map characteristics. In [165], an online exploration algorithm that maximizes expected information from unmapped space at the next best viewpoint is introduced. The authors in [113] maximize a mutual information reward function to encourage robots to explore new regions. Using an information-theoretic approach, [71] employs RRT*-IT to plan exploration paths for a Mars helicopter, aiming to reduce the standard deviation of the terrain type belief distribution.

In [12], a map is represented by obstacles and free-edge lists, where the free-edge list marks the boundary between the explored and unexplored regions. The expected information gain is quantified on the basis of the total length of segments in the free-edge list detected from a candidate observation point. In [218], an autonomous exploration task for a mobile robot in a 2D environment, expressed by an occupancy grid map, selects the next observation point to minimize the expected reduction in entropy.

Although adaptive environmental perception enhances the robot's ability to gather information and reduce uncertainty, it may also lead to unexpected obstacles or operational challenges. In such cases, fallback path planning ensures that the robot can safely recover and continue its task without external intervention, e.g., returning to charging stations, safe human interaction, or logistics tasks.

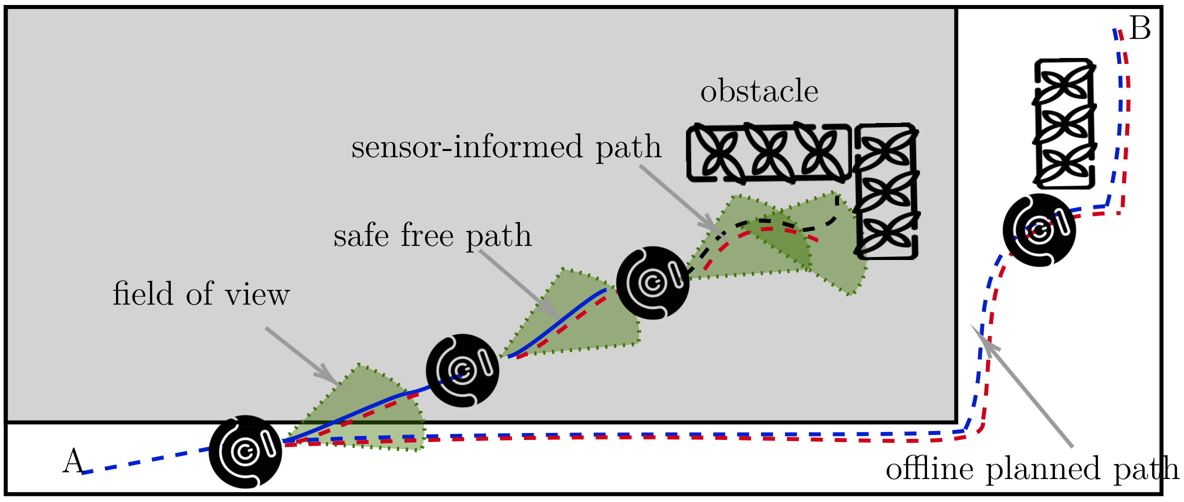


Figure 2.15: An autonomous mobile robot follows a preplanned safe path from point A to point B (dashed blue line). If an unknown area is encountered, the robot dynamically adjusts its route. If further obstacles prevent progress, the fallback controller reverts to a previously explored safe path to ensure continuity and safety (dashed red line).

2.6 Safe and Fallback Path Planning

In perception-aware and autonomous navigation, robots may encounter unforeseen situations that prevent them from proceeding safely. To address this, a fallback planner serves as a recovery mechanism, enabling the robot to return to a previously known safe state or adapt its path when necessary. Such situations may arise due to unexpected workspace constraints, environmental changes, or temporary sensor limitations. The fallback approach leverages either precomputed safe paths or recently explored routes to allow the robot to safely reposition itself and continue its operation (see Figure 2.15).

Several strategies have been proposed to enhance fallback planning for safety-critical applications. For instance, [226] introduces a method for handling sensor failures, ensuring that robots can continue operating safely despite temporary perception issues. Similarly, [82] presents a driving fallback controller designed to improve pedestrian safety in urban environments by mitigating unexpected perception errors. In [199], a safety-oriented fallback controller is proposed to protect autonomous systems from perception failures, enhancing overall reliability.

In the context of automated mobility, [232] emphasizes the importance of a multi-level fallback strategy for highly autonomous systems, detailing various degradation levels that allow for graceful performance reduction while maintaining operational safety. A similar approach is outlined in [227], where a fallback control mechanism is designed to assist automated vehicles in safely maneuvering to a designated parking area in case of sensor malfunctions.

By incorporating fallback planning, robots operating in hospitals, warehouses, and service environments can ensure safety, minimize disruptions, and recover from unexpected changes in their surroundings. These strategies enhance operational reliability and contribute to the trustworthiness of autonomous robotic systems in real-world applications.

2.7 Summary

Designing a safe and efficient path for an autonomous mobile robot is a complex task that requires consideration of factors such as robot dynamics, sensing capabilities, and operational safety, particularly in environments where robots interact with humans. This chapter provided an overview of the general path-planning problem, introduced key definitions, and categorized planning strategies based on available environmental information and system constraints.

Given the complexity of real-world navigation, various computational approaches have been developed to address the path-planning problem. These range from simplified models that approximate the robot as a point mass to full dynamic models that incorporate constraints such as nonholonomic motion and sensor-based adjustments. Additionally, perception-aware planning was discussed, emphasizing the role of sensor-driven navigation in handling uncertainties related to localization, motion execution, and environmental changes.

The chapter also introduced the concept of fallback path planning, which ensures that an autonomous robot can safely recover from unexpected obstacles or temporary sensor limitations by reverting to a previously known safe state. This enhances the robustness and reliability of robotic systems in dynamic environments such as warehouses, healthcare facilities, and public spaces.

Several optimization-based approaches have been developed to improve motion planning, including mathematical programming techniques that allow for structured decision-making under constraints. Among these, Model Predictive Control has emerged as a promising method, enabling rolling-horizon optimization for hierarchical planning and real-time adaptability. The subsequent chapters will explore the application of MPC-based approaches for efficient, perception-aware motion planning and control in autonomous robotic systems.

3 Model Predictive Control for Tracking, Path Following, and Planning

We should be taught not to wait for inspiration to start a thing. Action always generates inspiration. Inspiration seldom generates action.

Frank Tibolt

Model Predictive Control (MPC) has become a widely adopted control methodology for mobile robots due to its ability to handle constraints, optimize control actions, and ensure stability. This chapter introduces the fundamental principles of MPC for autonomous mobility in structured environments, focusing on applications such as warehouse automation, industrial logistics, and urban transportation systems.

The chapter provides a structured overview of MPC formulations for set-point tracking, trajectory tracking, and path following, emphasizing their role in improving efficiency, safety, and reliability in real-world applications. The discussion begins with an introduction to MPC's core principle: solving a constrained optimization problem in a receding horizon fashion. It then explores practical implementations in various robot control tasks: Set-point tracking enables a mobile robot to reach and stabilize at a desired location, improving precision in automated warehouses and industrial robotics. Trajectory tracking ensures a mobile robot follows a time-dependent reference, enhancing automated delivery robots and fleet coordination. Path following allows a robot to navigate along a predefined path, supporting urban mobility solutions and collaborative robots.

The chapter also discusses MPC-based path planning, which enables mobile robots to operate in partially known environments while accounting for constraints such as obstacle avoidance, motion feasibility, and real-time adaptation. To ensure computational efficiency, the chapter introduces approximate models and Mixed-Integer Programming formulations, making MPC scalable for real-world robotics applications.

The methodologies discussed here provide a robust foundation for safe, efficient, and adaptive control in structured civilian settings. The next sections present detailed mathematical formulations and case studies, demonstrating how MPC enables autonomous systems to perform complex tasks while ensuring safety and reliability.

3.1 Principle of Model Predictive Control

Model Predictive Control (MPC), also referred to as receding horizon control, has emerged as a prominent advanced control methodology widely utilized in both academic research and industrial applications. Initially developed and popularized in the 1970s and 1980s, MPC gained widespread adoption in the process industry due to seminal contributions such as dynamic matrix control, model predictive heuristic control, and generalized predictive control [107]. Its versatility in managing soft and hard constraints, which encapsulates physical system limitations and safety considerations, and its capability to handle nonlinear system dynamics and multi-input-multi-output (MIMO) systems makes MPC a superior control approach compared to alternative methods. Numerous studies offer comprehensive information on MPC principles and diverse applications, as documented in works such as [114, 141, 150, 185]. Simulation studies considering autonomous systems highlight the differences between the approaches and provide insights into the controller design for the different tasks.

Initially, the utilization of MPC algorithms was confined to systems characterized by simple or slow dynamics due to constraints imposed by available solvers and computational resources. However, with continued advancement of computational capabilities and the development of efficient formulations and tailored solution methods, MPC can now address more complex and fast tasks, spanning domains such as aerospace and robotics [53, 54, 70, 141]. The fundamental principle of MPC revolves around iteratively solving a finite-horizon optimization problem at each time instant, as illustrated in Figure 3.1 [36, 175, 180]. The optimization problem is subject to constraints imposed by the system dynamics, system limitations, and additional constraints that ensure system stability. The solution to the optimization problem involves determining an open-loop optimal control signal. The optimal control input minimizes a user-defined objective function while respecting the imposed constraints. This objective function typically minimizes or maximizes a specific metric, such as travel time or distance. In practice, only the first part of the optimal control input is applied to the system. Subsequently, the new state of the system is fed back into the optimization problem, which is solved iteratively in a rolling horizon framework. The accuracy of the mathematical model used in MPC formulations as well as the correct choice of the cost function is crucial: a more accurate model reduces prediction errors and improves control performance, possibly at the cost of increased computational overhead or the need to determine more parameters of the model. Both the reiteration/sampling time and the prediction horizon influence the computational load, underscoring the trade-off between control effectiveness and computational efficiency. In addition, the iterative real-time optimization characteristic of MPC can pose challenges, particularly when dealing with large-scale or very fast problems [70, 149]. These problems often require substantial computational resources to ensure that the optimization can be solved within the available sampling time. As a result, achieving a balance between computational complexity, optimization accuracy, and real-time performance is crucial for the practical implementation of MPC. Despite the challenges and considerations described, Model Predictive Control (MPC) is a widely applied control methodology that finds application across various domains, including aerospace and robotics, see, i.e.[15, 53, 176].

3.2 Basic MPC Formulation

We start by considering continuous-time formulations for MPC, following the expositions provided in [68] for the setpoint stabilization case. For the practical implementation, we will use sampled-data formulations, where the input will be updated or kept constant between sampling instances, where new measurements become available.

The core principle of MPC involves solving an optimization problem at each time $t \geq 0$ (or specific sampling times) to minimize or maximize a user-defined objective function $J(\cdot)$. This optimization yields a control signal over possibly a control horizon, and the first control input is then applied to the real system at the subsequent time step. In this section, we present the fundamental formulation of MPC. We consider nonlinear, continuous-time, time-invariant systems of the form:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \qquad x(0) = x_0, \\ y(t) &= h(x(t), u(t)). \end{aligned} \tag{3.1}$$

Where $t \in \mathcal{R}$ is the time, $x(t) \in \mathbb{R}^{n_x}$, $u(t) \in \mathbb{R}^{n_u}$ and $y(t) \in \mathbb{R}^{n_y}$ are the system states, control inputs and output states respectively with state, output, and control dimensions n_x, n_u and n_y. The nonlinear mapping function $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ represents the system dynamics. The output mapping function $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ represents the relation between the system states and the output $y(t)$. Notably, the output does not necessarily denote the measured variables; we rather use it for defining the control

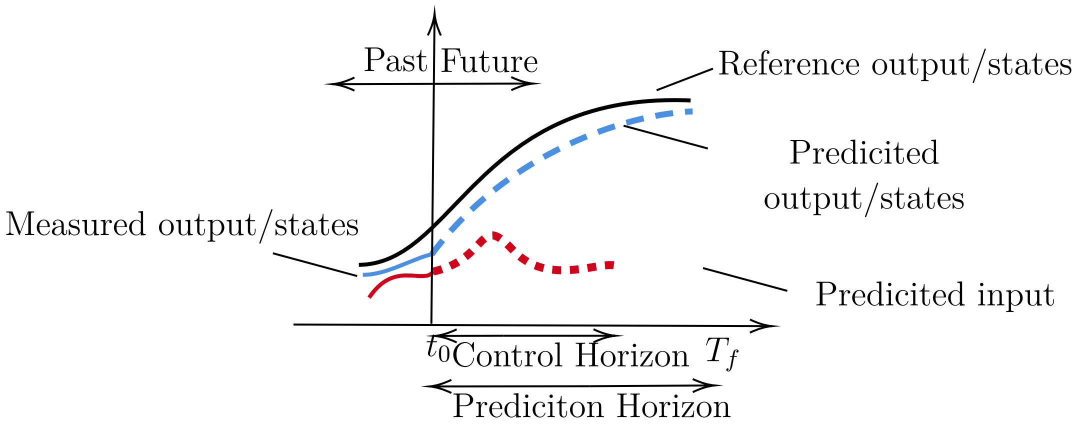


Figure 3.1: Moving horizon principle of MPC: The System behavior is forward predicted using a model of the system over a finite prediction horizon.

objective, or to capture important states or state projection for obstacle avoidance. Rather, we assume that all system states are measured and available at each time instant. The nonlinear system (3.1) is subjected to constraints. One might consider constraints due to safety reasons by excluding some regions of the state, input, or output space that are considered dangerous for the system under control or its environment. These restrictions can take the form of state, input and output constraints, which are represented in this work through the sets $\mathcal{X} \in \mathbb{R}^{n_x}$,$\mathcal{U} \in \mathbb{R}^{n_u}$, and $\mathcal{Y} \in \mathbb{R}^{n_y}$. Moreover, we rely on the following assumptions on the nonlinear system in (3.1):

Assumption 1. *The state and output constraint set $\mathcal{X}$ and $\mathcal{Y}$ are closed, the input constraint set $\mathcal{U}$ is compact.*

Assumption 2. *The system dynamics $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}^{n_x}$ and the output mapping function $h : \mathbb{R}^{n_x} \mapsto \mathbb{R}^{n_y}$ are assumed to be sufficiently continuously differentiable and locally Lipschitz for all $(x, u)^\mathsf{T} \in \mathcal{X} \times \mathcal{U}$.*

Assumption 3. *For any continuous input signal $u(\cdot)$ and for all initial points $x_0 \in \mathcal{X}$, the system (3.1) admits a unique absolutely continuous solution*

Model Predictive Control can address the control of the nonlinear system (3.1) considering the constraints such as system dynamics and the constraints on the input, states, and output sets. The optimal control problem minimizes a user-defined objective function to find an optimal open-loop control input sequence and the corresponding state trajectory.

The objective function to be optimized over the finite, in our considerations fixed, prediction horizon T_f is represented as follows:

$$J(\cdot) = \int_{t_o}^{T_f} F(x(\tau), u(\tau)d\tau + E(x(T_F)) \tag{3.2}$$

where t_0 is the current time and the function $F(\cdot)$ represents the stage cost that penalizes the state and the input with respect to the reference functions. The solution of the optimization problem minimizing (3.2) with respect to the input leads to an open loop control input $u(\cdot)$ over the prediction horizon. The predicted input signal could be infeasible, i.e., can't be achieved by the dynamical systems. Therefore, the optimal control problem (3.2) is augmented with constraints. These constraints may account for various physical limitations of the system, such as maximum speed/voltage or torque. Additionally, constraints can be imposed to regulate the rates of change in certain states, ensuring passenger comfort [23], or to ensure that the system operates within a safe region while avoiding obstacles; see, e.g. [7, 100]. Generally, the problem should be well defined to be solvable by an off-the-shelf solver in a reasonable time to meet the real-time requirements.

Summarizing, at each time instant t the following constrained continuous-time optimization problem is solved:

$$
\begin{aligned}
\min_{u(\cdot)} \quad & \int_{t_0}^{T_f} F(x(\tau), u(\tau)) d\tau + E(x(T_f)) && \text{(3.3a)}\\
\text{s.t.} \quad & \dot{x} = f(x(\tau), u(\tau)), && \text{(3.3b)}\\
& g(x(\tau), u(\tau)) \leq 0, && \text{(3.3c)}\\
& x(\tau) \in \mathcal{X}, u(\tau) \in \mathcal{U}, && \text{(3.3d)}\\
& x(T_f) \in \mathcal{E}. && \text{(3.3e)}
\end{aligned}
$$

The solution of (3.3) over the prediction horizon T_f yields a state trajectory $x^*(\cdot)$ and a feasible open loop control input trajectory $u^*(\cdot)$ that satisfies the system dynamical constraints in (3.3d) and safety constraints imposed in (3.3c). At the end of the prediction horizon, the predicted state is enforced to belong to the terminal region in (3.3e). The first part of the open loop optimal control signal is fed back to the real system and the new state is measured or estimated and the optimization problem (3.3) is solved at the next time in a rolling/moving horizon fashion.

As the objective function is central in determining the optimal control input $u^*(\cdot)$, one can formulate different tasks, such as setpoint-tracking, trajectory-tracking, path-following, or economic operation [148]. For example, one can use a time-dependent state or output reference trajectory. The cost function then penalizes the error between the predicted states and the reference at each stage or prediction step. This formulation leads to the *trajectory-tracking* MPC problem [106, 148]. Alternatively, in the *path following* MPC problem, the reference is defined by a parametrized path, via the path parameter. This introduces an additional degree of freedom, allowing the optimizer to adjust the path parameter to minimize the error between the predicted states and the reference path without explicitly adhering to specific states or outputs at particular times, as required in trajectory-tracking [63, 148]. This flexibility can be advantageous in applications where specifying a time-dependent reference trajectory is impractical or overly complex [63]. Furthermore, the reference can be defined as a constant state value that remains unchanged along the predictions. Doing so leads to a *set-point* MPC formulation. In the following sections, we underline the formulation of MPC for different problem formulations, spanning from set-point tracking to trajectory-tracking control, before outlining the moving-horizon optimization principle for path-planning problems.

3.3 Set-point Tracking

Set-point stabilization or tracking involves a steady state. The control objective is to steer the system state $x(t)$ to a reference state value x_s. The objective function in the case of the set-point stabilization penalizes the error between the predicted states and their reference. MPC set-point formulations are used in many applications [146, 193].

For the desired setpoint the following steady state condition needs to hold:

$$0 = f(x_s, u_s). \tag{3.4}$$

Here x_s is the set-point or desired state, and u_s is the corresponding control input. We use the subscript s to refer to set-point.

The optimal control problem for the set-point tracking can be expressed in continuous time as follows:

$$\begin{aligned}
\min_{\mathbf{u}(\cdot)} \quad & \int_{t_0}^{T_f} F_s(x(\tau), u(\tau), x_s) d\tau + E_s(x(T_f), x_s) && (3.5a) \\
\text{s.t.} \quad & \dot{x} = f(x(\tau), u(\tau)),\ x(t_0) = x(t), && (3.5b) \\
& x(\tau) \in \mathcal{X}, u(\tau) \in \mathcal{U}, && (3.5c) \\
& x(T_f) \in \mathcal{E}_s, && (3.5d) \\
& \forall \tau \in [t_0, T_f]. && (3.5e)
\end{aligned}$$

Here we assume zero control input at the reference point, i.e., $u_s = 0$. The stage cost $F_s(\cdot)$ penalizes the errors between the predicted states and input and the reference values, while the terminal cost $E_s(\cdot)$ penalizes the error at the end of the prediction horizon T_f. The solution of the open loop OCP (3.5) results in a feasible input trajectory $u^*(\cdot)$ and a state trajectory $x^*(\cdot)$ that respects the input and state constraints in (3.5c). Moreover, the state trajectory at the end of the prediction horizon is restricted to be in a set $\mathcal{E}_s$ in (3.5d), which is often used to ensure stability. This terminal region $\mathcal{E}_s$ should be a subset of the state constraint set $\mathcal{X}$ and the pointwise preimage of the output constraint set $\mathcal{Y}$. Only the first control input candidate in the predicted control trajectory is applied to the real system. The new states of the system are measured or estimated at the next instance $t + 1$ and the optimization problem is solved repetitively.

3.3.1 Stability of MPC Set-Point Tracking

The fundamental principle of MPC involves solving a constrained or unconstrained open-loop optimization problem at each time instant. However, as highlighted in [68, 149], achieving optimality in MPC does not necessarily guarantee stability, except in certain exceptional cases, such as the infinite horizon scenario where $N_\text{p} \to \infty$. However, solving an infinite open-loop optimization problem online is impractical and also often the finite-time behavior, not the asymptotic behavior, is important. Hence, since MPC solves a finite optimization problem, other properties must be carefully considered to ensure stability. Various approaches have been proposed in the literature to establish the stability of MPC [62, 68, 149], which generally involves the proper selection of the terminal cost, the terminal set, and the weighting matrices.

Similarly as in [68], to discuss stability of MPC for the system in (3.1) we relay on the following assumptions:

Assumption 4. *(Stage Cost)*
The stage cost $F_s : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}_0^+$ is continuous and $F_s(0,0) = 0$. It is further lower bounded by a class $\mathcal{K}_\infty$ function α_1, such that $F_s(x,u,x_s) \geq \alpha_1(\|x - x_s\|)$, $\forall(x,u,x_s)$

Assumption 5. *(Terminal Cost)*
The terminal cost $E_s : \mathbb{R}^{n_x} \mapsto \mathbb{R}_0^+$ is positive semi-definite and $E_s(0,0) = 0$. Furthermore, it is continuously differentiable.

Assumption 6. *(Terminal Set)*
The terminal set $\mathcal{E}_s \subseteq \mathcal{X}$ is closed.

Assumption 7. *(Existence of local controller)*
For any state $x \in \mathcal{E}_s$, there exist a local control input $u^{\mathcal{E}_s}(\cdot) \in \mathcal{U}$ such that $\forall \tau \geq 0$,

$$\frac{\partial E_s}{\partial x} . f(x(\tau), u^{\mathcal{E}_s}(\tau)) + F_s(x(\tau), u(\tau)) \leq 0.$$

This inequality ensures that the local controller $u^{\mathcal{E}_s}(\cdot)$ stays inside the terminal set $\mathcal{E}_s$, and the terminal cost $E_s(\cdot)$ will decrease by at least $F_s(x(\tau), u(\tau))$.

Assumption (4) is typically required for set point stabilization as it enforces that minimizing $F_s(\cdot)$ leads to convergence of the state to the desired set point. Whenever assumptions (1-7) hold, one can state the following theorem:

Theorem 1. *(Convergence of sampled-data NMPC for set point tracking)*
Given a system in (3.1), if the OCP (3.5) is feasible at t_0 with a proper stage, terminal costs as well as the terminal set satisfying assumptions (4-7), then it is recursively feasible, and the controller will lead to error convergence as the stage cost $F_s(\cdot)$ penalizes the system states with respect to the reference state value x_S.

The proof for convergence in the sampled-data NMPC case can be found in [68].

3.3.2 Illustrative Example

In the following, we employ a sampled-data model predictive control formulation to enable real-world application to a continuous-time system via a discrete-time implementation, which is necessary for controller design. The optimization problem to be solved at all sampling times looks like follows:

$$\min_{\{u_k\}} \quad \sum_{k=0}^{N_\mathrm{p}-1} l_s(x_k, x_s, u_k) + e_s(x_{N_\mathrm{p}}, x_s). \tag{3.6a}$$

$$\text{s.t.} \quad x_{k+1} = f_d(x_k, u_k), \;\; x_0 = \hat{x}(t), \tag{3.6b}$$

$$x_{k+1} \in \mathcal{X}, u_k \in \mathcal{U}, \tag{3.6c}$$

$$x_{N_\mathrm{p}} \in \mathcal{E}_s, \tag{3.6d}$$

$$\forall k \in [0, ..., N_\mathrm{p}]. \tag{3.6e}$$

The optimization problem (3.6) is solved over the prediction horizon N_p, utilizing the discrete-time stage cost function $l_s(\cdot)$ and the terminal cost function $e_s(\cdot)$. The system dynamics, as described in (3.6b), represent the discrete-time counterpart of the continuous-time dynamics in (3.5b), obtained through appropriate discretization or implicit integration within the chosen dynamic optimization scheme.

Similar to the continuous-time formulation, the predicted states and inputs are constrained to remain within the state and input sets specified in (3.6c). The predicted optimal control input accounts for system limitations, such as maximum acceleration and turning rates, ensuring feasibility with respect to the system dynamics. Additionally, the predicted states at the end of the prediction horizon must belong to the terminal set defined in (3.6d).

Solving the optimization problem (3.6) yields an optimal control trajectory $u^* = \left\{u_0, u_1, \ldots, u_{N_p-1}\right\}$ that minimizes the cost function, along with a corresponding state trajectory $x^* = \left\{x_0, \ldots, x_{N_p}\right\}$.

Only the first control input from the optimal sequence is applied to the system, after which the measured states are updated. The set-point tracking problem (3.6) is solved recursively until the system states converge to the desired reference values.

We demonstrate the MPC set-point tracking formulation (3.6) considering an mobile robot that can be mathematically represented by a kinematic bicycle model as outlined in [111]:

$$\dot{p}_x = v\cos(\psi). \tag{3.7a}$$
$$\dot{p}_y = v\sin(\psi). \tag{3.7b}$$
$$\dot{\psi} = v\tan(\delta)/L. \tag{3.7c}$$
$$\dot{v} = u_1. \tag{3.7d}$$
$$\dot{\delta} = u_2. \tag{3.7e}$$

Equations (3.7a) and (3.7b) represent the dynamics of the center of mass of the vehicle while the heading angle dynamics is given by (3.7c). The control inputs u_1 and u_2 are the acceleration and steering angle rates, respectively. The differential equations can be either implicitly integrated in the optimization, or the system equations can be discretized, leading to:

$$\begin{aligned} x_{k+1} &= f_d(x_k, u_k), \qquad x_0 = \hat{x}(t). \\ y_k &= h_d(x_k). \end{aligned} \tag{3.8}$$

Here, x represents the state vector $[p_x, p_y, \psi, v, \delta]^\top$ and u is the control vector $[u_1, u_2]^\top$ and y is the output of the system.

Example 1. *(Set-point tracking problem)*

The system should be steered from an initial point A to a final destination point B while avoiding obstacles. Therefore, the following optimal control problem is solved at

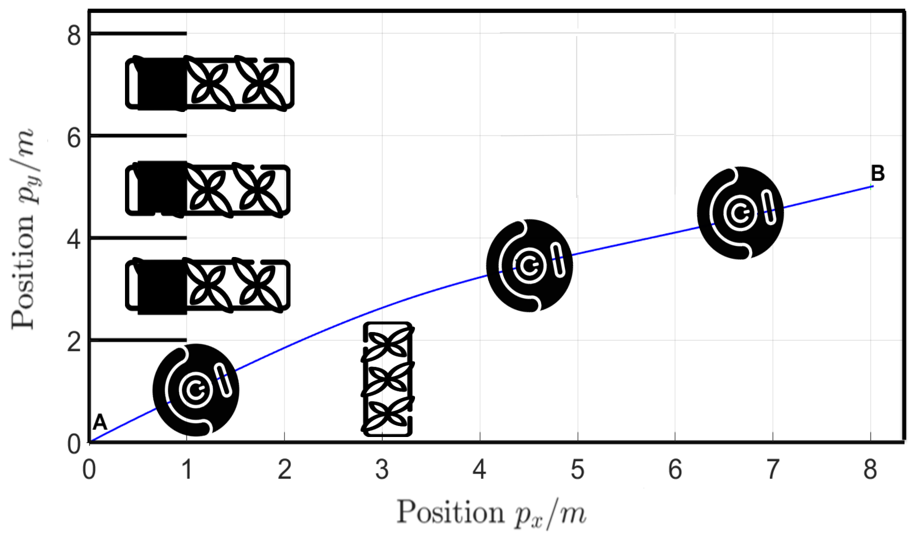


Figure 3.2: The autonomous mobile robot moves from a start given point A to a final destination point B while avoiding static obstacles.

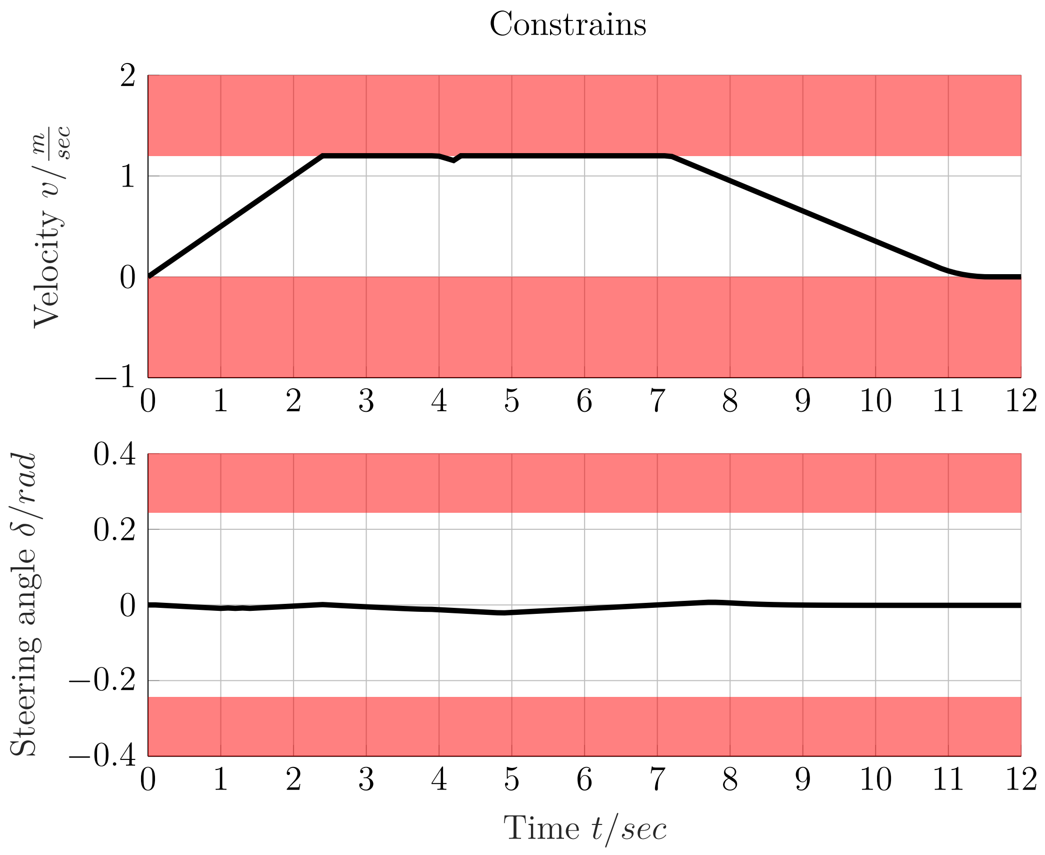


Figure 3.3: Velocity and steering angle. The red areas represent the input limitations.

sampling times t:

$$
\begin{aligned}
\min_{\{u_k\}} \quad & \sum_{k=0}^{N_p-1} \left(\|p_k - p^s\|_2^Q + \|u_k\|_2^R \right) && \text{(3.9a)} \\
\text{s.t.} \quad & x_{k+1} = f_d(x_k, u_k), x_0 = \hat{x}(t), && \text{(3.9b)} \\
& p_{min} \leq p_k \leq p_{max}, && \text{(3.9c)} \\
& v_{min} \leq v_k \leq v_{max}, && \text{(3.9d)} \\
& \delta_{min} \leq \delta_k \leq \delta_{max}, && \text{(3.9e)} \\
& U_{min} \leq u_k \leq U_{max}, && \text{(3.9f)} \\
& \forall k \in [0, ..., N_p]. && \text{(3.9g)}
\end{aligned}
$$

Here, p represents the center of mass coordinates with respect to the global frame, i.e., $p = [p_x, p_y]^\top$, while p_s denotes the final goal coordinates, which are time-independent. The objective function (3.9a) not only minimizes the difference between the predicted vehicle position and the goal position but also seeks to reach the final point with minimal control effort, thereby reducing the vehicle's energy consumption. The weighting matrices $Q \geq 0$ and $R > 0$ are positive semidefinite and positive definite, respectively, representing the trade-off between state and control input penalization.

Note that we focus only on reaching the point B, the speed is not constrained, thus it does not appear in the cost function. This does not allow to utilize Theorem 3.3.1 to proof stability.

In addition to the system dynamics constraints physical limitations such as maximum and minimum velocity and steering angle are incorporated, ensuring the feasibility of the predicted trajectory (3.9c) . The input constraints in (3.9f) allow for a trade-off between system aggressiveness and passenger comfort [23].

In this example, the reference is defined by constant reference states that are constants throughout the prediction horizon, such as the state at reference point B, as illustrated in Figure 3.2. Hard constraints account for the physical limitations of the mobile robot, including velocity and steering angle. Additionally, obstacle avoidance constraints are integrated into the optimization problem to ensure the autonomous mobile robot safety. By solving the constrained optimization problem (3.9) at each simulation step, the controller safely guides the robot from the initial point A to the destination point B, as shown in Figure 3.2. The controller optimizes acceleration to reach the final with maximum speed while adjusting speed and steering angle to navigate safely around known obstacles, as illustrated in Figure 3.3. Consequently, the planned path is both dynamically feasible and safe.

The MPC optimization problem (3.9) has been implemented numerically and translated into highly efficient C-code using the ACADO toolbox [98]. The implementation was executed on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz, achieving an average computation time of $t_{cpu} = 2.4732$ ms.

3.4 Path-Following Formulation

In the set-point tracking formulation so far presented, the reference is considered constant, i.e. to reach a specific vehicle position, as demonstrated in Example 1. However, often the system is required to follow a reference state trajectory or path. For example, an autonomous mobile robot may need to follow a predefined path to accomplish specific tasks, where the exact timing of reaching particular points along the path is less critical, provided the vehicle completes its task within a specified timeframe or as quickly as possible.

This requirement gives rise to the path-following problem, where the reference is represented as a geometric curve without predefined timing information [63, 148]. In path-following formulations, the reference velocity is not predetermined. Instead, the planner supplies a geometric curve, and the controller dynamically adjusts the velocity along the path during operation. MPC path-following formulations have been applied to autonomous systems in [66, 89, 198] and to robotic manipulators in [64, 215]. Typically, the reference path denoted as $\mathcal{P}$, is defined as a parameterized curve within the output space $\mathcal{Y}$, expressed as follows:

$$\mathcal{P} := \Big\{y^p \in \mathbb{R}^{n_y} | y^p = f^p(\theta(t))\Big\}. \tag{3.10}$$

In the context of the path-following formulation, we use the superscript p to refer to the path-following. In (3.10) the scalar variable θ is the path parameter. In the following, the mapping function f^p is assumed to be a continuous differentiability, denoted as C^∞. The reference defined in (3.10) is indirectly depending on the time via the path parameter $\theta(t) \in \Theta$. In turn, the evolution of the path parameter over time is not fixed a priori. We will consider constraints on the path parameter of the form $\theta := [\theta_{start}, \theta_{end}]$ and $\dot{\theta} \geq 0$. In the controller, the path parameter is increased, until the final value θ_{end} is achieved, such that the whole parametrized reference is traversed. The parameterization of the reference can be introduced by introducing a virtual dynamical single-input single-output system that delineates the evolution along the intended path as follows:

$$\begin{aligned} \dot{z}(t) &= r(z(t), q(t)), \\ \theta(t) &= l(z(t)). \end{aligned} \tag{3.11}$$

In this context, the variable $z(t) \in \mathbb{R}^{n_z}$ denotes the virtual system states, $q(t) \in \mathbb{R}^{n_q}$ represents the virtual system input, and $\theta(t)$ signifies the virtual system output. It is important to emphasize that the progression of the reference path is achieved upon the input of the virtual system q. The optimizer can freely choose the values of the virtual input to minimize the error between the real system states and the reference path. Constraints can also be imposed on the virtual states, inputs, and outputs. For simplicity, the virtual system dynamics (3.11) is typically chosen as an integrator chain.

In this case, forward motion can be enforced by choosing the virtual state constraints to restrict the derivative of the path parameter to be positive until reaching the end value θ_{end}. Therefore, for the system (3.1), the path-following optimal control problem is formulated as:

$$
\begin{aligned}
\min_{\mathbf{u}(\cdot),\mathbf{q}(\cdot)} \quad & \int_{t_0}^{T_f} F_p(x(\tau),u(\tau),z(\tau),q(\tau))d\tau + E_p(x(T_f),z(T_f)) && \text{(3.12a)}\\
\text{s.t.} \quad & \dot{x} = f(x(\tau),u(\tau)),\ x(t_0)=x(t), && \text{(3.12b)}\\
& \dot{z} = r(z(\tau),q(\tau)),\ z(t_0)=z(t), && \text{(3.12c)}\\
& \theta(\tau) = l(z(\tau)), && \text{(3.12d)}\\
& \theta(\tau) \in [\theta_{max},\theta_{min}], && \text{(3.12e)}\\
& x(\tau)\in\mathcal{X}, u(\tau)\in\mathcal{U}, && \text{(3.12f)}\\
& z(\tau)\in\mathcal{Z}, q(\tau)\in\mathcal{Q}, && \text{(3.12g)}\\
& (x(T_f),z(T_F))^{\mathsf{T}}\in\mathcal{E}_p, && \text{(3.12h)}\\
& \forall\tau\in[t_0,T_f]. && \text{(3.12i)}
\end{aligned}
$$

The objective function in (3.12a) minimizes the error between the system states and the parametrized reference path through the stage function $F_p(\cdot)$, and the terminal cost function $E_p(\cdot)$. The real system dynamics are considered in (3.12b) while the virtual system dynamics are represented in (3.12c). The evolution of path parameter is considered in (3.12d) and constrained in (3.12e). The real and virtual systems states and inputs are restricted to be in the corresponding safe sets in (3.12f), (3.12g). The terminal set represented in (3.12h) does not depend on time rather it depends on the virtual and real system states sets such that $\mathcal{E}_p \subseteq \mathcal{X}\times\mathcal{Z}$ that is influenced by the augmented system input $u^{pf}=[u,q]^{\mathsf{T}}$ with augmented states $x^{pf}=[x,z]^{\mathsf{T}}$. It is important to note that with the introduction of the virtual system input q, the controller gains an additional degree of freedom to adjust the parameterized evolution of the path. This flexibility allows the controller to minimize the error between the reference path and the actual system states. The solution of the OCP in (3.12) results in an optimal control input $u^{*pf}(\cdot)$ and optimal state trajectory $x^{*pf}(\cdot)$ over the prediction horizon $[t_0,T_f]$.

3.4.1 Stability of MPC Path-Following

To derive stability of the path-following formulation, the following assumptions are required to hold [63, 148]:

Assumption 8. *(Constraint consistent path)*
The path $\mathcal{P}$ in (3.10) respects the state constraints $\mathbb{R}^{n_x}$.

Assumption 9. *(Stage Cost)*
The stage cost $F_p : \mathbb{R}^{n_x}\times\mathbb{R}\times\mathbb{R}^{n_u}\times\mathbb{R}^{n_q}\mapsto\mathbb{R}_0^+$ is continuous. It is further lower

bounded by a class $\mathcal{K}_\infty$ function α_1 such that $l_p(x, z, u, q) \geq \alpha_1(\|(x - z\|)^\top \forall (x, z, u, q)$. (Here x is the real system state while z is the virtual system state representing the path. The real and virtual system inputs are u, q, respectively).

Assumption 10. *(Terminal Cost)*
The terminal cost $E_p : \mathbb{R}^{n_x} \times \mathbb{R}^{n_z} \mapsto \mathbb{R}_0^+$ is positive semi-definite. Furthermore, $E_p(\cdot)$ is continuously differentiable w.r.t. the states of the real and virtual systems x and z.

Assumption 11. *(Terminal Set)*
The terminal set $\mathcal{E}_p \subseteq \mathcal{X} \times \mathcal{Z}$ is closed.

Assumption 12. *(Existence of a local controller)*
For any state $(x_k, z_k) \in \mathcal{E}_p$, there exist a local control input $(u^{\mathcal{E}_p}, q^{\mathcal{E}_p}) \in \mathcal{U} \times \mathcal{Q}$ such that

$$\left(\frac{\partial E_p}{\partial x}, \frac{\partial E_p}{\partial z}\right) . \begin{pmatrix} f(x(\tau), u^{\mathcal{E}_p}(\tau)) \\ r(z(\tau), q^{\mathcal{E}_p}(\tau)) \end{pmatrix} + F_p(x(\tau), u(\tau), z(\tau), q(\tau)) \leq 0.$$

This implies that the local controller $(u^{\mathcal{E}_p}, q^{\mathcal{E}_p})$ keeps the states inside the terminal set $\mathcal{E}_p$, that is, it renders the terminal region invariant and ensures the convergence of the cost function.

Whenever assumptions (1-3) as well as assumptions (9-12) hold, one can obtain the following theorem.

Theorem 2. *(Convergence of sampled-data NMPC for path following)*
Given the system (3.1). If the OCP (3.12) is feasible at t_0 with a proper stage, terminal costs as well as the terminal set satisfying Assumptions (9-12), then the MPC OCP is recursively feasible, and the controller will lead to convergence of the path following error under sampled-data NMPC.

For the proof we refer to [62].

3.4.2 Illustrative Example

For the autonomous mobile robot described in (3.7) we adopt a discrete-time/sampled-data formulation (assuming fixed inputs between the equi-distant sampling times) of

(3.12):

$$\min_{u^p\{\cdot\}} \quad \sum_{k=0}^{N_\mathrm{p}-1} l_p(x_k, z_k, u_k^p, q_k) + e_p(x_{k+N_\mathrm{p}}, y_{k+N_\mathrm{p}}^p). \tag{3.13a}$$

$$\text{s.t.} \quad \dot{x}_{k+1} = f_d(x_k, u_k),\ x_0 = \hat{x}(t), \tag{3.13b}$$

$$z_{k+1} = g_d(z_k, q_k),\ z_t = \hat{z}(t), \tag{3.13c}$$

$$\theta_k = l(z_k), \tag{3.13d}$$

$$\theta_k \in [\theta_{max}, \theta_{min}], \tag{3.13e}$$

$$x_{k+1} \in \mathcal{X},\ u_k \in \mathcal{U},\ y_k \in \mathcal{Y}, \tag{3.13f}$$

$$z_{k+1} \in \mathcal{Z},\ q_k \in \mathcal{Q}, \tag{3.13g}$$

$$(x_{k+1}, z_{k+1})^\mathsf{T} \in \mathcal{E}_p, \tag{3.13h}$$

$$\forall k \in [0, ..., N_\mathrm{p}]. \tag{3.13i}$$

The objective function (3.13a) penalizes the error between the predicted system states x_k and the parameterized reference that is calculated based on the virtual system represented in (3.13c) and the path parameter in (3.13d) through the stage and terminal costs $l_p(\cdot), e_p(\cdot)$ respectively. The evolution of the path parameter is restricted in (3.13e). Constraints are applied to the states and inputs of the real and virtual systems in (3.13f) and (3.13g). Please note that the control input $u^p\cdot$ in the (3.13) represents not only the system input u but also the virtual system input q. The solution of the OCP (3.13) results in a feasible input trajectory $u^{*p} = \left\{u_t^p, u_{k+1}^p, \cdots u_{t+N_p}^p\right\}$ and a state trajectory $x^* = \left\{x_{t+1}, \cdots x_{t+N_p+1}\right\}$ that respect the input and state constraints. Only the first control input candidate in the predicted control trajectory is applied to the real system. The new states of the system are measured or estimated at the next time instance, and the optimization problem is solved repetitively.

We demonstrate the path-following MPC formulation using the mobile robot described by (3.7) to track a preplanned path. This path is designed by a suitable path planning algorithm to safely guide the autonomous mobile robot from a starting point A to a final destination point B. The planned path ensures safety, as obstacle avoidance constraints are explicitly considered during the path planning phase.

Example 2. *(Path following problem)*
The safe planned path $\mathcal{P}$ is parameterized by a path parameter θ as in (3.10). The evolution of path parameters can be expressed by virtual system dynamics as follows:

$$\theta(t) = l(z(t)).$$

In our example, the virtual system is expressed by a double integrator chain such that:

$$\begin{aligned} \dot{z}_1(t) &= z_2, \\ \dot{z}_2(t) &= q. \end{aligned} \tag{3.14}$$

Moreover, a sigmoid function denoted by $S(\cdot)$ is employed to guarantee a seamless transition and ensure the path's continuity in differentiation. A prevalent example of sigmoid functions is the logistic function, expressed as follows:

$$S(x) = \frac{1}{1+e^{-x}}.$$

Therefore, the optimal control input can be obtained by solving the following optimization problem:

$$\begin{aligned}
\min_{u^p\{\cdot\}} \quad & \sum_{k=0}^{N_p-1} \left(\|p_k - p_k^p\|_2^Q + \|u_k^p\|_2^R \right) && (3.15a)\\
\text{s.t.} \quad & x_{k+1} = f_d(x_k, u_k), x_0 = \hat{x}(t), && (3.15b)\\
& z_{k+1} = g_d(z_k, q_k), \, z_t = \hat{z}(t), && (3.15c)\\
& \theta_k = z_1, && (3.15d)\\
& p_k^p = l(\theta_k), && (3.15e)\\
& v_{min} \leq v_k \leq v_{max}, && (3.15f)\\
& \delta_{min} \leq \delta_k \leq \delta_{max}, && (3.15g)\\
& 0 \leq z_k \leq 1, && (3.15h)\\
& U_{min}^p \leq u_k^p \leq U_{max}^p, && (3.15i)\\
& \forall k \in [0, ..., N_p]. && (3.15j)
\end{aligned}$$

Where x is the system states capturing the center of mass, velocity, steering angle, etc...., p captures the center of mass coordinates w.r.t. the global frame, i.e., $p = [p_x, p_y]^\top$, and p^p is the parametrized such that $p^p = \left[p_x^p, p_y^p\right]^\top$. The objective function (3.15a) aims to minimize the error between the predicted position of the system and the parameterized reference outlined in (3.15e). The weighting matrices Q and R are chosen such that $Q \geq 0$ and $R > 0$. It is important to note that the path velocity or the velocity of the virtual system q is an optimization variable that remains flexible for the optimizer's selection. Analogously to Example (1), physical constraints are taken into account in (3.15f) and (3.15g). At the same time, the dynamics of the autonomous mobile robot are captured in (3.15b) based on the measured or estimated states $\hat{x}(t)$ at time t. In addition, constraints on the path parameter are enforced in (3.15h). It is noteworthy that u^p represents the comprehensive control input vector, defined as $u^p = [u_1, u_2, q]^\top$, and is subject to constraints as detailed in (3.15i).

In this example, upon solving the path following optimization problem (3.15), the resulting control inputs guide the autonomous mobile robot safely along the reference path, circumventing obstacles, as depicted in Figure 3.4. Leveraging the concept of virtual system dynamics, representing the evolution of the reference path, offers the controller an additional degree of freedom (cf. Figure 3.6). This enables the selection

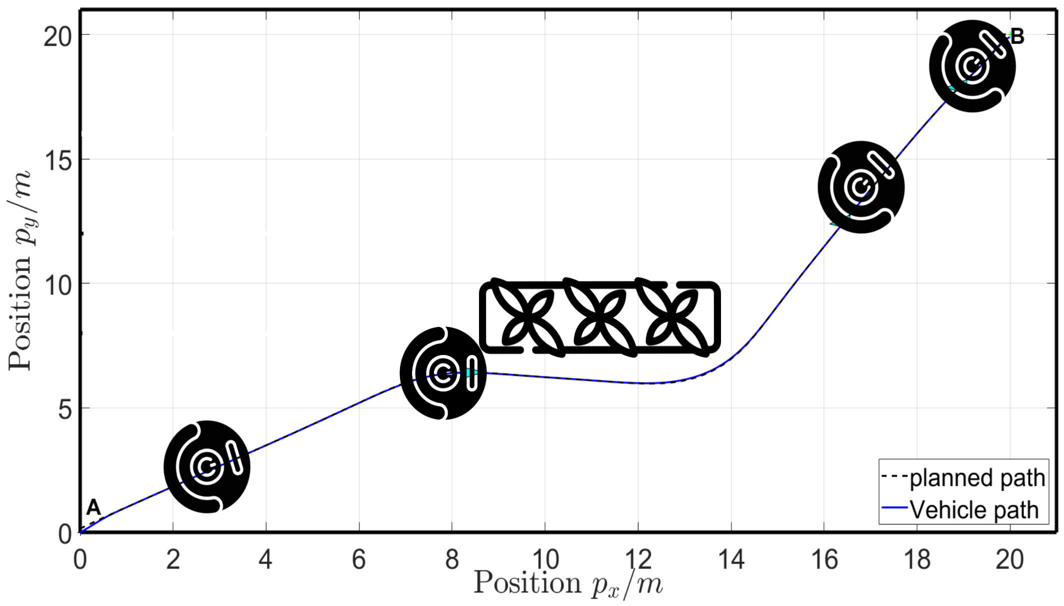


Figure 3.4: The autonomous mobile robot follows the safe path given by the planner using the NMPC path following formulation.

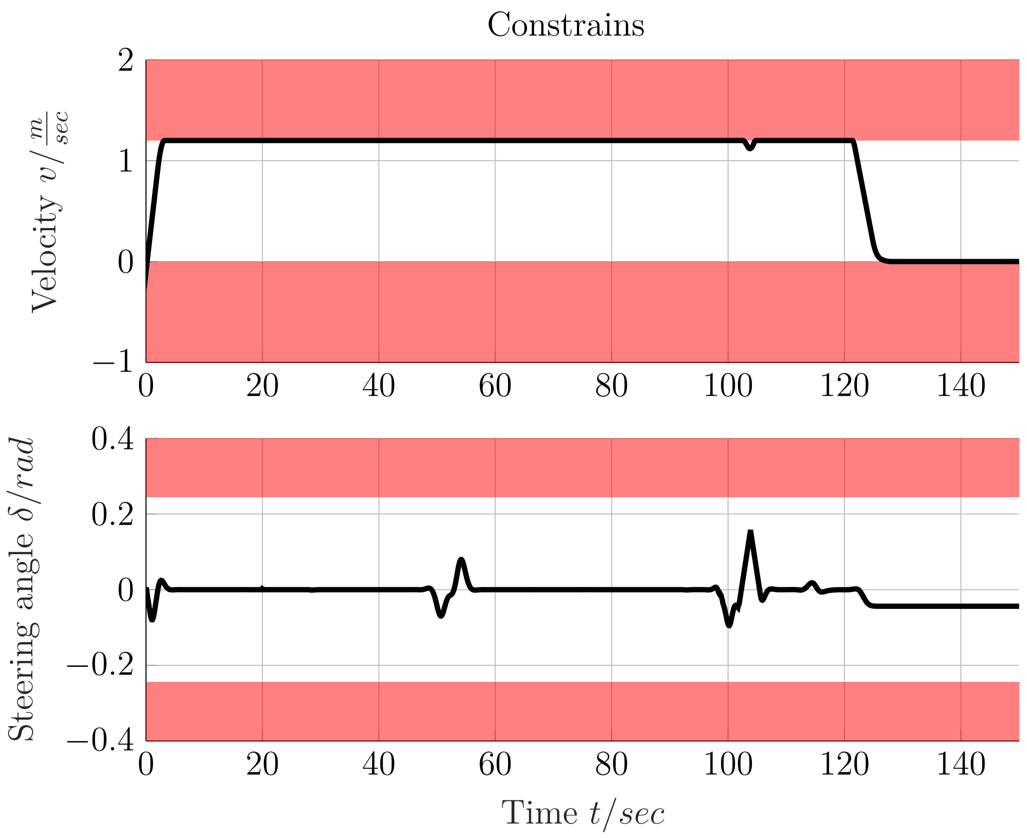


Figure 3.5: The controller is aware of the physical vehicle limitations in velocity and steering angle.

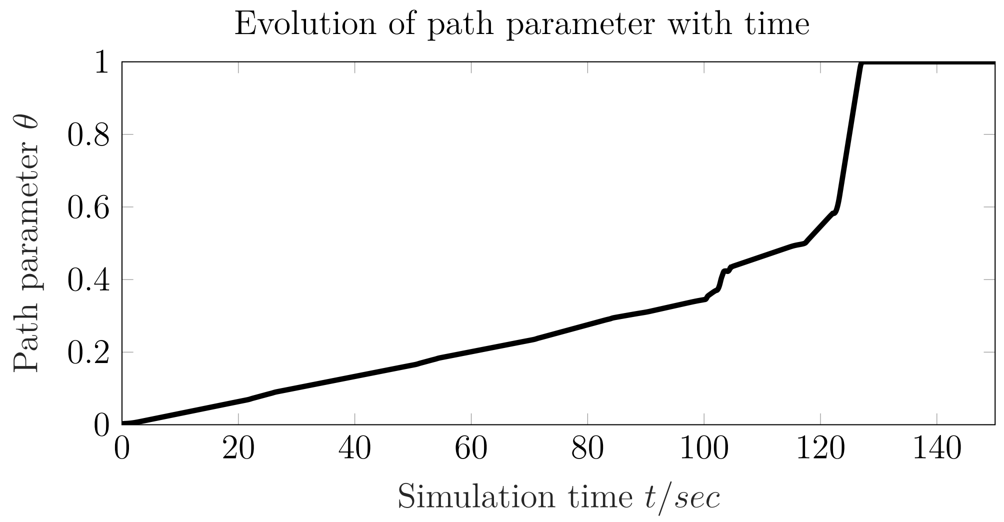


Figure 3.6: The evolution of the path parameter enables the vehicle to follow the safe path without colliding with the obstacles.

of values that minimize the discrepancy between the vehicle's state and the reference path while adhering to constraints. As illustrated in Figure 3.5, the control inputs comply with the limitations of the autonomous vehicle, speeding it at maximum velocity and adjusting the steering angle to avoid obstacles and keep on track with the reference path. Please note that autonomous vehicle safety and task completion are ensured as the path the vehicle follows is safe and steers the autonomous mobile robot to the destination point. The path following optimization problem (3.15) is tackled using the ACADO toolbox [98], executed on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz, with an average computation time of $t_{cpu} = 7.5381$ *ms.*

3.5 Trajectory Tracking Formulation

In the path following formulation, the evolution along the trajectory is not fixed a priori. In contrary, in the trajectory tracking formulation, the reference comprises a time-dependent curve that assigns specific values to the system's state/output at particular time intervals. This formulation is commonly applied to mobile ground robots [123, 155] and aerial vehicles [78, 205]. The challenge of designing a trajectory-tracking problem lies in defining the time-dependent reference, which can be crafted by an offline path planner or determined dynamically by online planning algorithms [148]. The trajectory tracking problem aims to determine an optimal control input u^* that minimizes the tracking error between the predicted system trajectory and a given time-dependent reference trajectory. This optimization must be performed while ensuring adherence to the imposed constraints on the system dynamics, control inputs, and states. However, applying trajectory tracking formulation could endanger task completion as some time-dependent reference can't be achieved while respecting the system constraints, e.g., maximum/minimum steering angles. The OCP for the trajectory tracking problem is formulated as follows while the superscript tt refers to the trajectory tracking:

$$\begin{aligned}
\min_{\mathbf{u}(\cdot)} \quad & \int_{t_0}^{T_f} F_{tt}(x(\tau), u(\tau), x^{tt}(\tau))d\tau + E_{tt}(x(T_f), x^{tt}(T_f)) && (3.16a)\\
\text{s.t.} \quad & \dot{x} = f(x(\tau), u(\tau)),\ x(t_0) = x(t), && (3.16b)\\
& g(x(\tau), u(\tau)) \leq 0, && (3.16c)\\
& x(\tau) \in \mathcal{X}, u(\tau) \in \mathcal{U}, && (3.16d)\\
& (x(T_f)) \in \mathcal{E}_{tt}, && (3.16e)\\
& \forall \tau \in [t_0, T_f]. && (3.16f)
\end{aligned}$$

The objective function (3.16a) minimizes the error between the system states and the reference time-varying states with stage cost $F_{tt}(\cdot)$ and the error at the end of the prediction horizon via the terminal cost function $E_{tt}(\cdot)$. System dynamical constraints, e.g., maximum acceleration and turning rates are considered in (3.16c). The predicted state and input are restricted to the state and input sets in (3.16d). The predicted state at the end of the prediction horizon is restricted to the time-varying terminal region $\mathcal{E}_{tt} \subseteq \mathcal{X}$ in (3.16e) which can be considered as an equality constraint forcing the predicted state to be on the state reference. The solution of the OCP (3.16) results in an optimal control input function $u^*(\cdot)$ and a state trajectory $x^*(\cdot)$ minimizing the tracking error between the system states and the reference states.

3.5.1 Stability of MPC Trajectory Tracking

In the trajectory tracking case, the time-dependent state reference leads to a time-varying error and a time-varying terminal region. Time-varying terminal regions provide a possibility to achieve trackability.

Definition 14. *(Trackability under Constraints)*
The reference state x_{tt} is said to be trackable for the system (3.1) if it fulfills the state constraints,i.e., $x^{tt} \in \mathcal{X}$ and can be tracked given the system dynamics once starting on it. This implies that once the system starts on the reference states there is a control input $u(t) \in \mathcal{U}$ driving the system to track the reference.

To discuss the stability of trajectory tracking OCP (3.16) for the non-linear system given in (3.1) considering the control input design according to Definition 14 the following assumptions ensures the convergence of the tracking error [63, 148]:

Assumption 13. *(Stage Cost)*
The stage cost $F_{tt} : \mathbb{R}^{n_y} \times \mathbb{R}^{n_u} \mapsto \mathbb{R}_0^+$ is continuous and $F_{tt}(0, 0) = 0$. It is further lower bounded by a class $\mathcal{K}_\infty$ function α_1 such that $l_{tt}(x - x^{tt}) \geq \alpha_1(\|x - x^{tt}\|)^\top \forall (x, x^{tt})$.

Assumption 14. *(Terminal Cost)*
The terminal cost $E_{tt} : \mathbb{R}^{n_x} \mapsto \mathbb{R}_0^+$ is positive semi-definite. Furthermore, $E_{tt}(\cdot)$ is continuously differentiable w.r.t. system states x.

Assumption 15. *(Terminal Set)*
The terminal set $\mathcal{E}_{tt} \subseteq \mathcal{X}$ is closed and time-varying.

Assumption 16. *(Existence of local controller)*
For any state $x_k \in \mathcal{E}_{tt}$, there exist a local control input $u^{\mathcal{E}_{tt}}(\cdot) \in \mathcal{U}$ such that

$$\frac{\partial E_{tt}}{\partial x} \cdot f(x(\tau), u^{\mathcal{E}_{tt}}(\tau)) + F_{tt}(x(\tau), x^{tt}(\tau), u(\tau)) \leq 0$$

This means that the local controller $u^{\mathcal{E}_{tt}}$ will keep the states inside the terminal set $\mathcal{E}_{tt}$ i.e., the terminal region is a control invariant set and ensures the convergence of the cost function.

When assumptions (1-3) as well as assumptions (13-16) hold, one can state the following theorem [62]:

Theorem 3. *(Convergence of sampled-data NMPC for trajectory tracking)*
Given a system in (3.1), if the OCP (3.16) is feasible at t_0 with proper stage, terminal costs, and the terminal set satisfying assumptions (13-16), then the OCP is recursively feasible, and the controller will lead to convergence of tracking error under sampled-data NMPC see [62] for the complete proof.

3.5.2 Illustrative Example

For the autonomous mobile robot that is described in (3.7) we adopt the discrete-time form of (3.16) that can formulated as follows:

$$\begin{aligned}
\min_{u\{\cdot\}} \quad & \sum_{k=0}^{N_\mathrm{p}-1} l_{tt}(x_k, x_k^{tt}, u_k) + e_{tt}(x_{N_\mathrm{p}}, x_{N_\mathrm{p}}^{tt}). && (3.17a)\\
\text{s.t.} \quad & \dot{x}_{k+1} = f_d(x_k, u_k),\ x_0 = \hat{x}(t), && (3.17b)\\
& x_{k+1} \in \mathcal{X},\ u_k \in \mathcal{U},, && (3.17c)\\
& x_{t+N_\mathrm{p}+1} \in \mathcal{E}_{tt}, && (3.17d)\\
& \forall k \in [0, ..., N_\mathrm{p}]. && (3.17e)
\end{aligned}$$

The stage cost $l_{tt}(\cdot)$ in the objective function (3.17a) minimizes the error between the predicted system state and the time-varying reference state x_k^{tt}, while the terminal cost $e_{tt}(\cdot)$ minimizes the error at the end of the prediction horizon. The objective function does not explicitly vary with time. However, the time dependence is introduced through the time-varying reference states this implies that a new reference is available to the controller at each prediction step. Consequently, the controller must guide the system to specific states at specific times while adhering to state and control input constraints (3.17c). Due to the time dependency of the reference states, some states may be infeasible due to system dynamics limitations resulting in higher deviation errors than the path-following formulation. This can potentially threaten the system's safety and overall task completion. The solution of the OCP (3.17) results in a feasible input trajectory $u^* = \left\{u_t, u_{t+1}, \cdots u_{t+N_p}\right\}$ and a state trajectory $x^* = \left\{x_{t+1}, \cdots x_{t+N_p+1}\right\}$ that respect the input and state constraints. Only the first control input is applied to the real system. The new states/output of the system are measured or estimated at the next instance, and the optimization problem is solved repetitively.

Example 3. *(Trajectory tracking problem)*
In contrast to Example (2), where the reference path is defined by a path parameter θ and the problem is formulated as a path following optimization problem, in this example, we tackle a trajectory tracking problem. Specifically, the autonomous mobile robot described by the nonlinear vehicle dynamics (3.7) is assigned to track a trajectory based on the trajectory tracking formulation (3.17). The optimal control input u^* has to drive the autonomous mobile robot to a specific state at a specific time while respecting the vehicle constraints such as maximum and minimum turning rates and acceleration.
The optimal control problem for trajectory tracking is expressed as follows:

$$\min_{u\{\cdot\}} \quad \sum_{k=t}^{t+N_p} \left(\|p_k - p_{k+1}^{tt}\|_2^Q + \|u_k\|_2^R\right) \tag{3.18a}$$

$$\text{s.t.} \quad x_{k+1} = f_d(x_k, u_k), \quad x_0 = \hat{x}(t), \tag{3.18b}$$

$$v_{min} \leq v_k \leq v_{max}, \tag{3.18c}$$

$$\delta_{min} \leq \delta_k \leq \delta_{max}, \tag{3.18d}$$

$$x_k \in \mathcal{X}, \; u_k \in \mathcal{U}, \; h(x_k) \in \mathcal{Y}, \tag{3.18e}$$

$$\forall k \in [t, ..., t+N_p]. \tag{3.18f}$$

Where x is the system states capturing the center of mass, velocity, steering angle, etc...., p captures the center of mass coordinates w.r.t. the global frame, i.e., $p = [p_x, p_y]^\top$, and p^{tt} is the reference at each time step, such that $p^{tt} = \left[p_x^{tt}, p_y^{tt}\right]^\top$. In this formulation, the control objective (3.18a) aims to minimize the tracking error between the time-dependent reference p^{tt} and the predicted vehicle trajectory p_k. The weighting matrices Q and R are chosen such that $Q \geq 0$ and $R > 0$. Constraints (3.18c) and (3.18d) enforce the mechanical limitations on the velocity and steering angle of the autonomous vehicle, respectively. Nonlinear vehicle dynamics are considered in (3.18b) with states $\hat{x}(t)$ measured or estimated at time t. Solving the trajectory tracking optimization problem (3.18) yields optimal control inputs u^* that guide the vehicle states to specific values at each prediction step. However, while the control inputs steer the vehicle to track the reference, some states can't be achieved with limited vehicle capabilities such as steering angle. Consequently, the autonomous mobile robot

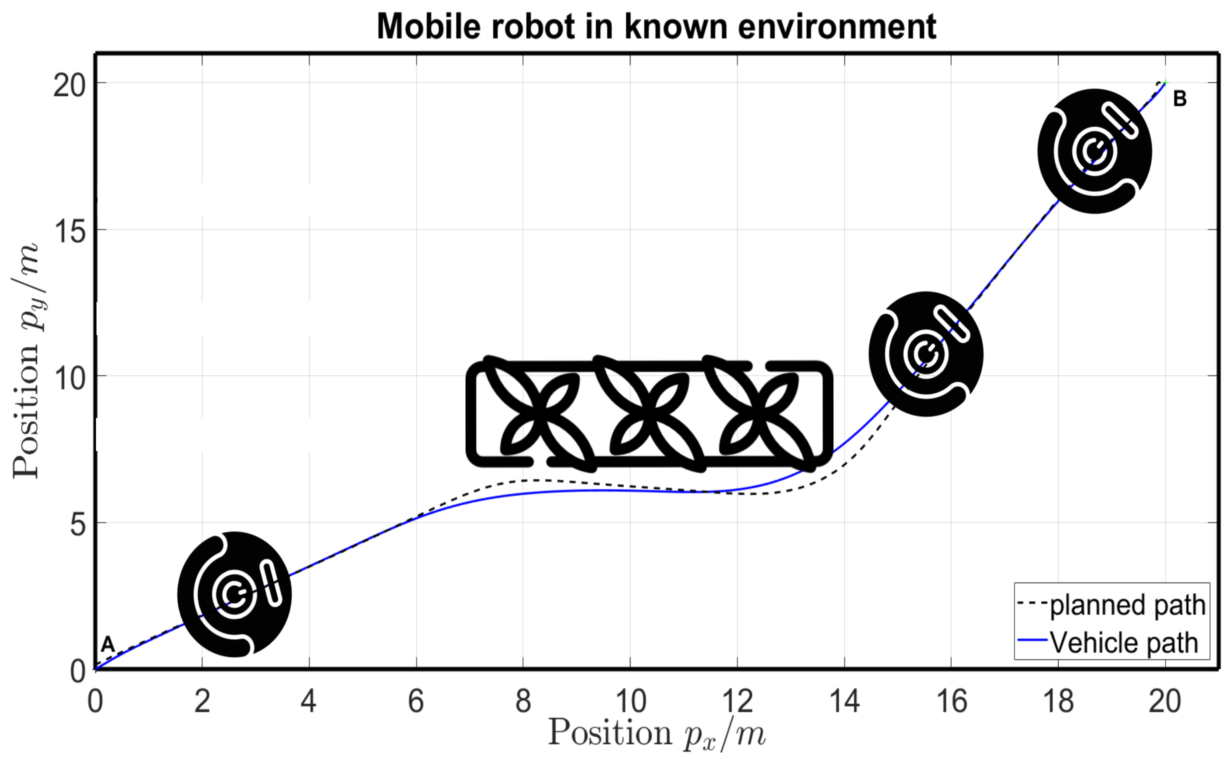


Figure 3.7: The mobile robot tracks a reference path from a start given point A to a final destination point B, which leads to hitting the obstacle.

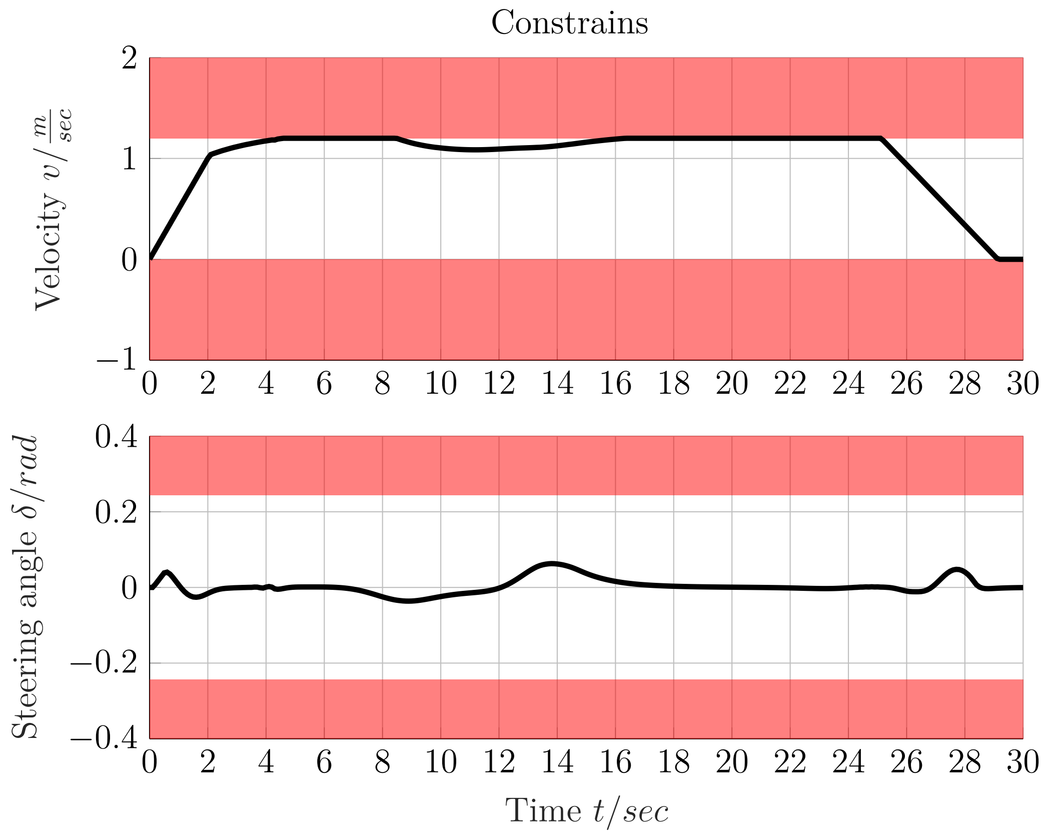


Figure 3.8: Velocity and steering angle. The red areas represent the input limitations.

may collide with an obstacle, resulting in a higher error between the vehicle path and the reference trajectory compared to the path following formulation in Example (2), as depicted in Figure 3.7. This arises from the controller's attempt to minimize the error between the current vehicle states and the time-varying reference while adhering to the vehicle's dynamic constraints, as illustrated in Figure 3.8.

Similarly to the previous examples, problem (3.18) is solved using the ACADO toolbox [98] on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz, with an average computation time of $t_{cpu} = 1.5724$ *ms.*

3.6 Model Predictive Control for Path Planning–A Mixed-Integer Formulation

The previous sections reviewed various Model Predictive Control (MPC) formulations for tracking or following a predetermined safe path. In this section, we establish the foundation for the subsequent chapters by demonstrating that the path-planning problem for an autonomous mobile robot (1) can also be effectively addressed within the MPC framework. This approach involves solving a constrained optimization problem at each time step in a rolling-horizon or moving-horizon manner. The constraints embedded in the optimization problem ensure vehicle safety by incorporating obstacle avoidance and guaranteeing the feasibility of the planned path while considering the vehicle's dynamics and physical limitations, such as steering angle constraints, speed limits, and turning rate restrictions. Furthermore, additional constraints can be included to enhance vehicle stability and passenger comfort. The formulation of the optimization problem, along with the imposed constraints, plays a crucial role in determining both the quality of the planned path and its suitability for real-time implementation.

However, a fundamental trade-off exists between achieving an optimally planned path according to a given metric and ensuring that the computational complexity remains manageable for real-time execution. Thus, balancing optimization accuracy and computational efficiency is crucial when designing effective path-planning algorithms for autonomous vehicles.

The extent of available environmental information significantly influences the approach to solving the path-planning problem for autonomous mobile robots, cf. Figure 3.9. On the one hand, global or offline path planners leverage all accessible environmental data before the robot's motion execution. In this approach, an optimization problem is formulated to minimize or maximize a user-defined objective function while considering factors such as obstacle geometry, position, and velocity. Additionally, constraints related to the autonomous robot's motion capabilities and dynamics are integrated into the planning process.

On the other hand, online or sensor-based path planners operate by utilizing real-

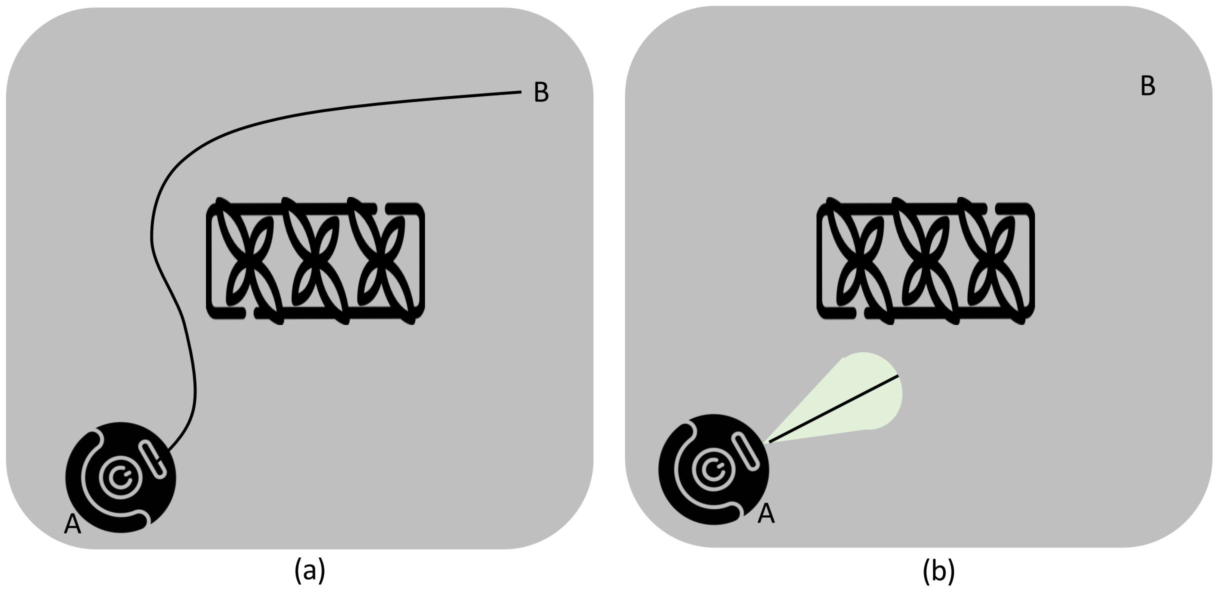


Figure 3.9: A global/offline path planner (a) considers all available offline environmental information to generate a complete, safe path connecting the starting point (A) to the goal point (B). A local path planner (b) primarily utilizes real-time environmental data within the sensor range (depicted in green) to compute a safe but locally constrained path.

time environmental information captured by onboard sensors within their field of view. This approach involves continuously sensing new environmental data at each time step and dynamically planning a safe path accordingly (see Sections 2.4.1, 2.4.2).

To ensure a safe transition from a starting point A to a final destination B when solving a local planning problem, two main strategies can be employed. One approach integrates local and global planning in a hierarchical manner, where the local planner, operating over a shorter horizon, refines its decisions based on the global path and repeatedly solves the local planning problem. Alternatively, a purely local planning strategy may be employed, where local paths are computed iteratively without reference to a global plan. However, this approach may result in the vehicle becoming trapped in an obstructed or infeasible region.

MPC has gained significant attention in the context of path planning. For instance, the study conducted in [204] employed a global path planner within the MPC framework to determine the fastest and safest path for an autonomous go-kart. In [133], an MPC-based path-planning approach was proposed to enhance safety by modeling other vehicles as polygons. The work in [110] introduced an MPC path-planning method for coordinating multiple mobile robots in a warehouse setting. Similarly, [59] presented a global path-planning strategy for autonomous mobile robots, incorporating potential functions to ensure safety in the presence of adversaries and road boundaries.

In [164], a global path-planning approach for autonomous vehicles was developed using obstacle approximations based on circular representations. Furthermore, [197] focused on online route planning for autonomous underwater vehicles operating in dynamic environments. Additionally, [159] proposed an MPC-based online path-planning method for articulated vehicles navigating partially known environments. In [170], an online three-dimensional path-planning algorithm was introduced for UAVs operating in environments with emerging threats.

3.6.1 Special Case: Path Planning for Mobile Robots on the Ground

For clarity of exposition, this section explores path planning using an MPC formulation for an autonomous mobile robot operating on a planar surface (2D). However, the outlined concepts can be readily extended to more complex 3D environments with additional dynamic considerations. Specifically, we focus on an autonomous mobile robot navigating a partially known environment. The vehicle dynamics are assumed to be given in discrete time:

$$x_{k+1} = f_d(x_k, u_k). \tag{3.19}$$

Continuous-time dynamics can be considered by integrating the differential equations (numerically) to obtain f_d. Here, $x \in \mathbb{R}^{n_x}$ and $u \in \mathbb{R}^{n_u}$ represent the mobile robot's states and control inputs, respectively, with $f_d : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$. The state vector is given by:

$$x_k = \begin{bmatrix} p_k^\mathsf{T}, & \cdots \end{bmatrix}^\mathsf{T}, \tag{3.20}$$

where $p_k \in \mathbb{R}^2$ represents the coordinates of the vehicle's center of mass, and possibly additional states such as heading angle, velocities, and pitch are included.

The autonomous mobile robot operates in a partially known environment with static obstacles whose positions and geometries are known *a priori*. Any region that remains unmapped is treated as a no-go zone and incorporated into the obstacle avoidance constraints of the optimization problem. The global path planner is responsible for generating a safe path from the initial starting point (A) to the final destination (B) while ensuring obstacle avoidance and path feasibility.

Definition 15 (**Obstacle Avoidance**). *Obstacle avoidance for an autonomous mobile robot j is achieved if its position satisfies $p_j \notin \mathcal{O}_i$ for all time instances. Here, $p_j \in \mathbb{R}^n$ denotes the position of the vehicle j with respect to a given reference frame, and $\mathcal{O}_i \subset \mathbb{R}^n$ represents the set of all forbidden regions in the environment that must be avoided. Each element of $\mathcal{O}_i$ corresponds to an obstacle that has been enlarged by the maximum vehicle dimensions to ensure safety [189].*

Notably, incorporating obstacle avoidance constraints leads to a nonconvex optimization problem, significantly increasing computational complexity[190].

Definition 16 (**Convex Set**). *A set $\mathcal{C}$ is convex if the line segment between any two points in $\mathcal{C}$ remains entirely within $\mathcal{C}$, as illustrated in Figure 3.10. Formally, for any* *two arbitrary points $x_1, x_2 \in \mathcal{C}$ and any scalar θ satisfying $0 \leq \theta \leq 1$, the following relation holds [30]:*

$$\theta x_1 + (1 - \theta) x_2 \in \mathcal{C}.$$

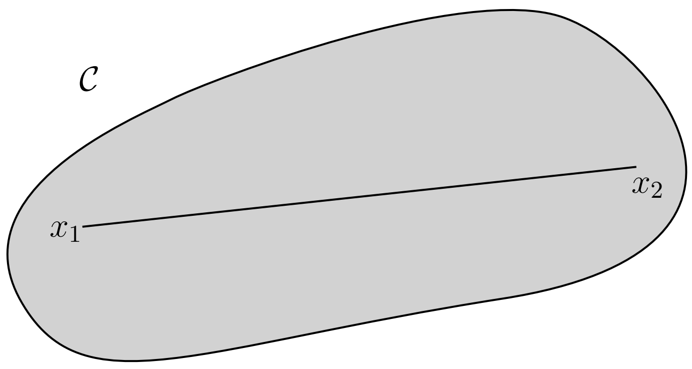


Figure 3.10: Convex set $\mathcal{C}$.

Definition 17 (**Nonconvex Set**). *A set $\mathcal{C}$ is said to be nonconvex if there exist two points $x_1, x_2 \in \mathcal{C}$ such that the line segment connecting them contains at least one point that does not belong to $\mathcal{C}$. This concept is depicted in Figure 3.11.*

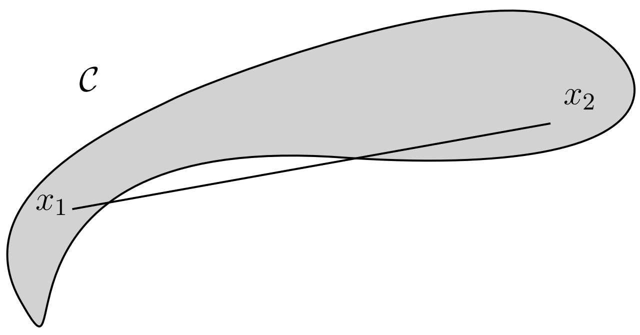


Figure 3.11: Nonconvex set $\mathcal{C}$.

In the following, the convex polyhedron $\mathcal{O}_i$ represents the known i^{th} obstacle. It is defined as:

$$\mathcal{O}_i := \{p \in \mathbb{R}^{n_o} \mid E_i p \leq e_i\}, \tag{3.21}$$

where n_o denotes the dimension of the i^{th} obstacle. If an obstacle is nonconvex, a convex hull enclosing the obstacle can be used instead [189]. To ensure robust obstacle avoidance, the obstacle representation may be enlarged by incorporating the vehicle's maximum dimensions.

In a moving horizon optimization setting, at each sampling time t, the optimization problem for planning a safe path from the initial point to the final destination point can be formulated as follows:

$$\min_{p} \quad \sum_{k=0}^{N_\mathrm{p}-1} (\|p_k - p_\mathrm{B}\|_\infty + \|u_k\|_\infty) \tag{3.22a}$$

$$\text{s.t.} \quad x_{k+1} = f_d(x_k, u_k), \; x_0 = \hat{x}(t), \tag{3.22b}$$

$$p_k \notin \mathcal{O}_i, \tag{3.22c}$$

$$g(x_k, u_k) \leq 0, \tag{3.22d}$$

$$\forall k \in [0, ..., N_\mathrm{p}], \tag{3.22e}$$

$$\forall i \in [1, ..., N_o]. \tag{3.22f}$$

Where $x_k = \left[{p_k}^\mathsf{T}, \cdots\right]^\mathsf{T}$ contains all system states, i.e., the coordinate of the center of mass p_k and other states, e.g., heading angle and pitch. $\hat{x}(t)$ is the measured/estimated system state at the current time t and u is the control input. We consider an infinity norm for penalizing the distance to the goal location, which allows later on formulation as a linear programming problem if all other components are linear. The safety of the autonomous vehicle is considered by enforcing the obstacle avoidance constraint (3.22c). Equation (3.22d) considers the mechanical limitations of the autonomous vehicle on velocities and acceleration. The resulting path planning problem (3.22) exhibits nonlinearity due to the possible non-linear equations (3.22b) and (3.22d), and is in general nonconvex, resulting in a computationally expensive constrained optimization problem. To alleviate the computational burden, we introduce approximated models and obstacle avoidance constraints.

3.6.2 Obstacle Avoidance Approximation

The reduce the computational complexity, obstacle avoidance constraints (3.22c) can be approximated by a set of linear inequalities; each of them represents a plane, and the intersection between these inequalities represents the obstacle, see Figure 3.12 [73, 189]. To furthermore simplify the problem and formulate it as a MILP, if linear dynamics are considered, one can use the *Big-M* method and exploit *binary/integer variables* [73, 189]. To illustrate the main idea of the Big-M method and binary variables, let us consider the following simple example:

Example 4. *Assume that a user-defined objective function $J(x)$ is minimized subject to either one of the following constraints $h_1(x)$ and $h_2(x)$. Then, the following OCP*

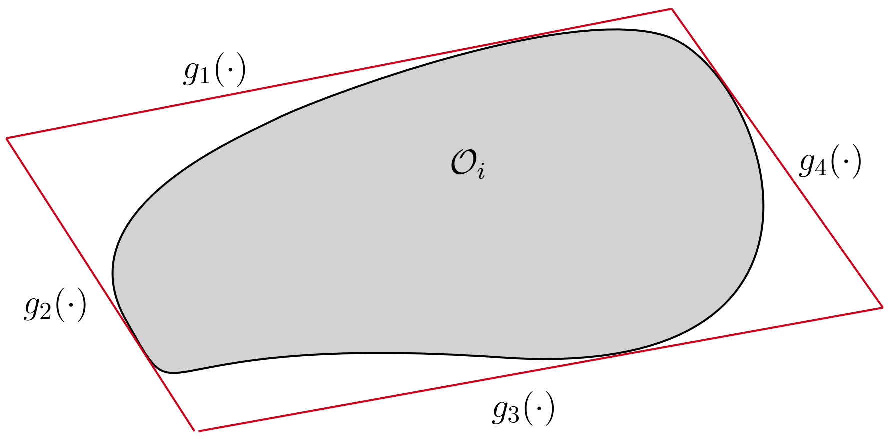


Figure 3.12: Obstacle $\mathcal{O}_i$ is approximated by a set of linear inequalities $g(\cdot)$

can be formulated:

$$\begin{aligned} \min_{x} \quad & J(x) && (3.23a)\\ \text{s.t.} \quad & h_1(x) \le 0 \quad OR, && (3.23b)\\ & h_2(x) \le 0 && (3.23c) \end{aligned}$$

By introducing a sufficiently large positive integer M and a binary variable d, the optimization problem (3.23) can be rewritten as follows:

$$\begin{aligned} \min_{x} \quad & J(x) && (3.24a)\\ \text{s.t.} \quad & h_1(x) \le Md \quad AND, && (3.24b)\\ & h_2(x) \le M(1-d), && (3.24c)\\ & d \in \{0,1\}. && (3.24d) \end{aligned}$$

Here, by choosing $d = 0$, the constraint in (3.24b) is activated, i.e., it has to be satisfied while the other constraint in (3.24c) is relaxed. The situation is reversed when $d = 1$. As d is a binary variable, d only takes the value 0 or 1, only one constraint $h_1(\cdot)$ or $h_2(\cdot)$ is activated at a time. One should also be careful when using binary variables as the complexity of the problem increases with increasing numbers of binary variables and prediction horizon N_p [107].

Consequently, the obstacle $\mathcal{O}_i$ can be represented by an inequality constraint such that $\forall i \in [1, 2, \cdots, N_o]$ where N_o is the number of known obstacles that are deployed in the working environment of the autonomous vehicle,

$$\mathcal{O}_i := \{(x, y) \mid E_i[x\ y] \le e_i\}. \tag{3.25}$$

Please note in (3.25), the row entry of the matrices E_i and e_i represent a plane and the intersection between these planes represents the obstacle i^{th}. In addition, the direction of the inequality reflects the definition of the obstacle region $\mathcal{O}_i$. Therefore, the obstacle avoidance constraint (3.22c) can be formulated using the *Big-M* method introducing binary variables d as follows: $\forall k \in [0, N_\mathrm{p}]$

$$E_i p_k \geq e_i + M(1 - D). \tag{3.26}$$

Here, $p \in \mathbf{R}^2$ captures the mobile robot coordinates. Please also note the change in the inequality direction because we enforce the mobile robot position to be outside the obstacle set, and the vector D contains the binary variables such that:

$$D = [d_1 \; d_2 \cdots d_n]^\top. \tag{3.27}$$

Here n is the number of rows in E_i or e_i. To ensure that one constraint is active, i.e., must be satisfied while the others are relaxed at each prediction step, the following constraint is imposed: $\forall i \in [1, ..., n]$:

$$\sum_{k=t}^{t+N_\mathrm{p}} d_{i,k} = 1. \tag{3.28}$$

3.6.3 Approximating Nonlinear Vehicle Dynamics

The primary objective of the path planner for an autonomous mobile robot is to formulate a safe path that establishes a connection between the initial starting point and the final destination goal. This path must account for the robot's equation of motion and adhere to its capabilities to ensure feasibility for execution by the autonomous robot. However, incorporating the full complexity of the vehicle's dynamics could significantly increase the challenge of finding a safe path. Consequently, a trade-off exists between the vehicle model's complexity and the problem's complexity, ultimately impacting the quality of the planned path compared to what could be achieved with a more comprehensive consideration of the vehicle's dynamics. In the following, we limit the derivations to mobile robots moving in a plane and approximate the dynamics of the mobile robot by a linear, double-integrator model:

$$\begin{bmatrix} p_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{I}_2 & T_s\boldsymbol{I}_2 \\ \boldsymbol{O}_2 & \boldsymbol{I}_2 \end{bmatrix} \begin{bmatrix} p_k \\ v_k \end{bmatrix} + \begin{bmatrix} \frac{T_s^2}{2}\boldsymbol{I}_2 \\ T_s\boldsymbol{I}_2 \end{bmatrix} \begin{bmatrix} ax_k \\ ay_k \end{bmatrix}. \tag{3.29}$$

The state vector p represents the coordinates of a point mass, defined as $p = [p_x \; p_y]^\top$. The vector $v = [v_x \; v_y]^\top$ characterizes the vehicle's velocity. The matrices $\mathbf{O}$ and $\mathbf{I}$ denote zero and identity matrices of size 2×2, respectively. The acceleration vector $[a_x \; a_y]^\top$ encapsulates the control inputs. Generally, the vehicle dynamics expressed in (3.29) can be written as follows:

$$x_{k+1} \; = \; Ax_k \; + \; Bu_k. \tag{3.30}$$

Similar simplified vehicle dynamics have been used before, see [42, 108, 134, 216] to alleviate computational load.

3.6.4 Constraints on the Vehicle Dynamics

Very often there are additional constraints required on the vehicle dynamics, such as maximum speed and acceleration. These constraints are often represented by a set of nonlinear inequality constraints:

$$\sqrt{v_x^2 \; + \; v_y^2} \leq v_{max}, \tag{3.31a}$$

$$\sqrt{a_x^2 \; + \; a_y^2} \leq a_{max}. \tag{3.31b}$$

These constraints (3.31) are nonlinear and do not conform to the desired linear mixed-integer representation. To address this, we approximate them using linear inequality constraints based on an M-sided polygon [73].

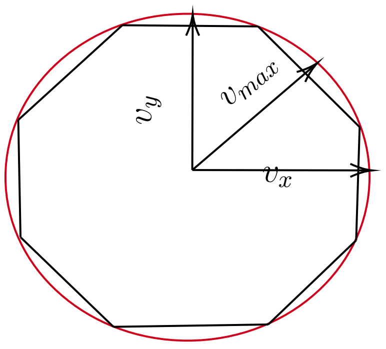


Figure 3.13: Nonlinear constraints, such as speed limitations, are approximated using polygons.

$\forall m \in \{1, \cdots, M\}$

$$v_x \cos\frac{2\pi m}{M} \; + \; v_y \sin\frac{2\pi m}{M} \; \leq v_{max}, \tag{3.32a}$$

$$a_x \cos\frac{2\pi m}{M} \; + \; a_y \sin\frac{2\pi m}{M} \; \leq a_{max}. \tag{3.32b}$$

Remark 1. *Minimum velocity constraints, such as $v_{min} = 0$, can also be incorporated. While they fit within the outlined formulations, they significantly increase computational complexity.*

Moreover, the smoothness of the planned path, i.e., the transitions between the waypoints, can be ensured by considering the rate of change of acceleration per each prediction step such that: $\forall k \in \{1, \cdots N_\mathrm{p}\}$

$$\sqrt{(a_{x_k} - a_{x_{k-1}})^2 \; + \; (a_{y_k} - a_{y_{k-1}})^2} \; \leq \; T_s \Delta a_{max}. \tag{3.33}$$

Where a_{x_k} and $a_{x_{k-1}}$ are the accelerations in x at the current and previous prediction steps, while $(a_{y_k} - a_{y_{k-1}})$ is the difference between the acceleration in y direction at two successive prediction steps. By approximating the nonlinear constraints in (3.33) using M-sided polygons, i.e. 4 sided ones setting $M = 4$, a set of linear inequalities can be formulated:

$$a_{x_k} - a_{x_{k-1}} \leq T_s \Delta a_{max}, \quad -a_{x_k} + a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{3.34a}$$

$$a_{y_k} - a_{y_{k-1}} \leq T_s \Delta a_{max}, \quad -a_{y_k} + a_{y_{k-1}} \leq T_s \Delta a_{max}. \tag{3.34b}$$

3.6.5 Mixed-Integer Programming Path Planning

Mixed-Integer Programming (MIP) refers to solving a mathematical optimization problem in which some or all decision variables are constrained to take integer values. The objective function can be linear, quadratic, or nonlinear, aiming at either minimization or maximization. Constraints may include linear or nonlinear equalities and inequalities, with certain variables restricted to binary values (0 or 1) [109].

Notably, MIP formulations belong to the class of NP-hard problems, meaning that computational complexity grows exponentially with the number of integer variables involved. This complexity is highly dependent on the problem formulation, influenced by factors such as the number of obstacles, vehicles, and the length of the prediction horizon [107].

MIP-based formulations have been widely used in mobile robot applications. For instance, [191] addresses a Mixed-Integer Linear Programming (MILP) optimization problem for single and multiple mobile robots vehicles, optimizing energy consumption and travel time while ensuring not to collide with static and dynamic obstacles. Similarly, [91] presents a multi-objective MILP approach for optimizing both fuel efficiency and arrival time for a UAV flying at a fixed altitude.

MILP formulations have also been applied in search-and-rescue operations. In [72], an MILP-based approach facilitated efficient planning, where an onboard camera system was used to cover a designated search area. Additionally, MILP techniques have been utilized for coverage path planning in UAV operations, as demonstrated in [41, 108].

Beyond aerial applications, MILP has been employed for mobile robot trajectory planning. In [14], a nonconvex safe operational region was defined to account for vehicle motion constraints, such as speed and jerk limitations, while ensuring safety through MIP-based trajectory planning. To improve computational efficiency, [157] proposed an optimized MILP formulation that reduces the number of binary variables, thereby streamlining online computations. Moreover, MILP has been used for coordinated path planning of autonomous vehicle fleets, incorporating obstacle avoidance directly into the problem formulation [8].

Building upon these concepts, we now define the base path-planning problem for an autonomous mobile robot navigating in a partially known environment with static obstacles (3.22), derived from the outlined approximations.

$$\min_{\{u\},d} \quad \sum_{k=0}^{N_\mathrm{p}-1} \left(\|p_k - p_\mathrm{B}\|_2^Q + \|u_k\|_2^R \right) \tag{3.35a}$$

$$\text{s.t.} \quad x_{k+1} = Ax_k + Bu_k,\ x_0 = \hat{x}(t), \tag{3.35b}$$

$$E_i p_k \geq e_i + M_\mathrm{big}(\mathbf{1} - d_{i,k}), \tag{3.35c}$$

$$a_{x_k} - a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{3.35d}$$

$$-a_{x_k} + a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{3.35e}$$

$$a_{y_k} - a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{3.35f}$$

$$-a_{y_k} + a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{3.35g}$$

$$v_x \cos\frac{2\pi m}{M} + v_y \sin\frac{2\pi m}{M} \leq v_{max}, \tag{3.35h}$$

$$a_x \cos\frac{2\pi m}{M} + a_y \sin\frac{2\pi m}{M} \leq a_{max}, \tag{3.35i}$$

$$\sum_{k=1}^{n} d_{i,k} = 1, \tag{3.35j}$$

$$\forall k \in [0, ..., N_\mathrm{p}], \tag{3.35k}$$

$$\forall i \in [1, ..., N_o], \tag{3.35l}$$

$$\forall m \in [1, ..., M]. \tag{3.35m}$$

The objective function (3.35a) determines an optimal control input sequence $\overset{*}{u}$ that minimizes the error between the vehicle position p_k and the destination point position p_B at each prediction step k, utilizing the approximated linear autonomous vehicle dynamics (3.35b). The nonlinear state constraints on velocity and acceleration are approximated using M-sided polygons and expressed as linear constraints (3.35i and (3.35h)). The safety of the autonomous vehicle is ensured by incorporating obstacle avoidance constraints using the Big M method and binary variables in (3.35c and (3.35j)). Solving the optimal control problem (3.35) yields an optimal obstacle-free state trajectory $x^* = \left[{x_0^*}^\mathsf{T}, ..., {x_{N_\mathrm{p}+1}^*}^\mathsf{T}\right]^\mathsf{T}$ and optimal control input sequence $u^* = \left[u_0, u_1, \cdots u_{N_p-1}\right]$. The optimization problem (3.35) is solved at each time step in a rolling horizon manner, resulting in a safe path from the starting point to the final destination oint. In the subsequent section, we demonstrate the global path planner based on the MIP formulation through an illustrative example.

3.6.6 Illustrative Example

We consider the autonomous mobile robot represented in (3.7) operating in a partially known environment. Static obstacles with known positions and geometries obstruct the operational environment. Some information about the environment is unknown during the path planning process and is represented as a no-go region cf. Figure 3.14. To ensure the safety of the autonomous mobile robot, the no-go region is considered in the obstacle avoidance constraints forcing the vehicle to avoid it.

Example 5. *(Global path planner)*
A global or offline path planner is employed. The planner utilizes the MIP formulation (3.35) to effectively leverage the environmental data, generating a safe path that guides the autonomous mobile robot from the initial starting point (A) to the final destination point (B), as illustrated in Figure 3.14.

Problem (3.35) is solved using the YALMIP toolbox [139] on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz, with an average computation time of $t_{cpu} = 2.3264$ *ms.*

3.7 Summary

This chapter introduced Model Predictive Control and outlined its key principles for tracking, path following, and planning. MPC provides a powerful approach for controlling mobile robots while handling constraints, ensuring stability, and optimizing decisions in real-time.

The discussion covered different MPC formulations. In set-point tracking, the mo-

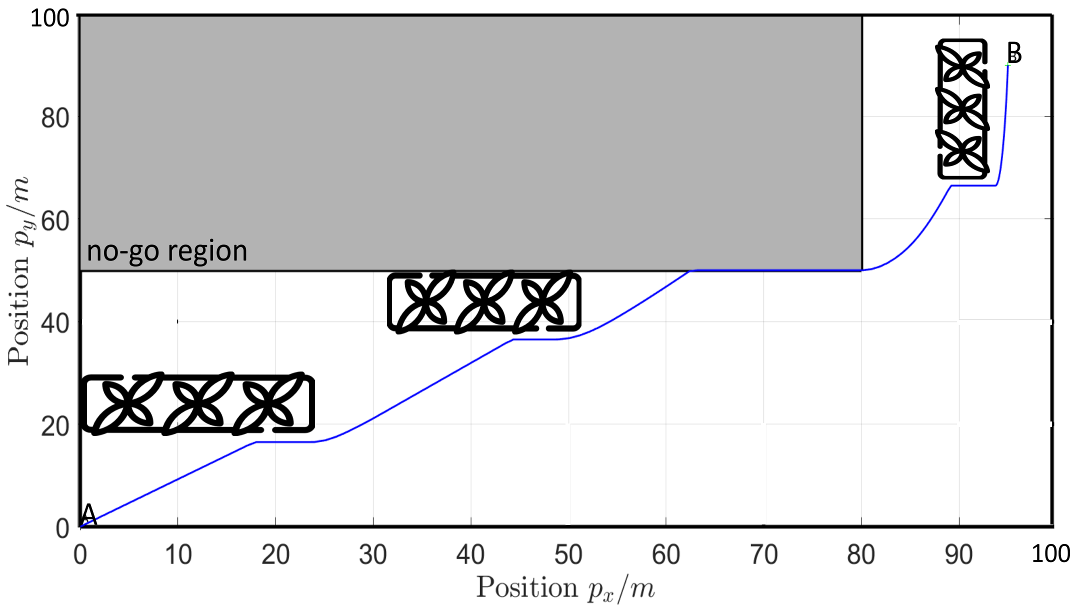


Figure 3.14: The offline planner plans a safe path from the start point to the final destination point, taking into account known obstacles while the no-go region is considered in the obstacle avoidance constraints .

bile robot reaches and stabilizes at a fixed reference state. In path following, a parameterized reference path allows flexible navigation while maintaining smooth motion. In trajectory tracking, the mobile robot follows a predefined time-dependent reference, ensuring precise execution but facing feasibility challenges due to dynamic constraints and physical limitations. The examples used a nonlinear mobile robot model to illustrate the trade-offs between computational complexity and real-time performance. The chapter also examined stability conditions and recursive feasibility to ensure robust and reliable control for various tasks.

The second part of the chapter focused on MPC-based path planning. A mobile robot must navigate an environment with static obstacles and limited prior information. To improve computational efficiency, the chapter introduced approximations such as linearized mobile robot dynamics and obstacle avoidance constraints formulated with Mixed-Integer Programming using the Big-M method. These techniques ensure safe and feasible motion planning while considering physical constraints like acceleration limits and system dynamics.

4 Efficient Hierarchical Planning and Control: Mathematical Programming and Local Sensing

The greatest challenge to any thinker is stating the problem in a way that will allow a solution.

Bertrand Russell

This chapter presents a hierarchical path-planning and control framework for autonomous mobile robot operating in partially or entirely unknown environments, such as warehouses with moving objects or dynamic indoor spaces. Traditional global path planning methods, which require prior knowledge of the environment, are unsuitable for such applications. Instead, online path planning is employed, dynamically adapting to real-time environmental data from onboard sensors, such as cameras and LiDAR. However, these sensors have inherent limitations in range and field of view, requiring continuous updates to the planned trajectory.

Obstacle detection and avoidance remain the primary challenges in online path planning. When no obstacles are present within the sensor range, a direct path to the destination can be efficiently computed. However, when obstacles are detected, the problem becomes computationally demanding, often requiring complex formulations to ensure safe and feasible trajectory generation in real-time. To alleviate this complexity, we propose a hierarchical approach where the planning problem is formulated using simplified linear vehicle dynamics, and obstacles are approximated via mixed-integer linear inequality constraints. This enables an efficient solution while guaranteeing obstacle avoidance within the sensor's field of view.

A local high-level planner computes a safe trajectory based on the latest sensor data, while a low-level controller employs trajectory-tracking MPC to follow the planned path, ensuring constraint satisfaction and optimized performance. Importantly, the limited sensor field of view is explicitly incorporated into the planning formulation, restricting the trajectory selection to areas where sensor data is available, thereby ensuring that the vehicle operates within a well-perceived environment.

At each time step, replanning and control are executed iteratively based on new sensor data, until the vehicle reaches its final destination. The hierarchical scheme is activated only when an obstacle is detected within the field of view (see Figure 4.1b). In the absence of obstacles, the low-level controller solves a set-point stabilization problem, incorporating vehicle dynamics, destination constraints, and sensor limitations (see Figure 4.1a).

This approach effectively balances computational efficiency and path safety in dynamic environments, ensuring real-time feasibility while adapting to sensor constraints and environmental uncertainties.

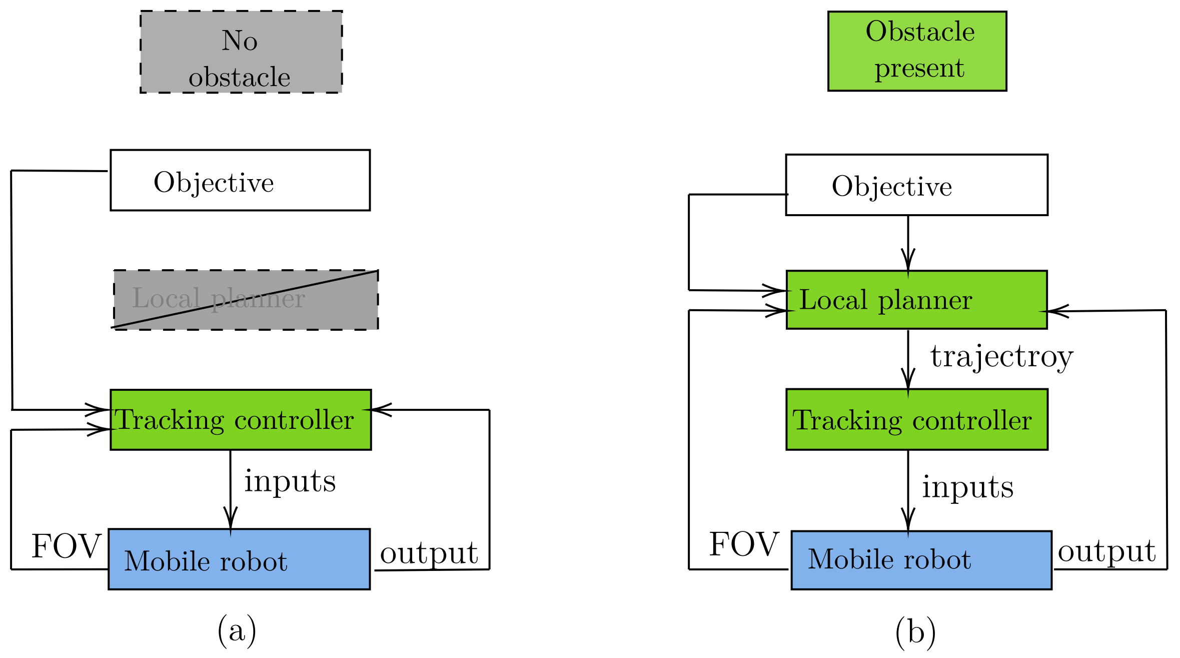


Figure 4.1: If no obstacles are detected (a), the low-level controller steers the vehicle using a setpoint-tracking formulation, considering the current field of view, destination, and vehicle dynamics, to generate a safe trajectory. Upon detecting an obstacle (b), the hierarchical scheme is activated: the high-level planner computes a safe trajectory within the sensor's field of view, incorporating the detected obstacle, and transmits it to the low-level controller for execution until the destination is reached.

4.1 Local Path Planning - Existing Approaches

We employ a local "High-level" planner for finding a path based on the local sensor information. The overarching control objective entails guiding the autonomous vehicle from its initial starting point to a final destination goal. The selection of path planners hinges upon the available information concerning the vehicle's working environment, as sketched in Chapter 2.

In the following, we summarize path planning approaches with a focus on local obstacle avoidance, complementing the general review of path planning provided in Chapter 2.

A global or offline path planner is utilized when complete knowledge of the environment is available before the autonomous mobile robot begins its motion. This knowledge includes the spatial dimensions of the environment, the locations, and shapes of obstacles, as well as their velocities. The offline-planned path, free from obstacles, establishes a connection between the initial starting position and the final destination point. In contrast, the online or local path planner primarily depends on information gathered within the immediate field of view of the mobile robot's onboard sensors. These sensors continuously acquire updated environmental data at each time step, enabling the planner to generate a safe path or trajectory in real-time. The planned path must remain confined to the current field of view to ensure the autonomous robot's safety. Extending the planning horizon beyond the sensor's field of view could pose risks to the robot's safety and hinder the successful completion of its task, see Figure 4.2.

Motion planning methods have been shaped by contributions from two primary fields: robotics, and dynamics and control. From the robotics perspective, the motion planner places greater emphasis on computational challenges and real-time robot control, while the dynamics and control developments focus on dynamic behavior and might simplify problem formulations to suit real-time applications. **Graph-Search** methods, such as roadmaps and the Dijkstra algorithm, represent safe paths as sequences of states, also known as nodes or waypoints, within free space [130, 144]. **Sampling-Based** approaches discretize the state and input spaces into a predefined library of steady states and transient trajectories that connect these states. This approach reduces computational time by limiting the planner's search to a finite set of precomputed states and inputs. Another category of planners includes **Heuristic Search** algorithms or Informed Search strategies, such as A* and D*. Over the past two decades, **Evolutionary Algorithms**, including Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO), have also been developed and applied to path-planning problems. While these methods generate obstacle-free paths, the resulting paths are often suboptimal and require significant memory resources [186]. Additionally, randomized planning approaches, such as **Rapidly-Exploring Random Trees** (RRT), can quickly identify feasible paths in cluttered environments by expanding a tree through random sampling of the configuration space. However, these methods face challenges such as high computational costs as the configuration space grows, lack of guarantees for convergence rate or optimality, and the generation of numerous redundant points during path planning [84, 212].

Several approaches have been proposed in the literature to address the local obstacle avoidance problem. For example in so-called **Bug Algorithms**, the vehicle is required to follow the contour detected by its sensors and navigate around obstacles [160]. A significant limitation of this method is that the vehicle's position relies mainly on sensor readings, which can result in situations where the sensor provides insufficient information about an obstacle, potentially leading to unexpected collisions. In the **Vector Field Histogram** (VFH) approach, the environment is represented using a 2D histogram grid [26]. This technique constructs a polar histogram around the vehicle at each time step, incorporating sensor data. The histogram sectors represent polar obstacle density, with values below a predefined threshold indicating free space

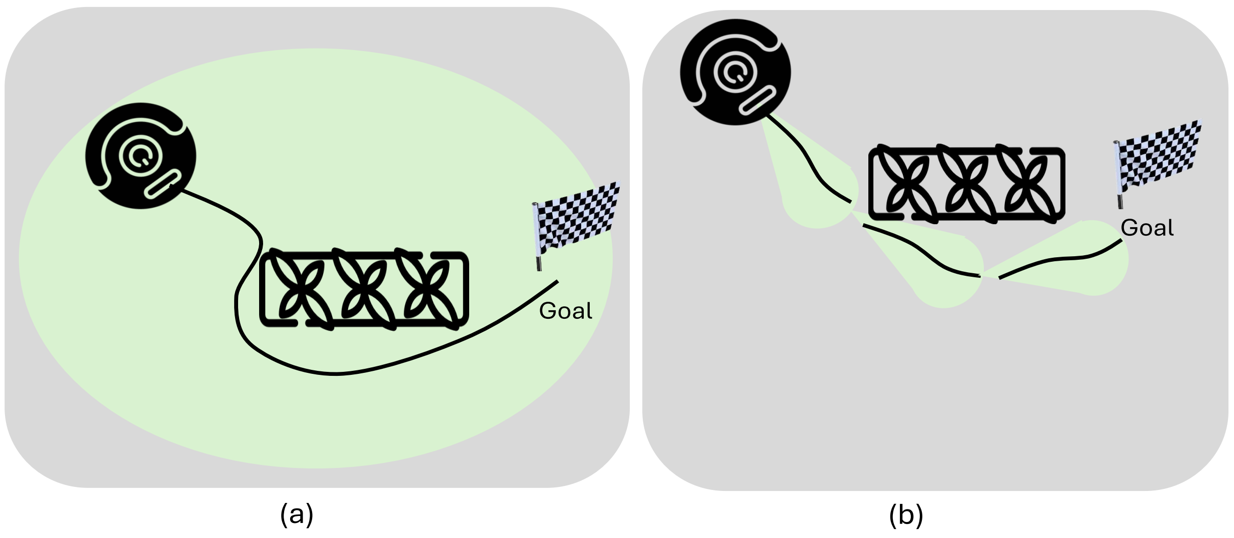


Figure 4.2: (a) Offline or global path planners use available offline information, while (b) online path planners calculate a path based on local, limited sensor information.

and values above the threshold marking obstacles. The algorithm then guides the vehicle toward safe regions based on this information.

The **Dynamic Window Approach** (DWA) is another method for obstacle avoidance, which focuses on generating a safe trajectory by selecting control inputs within the velocity space. In this approach, the trajectory is represented as a sequence of circular arcs [75]. Similarly, **Curvature Velocity** techniques incorporate vehicle dynamics into the planning process, taking into account the robot's velocity and acceleration constraints while modeling obstacles as circular objects. This method assumes that the vehicle's trajectory follows a circular arc, and the control velocity is selected based on an objective function. Another approach, the **Nearness Diagram** (ND) method, partitions the working environment into distinct regions to represent the positions of obstacles. Rooted in the situated activity paradigm, which defines specific situations and their corresponding appropriate actions, the algorithm determines the current state during execution and performs the corresponding action [156].

In **Potential Field** methods, system is treated as a particle influenced by a potential field generated by the goal location and obstacles [105]. The core concept of this approach is that the final point exerts an attractive force to draw the vehicle toward it, while obstacles generate repulsive forces to keep the vehicle at a safe distance. Although these methods are computationally efficient and well-suited for real-time applications, they are prone to issues such as becoming trapped in local minima and providing imprecise representations of obstacle shapes and dimensions, as these factors are not explicitly treated as constraints.

In [96], the authors represent the online path planning based on Rapidly-Exploring Random Tree (RRT*). The authors in [230] address the challenge of planning optimal paths in complex, dynamic urban environments. The authors propose a hybrid approach that combines offline pre-planning with online adaptation to handle multiple objectives and real-time changes in the environment based on heuristic search and predictive modeling to anticipate the position of the moving obstacles. A modified potential field method is presented in [28]. The online path planning relies on continuously updating the repulsive and attractive forces based on the robot's current state and the environment. In the work presented in [188], the authors present a novel approach for autonomous robots to explore and map unknown environments efficiently. The key focus is online informative path planning, where the robot must dynamically decide its path to maximize information gain while operating in real-time. The proposed method uses a sampling-based strategy to generate candidate paths that are continuously updated as new sensor data is collected. In [161], the authors present an online adaptation of the RRT algorithm for real-time path planning in dynamic and uncertain environments, by incrementally updating the tree based on the sensor information. Notably, most of the planning algorithms mentioned earlier do not incorporate the differential constraints of the vehicle, such as its dynamics, during the planning process. This omission can result in infeasible paths and undesirable vehicle behavior, particularly in scenarios involving fast dynamics [107].

From a dynamics and control perspective, a widely used approach for solving path planning problems while incorporating vehicle dynamics is **Mathematical Programming** [84]. In this framework, the path planning problem is formulated as an optimal control problem, where the objective function is designed to minimize a specific performance metric, such as travel time to a goal state, control energy consumption, or deviation from a reference trajectory.

By solving the resulting constrained optimization problem, which accounts for both vehicle dynamics and obstacle avoidance, an optimal path is generated based on the defined performance metric. We propose using mathematical programming for local path planning, leveraging real-time sensor data and taking sensor limitations, such as limited field of view, into account. To improve efficiency, we distinguish between obstacle-free and obstructed scenarios. Furthermore, we formulate the planning problem in mixed-integer form, reducing computational complexity by approximating nonlinear functions with linear equations and inequalities.

4.2 Efficient Local Path Planning via Mathematical Programming Using Sensor Field of View Data

The moving-horizon path planner presented in Chapter 3 computes a safe trajectory in a known environment. However, solving such problems in real time remains computationally challenging, especially for efficient on-board re-planning. In contrast, local re-planning, which is the focus of this chapter, requires high computational efficiency. At each time step, onboard sensors acquire new but inherently limited environmental data due to their restricted sensing capabilities, see Figure 4.3 and Figure 4.4. The sensors only provide information in the field of view set $\mathcal{Z}_t$.

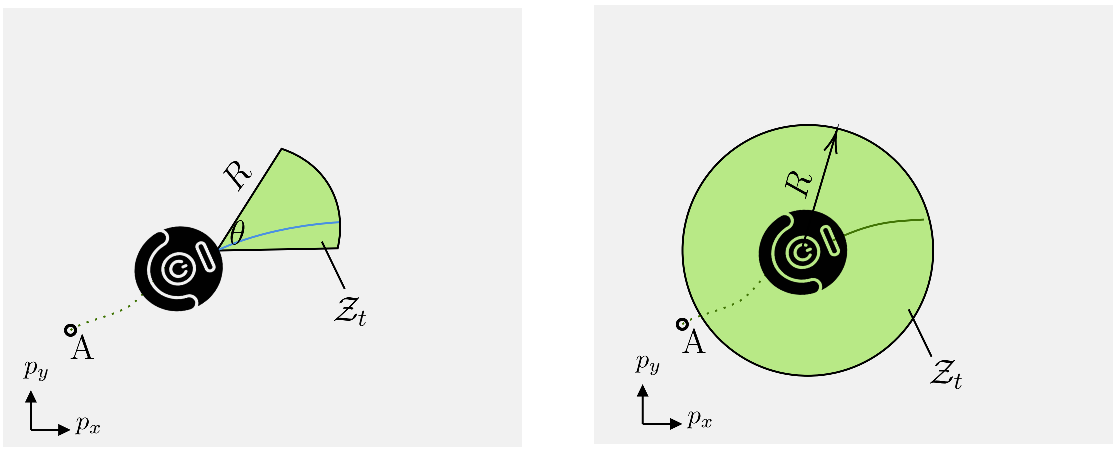


Figure 4.3: Autonomous vehicles are typically equipped with onboard sensors, such as cameras or LiDAR, which have a limited field of view, characterized by parameters such as the viewing angle θ and the sensor range R. This defines the field of view $\mathcal{Z}_t$. In the illustrated examples, the field of view is entirely free of obstacles, meaning that the detected obstacle-free region coincides with the entire sensing area, i.e., $\mathcal{Z}_t^f = \mathcal{Z}_t$. Consequently, the local path planner can generate a safe trajectory within the entire field of view, as no obstacles are present at time t.

When relying solely on local sensor data, the planner must ensure that trajectory generation is restricted to areas where sensor information is available, guaranteeing safety. Therefore, we first address how sensor limitations can be incorporated into the planning process, distinguishing between obstacle-free and obstructed scenarios. This data includes partial or complete obstacle detections or may indicate a clear path with no obstacles in the field of view/under the local sensor information. Additionally, to enhance real-time feasibility, we propose using suitably formulated and approximated representations for both vehicle dynamics and constraints.

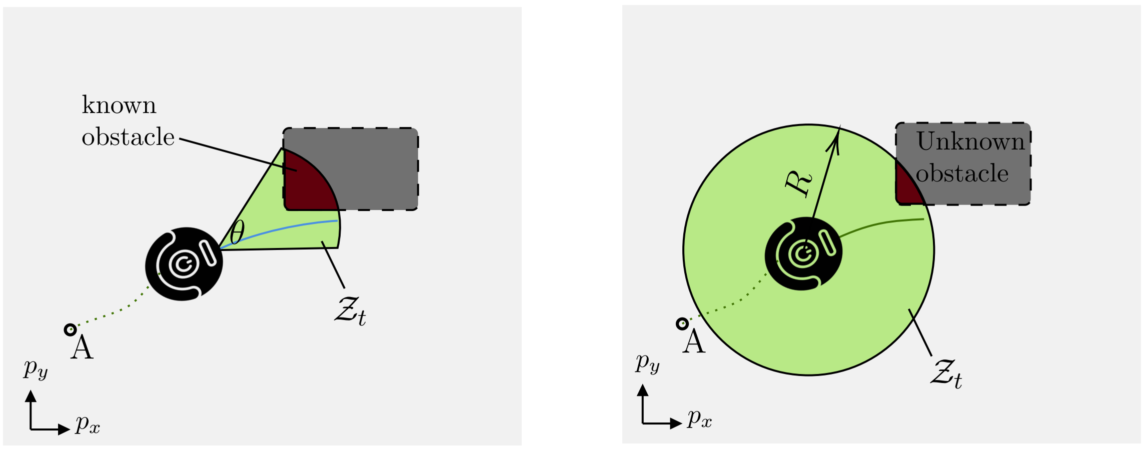


Figure 4.4: Limited sensing capabilities will lead to only partially detected obstacles (shown in red) and will lead to a set $\mathcal{Z}_t^f = \mathcal{Z}_t \setminus \bigcup_i \mathcal{O}_{i,t}$, which describes the obstacle free region in the field of view. Here $\mathcal{O}_{i,t}$ represents an obstacle i detected at time t. The local path planner is only allowed to generate a safe trajectory within the obstacle free field of view $\mathcal{Z}_t^f$.

4.2.1 Planning with no Obstacles in the Field of View

In the obstacle-free case, we propose that at each time step, the online path planner solves the following optimization problem:

$$\begin{aligned}
\min_{\{u\}} \quad & \sum_{k=0}^{N_\mathrm{p}-1} (\|p_k - p_B\|_\infty + \|u_k^\mathrm{p}\|_\infty) && (4.1a)\\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k), \;\; x_0 = \hat{x}(t), && (4.1b)\\
& p_k \in \mathcal{Z}_t, && (4.1c)\\
& g(x_k, u_k) \leq 0, && (4.1d)\\
& \forall k \in [0, ..., N_\mathrm{p}]. && (4.1e)
\end{aligned}$$

Here $x_k = \left[{p_k}^\mathsf{T}, \cdots\right]^\mathsf{T}$ contains all system states, i.e., the coordinate of the center of mass p_k and other states, e.g., heading angle and pitch. $\hat{x}(t)$ is the measured/estimated system state at the current time t, and u_k is the input. The objective function (4.1a) minimizes the distance in the infinity norm from the desired point while minimizing the energy. While the objective function could be anything, we focus in this section on the infinity norm, as this will simplify the optimization later on. Equation (4.1b) represents the system dynamics. Equation (4.1d) represents constraints, e.g., on velocities and acceleration. Equation (4.1c) enforces that the planned path remains within the current field of view $\mathcal{Z}_t$. The solution of the online path planning problem (4.1) results in a path that is executable by the autonomous vehicle dynamics and safe, as it belongs to the current field of view.

4.2.2 Planning Subject to Obstacles in the Field of View

When an obstacle is detected, either fully or partially, the planner is required to generate a feasible path that adheres to the autonomous mobile robot's dynamic constraints while ensuring safe obstacle avoidance, see Figure 4.4. Notably, the detection of obstacles within the field of view leads to a time-varying obstacle-free set:

$$\mathcal{Z}_t^f = \mathcal{Z}_t \setminus \bigcup_i \mathcal{O}_{i,t}. \tag{4.2}$$

Here $\mathcal{O}_{i,t}$ represents the obstacle i detected at time t. To ensure safety, an obstacle-free path can only be planned within the current field of view. To do so, we propose that the online path planning problem (4.1) is augmented by obstacle avoidance constraints, formulated as follows:

$$\begin{aligned}
\min_{\{u\}} \quad & \sum_{k=0}^{N_\mathrm{p}-1} (\|p_k - p_B\|_\infty + \|u_k^\mathrm{p}\|_\infty) && \text{(4.3a)}\\
\text{s.t.} \quad & x_{k+1} = f(x_k, u_k),\ \ x_0 = \hat{x}(t), && \text{(4.3b)}\\
& p_k \in \mathcal{Z}_t, && \text{(4.3c)}\\
& p_k \notin \mathcal{O}_{i,t}, && \text{(4.3d)}\\
& g(x_k, u_k) \leq 0, && \text{(4.3e)}\\
& \forall k \in [0, ..., N_\mathrm{p}]. && \text{(4.3f)}
\end{aligned}$$

Here (4.3d) represents the obstacle avoidance constraint. The overall problem is in general nonlinear and non-convex.

Similar to the global moving horizon planner, complexity of the overall problem is reduced by suitable approximations, see Section 3.6.1. The high-level planner will only consider linear vehicle dynamics and constraints approximated by linear inequalities and mixed-integer formulations, see Section 3.6.3 and 3.6.4. In the following we focus on approximating and modeling the field of view

4.2.3 Modeling and Approximating the Field of View

Autonomous mobile robots are commonly outfitted with onboard sensors such as camera systems or LiDAR. These sensor systems play a crucial role in furnishing the controller/planner layer with environmental information essential for safe navigation. Specifically, this information delineates the intersection between obstacles and the current sensor field of view, see Figure 4.4. These sensors have limited sensing capabilities, i.e., limited range R and/or limited sensing angle θ, and can be approximated by a set of time-varying inequality constraints (cf. Figure 4.5) such that:

$$\mathcal{Z}_t := \{(x, y) \mid A_t[x\ y] \leq B_t\}. \tag{4.4}$$

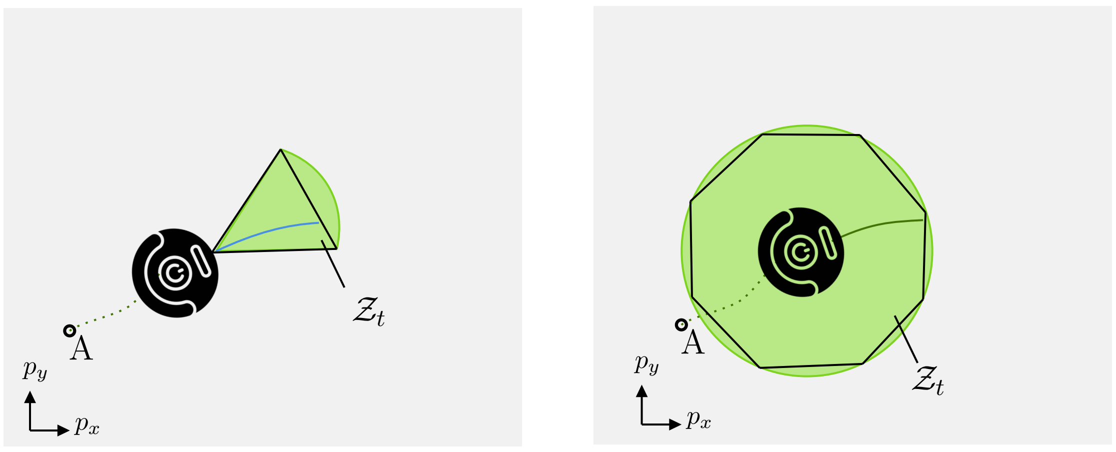


Figure 4.5: The field of the onboard sensors can be approximated by the intersection between a set of inequalities.

A_t and B_t are time-varying matrices with proper dimensions representing the polytopic approximation of the field of view at time step t. It is worth noting that the entries of A_t and B_t matrices vary at each time instant according to the vehicle position. Moreover, the field of view region $\mathcal{Z}_t$ is assumed to be a safe convex set as long as no obstacle is detected, i.e.,

$$\mathcal{Z}_t \cap \mathcal{O} = \emptyset. \tag{4.5}$$

By replacing the field of view in (4.1c) or in(4.3c) with the approximation (4.4), the planner predicts a state trajectory that belongs to $\mathcal{Z}_t$.

Remark 2. *The field of view approximations depicted, see Figure 4.5, are conservative, resulting in some regions being clipped. However, doing so guarantees the safety of the planned path. Additionally, the field of view is dynamically updated at each subsequent time instance, and the corresponding approximation is adjusted accordingly.*

4.2.4 Obstacle Approximation in the Field of View

The intersection of an obstacle with the field of view results in a time-varying non-convex set $\mathcal{Z}_t^f$ (4.2) see Figure 4.6. This increases computational complexity. To alleviate the computational budget, the obstacles $\mathcal{O}_i$ that are detected within the current field of view are represented by linear inequality constraints as follows:

$$\mathcal{O}_i := \{(x, y) \mid A_t^o[x\ y] \leq B_t^o\}. \tag{4.6}$$

$\forall i \in [1, 2, \cdots, N_o]$ where N_o is the number of the detected obstacles within the field of view. Please note that in (4.6), the row entry of the matrices A_t^o and B_t^o represents a "plane", and the "intersection between these planes" represents the approximated detected obstacle. Also, the direction of the inequality reflects the definition of the obstacle region $\mathcal{O}_i$. To ensure the safety of the planned path despite the detected or

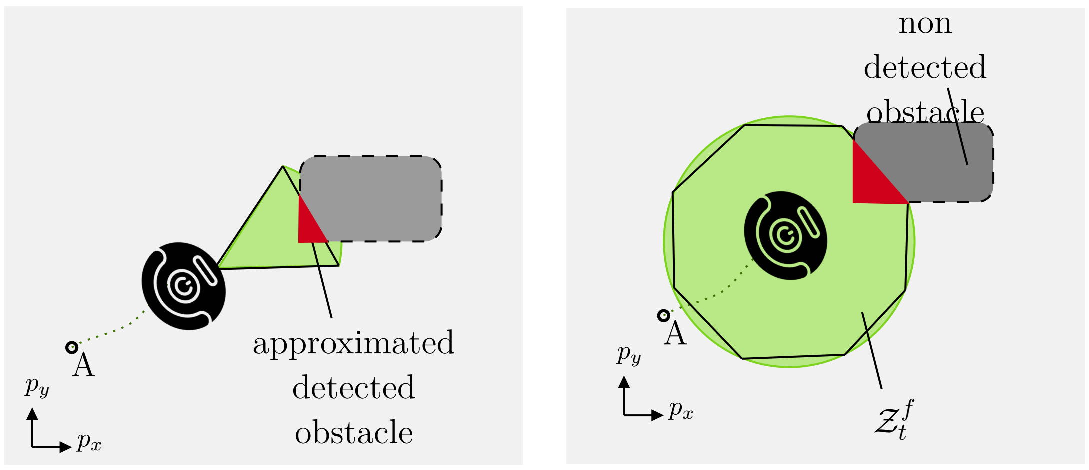


Figure 4.6: The field of view is approximated and the detected obstacle (in red) is considered in the optimization problem.

only partially detected obstacle, obstacle avoidance is enforced via the Big-M method and binary variables d as follows: $\forall k \in [t, t+N_\mathrm{p}]$

$$A_t^o p_k \geq B_t^o + M(1-D). \tag{4.7}$$

Here, $p \in \mathbf{R}^2$ represents the coordinates of the autonomous robot. It's essential to note the change in the direction of the inequality because we aim to ensure that the ground vehicle's position lies outside the obstacle set. Additionally, the vector D encompasses binary variables, denoted as follows:

$$D = [d_1 \; d_2 \cdots d_n]^\top. \tag{4.8}$$

Here n is the number of rows in A_t^o or B_t^o. To ensure that one constraint is active, i.e., must be satisfied while the others are relaxed at each prediction step, the following constraint is imposed: $\forall i \in [1, ..., n]$:

$$\sum_{k=t}^{t+N_\mathrm{p}} d_{i,k} = 1. \tag{4.9}$$

Therefore, the obstacle avoidance constraints (4.3d) can be approximated by binary variables in (4.8) and (4.7).

Practically, the sampling rate in the planner layer plays an important role in avoiding the obstacle, i.e., with a fast sampling/re-planning rate, the planner could provide waypoints/states that indeed avoid the obstacle but intersect with the obstacle. However, a shorter re-planning rate could increase the computational complexity. Therefore, one should consider obstacle enlargement to ensure obstacle avoidance [73, 107]. Thus, the detected obstacle within the field of view is enlarged by a safety margin δ_{safe}, cf. Figure 4.7. This safety margin is a function of vehicle dynamics, e.g., v_{max}

and the sampling rate in the planner layer T_s such that [73]:

$$\delta_{safe} \geq \frac{v_{max} T_s}{2} \, sin(\alpha). \tag{4.10}$$

where α is the angle between the boundary and the path between two waypoints.

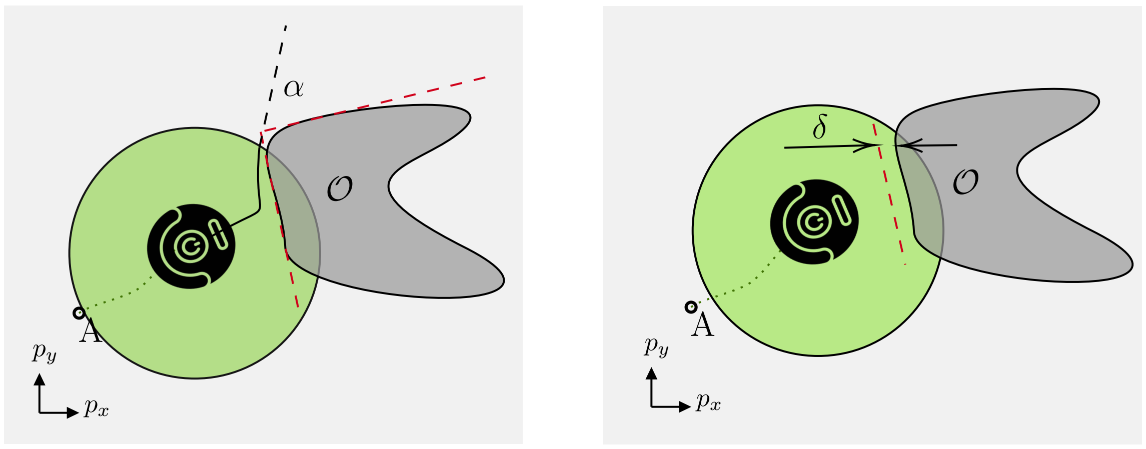


Figure 4.7: Obstacle enlargement by a safety margin δ to ensure vehicle safety despite corner-cutting. The executed path is depicted in solid green, whereas the planned path is shown in dotted black.

4.2.5 Overall Linear Mixed-Integer Moving-Horizon Local Planner Formulation

The outlined approximations and reformulations enable the formulation of a linear, mixed-integer local planner operating on a moving horizon, which integrates and accounts for all available local sensor and obstacle data.

$$\min_{\{u^{\mathrm{p}}\},\, d} \quad \sum_{k=0}^{N_{\mathrm{p}}-1} \left(\|p_k^{\mathrm{p}} - p_{\mathrm{B}}^{\mathrm{p}}\|_2^Q + \|u_k^{\mathrm{p}}\|_2^R \right) \tag{4.11a}$$

$$\text{s.t.} \quad x_{k+1}^p = Ax_k^{\mathrm{p}} + Bu_k^{\mathrm{p}},\ x_0^{\mathrm{p}} = \hat{x}(t), \tag{4.11b}$$

$$A_{i,t}^o p_k^{\mathrm{p}} \geq B_{i,t}^o + M_{\mathrm{big}}(\mathbf{1} - d_{i,k}), \tag{4.11c}$$

$$A_t p_k^{\mathrm{p}} \leq B_t, \tag{4.11d}$$

$$a_{x_k} - a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{4.11e}$$

$$-a_{x_k} + a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{4.11f}$$

$$a_{y_k} - a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{4.11g}$$

$$-a_{y_k} + a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{4.11h}$$

$$v_x \cos \frac{2\pi m}{M} + v_y \sin \frac{2\pi m}{M} \leq v_{max}, \tag{4.11i}$$

$$a_x \cos \frac{2\pi m}{M} + a_y \sin \frac{2\pi m}{M} \leq a_{max}, \tag{4.11j}$$

$$\sum_{k=1}^{n} d_{i,k} = 1, \tag{4.11k}$$

$$\forall k \in [0, ..., N_{\mathrm{p}}], \tag{4.11l}$$

$$\forall i \in [1, ..., n], \tag{4.11m}$$

$$\forall m \in [1, ..., M]. \tag{4.11n}$$

In the optimization problem(4.11), we use the superscript p to refer to the planner layer. The objective function (4.11a) tries to find an optimal control input sequence $\overset{*}{u}{}^{\mathrm{p}}$ that minimizes the error between the vehicle position p_k^{p} and the final point position $p_{\mathrm{B}}^{\mathrm{p}}$ at each prediction step k considering linear system dynamics (4.11b). The nonlinear state constraints on velocity and acceleration are approximated using an M-sided polygon and expressed by linear constraints (4.11j, and 4.11i). Vehicle safety is ensured by enforcing obstacle avoidance using the Big-M method and binary variables in (4.11c). Moreover, the planned state trajectory has to be inside the current field of view in (4.11d). The solution of the optimization problem (4.11) results in an optimal obstacle-free state trajectory $x_t^{*p} = \left[x_t^{\mathrm{p}*\mathsf{T}}, ..., x_{t+N_{\mathrm{p}}}^{\mathrm{p}*}{}^{\mathsf{T}} \right]^{\mathsf{T}}$.

In summary, the local planning layer is responsible for generating an obstacle-free path within the current field of view $\mathcal{Z}_t^f$ while minimizing a user-defined objective function. This path must satisfy nonlinear vehicle dynamics and nonconvex obstacle avoidance constraints, making the optimization problem (4.3) both nonlinear and nonconvex, and consequently computationally intensive. To address this complexity, several approximation techniques are introduced. Linear vehicle dynamics are considered, and constraints are approximated using polygons. Obstacle avoidance constraints are enforced through the Big-M method. Additionally, the current field of view of the onboard sensor is approximated by a set of time-varying linear inequality constraints, ensuring that the planned path remains within the sensor's range. These approximations reduce the computational burden while maintaining the feasibility and effectiveness of the planned path.

Solving this problem yields an optimal, obstacle-free path that is constrained to the current sensor range. The computed path is then transmitted to the low-level controller for execution.

4.3 Low Level Receding Horizon Control

As shown in Figure 4.1, a safe planned path is sent from the high-level planner to the low-level controller, which should steer the autonomous mobile robot within the current field of view along this path. The low-level controller exploits the MPC trajectory tracking formulation presented in Section 3.5, incorporating nonlinear vehicle dynamics to ensure accurate path execution. New sensor data is continuously processed to detect new obstacles and switch the operational mode if required, until the final position is reached.

Specifically, the following MPC trajectory tracking formulation is used, based on the optimal obstacle-free path $x_t^{*p} = \left[x_t^{\mathrm{p}*\mathsf{T}}, ..., x_{t+N_\mathrm{p}}^{\mathrm{p}*}{}^{\mathsf{T}} \right]^{\mathsf{T}}$:

$$
\begin{aligned}
\min_{\{u\}} \quad & J(\mathbf{x}_t^{*p}, x_k, u_k). && \text{(4.12a)} \\
\text{s.t.} \quad & x_{k+1}^c = f(x_k^c, u_k^c),\ x_t^c = \hat{x}(t), && \text{(4.12b)} \\
& x_{k+1} \in \mathcal{X},\ u_k \in \mathcal{U},\ h(x_{k+1}) \in \mathcal{Y}, && \text{(4.12c)} \\
& \quad \forall k \in [0, ..., N_\mathrm{p}]. && \text{(4.12d)}
\end{aligned}
$$

In the trajectory tracking optimization problem (4.12), the superscript c denotes the controller layer. While the objective function in (4.12a) remains constant over time, the time dependency enters the optimization problem through the time-varying reference $\mathbf{x}_t^{*p}$. This implies that a new reference is available to the controller at each time instant, incorporating new information captured by the onboard sensor. The role of the low-level controller is to guide the autonomous robot along the reference trajectory sent by the high-level planner while considering potential nonlinear vehicle dynamics (4.12b), as well as constraints on predicted states, inputs, and outputs, as outlined in (4.12c). The safety of the autonomous robot is ensured as long as the obstacle avoidance constraints are respected by the high-level planner, resulting in a safe, obstacle-free path. The effectiveness of the proposed hierarchical approach is validated through the simulation examples presented below.

4.4 Simulation Examples

In the subsequent examples, we examine a scenario involving an autonomous mobile robot equipped with onboard sensors, such as LiDAR or a camera system, as illustrated in Figure 4.3. The vehicle operates in a partially known environment, where certain environmental details, including the positions and geometries of obstacles, are known prior to the execution of the vehicle's motion. However, some areas of the environment remain unknown or unexplored. By utilizing the global path planner and the mixed-integer programming formulation (3.35), a safe path is generated to connect the initial starting point A with the final destination point B.

Using the model predictive control path-following formulation (3.13), the autonomous ground robot safely follows the offline planned path until the onboard sensors detect a no-go zone or an unknown region. When such a region is detected, the online path-planning algorithm is activated, and the MPC path-following is deactivated to facilitate exploration of the unknown area. The proposed online path planning algorithm is responsible for generating a safe path based on the sensor data available at each time instant t.

The provided examples highlight the efficacy of the proposed hierarchical approach, which activates upon the detection of obstacles by the onboard sensors and deactivates in obstacle-free scenarios, as depicted in Figure 4.1. This method ensures efficient and adaptive navigation in dynamic and partially unknown environments. The dynamics of the unmanned ground vehicle are modeled using a kinematic bicycle model, described as follows [111]:

$$\dot{p}_x = v\cos(\psi), \tag{4.13a}$$
$$\dot{p}_y = v\sin(\psi), \tag{4.13b}$$
$$\dot{\psi} = v\tan(\delta)/L, \tag{4.13c}$$
$$\dot{v} = u_1, \tag{4.13d}$$
$$\dot{\delta} = u_2. \tag{4.13e}$$

Equations (4.13a) and (4.13b) represent the dynamics of the center of mass of the vehicle while the heading angle dynamics is given by (4.13c). The control inputs u_1 and u_2 are the acceleration and steering angle rates, respectively. Generally, (4.13) can be expressed as:

$$\begin{aligned} \dot{x}(t) &= f(x(t), u(t)), \qquad x(0) = x_0, \\ y(t) &= h(x(t)). \end{aligned}$$

where x represents the state vector $[p_x, p_y, \psi, v, \delta]^\top$ and u is the control vector $[u_1, u_2]^\top$ and y is the system output.

Example 6. *In this scenario, the autonomous ground robot is equipped with a LiDAR sensor. The LiDAR has a limited field of view as illustrated in Figure 4.3. The*

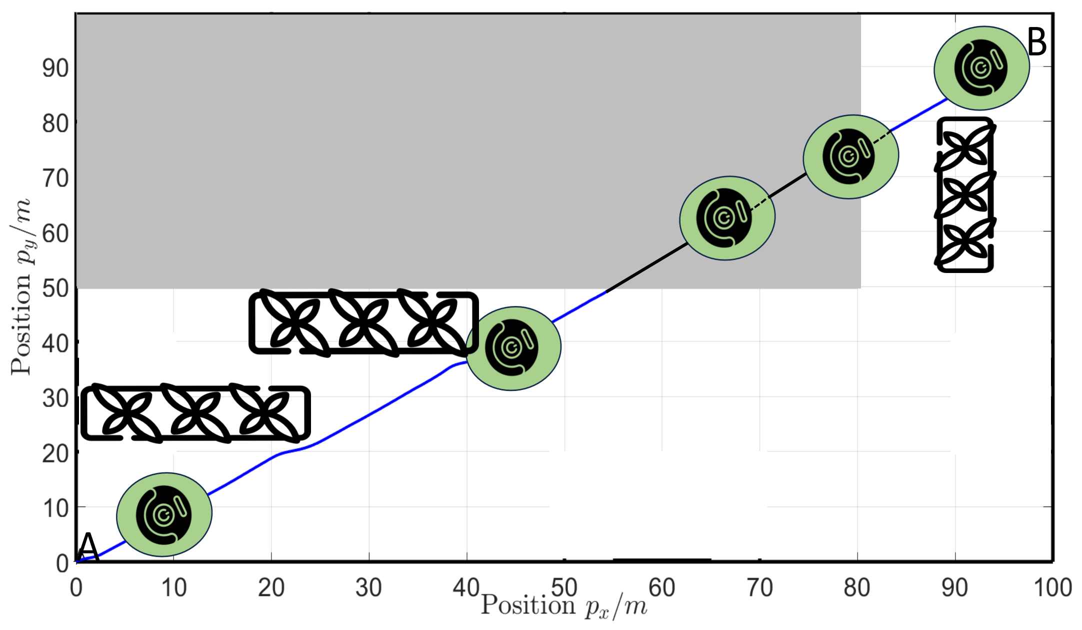


Figure 4.8: The vehicle followed the offline planned path until detecting the unknown region (in gray). Upon unknown region detection, the proposed online path planner is activated to plan a safe path (dashed black line) within the field of view represented in green.

.

controller leverages the maximum capabilities of the autonomous robot to follow the preplanned path until the unknown region is detected. It is important to note that the controller will not respond to obstacles the onboard sensor detects during the path following execution, as obstacle avoidance is inherently addressed during the planning phase. This ensures that the planned path is already obstacle-free, allowing the controller to focus solely on accurately following the precomputed path without the need for real-time obstacle avoidance adjustments. Upon exploring the unknown region, the proposed online path planning is activated, and the robot's speed is reduced to safely navigate the unknown region such that $v_{safe} < v_{max}$. *This approach is essential to ensure the safety of the autonomous robot, as it allows the vehicle to stop within the current field of view if an unknown obstacle is detected. As long as no obstacles are detected within the field of view, the controller/planner solves the MPC formulation (3.6), considering the final destination point. The controller/planner must generate a safe path within the current field of view, as outlined in (4.1), while accounting for the field of view approximation described in (4.4). Once the robot re-enters the known region, the controller accelerates the autonomous robot to its maximum velocity until the destination point is reached, as illustrated in Figure 4.9. It is important to note that the problem of online path planning becomes significantly more challenging when mul-*

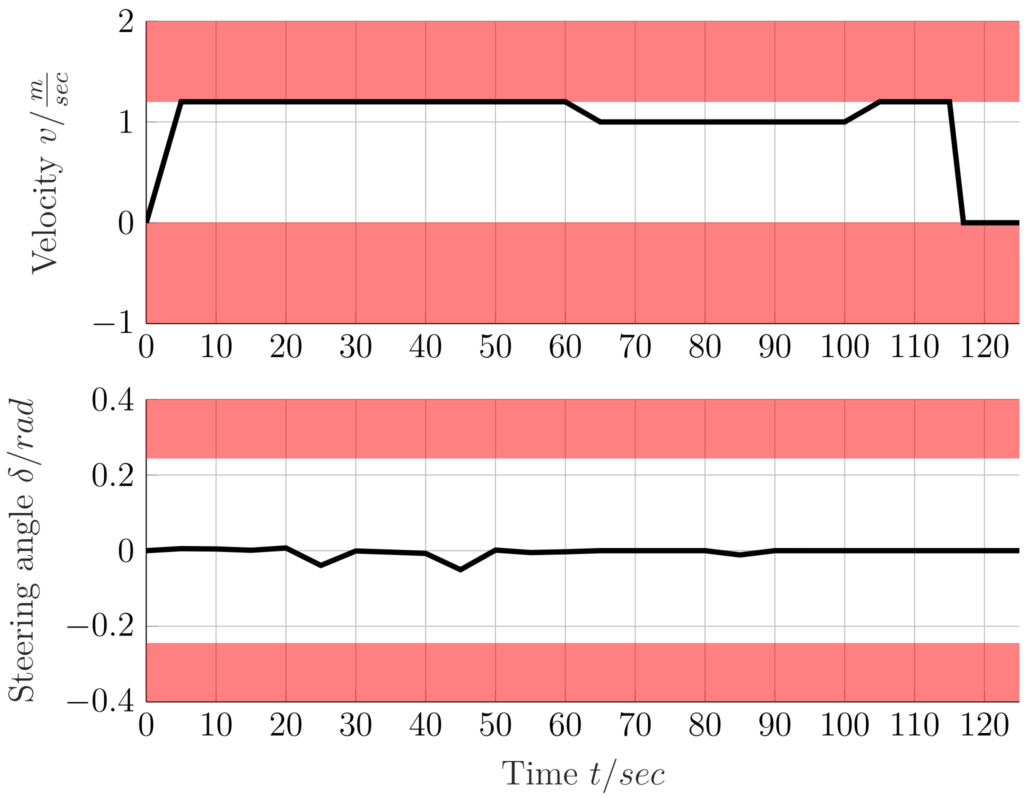


Figure 4.9: The controller is aware of the physical vehicle limitations in velocity and steering angle. The autonomous robot speed is reduced to ensure its safety while exploring the unknown region.

tiple obstacles are detected during the exploration of unknown regions, as demonstrated in the following examples.

Example 7. *In this example, we investigate the activation of the proposed hierarchical online path planning and control framework. The autonomous ground vehicle is equipped with a LiDAR sensor, which captures environmental data within its vicinity, as illustrated in Figure 4.3. Unlike the scenario presented in Example 6, where no obstacles are present, this example involves the detection of obstacles within the LiDAR's field of view. Upon obstacle detection, the hierarchical receding horizon control approach is activated (cf. Figure 4.1). The high-level planner generates a safe path within the current field of view, taking into account the detected or partially detected obstacles, the linearized dynamics of the autonomous vehicle, and the field of view constraints. The solution to the mixed-integer programming optimization problem (4.11) in the high-level planner results in an obstacle-free trajectory within the current field of view. This safe trajectory is then passed to the low-level controller.*

The low-level controller employs the nonlinear model predictive control trajectory tracking formulation (4.12) to follow the safe trajectory provided by the high-level planner (cf. Figure 4.10). To ensure the vehicle's safety, the detected obstacle is enlarged according to the criterion specified in (4.10), accounting for the vehicle's dimensions.

Figure 4.10 illustrates the autonomous ground vehicle accelerating to its maximum speed while adhering to the offline planned path. When the controller identifies a no-go region, it navigates the unexplored area at a reduced speed, as demonstrated in Example 6. Due to the detection of obstacles by the onboard LiDAR sensor, the vehicle's speed decreases to satisfy the condition outlined in (4.10), and it resumes acceleration once no obstacles are detected. New environmental information is gathered at each subsequent time step, prompting the system to reevaluate and adjust the planned path accordingly. The hierarchical problem is solved on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz with average computation time $t_{cpu} = 9.1702$ *ms.*

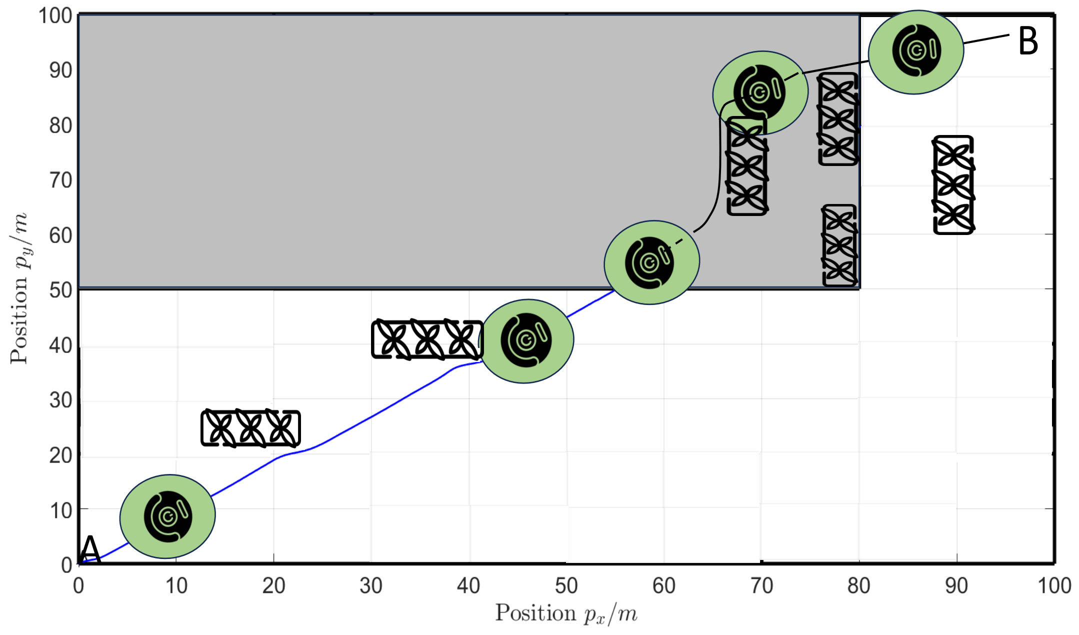


Figure 4.10: The vehicle, equipped with a LiDAR sensor represented by a green circle, is guided by an MPC path-following controller along a safe planned path until an unknown region is detected. The proposed online planning scheme is then activated. Upon obstacle detection, the hierarchical framework is activated to plan a safe path (dashed black line) in the field of view.

Example 8. *In this example, we consider an autonomous ground robot equipped with an onboard camera with limited sensing capabilities, i.e., limited sensing range and angle (cf.Figure 4.3), rendering a more challenging online path planning problem. The onboard sensor has a limited range R and sensing angle θ with zero pitching angle. Similar to Example 7, the autonomous ground robot follows the safe planned path till detecting the no-go region. The online path planning and control steers the autonomous robot to explore the region by solving the optimization problem (4.1) as no obstacle is*

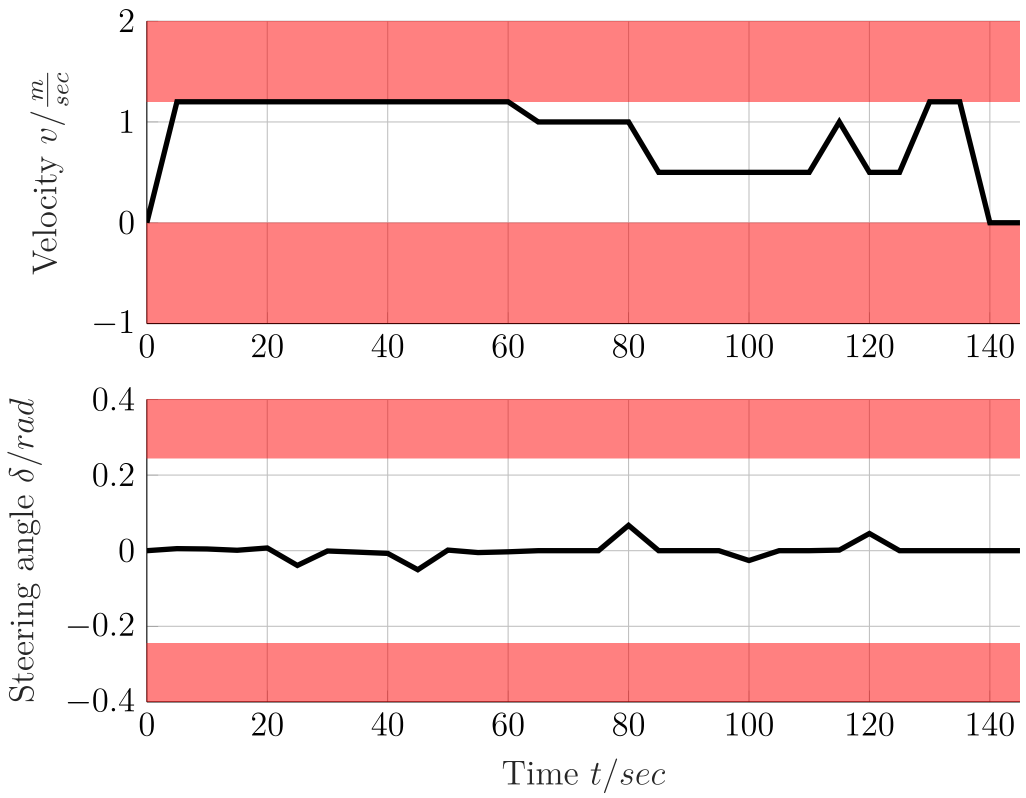


Figure 4.11: The controller maximizes the robot's capabilities to follow the safe planned path. The proposed online path planning enforces a safe robot velocity to ensure safety during the exploration of the unknown region. The controller accelerates it to its maximum speed in the known region.

detected. The proposed hierarchical scheme is activated when an obstacle is detected via the onboard camera system.

The hierarchical problem is solved on an Intel® Core^{TM} i7-6700 CPU @ 3.40 GHz with average computation time in the planner layer $t_{cpu} = 83.7$ ms and $t_{cpu} = 0.33733$ ms in the controller layer.

Note that in the provided examples, the obstacles are assumed to have a rectangular geometry, but the formulation remains valid for any obstacle shape, as detected obstacles can be approximated by linear inequalities based on the sensor's readings.

In the previous examples 8 and 7, the controller/planner does not explicitly incentivize the ground mobile robot to explore the detected obstacle, which could potentially improve the overall objective of reaching the goal point in a shorter time frame.

4.5 Summary

This chapter introduced an online path planning framework in which the planner generates a safe trajectory strictly within the sensor's field of view, ensuring that the autonomous mobile robot only navigates in regions where reliable sensor data

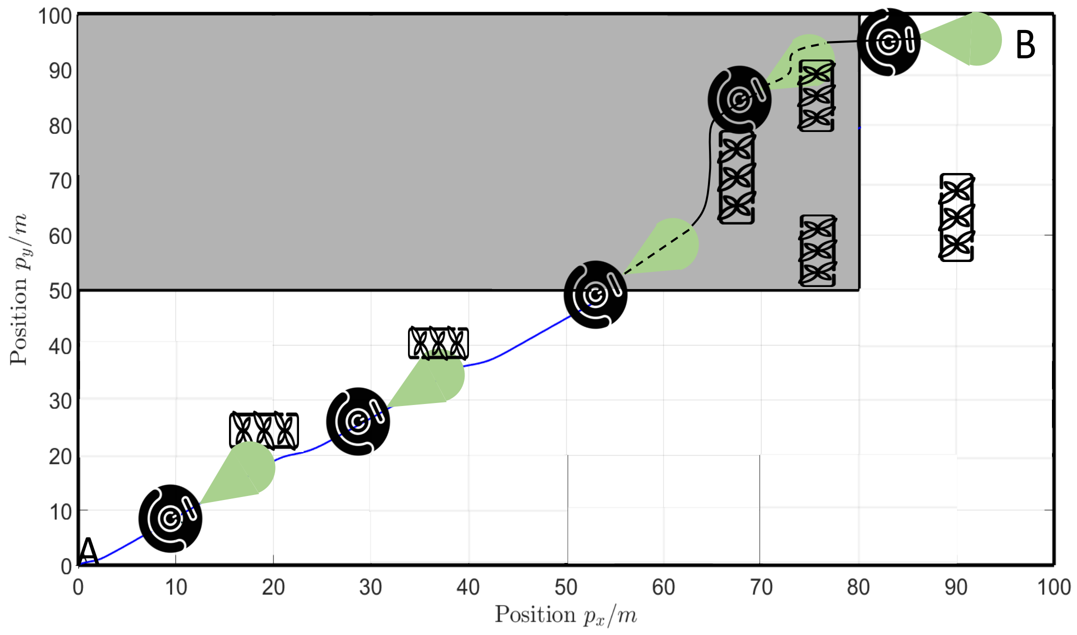


Figure 4.12: The autonomous mobile robot is equipped with a camera system represented in green. The proposed scheme is activated when the onboard sensor detects the unknown region (depicted in gray) to plan a safe path (dashed black line) in the field of view. The hierarchal scheme is active upon obstacle detection.

is available. This restriction guarantees that the vehicle always operates in a well-perceived environment, preventing it from making decisions based on uncertain or extrapolated data.

A key challenge in real-time execution is computational efficiency, as both the local planner and the low-level controller must be executed onboard the vehicle with limited computational resources. To address this, we proposed a hierarchical structure that efficiently balances path planning and trajectory tracking while ensuring real-time feasibility. To enable efficient and safe trajectory generation, the sensor's field of view and the unobstructed (obstacle-free) portion of the field of view were explicitly modeled. This distinction allows the planner to incorporate both sensor limitations and detected obstacles in its decision-making process.

The local planner employs a linear mixed-integer programming formulation, which leverages simplified linear vehicle dynamics and approximated constraints to reduce computational complexity. Obstacle avoidance constraints are enforced using the Big-M method, ensuring that feasible paths are efficiently computed at each planning step. A fundamental property of the proposed approach is that collision avoidance is guaranteed, as the detected obstacles are conservatively approximated using polytopic representations. This conservative modeling ensures that the planned path remains safe even when obstacles are only partially detected within the field of view.

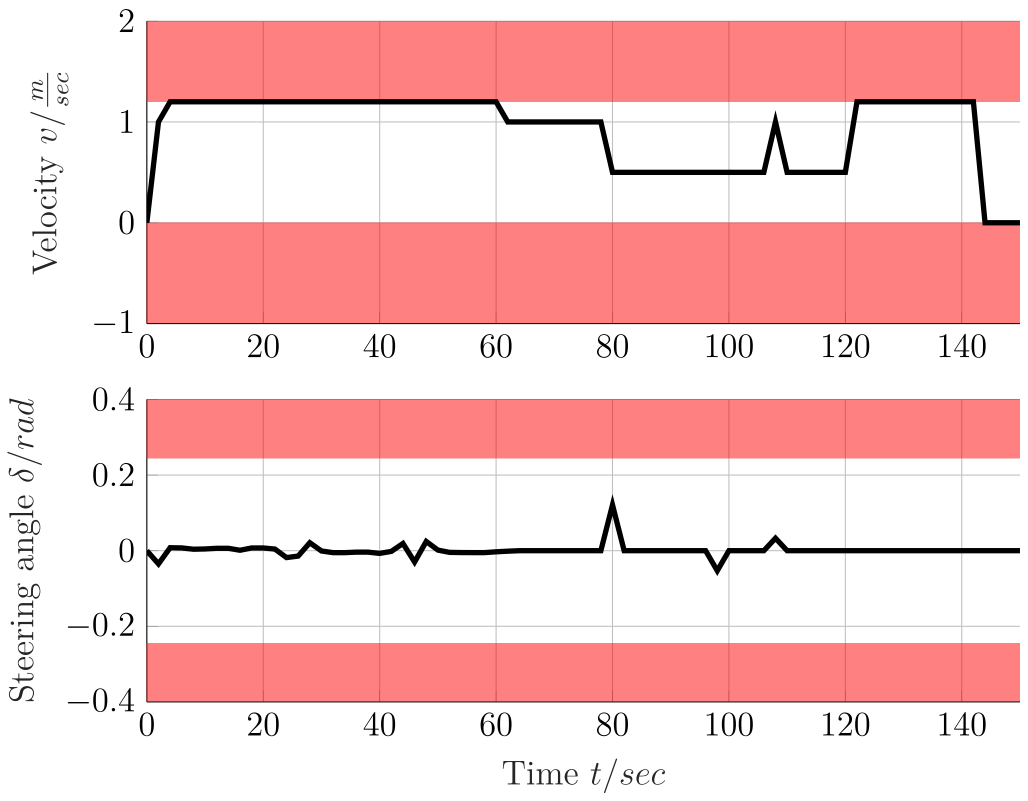


Figure 4.13: The controller exploits the maximum vehicle capabilities to follow the safe path. Once the onboard sensor detects the unknown region, the vehicle speed is reduced to a safe value. Once an obstacle is detected, the vehicle's velocity is reduced to safely avoid the partially detected obstacle.

The low-level controller is a model predictive controller that ensures accurate trajectory tracking by incorporating detailed nonlinear vehicle dynamics. While the planner operates with simplified dynamics for computational efficiency, the controller refines the trajectory execution by accounting for the full complexity of the vehicle's motion.

The effectiveness of the proposed hierarchical framework was demonstrated in simulation examples, showcasing its ability to safely navigate unknown environments while efficiently avoiding obstacles.

Summarizing, this chapter presents a hierarchical planning and control scheme which is real-time capable, yet provides safety guarantees despite limited sensor information. The local high-level planner is formulated as a mixed integer programming problem, generating feasible trajectories while incorporating obstacle avoidance as hard constraints. The low-level controller operates solely within the robot's sensor field of view, ensuring computational efficiency by restricting optimization to immediately perceivable obstacles. This reduces complexity, enabling fast, reactive control without global re-planning. The Nonlinear MPC controller tracks the planned path precisely while handling nonlinear robot dynamics. The hierarchical approach balances global planning consistency with local real-time optimization, ensuring robust obstacle avoidance and efficient trajectory tracking in dynamic settings.

The presented methodologies are intended to advance safe, responsible, and human-centric robotics, ensuring that autonomous systems operate in a transparent and socially beneficial manner.

The results also suggest that actively exploring partially detected obstacles could further improve performance, motivating the development of a Dual MPC approach, which will be the focus of the next chapter.

5 Hierarchical Dual Rolling Horizon Control for Path Planning and Control

We live on an island surrounded by a sea of ignorance. As our island of knowledge grows, so does the shore of our ignorance.

John Archibald Wheeler

Chapter 4 introduced the concept of online path planning and control for autonomous service robots equipped with onboard sensors, often deployed in partially or entirely unknown environments such as hospitals, elder care facilities, or logistics hubs. The proposed methodology enables the robot to generate a safe and adaptive path based on the available environmental information within its current field of view. As the robot follows the planned trajectory, new information is acquired, allowing for continuous re-planning and improved navigation efficiency.

However, optimizing the movement of an assistive robot is not simply about reaching a destination–it also requires adapting to evolving conditions in a way that prioritizes safety, efficiency, and interaction with human users. For instance, a hospital service robot may need to navigate through a corridor where patients, medical staff, or mobile equipment introduce temporary obstacles. Rather than rigidly following a pre-planned path, the robot can adaptively refine its route by incorporating newly observed environmental details, improving both efficiency and safety.

At the same time, adaptive navigation introduces potential challenges. The robot may encounter unexpected obstructions, such as a wheelchair suddenly blocking its path. To address this, we propose an approach that integrates active perception with built-in safety mechanisms, ensuring that the robot remains predictable and trustworthy in dynamic settings.

To achieve both adaptability and safety, we employ a hierarchical motion planning and control strategy. The local path planner continuously provides the low-level controller with a safe exploration region, defining where the robot is allowed to probe its environment for additional information. This enables the robot to refine its understanding of obstacles while ensuring it remains in a secure, predefined workspace.

Moreover, the control system actively incentivizes exploration by introducing an additional cost function component that guides the robot toward unknown regions and partially observed obstacles. In the case of a hospital service robot, this could mean actively adjusting its trajectory to improve localization in a poorly mapped or

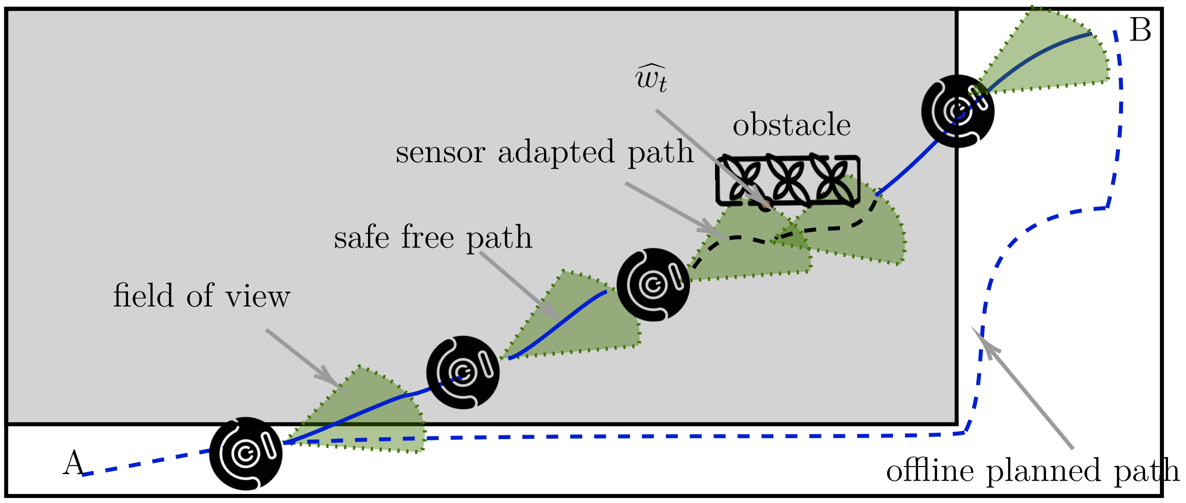


Figure 5.1: A robot navigating from point A to B based on uncertain and incomplete information. The available sensor data (green shading) is limited, requiring the robot to actively explore unknown areas by adjusting its orientation to improve environmental awareness.

illuminated corridor or identifying an alternative route when patient beds are moved unexpectedly.

This hierarchical structure, where probing and adaptation occur at the lowest level, allows for efficient real-time implementation without excessive computational overhead. The proposed framework is validated through realistic scenarios, demonstrating its effectiveness in safe, perception-aware navigation for service robots in dynamic, human-centered environments.

5.1 Control, Planning, and Perception-Aware Probing

Navigating an autonomous robot from a starting point to a goal while accounting for uncertainty and obstacle avoidance is a fundamental challenge, especially in partially known environments. Onboard sensors provide critical environmental information, but their limited field of view and measurement uncertainty require adaptive strategies for safe and efficient movement.

To address this, perception-aware planning balances immediate path efficiency with active exploration, ensuring that the robot refines its understanding of obstacles as it moves. As illustrated in Fig. 5.1, taking a short detour to improve environmental awareness may yield a more optimal long-term trajectory. This approach is particularly relevant for robots operating in dynamic, human-centered environments, where reactive path adaptation is essential for safety and efficiency.

Recent research has explored various approaches to perception-aware path planning, where environmental learning enhances trajectory optimization. For instance, [48] proposes a framework that integrates geometric and photometric information to improve localization. Similarly, [51] maximizes area coverage while accounting for obstacles, localization errors, and sensing uncertainties. Other works leverage entropy-based reward functions [92] or Fisher information matrices [132] to enhance active learning about obstacles.

In [166, 167], an exploration strategy is developed to minimize the covariance of the estimated robot pose and tracked landmarks, while [237] evaluates information gain by measuring newly covered free voxels detected by the robotâ€™s camera model. These approaches highlight the importance of balancing trajectory optimization with real-time environmental exploration.

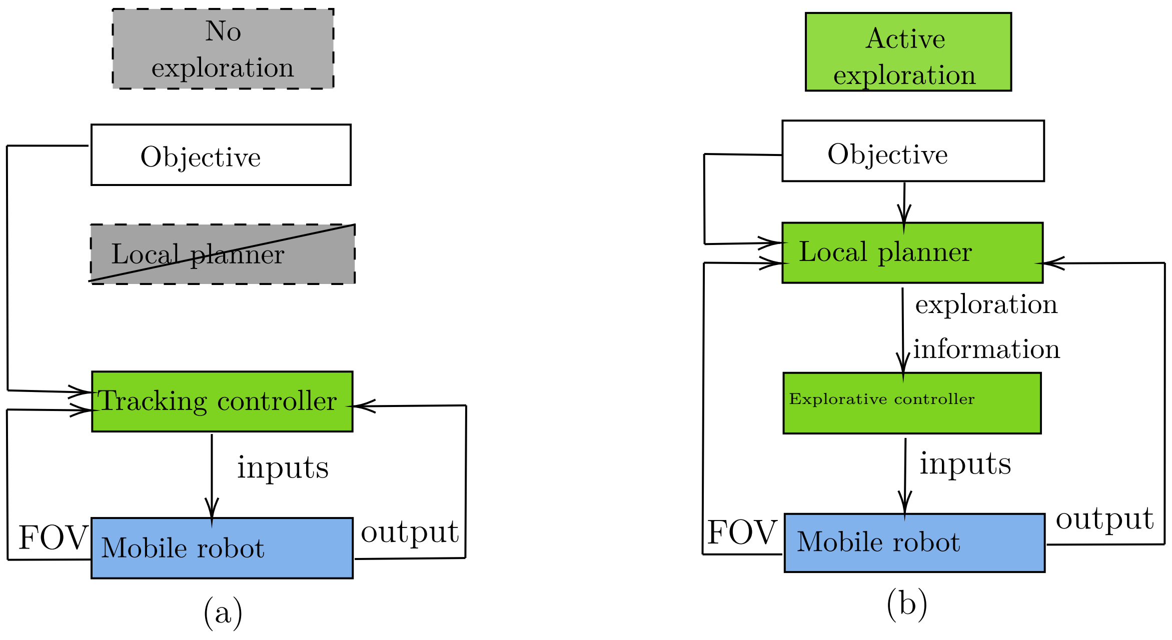


Figure 5.2: Hierarchical dual control scheme: Exploration is only activated upon obstacle detection. The high-level planner generates a safe trajectory and exploration set, while the low-level controller integrates uncertainty reduction.

5.1.1 Dual Control and Hierarchical Rolling Horizon Planning

Control strategies that integrate active exploration and uncertainty reduction are closely related to dual control theory [65, 153]. Dual control ensures that control actions not only regulate the system states but also influence environmental knowledge. This is particularly relevant for autonomous robots in complex, dynamic environments, where uncertainty-aware planning is necessary.

This chapter introduces a Hierarchical Dual Rolling Horizon Control approach to address this challenge. The high-level planner generates a safe reference trajectory based on currently available information and constructs a convex exploration set (Figure 5.2). The low-level controller, in turn, optimizes the control actions by balancing task execution with environmental learning, using a covariance-based dual cost function.

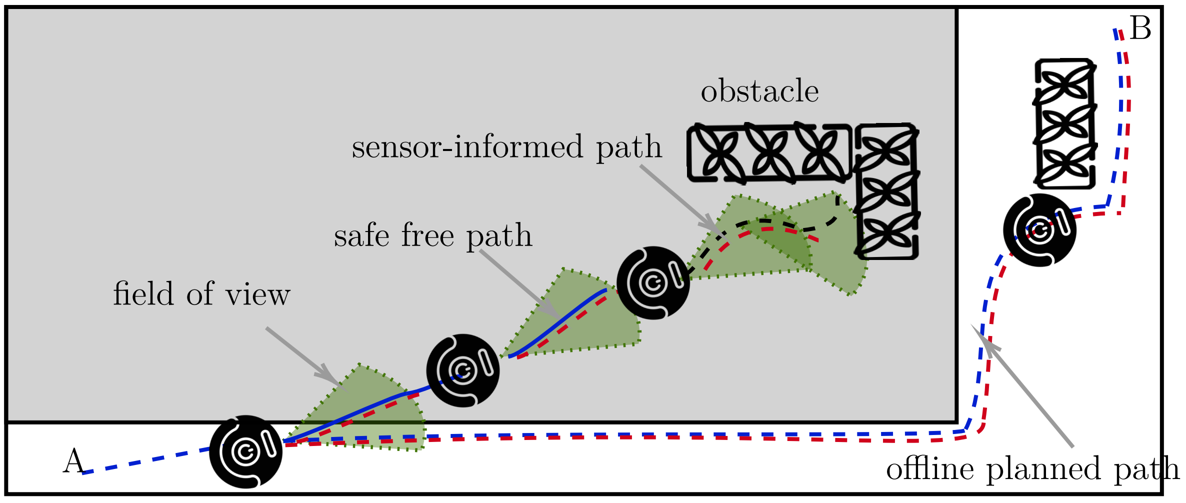


Figure 5.3: Fallback control mechanism: A robot follows a preplanned path (dashed blue). If unexpected obstacles appear, the online path planner activates (dashed black). If navigation is blocked, the fallback controller guides the robot back to a safe trajectory (dashed red).

Furthermore, we formulate a dual optimal control problem specifically designed for autonomous robots with onboard sensors. This framework integrates the regulation of system states relative to reference values with an additional exploration objective aimed at actively improving environmental awareness. However, solving such a dual control problem is inherently nonconvex and computationally intensive, particularly in dynamic, partially known environments.

To address this, we propose a Hierarchical Dual Rolling Horizon Controller [203], which is selectively activated upon obstacle detection (Figure 5.2). This hierarchical approach enables real-time decision-making while ensuring safe and efficient exploration. The high-level planner generates a safe trajectory within the robotâ€™s current field of view, while also defining a time-varying exploration set. This set represents regions where additional environmental information can be gathered to improve navigation efficiency. The planner transmits this data, along with obstacle information, to the low-level controller.

The low-level controller incorporates this environmental data into the control process by modeling it as a virtual system dynamics subject to process noise, which accounts for the uncertain shape and location of obstacles beyond the current sensing range. Since this uncertainty is influenced by the robot|s motion, the system dynamics exhibit state-dependent uncertainty. By augmenting the control objective with the trace of the covariance matrix, the controller balances safe navigation with active exploration, allowing the robot to efficiently probe its surroundings while maintaining safety constraints.

While probing enhances navigation by acquiring more environmental information, it can also introduce additional costs, as the robot may deviate from the optimal planned path. Furthermore, excessive probing may lead to situations where the system becomes blocked by unforeseen obstacles due to the limited sensor range. To address these challenges, we propose an active exploration scheme with integrated safety mechanisms [202]. This scheme continuously monitors both the overall cost and the information gained at each time step. If the detected information is insufficient to justify further probing, or if new obstacles prevent forward progress, the robot safely reverts to a previously explored, collision-free path leading to the final destination. This approach ensures task completion, prevents deadlocks, and maintains safe and reliable operation, as illustrated in Figure 5.3.

5.1.2 Dual Model Predictive Control

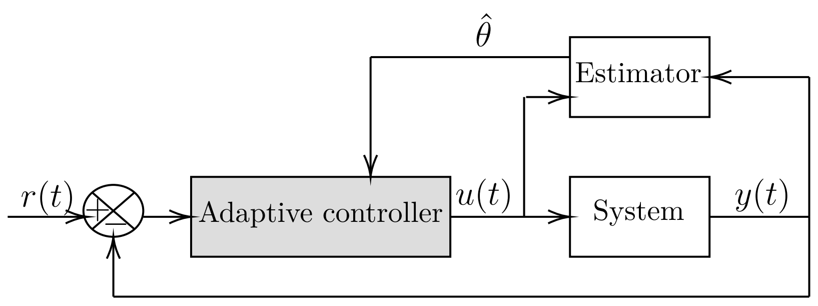
To effectively implement the Hierarchical Dual Rolling Horizon Controller, we must account for both task execution and uncertainty reduction in the planning and control process. This uses the concept of dual model predictive control, which we briefly introduce, as it explicitly balances safe navigation with active learning about the environment.

Dual control is related to adaptive control, which also aims to improve control performance over time. Traditional adaptive control strategies typically assume a certainty equivalence principle, where estimated parameters are used as if they were the true values. While this approach simplifies control design, it often fails in rapidly changing or uncertain environments, particularly in scenarios where environmental information is incomplete or unreliable. In such cases, the control system may overshoot or underperform, as it does not explicitly account for the accuracy of its own estimates [213].

In contrast, dual control addresses this limitation by simultaneously considering both the estimated parameters and their associated uncertainty. Originally introduced by Feldbaum [65], the key insight behind dual control is that control actions must serve a dual purpose: Directing Effect – Steering the robot toward its goal while maintaining safety and efficiency. Probing Effect – Actively gathering information to improve state estimation and environmental awareness.

This dual nature is particularly relevant in autonomous robot navigation, where perception-aware planning plays a critical role. For instance, when navigating through a hospital or warehouse, an assistive robot must balance following its optimal path with the need to adapt to new obstacles and dynamically update its map. As illustrated in Figure 5.4, while certainty-equivalent controllers treat parameter estimates as fixed, Dual MPC explicitly accounts for estimation uncertainty and adjusts control actions accordingly.

By integrating dual MPC into the hierarchical planning framework, we ensure that the robot not only reacts to environmental changes but also proactively gathers information to enhance its navigation capabilities. The following sections formalize the



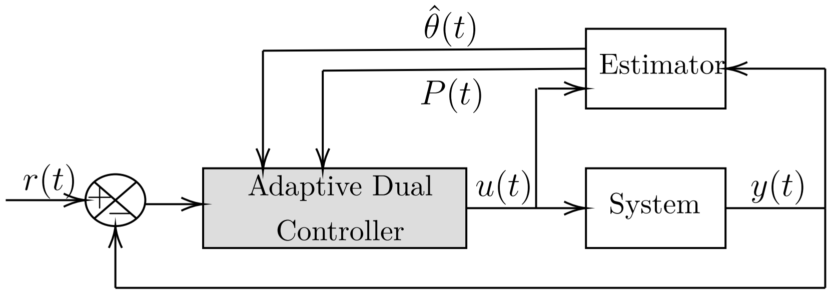


Figure 5.4: The adaptive control scheme takes into account the estimated parameters $\hat{\theta}$ as if it represents the real value, while the dual adaptive controller considers not only the estimation but also the accuracy of estimation P in addition to the reference r and the system output y.

dual MPC problem and discuss its computational challenges, as well as practical approximations for real-time implementation.

In standard control frameworks, the controller regulates system states but does not influence uncertainty reduction. To illustrate this limitation, consider the following example, where control inputs do not affect estimation accuracy. This example highlights why a conventional control approach fails to balance state regulation with active uncertainty learning.

Example 9. *(System without dual effect)*
Let us consider the following system:

$$x(t) = ax(t-1) + bu(t-1) + d. \tag{5.1}$$

with states and controls x and u respectively, b and a are known constants. The conditional distribution of d, given the output measurements up to and including time t, is Gaussian with mean $\hat{d}(t)$ and variance $P(t)$ satisfying the following equations:

$$\hat{d} = \hat{d}(t-1) + K(t)(x(t) - ax(t-1) - bu(t-1) - \hat{d}(t-1)), \tag{5.2a}$$

$$K(t) = P(t-1)(1 + P(t-1))^{-1}, \tag{5.2b}$$

$$P(t) = (1 - K(t))P(t-1). \tag{5.2c}$$

with $\hat{d}_0 = \hat{d}(t_0)$ and $P(t_0) = P_0$. Furthermore, the covariance update can be formulated *using (5.2b) and (5.2c) as follows:*

$$P(t) = P(t-1) - P^2(t-1)/(1 - P(t-1)). \tag{5.3}$$

It is clear that the second central moment (5.3) is not affected by the control input $u(t)$. Therefore, the control input has no dual effect as it does not influence the $\hat{d}$ estimate.

To overcome these limitations, Dual MPC extends standard control methods by incorporating an active learning mechanism, ensuring that control actions not only regulate the system but also gather additional information about the environment. This dual effect enables the system to make more informed decisions in uncertain conditions, improving long-term performance. In particular, Dual Control provides a framework where control inputs influence both system dynamics and the accuracy of state estimation.

In conventional MPC formulations, control inputs are designed to regulate system dynamics but do not actively influence the learning process of system uncertainty. As a result, these formulations may struggle to address uncertainties that evolve dynamically, leading to degraded control performance in unpredictable environments. Dual MPC, in contrast, introduces the concept of the dual effect, where control inputs serve both as a means of regulation and as a mechanism for reducing uncertainty. This necessitates a balance between probing actions (which gather new information) and directing actions (which achieve control objectives). The challenge lies in integrating uncertainty-aware control within an optimization framework while maintaining computational efficiency.

Definition 18. *(Dual effect)*
A control input has a dual effect if it can, with nonzero probability, affect at least one r^{th} central moment of the state, with $r \geq 2$. Conversely, if the future uncertainty is unaffected by the control with probability one, i.e., there are no central moments of order $r \geq 2$ of any state that is affected by the input signal, the control has no dual effect [16].

Solving the original Feldbaum dual control problem poses a significant computational challenge. It involves quantizing hyperstates into a grid and iterating over all values, leading to exponential growth in computation-a phenomenon known as the curse of dimensionality [93, 153]. To tackle this complexity, various approximations in the context of MPC have been proposed. These approximations can be classified into two main categories: 1. Implicit Dual MPC: This approach aims to approximate the stochastic dynamic programming problem associated with dual control. 2. Explicit Dual MPC: This method reformulates the original optimal control problem into a more tractable form. The objective function is also augmented with a heuristic-based probing effect to actively learn system uncertainty.

Implicit Dual MPC: The formulation of implicit dual MPC (IDC) relies on approximating the original Feldbaum dual control problem. As proposed by [57], The early works on IDC aimed to approximate the infinite-dimensional hyperstates resulting from the probability distribution of states and/or outputs by their mean and variance. Another approach to mitigate the computational burden is based on approximate dynamic programming (ADP). ADP primarily relies on forward simulation rather than a backward approximation of the value function. However, while it reduces computational demands, it tends to yield suboptimal solutions [93, 153].

In [83], an implicit Dual MPC framework was proposed where the influence of constraints on the objective indirectly incentivizes uncertainty reduction. Implicit dual MPC within the framework of multi-stage and Ensemble Kalman filter is proposed in [90], where uncertainty is expressed as a scenario tree branching up to $n^r \leq N_\mathrm{p}$ possible realizations of the uncertain system. However, computational complexity increases with the number of branches. However, this formulation showed promising results for simple cases that may not be suitable for online controllers.

In [209], an implicit dual MPC based on scenario trees is introduced. In [140], the authors leverage the concept of Multi-stage MPC to address system parameter uncertainty, expressed by a tree representing possible realizations of the uncertain dynamical system. They enforce the probing action by solving a bi-level optimization problem. The lower-level optimization utilizes the Karush-Kuhn-Tucker conditions, assuming the predicted measurement is the future measurement, to obtain the exact confidence region for uncertain parameters. The tree is then updated accordingly along the robust horizon.

The implicit formulation presented in [210] is based on a multi-stage formulation and Fisher Information Matrix (FIS). Here, the controller is informed of the estimated confidence region computed via FIS at each stage k, assuming the least square estimate of uncertain parameters remains constant along the prediction horizon. This approach is extended in [211], which provides an approximation for future parameter estimation instead of assuming constancy along the prediction. The estimation problem is divided into two parts: estimation based on available information up to time m and estimation based on predicted information. It's important to note that problem complexity grows exponentially with an increase in the number of uncertain parameters and/or longer prediction horizons [17].

Despite the computational complexities hindering their applications, IDC methods can still find applicability in the fields of economics and mobile robot trajectory tracking controllers based on neural networks. For a comprehensive review of implicit dual control methods, readers are referred to [153, 213].

Explicit Dual MPC: In explicit dual MPC (EDC), the concept of active uncertainty learning is realized by incorporating a heuristic function instead of directly solving the Feldbaum dual problem. This can be accomplished by introducing system excitation into the control input or reformulating the Optimal Control Problem (OCP) to include a measure of model uncertainty regarding the predicted control inputs. However, probing the system with a control signal may result in a loss of control performance due to unnecessary excitation. Nevertheless, overall performance is generally expected to be better compared to cases without active learning [153].

The work in [154] achieved the probing effect for uncertain system parameters by augmenting the objective function with a reward term expressed by the variance of the uncertain system parameter. Similarly, in [11], the reward term was treated as a constraint by constraining the estimated covariance error to remain within certain bounds. Further enhancement of the probing effect or uncertainty reduction can be attained by incorporating a "persistence excitation" condition. This condition, defined in terms of the information obtainable about uncertain system parameters through system excitation over the control horizon, is integrated into the OCP as a non-convex constraint. The resulting optimization problem is typically solved using successive semidefinite programming.

In [178], a persistence excitation constraint was considered to reduce uncertainty for a system parameter. The excitation constraint is formulated based on the increase of the system parameter information matrix. For the Auto-Regressive external (ARX) model structure, the persistence excitation condition is expressed by maximizing the minimal eigenvalue of the information matrix [214].

While Model Predictive Control with persistence excitation offers the advantage of exponentially converging parameter estimates when utilizing a recursive least-squares algorithm, it may lead to a degradation in output regulation, impacting the primary control objective [95]. Additionally, the excitation level relies on a tuning parameter, and the controller lacks the ability to determine whether excitation is necessary [153].

An alternative approach is presented in [145], where the condition of persistence excitation is imposed on the first control input. In this method, the open-loop optimization problem is augmented by nonlinear excitation constraints considering the previous and current control input vectors. This ensures that the first control candidate satisfies the excitation constraints by looking back over a specified horizon, obviating the need for explicit enforcement of periodic system excitation.

Another methodology for excitation is introduced in [86], where the system is steered to an invariant set $\mathcal{X}^e \subset \mathcal{X}$ where excitation can be conducted safely, ensuring system stability and recursive feasibility when system states are outside the excitation set.

In addition to the methods discussed earlier for enforcing probing actions of control inputs, an explicit dual control problem can be viewed as an instance of "optimal experimental design" (OED). In OED, parameters are estimated without bias and with minimum variance, enabling precise parameter estimations with a minimum number of experimental runs. Optimal experiments can significantly reduce experimentation costs. Integrating OED with control model-based methodologies results in control inputs that regulate system outputs or states and initiate probing actions on the uncertain dynamics of the system [153]. The fundamental concept of MPC combined with OED is to explicitly augment the objective function with a measure of estimation quality. Unlike persistence excitation, control inputs in this approach only excite the system if the quality of parameter estimation, quantified by covariance, is large.

In [94], authors extended certainty equivalence (CE) based controllers by considering that future measurements will resolve uncertainty. This is achieved by augmenting the objective function with a reward term, such as minimizing the trace of the covariance matrix or maximizing the trace of the information matrix, to reduce parametric uncertainty for single-input, single-output (SISO) systems. As discussed in [122], the dual effect is introduced by expressing the covariance of uncertain parameters as a function of control inputs for multi-input, multi-output systems. Thus, the modified objective rewards control inputs that minimize future covariance.

The work in [126] introduced explicit dual MPC by integrating the certainty equivalence formulation with optimal experimental design. This formulation aims to minimize the weighted sum between the control cost function and its variance, expressed by the Fisher Information Matrix. Another approach for OED combined with MPC is presented in [127], where probing action is achieved by considering a G-Criterion. Here, the covariance matrix of the estimate is weighted by the derivative of the nominal objective value with respect to the unknown parameters and the initial states, reflecting the sensitivity of the optimal objective function to uncertainty in the initial states.

The field of optimal experimental design motivated the development of "control-oriented OED" for identifying uncertain systems based on a predefined control performance criterion. In this approach, the standard MPC formulation is augmented with a control-oriented constraint expressed by an "application cost," quantifying the control performance degradation between the uncertain system and a predefined one. The control input excites the system until the information content about uncertain system parameters, expressed by the Fisher Information Matrix, does not result in unacceptable control performance degradation. The work in [128] presented the idea of an application-oriented dual control problem, where excitation is only performed until a predefined control performance level is achieved. The control input maximizes the information matrix to meet an application-oriented constraint representing the accepted degradation between the well-known and identified models. It's important to note that the choice of this constraint heavily depends on the application.

The concept of the Fisher Information Matrix and loss of optimality is discussed in [208]. Here, the solution obtained from known system parameters differs from the one with estimated parameters, resulting in an optimal gap expressed as an economic loss constraint. System excitation is conducted until the accepted economic loss is satisfied. A key challenge in control-oriented dual MPC is the requirement for true system parameters, which are uncertain [153]. To address the probabilistic nature of uncertain system parameters, the work in [19] introduced an MPC formulation with a control-oriented experimental design. Here, the control-oriented constraint is satisfied in a probabilistic sense, imposing chance constraints and yielding a stochastic MPC formulation.

The mentioned works show that the central consideration in explicit dual control applications lies in the trade-off between probing action, i.e., system input excitation to gain more information about uncertain system parameters, and control action, which regulates the system states and/or output. Although explicit dual MPC requires fewer computational resources and is easier to implement than implicit dual MPC, it cannot guarantee enhancement in control performance due to the simplification of the formulation, which relies on performance considerations. Furthermore, tuning the Explicit dual controller presents challenges due to the inherent conflict between learning and control objectives.

5.2 Efficient Hierarchical Dual Control for Sensor-Aware Path Planning

A major challenge in sensor-based path planning and control, as outlined in Chapter 4, arises when obstacles are detected within the sensor's limited field of view. To ensure efficient and safe navigation, we propose a hierarchical framework that separates planning and control into two layers. The local planner formulates a safe path within the visible environment using a simplified robot model, while obstacle avoidance is handled through a Mixed Integer Programming approach. The computed trajectory is then passed to the low-level controller, which employs a Nonlinear Model Predictive Control formulation to track the planned path. This structured approach reduces computational complexity while ensuring real-time feasibility.

A key limitation of this passive exploration strategy is that the robot only reacts to newly detected obstacles without actively seeking additional information. As a result, the planned path may not be optimal in terms of efficiency metrics such as travel time or energy consumption. Additionally, control inputs tend to be conservative, prioritizing safety over exploration due to the limited sensing horizon. This highlights the need for perception-aware path planning, where the robot actively acquires environmental information to enhance decision-making.

To address this, we introduce an exploration incentive into the control objective. The controller is rewarded for investigating detected obstacles, thereby improving environmental knowledge and potentially discovering more efficient paths (see Figure 5.1). However, solving this dual control problem in real time is computationally demanding due to its nonconvex and nonlinear nature. To mitigate this, we propose now the idea of Hierarchical Dual Rolling Horizon Controller (HDRHC), which efficiently balances exploration and task performance while reducing computational complexity [202].

This approach actively integrates exploration into the low-level control by incorporating an excitation condition in the objective function. The high-level planner computes a dynamically feasible and obstacle-aware path while defining a safe exploration region. The low-level controller, in turn, is incentivized to explore within this predefined region, ensuring that the robot remains safe while acquiring new information. This structure allows the system to refine and optimize its path adaptively while maintaining safety constraints.

The hierarchical approach effectively balances safe navigation and active exploration by leveraging hierarchical planning and control. Unlike full global re-planning, which is computationally prohibitive, this approach allows incremental updates based on newly acquired sensor data, making it computationally feasible for real-time applications. By integrating learning into the control process, this method enables efficient, perception-aware motion planning, allowing the robot to make more informed decisions even in uncertain environments. This hierarchical formulation ensures real-time feasibility while incorporating active exploration, striking a balance between efficiency, safety, and adaptability in uncertain environments. The following sections detail its implementation and demonstrate its effectiveness in autonomous navigation. We now outline the main components of the approach.

5.2.1 Local Planner

The local planner in the proposed scheme integrates multiple approximations to efficiently compute a safe path, which is then communicated to the low-level controller, similar to in Chapter 4. These approximations include simplifying the autonomous robot's dynamics, incorporating field-of-view constraints, and accounting for the detection or partial detection of obstacles. The planner streamlines complex robot dynamics into a more manageable linear model, ensuring computational efficiency. Additionally, the field-of-view constraints guarantee that the planned path remains within the robot's sensory capabilities, effectively addressing visibility limitations. Finally, detected or partially detected obstacles are incorporated into the planning process using binary variables and the Big-M method, ensuring both safety and task completion.

For the sake of completeness and as some of the derivations require insights into the formulations, we repeat some of the calculations of Chapter 4.

As in Chapter 4 we approximate the autonomous ground robot's dynamics in the planner by a linear model, expressed as:

$$x_{k+1} = Ax_k + Bu_k. \tag{5.4}$$

The onboard sensors like LiDAR or camera systems that are tasked with capturing the environmental data with the sensor of view can be approximated by a set of time-varying inequality constraints such that:

$$\mathcal{Z}_t := \{(x, y) \mid A_t[x\ y] \leq B_t\}. \tag{5.5}$$

for details see Section 4.2.3. It is worth noting that the entries of A_t and B_t matrices vary at each time instant according to the vehicle position (see Section 4.2.3). The high-level planner must plan a safe path that belongs to the field of view at the current time $\mathcal{Z}_t$.

Upon obstacle detection, the planner must consider the obstacle avoidance constraints to ensure the safety of the autonomous mobile robot. Consequently, the obstacles $\mathcal{O}_i$ that are detected within the current field of view $\mathcal{Z}_t$ can be represented by time-varying inequality constraints as follows:

$$\mathcal{O}_i := \{(x, y) \mid A_t^o[x\ y] \leq B_t^o\}. \tag{5.6}$$

$\forall i \in [1, 2, \cdots, N_o]$ where N_o is the number of the detected obstacles within the field of view. In (5.6), the row entry of matrices A_t^o and B_t^o represents a plane, and the intersection between these planes represents the approximated detected obstacle. Also, the direction of the inequality reflects the definition of the obstacle region $\mathcal{O}_i$. To ensure the safety of the planned path despite the detected/partially detected obstacle, obstacle avoidance is enforced via the Big-M method and binary variables$d$ as follows: $\forall k \in [t, t + N_\mathrm{p}]$

$$A_t^o p_k \geq B_t^o + M(1 - D). \tag{5.7}$$

Here, $p \in \mathbf{R}^2$ represents the coordinates of the autonomous ground vehicle. The vector D encompasses binary variables, denoted as follows:

$$D = [d_1\ d_2 \cdots d_n]^\top. \tag{5.8}$$

where n is the number of rows in A_t^o or B_t^o. For practical reasons, one should consider obstacle enlargement to ensure obstacle avoidance [73]. Thus, the detected obstacle within the field of view is enlarged by a safety margin δ_{safe}, cf. Figure 4.7. This safety margin is a function of vehicle dynamics, e.g., v_{max} and the sampling rate in the planner layer T_s such that [73]:

$$\delta_{safe} \geq \frac{v_{max} T_s}{2}\ sin(\alpha). \tag{5.9}$$

where α is the angle between the boundary and the path between two waypoints (see. Section 4.2.4. Constraints on the robot itself, such as maximum speed and acceleration can be integrated as in the planning process of Chapter 4. These constraints can be represented by a set of nonlinear inequality constraints as follows

$$\sqrt{v_x^2\ +\ v_y^2}\ \leq v_{max}, \tag{5.10a}$$

$$\sqrt{a_x^2 \ + \ a_y^2} \ \leq a_{max}. \tag{5.10b}$$

As these constraints are (5.10) nonlinear and computationally expensive we approximate them by M-sided polygons as follows:
$\forall m \in \{1, \cdots, M\}$

$$v_x \cos \frac{2\pi m}{M} \ + \ v_y \sin \frac{2\pi m}{M} \ \leq v_{max}, \tag{5.11a}$$

$$a_x \cos \frac{2\pi m}{M} \ + \ a_y \sin \frac{2\pi m}{M} \ \leq a_{max}. \tag{5.11b}$$

The smoothness of the planned path can be ensured by considering the rate of change of acceleration per each prediction step such that: $\forall k \in \{1, \cdots N_{\mathrm{p}}\}$

$$\sqrt{(a_{x_k} - a_{x_{k-1}})^2 \ + \ (a_{y_k} - a_{y_{k-1}})^2} \ \leq \ T_s \Delta a_{max}. \tag{5.12}$$

Here a_{x_k} and $a_{x_{k-1}}$ are the accelerations in x at the current and previous prediction steps while $(a_{y_k} - a_{y_{k-1}})$ is the difference between acceleration in y direction at two successive prediction steps. The constraint (5.12) can be approximated using M-sided polygon with $M = 4$ as follows:

$$a_{x_k} - a_{x_{k-1}} \ \leq \ T_s \Delta a_{max}, \quad -a_{x_k} + a_{x_{k-1}} \ \leq \ T_s \Delta a_{max}, \tag{5.13a}$$

$$a_{y_k} - a_{y_{k-1}} \ \leq \ T_s \Delta a_{max}, \quad -a_{y_k} + a_{y_{k-1}} \ \leq \ T_s \Delta a_{max}. \tag{5.13b}$$

Summarizing, the the optimization problem in the local planing layer can be formulated as an mixed-integer optimization problem where we use the superscript p to refer to the planner layer :

$$\min_{\{u^{\mathrm{p}}\},\, d} \quad \sum_{k=0}^{N_{\mathrm{p}}} \left(\|p_k^{\mathrm{p}} - p_{\mathrm{B}}^{\mathrm{p}}\|_Q^2 + \|u_k^{\mathrm{p}}\|_R^2 \right) \tag{5.14a}$$

$$\text{s.t.} \quad x_{k+1}^p = Ax_k^{\mathrm{p}} + Bu_k^{\mathrm{p}}, \; x_t^{\mathrm{p}} = \hat{x}(t), \tag{5.14b}$$

$$A_{i,t}^o p_k^{\mathrm{p}} \geq B_{i,t}^o + M_{\mathrm{big}}(\mathbf{1} - d_{i,k}), \tag{5.14c}$$

$$A_t p_k^{\mathrm{p}} \leq B_t, \tag{5.14d}$$

$$a_{x_k} - a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{5.14e}$$

$$-a_{x_k} + a_{x_{k-1}} \leq T_s \Delta a_{max}, \tag{5.14f}$$

$$a_{y_k} - a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{5.14g}$$

$$-a_{y_k} + a_{y_{k-1}} \leq T_s \Delta a_{max}, \tag{5.14h}$$

$$v_x \cos \frac{2\pi m}{M} + v_y \sin \frac{2\pi m}{M} \leq v_{max}, \tag{5.14i}$$

$$a_x \cos \frac{2\pi m}{M} + a_y \sin \frac{2\pi m}{M} \leq a_{max}, \tag{5.14j}$$

$$\sum_{i=1}^{n} d_{i,k} = 1, \tag{5.14k}$$

$$\forall k \in [0, ..., N_{\mathrm{p}}], \tag{5.14l}$$

$$\forall i \in [1, ..., n], \tag{5.14m}$$

$$\forall m \in [1, ..., M]. \tag{5.14n}$$

The high-level planner sends the safe trajectory x_t^{*p}, obtained from solving the optimization problem (5.14), along with obstacle information $\hat{w}_t$ (as outlined in the following) and the time-varying convex exploration set $\mathcal{L}_t$ to the controller.

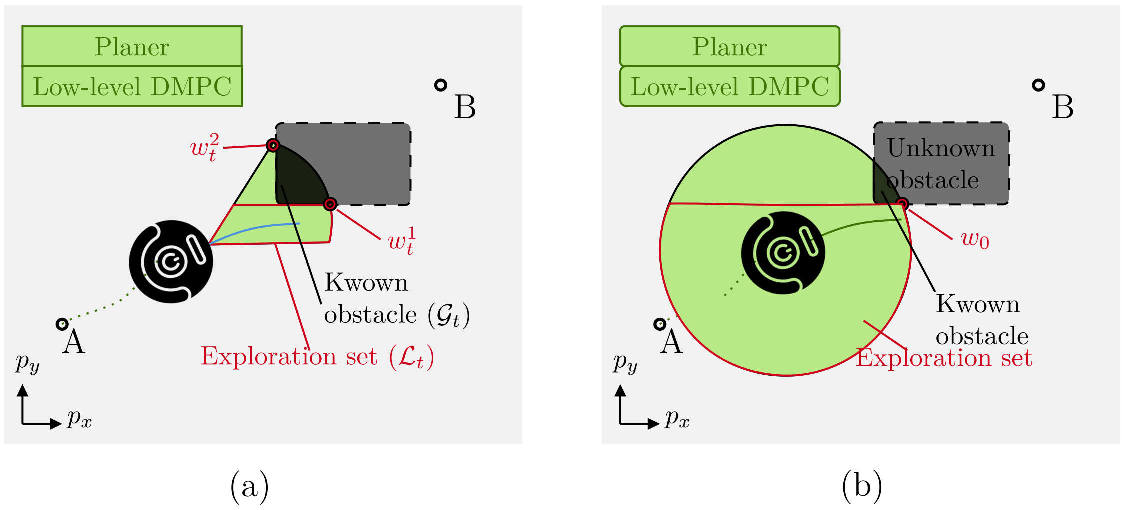


Figure 5.5: The planner sends the safe convex set $\mathcal{L}_t$, the safe trajectory $\mathbf{x}_t^*$, as well as the intersection point closest to the goal point w_t^1 or w_t^2 to the low-level Dual MPC (DMPC).

5.2.2 Obstacle and Field of View Information Sent to the Controller

To enable safe probing and enhance obstacle information, the planner transmits obstacle and field-of-view data to the controller, allowing it to explore the most relevant obstacles while ensuring safety (see Figure 5.5). The provided obstacle information $\hat{w}_t$ includes the intersection points where the current field of view $\mathcal{Z}_t$ overlaps with a partially detected obstacle $\mathcal{O}_{i,t}$ (cf. Figure 5.5). To simplify the probing process, only the intersection points closest to the goal are communicated. The rationale is that extending knowledge about these obstacles heuristically maximizes information gain, ultimately improving the likelihood of reaching the goal efficiently.

The controller also receives from the planner the safe convex exploration set $\mathcal{L}_t$. This set is extracted from the field of view while ensuring that possible obstacles are excluded (see Figure 5.5). It defines the region within which the autonomous robot can safely explore obstacles. The set $\mathcal{L}_t$ is given by:

$$\mathcal{L}_t := \left\{ p \in \mathbb{R}^2 \;\middle|\; \begin{array}{l} E_{i,t}p \geq e_{i,t} + M_{\text{big}}(\mathbf{1} - d^*_{i,t}), \\ Z_t p \leq z_t, \\ \forall i \in [1, ..., N_t] \end{array} \right\}. \tag{5.15}$$

Since $d_{i,k}$ may change for any $k \in \{0, ..., N_\text{p}\}$, multiple feasible obstacle-free sets may exist, depending on the configuration of obstacles at different time steps.

5.2.3 Low-Level Dual Model Predictive Controller

The basic idea is to augment the optimization criteria in the local controller, which penalizes the deviation from the reference trajectory, by an exploration term. This exploration or rewarding term incentivizes the autonomous robot to 'detour' from the planned path to explore the environment and gather additional information, resulting in a dual optimal control formulation that explicitly balances control and exploration objectives.

Common rewarding functions in this context are derived from the parameter estimate covariance matrix or its inverse, known as the information matrix, with respect to the gain obtained from new information. These functions are often based on optimal experimental design criteria. For instance, D-optimal design aims to either minimize the determinant of the covariance matrix or maximize the determinant of the information matrix. E-optimal design focuses on minimizing the largest eigenvalue of the covariance matrix or maximizing the smallest eigenvalue of the information matrix. Similarly, A-optimal design seeks to minimize the trace of the covariance matrix or maximize the trace of the information matrix. By incorporating one of these design criteria into the control objective, the problem is transformed into a dual optimal control problem, where control inputs are rewarded for navigating the system in a way that reduces uncertainty and enhances information acquisition.

Our proposed model, drawing inspiration from [25], does not assume parametric model uncertainty. Instead, uncertainty arises from sensor limitations, particularly the partial detection of obstacles. To model this, we treat the obstacle's impact as a virtual system dynamic influenced by Gaussian process noise. This noise represents the unseen or undetected parts of the obstacle that may exist beyond the sensor's current field of view. This approach allows for flexible adaptation to previously unknown environmental features, enabling more informed decision-making during navigation.

To this end, we assume that the obstacle dynamics (which may be stationary or slowly moving) can be formulated as follows:

$$w_{k+1} = A^w w_k + B^w \nu_k, \quad \nu_k \sim \mathcal{N}(0,\, Q_k). \tag{5.16}$$

where $w \in \mathbb{R}^2$ represents the state of the virtual obstacle dynamics with initial condition $w_t = w_t^1$ or $w_t = w_t^2$, depending on which intersection point is closest to the goal (see Figure 5.5). The overall system states in the controller layer thus consist of the nonlinear robot dynamics and the uncertain virtual system dynamics.

The uncertainty function plays a pivotal role in the formulation of our proposed dual control problem. Here, the uncertainty associated with the virtual system dynamics representing the obstacle, Q_k, is modeled as a function of the robot's state. This formulation captures the principle that control inputs not only maneuver the robot but also improve its perception by covering more of the environment, thereby reducing uncertainty.

We adopt the A-optimal design criterion, incorporating an exploration term into the objective function-specifically, the trace of the covariance matrix. This ensures that control inputs serve a dual purpose: they both direct the robot along a safe trajectory and actively probe the uncertain environment, thus imparting a dual effect to the control strategy.

The newly acquired information from active exploration is then leveraged in the planning layer to generate a safe and optimized path toward the destination, as illustrated in Figure 5.1. This dynamic integration of exploration and regulation ensures that the autonomous robot not only follows its trajectory but also continuously adapts and responds to newly detected aspects of its surroundings, enhancing both safety and efficiency in navigation.

Due to the probabilistic nature of the virtual system dynamics, the propagation of w can be computed by its mean and variance as follows:

$$\mu_k = w_t, \tag{5.17a}$$

$$\sigma_{k+1} = g(\sigma_k, Q(x,u)). \tag{5.17b}$$

The function g in (5.17b) is a general estimator that reflects that the predicted control inputs affect the uncertainty propagation. Note that we assumed that the mean value of the obstacle dynamics is fixed at $\hat{w}_t$, while its uncertainty varies with the robot's dynamics.

State-dependent Modeling of the Uncertainty Impact: In the formulation of the Dual MPC problem, accurately modeling the impact of control inputs on the propagation of uncertainty is crucial. We use a state-dependent uncertainty model in our formulation, meaning that changes in the robot's states directly affect the uncertainty of the surrounding environment, denoted as $Q(\cdot)$. This approach acknowledges that different control actions can influence how much the environment is sensed and understood.

For instance, an angle-based uncertainty function might be utilized, where changes in the robot's heading angle ψ alter the area covered by the onboard sensors, thus affecting the detected region of interest. This modification in the heading angle leads to the acquisition of new information during the planning phase, which is crucial for updating the robot's trajectory based on the latest environmental data.

Alternatively, a distance-based uncertainty function may be more appropriate in scenarios involving sensors like LiDAR, where the orientation of the robot may not significantly affect the reduction in uncertainty. In these cases, the formulation reflects that the closer the robot is to an uncertain area, the more information it can gather, regardless of the robot's orientation. This approach is particularly effective in dense or complex environments where proximity directly correlates with the accuracy and volume of data acquired.

In the case of an angle-based uncertainty function, the uncertainty can be expressed as a function of the angle of the onboard sensors relative to an obstacle or area of interest. This might take the form of an equation or a set of rules that link the robot's heading angle to the detection efficacy of the sensors, thus capturing the dynamic interplay between robot orientation and sensor output. This method emphasizes that the presence of sensors and their directed use is vital in reducing uncertainty and enhancing the robustness of the path-planning process. Therefore, the angle-based uncertainty function can be formulated as follows:

$$Q = e_1(\psi, \theta). \tag{5.18}$$

Here $e_1(\cdot)$ depends on θ which is the angle of w_t at time t w.r.t the ground fixed frame and the heading angle ψ of the autonomous robot. In (5.18), the control signal has a direct effect on uncertainty propagation via robot heading angle state ψ. By injecting the first control input to the mobile robot, the heading angle will change to gain more information about the detected obstacle, which is indeed encouraged by the dual formulation, cf. Figure 5.6 left.

In the context of a distance-based uncertainty function, the uncertainty can be quantified by the proximity of the autonomous robot to a detected or partially detected

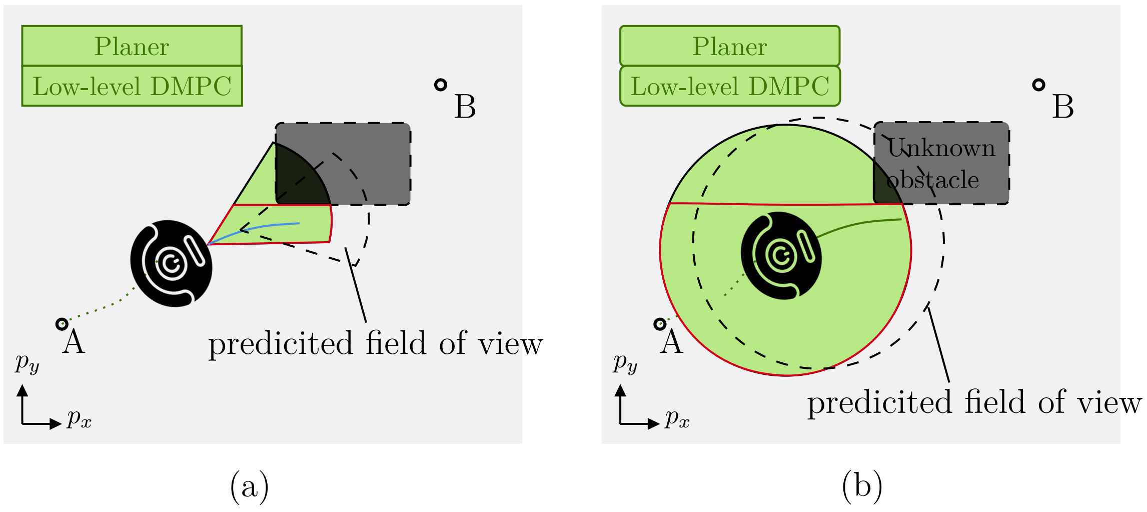


Figure 5.6: Altering the robot heading will change the region of interest detected by the sensor. In case of limited range approaching, on the right, the uncertain obstacle will indeed gain more information as the sensor will cover a new area.

obstacle and can be expressed as follows:

$$Q = e_2(x^{\mathrm{c}}, w_t). \tag{5.19}$$

Here, $x^{\mathrm{c}} \in \mathbb{R}^n$ denotes the center of mass of the autonomous robot with superscript c referring to the controller layer, and w_t represents the position or state of the detected obstacle. The function $e_2(\cdot)$ captures the relationship between the robot's proximity to the obstacle and the level of uncertainty. This relationship posits that as the robot approaches the obstacle, the uncertainty diminishes, given that the robot will cover new areas in its field of view in subsequent time steps, as depicted in Figure 5.6 right.

The underlying principle is that the closer the robot is to the uncertain environment w_t, the more detailed and expansive the sensor data becomes, thereby enhancing the accuracy of the robot's environmental model and reducing uncertainty. This dynamic interplay between the robot's position and the uncertainty of its environment emphasizes the dual effect of control inputs: they direct the robot along a desired trajectory and actively engage the robot's sensors in mapping and understanding the environment more comprehensively. This integration of navigation and sensing underscores the sophisticated nature of dual control strategies in autonomous robot systems.

Remark 3. *To ensure robot safety during exploration and reduce the computational burden in the controller layer, the exploration is conducted within a safe convex exploration set (5.15) sent by the planner layer*

Remark 4. *The proposed hierarchical scheme is active when obstacles are detected within the field of view, while in case of no obstacle, both the planner and exploration are deactivated, and only a non-linear convex optimization problem (4.1) is solved in the controller layer.*

Remark 5. *In the case of obstacle detection, the exploration is activated if it enhances the overall task performance, which could be the case when an obstacle obstructs the robot's path toward the goal. Otherwise, the controller follows the safe trajectory sent by the planner. This is due to the fact that unnecessarily exploring the environment will increase the control task objective by deviating from the reference trajectory.*

Choice of the Probing Cost Function: In the dual optimal control formulation for autonomous robots, the cost function plays a crucial role by balancing trajectory stabilization and information gain. In explicit dual control approaches based on Optimal Experiment Design, an additional exploration objective is incorporated into the cost function. Specifically, in A-optimal design, this objective is realized by minimizing the trace of the covariance matrix.

This formulation highlights that the control system's objective is not limited to stabilizing the system states or outputs relative to a reference trajectory (control objective). Instead, it also actively encourages control inputs that reduce uncertainty (exploration objective). The resulting dual objective function can be expressed as:

$$J = W_1 \times (\text{Control Objective Term}) + W_2 \times (\text{Trace of Covariance Matrix}) \tag{5.20}$$

The covariance matrix in our case represents how much we are certain about the edge location that is detected by the sensor and what would be the shape of the environment behind the field of view. Therefore, minimizing the trace of the covariance matrix is synonymous with maximizing information gain about the system's environment. This often results in a trade-off where increasing information gain might lead to deviations from the reference trajectory as the robot engages in actions that enhance its environmental understanding, depicted in Figure 5.1.

In this equation, W_1 and W_2 are weighting matrices in the objective function that represent the trade-off between achieving stabilization (information loss) and enhancing information gain (control performance degradation). These weights help balance the dual objectives; Stabilization Gain: Maintaining the robot's path or behavior aligned with the desired trajectory or behavior. Information Gain: Actively seeking and incorporating new environmental data to reduce uncertainty, which is quantified by the covariance matrix's trace. The selection of these weights is crucial, as it dictates the emphasis placed on either stabilization or exploration. Adjusting these parameters allows the control system to be tailored to specific operational priorities, such as prioritizing path accuracy over exploration in safety-critical applications or favoring extensive environmental mapping. The proposed cost function J can be written as follows:

$$\begin{aligned} J(x_k, u_k, x_s, \sigma_k) := \sum_{k=0}^{N_\mathrm{p}} & W_1 F_1(x_k^c, u_k^c, x_k^*) \\ & + W_2 F_2(\sigma_k) \\ & + W_3 F_3(x_{t+N_\mathrm{p}}^c, u_{t+N_\mathrm{p}}^c). \end{aligned} \tag{5.21}$$

Here F_1 is a quadratic cost that penalizes the states with respect to trajectory x_t^* given by planner, F_2 is an exploration function that can be expressed by the trace of the covariance matrix $(\mathrm{tr}(\sigma_{k+1}))$ and F_3 is a terminal penalty function. Additionally, the weighting matrices W_1 and W_2 represent the trade-off between the control task objective and the uncertainty learning objective, while W_3 is the terminal penalty weighting matrix. The dual objective function (5.21) is restricted by the non-linear system dynamics and robot physical constraints, and the predicted state trajectory is restricted to be in the exploration set $\mathcal{L} \subset \mathcal{Z}_t$ where $\mathcal{Z}_t$ is the current field of view.

Overall resulting Dual Model Predictive Control Formulation Combining all elements, we obtain the following, overall constrained dual model predictive control problem:

$$\min_{u^c} \quad J(x_k, u_k, x_s, \sigma_k) \tag{5.22a}$$

$$\text{s.t.} \quad x_{k+1}^c = f(x_k^c, u_k^c), x_0^c = \hat{x}(t), \tag{5.22b}$$

$$w_{k+1} \sim \mathcal{N}(\mu_k, \sigma_k), \tag{5.22c}$$

$$\sigma_{k+1} = g(\sigma_k, Q_k), \sigma_k = \sigma_t, \tag{5.22d}$$

$$Q_k = e_1(\psi_k, \theta_t), \tag{5.22e}$$

$$x_k^c = \hat{x}(t), \tag{5.22f}$$

$$\mu_k = \hat{w}_t, \tag{5.22g}$$

$$x_{k+1}^c \in \mathcal{L}_t, \ u_k^c \in \mathcal{U}, \tag{5.22h}$$

$$\forall k \in [0, ..., N_\mathrm{p}]. \tag{5.22i}$$

where $u^c = \{u_t^c, ..., u_{t+N_\mathrm{p}}^c\}$ is the sequence of control actions. Only the first piece of the optimal control sequence is applied to the system, and the optimization is repeated at each simulation step. Note that, the predicted control trajectory affects the uncertainty propagation through the constraints (5.22d) and (5.22e) where the function $g(\cdot)$ is a general estimator. Furthermore, note that the state-dependent uncertainty (5.22e) can also take the form of (5.19). The robot safety is guaranteed by restricting the predicted system trajectory to be within the safe exploration convex set $\mathcal{L}_t$ sent by the planner layer in (5.22h). Furthermore, the control signal u_k^c is constrained to be within a set $\mathcal{U}$ that can be chosen as a trade-off between the allowed level of aggressiveness and driving comfort[23]. The optimization problem is solved using system states information measured/estimated at t through the equality constraints in (5.22f).

The overall Dual Model Predictive Control (DMPC) scheme integrates trajectory tracking with active exploration under uncertainty. The control problem is solved iteratively in a receding horizon fashion, where only the first optimal control input is applied at each step before re-optimizing.

The DMPC formulation ensures that the robot follows a dynamically feasible trajectory while gathering additional environmental information. The robot's motion is governed by nonlinear system dynamics, with control inputs optimized over a finite horizon. At each step, the planner provides a safe trajectory, obstacle information, and a convex safe exploration region, which the controller uses to guide the robot's motion.

Uncertainty in the obstacle representation is modeled using a Gaussian process, where the mean represents the estimated obstacle position and the covariance characterizes uncertainty in the obstacle's shape and extent. The evolution of this uncertainty depends on the robot's trajectory, as its motion determines how much new information is gained. The control problem explicitly accounts for this effect by incorporating an exploration term in the objective function. This allows the robot to deviate from the nominal path when such deviations lead to better environmental awareness.

Safety constraints ensure that the explored trajectory remains within the exploration region defined by the planner while avoiding detected obstacles. The control inputs are further constrained to balance maneuverability and smoothness. The optimization problem is solved at each time step using real-time state estimates, ensuring adaptability to newly detected obstacles.

This hierarchical DMPC framework enables safe navigation while actively reducing uncertainty in the environment. By balancing control objectives with exploration incentives, it enhances decision-making in unknown environments while remaining computationally efficient.

In the following Section we provide illustrative simulation examples with respect to our approach.

5.3 Simulation Examples Planning and Exploration

In this section, we apply the hierarchical planning and dual MPC formulation for an autonomous ground robot equipped with onboard sensors, navigating in an environment that might be partially or completely unknown. These sensors provide crucial environmental data that the planner/controller layers utilize to formulate and execute the robot's path planning and control.

We present examples that demonstrate how the dual MPC formulation is implemented in various scenarios. These examples highlight the practical application of the theoretical concepts discussed and provide insight into the controller's decision-making process when faced with dynamic and uncertain environmental conditions. The examples are designed to showcase both the robustness of the dual MPC strategy in maintaining a safe and efficient path and its adaptiveness in incorporating new data to refine and optimize the robot's trajectory continuously.

The equation of motion of the adopted autonomous ground robot are the same as in Section 4.4.

Example 10. *(Distance-based uncertainty)*
The example illustrates the concept of active planning and control, where both the

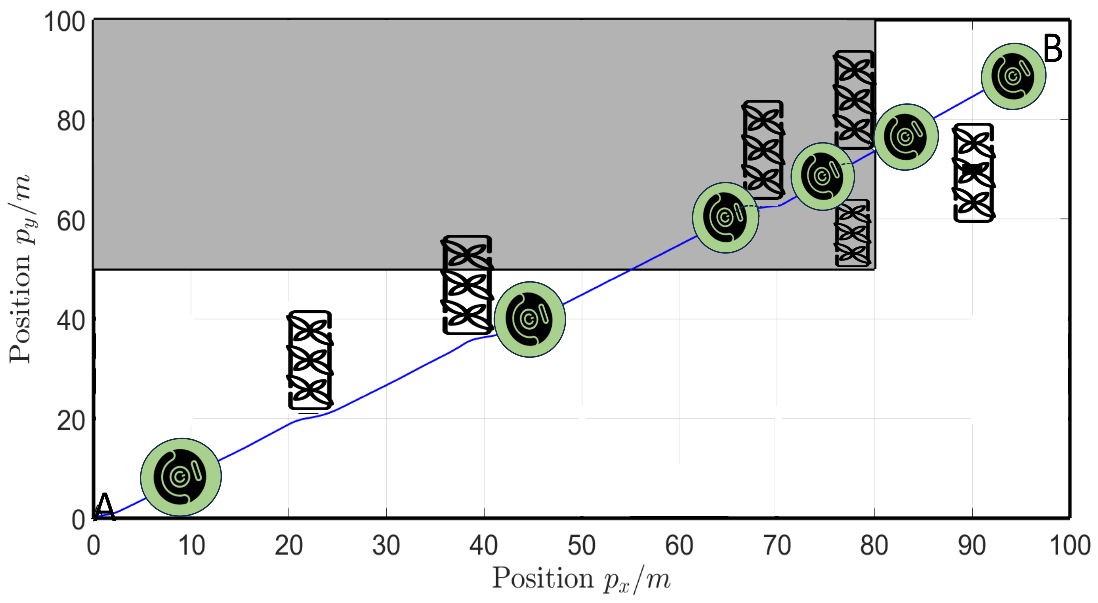


Figure 5.7: The autonomous robot follows the offline path till the detection of the unexplored region (in gray). Then the online path planner is activated. Once an obstacle is detected, the HDRHC is activated. The path of the autonomous robot equipped with exploration is depicted in dashed black. The LiDAR field of view is represented by a green circle.

high-level planner and the low-level controller contribute to environment exploration. The planner provides a safe trajectory, a convex exploration region, and initial conditions for the virtual system dynamics. The low-level controller then solves an explicit dual optimal control problem, balancing trajectory tracking with active exploration of detected obstacles.

The robot updates its environment model by steering toward detected obstacle edges to maximize information gain. This results in a trade-off between the primary control objective and exploratory deviations from the planned path. The resulting uncertainty evolution is formulated accordingly (cf. Figure 5.6 right), with the state propagation

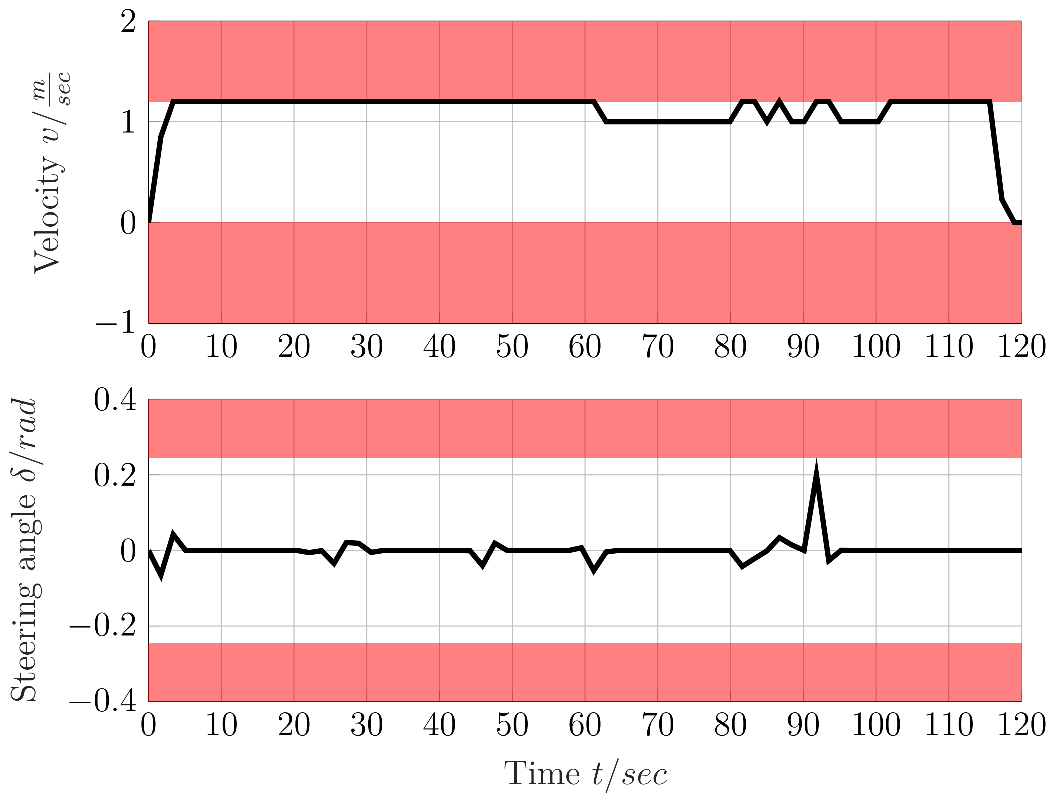


Figure 5.8: The autonomous robot accelerates with max velocity to follow the safe offline path. While exploration, the velocity is reduced to safely navigate the robot. In the case of exploration, the robot accelerates with maximum speed as it is safe considering the exploration is conducted in the exploration set $\mathcal{L}_t$.

of $\hat{w}_t$ given by:

$$\mu_k = \hat{w}_t \tag{5.23a}$$

$$\sigma_{k+1} = A^\top \sigma_k A + Q(x,u), \tag{5.23b}$$

where the state-dependent uncertainty is given by:

$$Q(x,u) = (p_k - w_t)^2. \tag{5.24}$$

Here $p \in \mathbb{R}^2$ represents the position of the center of the robot's center of mass, matrix $A = \boldsymbol{I}_2$ and $w_t \in \mathbb{R}^2$ is the initial virtual condition for the virtual system (cf. Figure 5.5). Notably, increasing the weight of the exploration objective leads to more explorative behavior at the expense of the control objective. Therefore, the trading-off between the control objective and exploration objective is a user-defined tuning parameter.

Using the MPC-based path-following formulation (3.13), the autonomous ground robot initially follows the offline-planned path (cf. Figure 5.7). When an unexplored region is detected, online path planning is activated, solving a convex nonlinear optimization problem in the planner layer (see Figure 5.2). The hierarchical scheme is ac-

tivated upon obstacle detection. Unlike the scenario in Example 7, where the controller must navigate cautiously to avoid obstacles, in the case of exploration, the controller accelerates the robot at maximum velocity since safety is ensured by constraining exploration within the safe convex set $\mathcal{L}_t$. This approach results in a shorter travel time to the final point B compared to the scenario without exploration, as illustrated in Figure 5.8.

Example 11. *(Angle-based uncertainty)*
In this example, we adopt an autonomous ground robot with a camera system featured

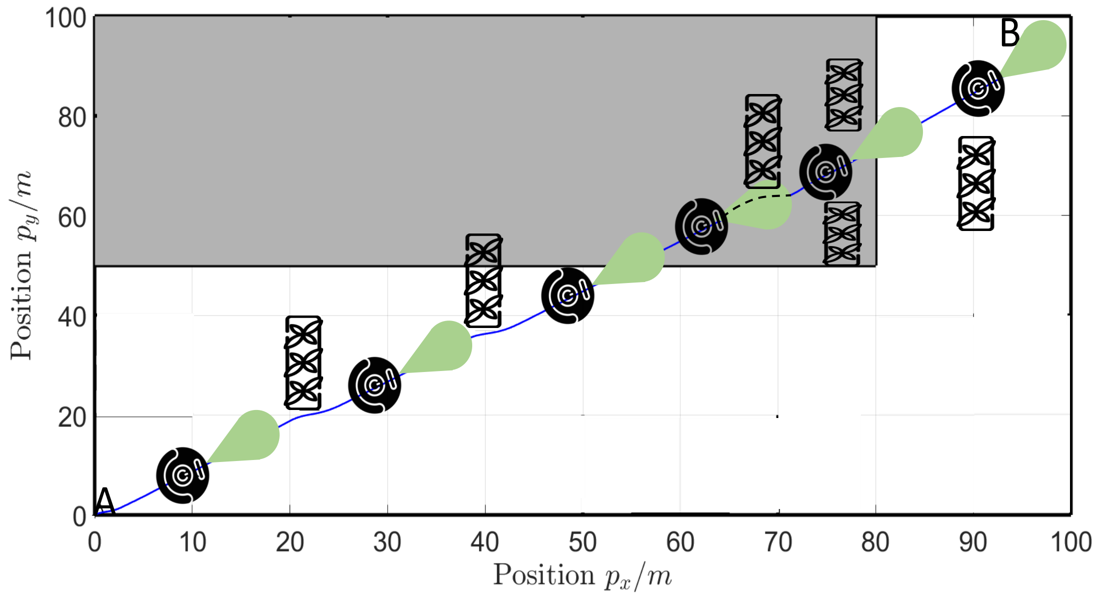


Figure 5.9: The autonomous robot follows the offline planned path till detecting the unexplored region (in gray). The online path planning and control is activated to plan a path within the current field of view (in green). The proposed hierarchical strategy is activated upon the obstacle detection.

by limited range $R = 3m$ and a limited field of view $\theta_c = 60°$. The uncertainty reduction about the environment/information gained about the detected obstacle can be obtained by altering the sensor's field of view region of interest. This can be achieved by making a "detour" by the autonomous ground robot (cf. Figure 5.6 left). Therefore, state-dependant uncertainty can be expressed in terms of the autonomous robot heading angle as follows:

$$Q(x, u) = (\psi - \theta_t)^2 \tag{5.25}$$

where ψ is the robot's heading angle while θ_t is the angle of the the initial condition w_t with respect to the fixed frame of reference with $A = \boldsymbol{I}_2$. The propagation of w_t

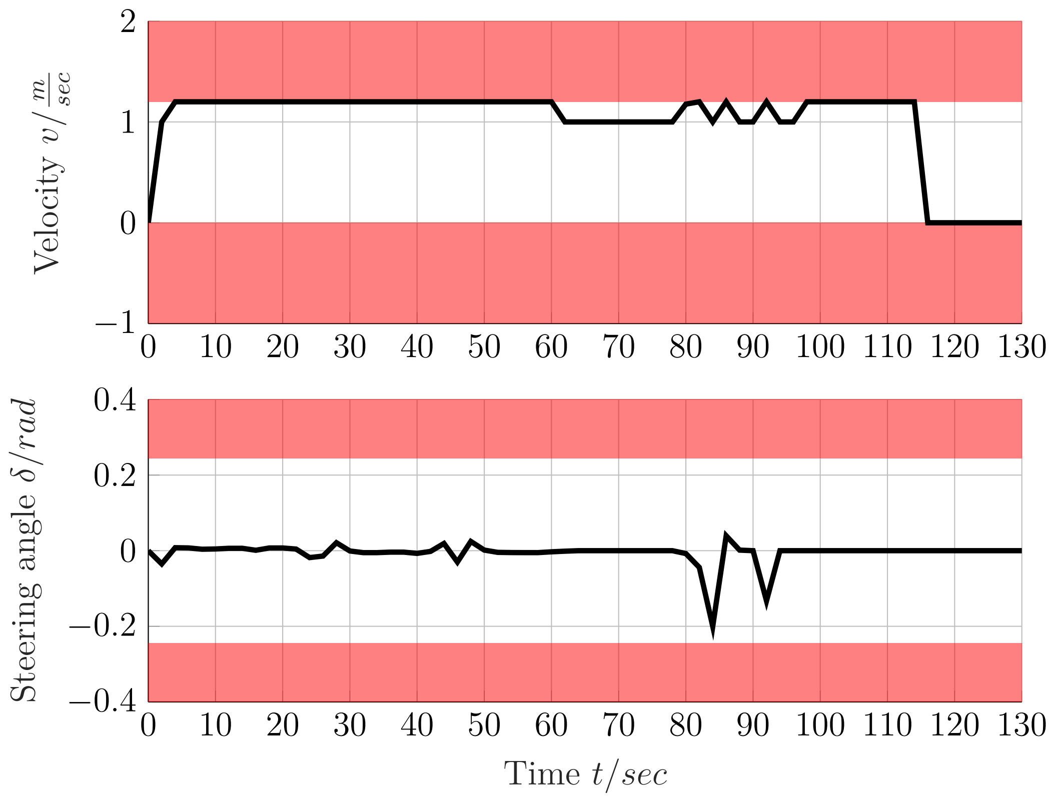


Figure 5.10: The autonomous robot accelerates with maximum velocity following the offline planned path. The velocity drops while exploring the unexplored region. Upon the activation of the proposed hierarchical strategy, the dual control input accelerates the autonomous robot to explore the environment with maximum velocity as the robot safety is guaranteed by exploring within the exploration set.

(5.17b) is expressed as follows:

$$\mu_k = \hat{w}_t \tag{5.26a}$$

$$\sigma_{k+1} = A^\top \sigma_k A + Q(x, u) \tag{5.26b}$$

The autonomous ground robot safely follows the offline planned path, adopting the MPC path following formulation (see. Example 2). Upon detecting the no-go region, the online path planning and control are activated to plan a safe path within the current field of view. The proposed strategy is activated when an obstacle is captured by the onboard camera system (cf. Figure 5.9). Due to the exploration term in the objective function, the control input changes the heading angle of the autonomous robot and, consequently, changes the region of interest of the onboard sensor (cf. Figure 5.10). Due to the dual feature of the control input, more information about the detected obstacle is available to the planner layer to plan a more efficient path. During the exploration, the autonomous robot safety is guaranteed by exploring the environment within the safe exploration set sent by the high-level planner $\mathcal{L}_t$.

Remark 6. *The autonomous robot follows the safe planned path, and the controller/planner will not react to the obstacles that might be detected outside the no-go region as these obstacles are considered during the planning phase before the autonomous robot motion execution.*

Remark 7. *The exploration set is a time-varying convex safe set; therefore, the controller can accelerate the robot with maximum velocity v_{max} as long as the robot is in $\mathcal{L}_t$.*

Remark 8. *The proposed HDRHC is activated if the detected obstacle hinders the robot's path toward the destination point; otherwise, the robot follows the safe trajectory planned by the planner.*

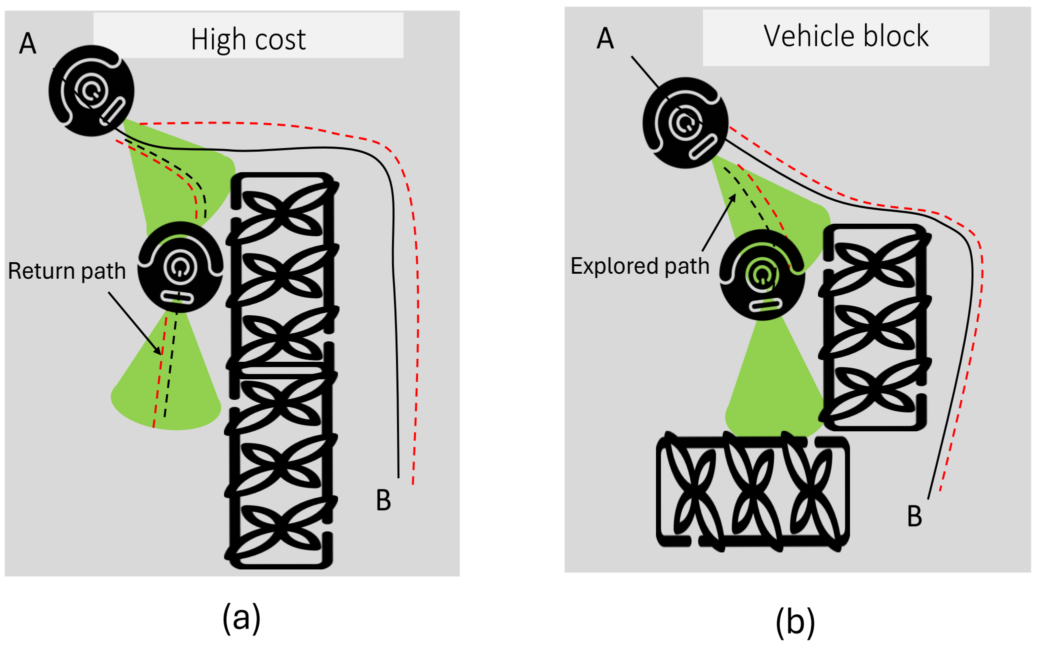


Figure 5.11: Exploration can lead to high overall cost in (a) or to a situation where the autonomous ground robot is blocked by other obstacles in (b). The fallback controller steers the autonomous robot to follow the safe path to the final destination point B. The preplanned path is in black while the fallback path is in red

5.4 Moving Horizon Planning and Control for Autonomous robots with Active Exploration and Safety Strategies

The objective function of the proposed hierarchical dual rolling horizon controller includes a rewarding term that encourages the autonomous robot to explore detected obstacles or environmental features. While this exploration improves environmental awareness, it may also increase control costs by deviating from the planned trajectory. Moreover, extended exploration can raise the risk of encountering additional obstacles, complicating navigation and jeopardizing both safety and task completion (cf. Figure 5.3). Balancing information acquisition with control objectives is therefore critical.

To address these challenges, a fallback controller is introduced to monitor the cost-benefit trade-off of exploration and ensure safe navigation. If exploration becomes excessively costly or leads to an impassable situation, the fallback mechanism redirects the robot to a preplanned safe path (cf. Figure 5.11). Prior work has explored safety mechanisms in autonomous systems, such as sensor failure handling [226], defensive driving strategies [82], fallback controllers for perception failures [199], and active planning approaches based on dual optimal control [203].

Building upon the hierarchical explorative path planning and control scheme introduced in Section 5.2, this section presents a moving horizon planning and control framework integrating active exploration and fallback strategies [202]. The fallback controller continuously assesses the cost of exploration and monitors potential obstructions. If exploration becomes too costly or the robot encounters obstacles that hinder progress, the system halts further environmental investigation. Additionally, the fallback mechanism leverages newly acquired information to guide the robot back to a safe trajectory, ensuring efficient navigation toward the final destination (cf. Figure 5.11). By combining active exploration with a safety strategy, the proposed approach enhances both adaptability and reliability, guaranteeing task completion while maintaining safe, obstacle-free navigation.

5.4.1 Illustrative Example

This example examines an autonomous ground robot equipped with a camera system with limited sensing capabilities. Initially, an offline path planner generates a safe trajectory based on known environmental information, including obstacle geometry and locations, as in Example 5. The robot follows this preplanned path until its onboard sensor detects an unexplored region. Upon detection, the planner/controller deviates from the preplanned path to explore the newly detected area using an online path planning and control strategy.

If no obstacles are detected within the robot's field of view, the controller focuses on generating a safe path within the current sensing range, as illustrated in Example 6. If obstacles are detected, the hierarchical dual control strategy is activated to encourage exploration and gather additional environmental information, as demonstrated in Examples 10 and 11. If the robot encounters additional obstacles that impede its progress, the fallback safety mechanism is triggered, guiding the robot along a safe path toward the destination (cf. Figure 5.3).

This approach ensures adaptability to new environmental information while maintaining steady progress toward the goal, effectively balancing exploration with safe navigation.

Example 12. *(Fallback-Exploration Moving Horizon Controller)*
In this example, we demonstrate the application of the proposed hierarchical dual receding horizon control framework, integrated with a fallback controller, for an autonomous ground robot equipped with a camera system. The robot must navigate from an initial point A to a final destination B. Initially, an offline path planner computes a safe path, which the robot follows using the MPC path-following formulation (3.13). Upon detecting an unexplored region, an online path planner is activated to generate a safe trajectory within the current field of view, solving the optimal control problem (4.1). When an obstacle is encountered, the hierarchical framework, augmented with a fallback controller, engages to explore it (cf. Figure 5.12).

Due to limited sensing capabilities, the robot may become obstructed by additional obstacles (cf. Figure 5.1), endangering both safety and task completion. In such cases, the fallback controller intervenes, guiding the robot along a safe trajectory using the MPC tracking (3.17) or path-following (3.13) formulation at maximum capability, ensuring it reaches its destination safely (cf. Figure 5.13). This approach enables robust navigation, effectively balancing exploration, safety, and task execution.

5.5 Summary

Building on the findings from Chapter 4, this work extends the task objectives for an autonomous robot equipped with onboard sensors by integrating active exploration into path planning and control. A Hierarchical Dual Rolling Horizon Controller is proposed to balance computational efficiency with the benefits of explicit dual control. Instead of assuming parametric model uncertainty, this framework models environmental uncertainty as a virtual system influenced by Gaussian process noise.

At the high level, the planner formulates a mixed-integer optimization problem using simplified robot dynamics, incorporating obstacles as hard constraints via the Big-M method. It generates a safe trajectory, initializes virtual environment dynamics, and defines a time-varying exploration set, which is communicated to the low-level controller.

The low-level controller extends the system state representation by integrating artificial system dynamics subject to process noise. These dynamics account for unseen obstacles beyond the sensor's field of view. Two uncertainty formulations are introduced: distance-based uncertainty, where uncertainty is reduced by approaching detected obstacles, and angle-based uncertainty, where sensing limitations necessitate heading angle adjustments. By incorporating the trace of the covariance matrix as a reward in the control objective, an explicit dual MPC formulation emerges, enabling the robot to balance trajectory tracking with information acquisition.

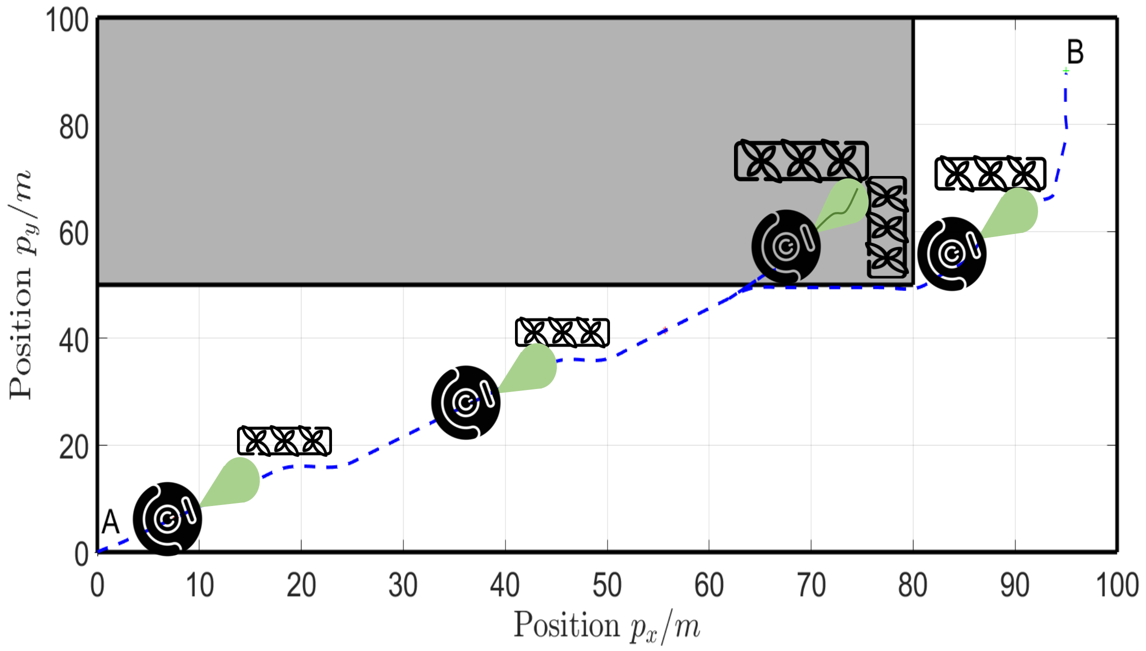


Figure 5.12: An autonomous robot equipped with a limited onboard sensor (in green) is moving in a partially known environment. The gray region represents the completely unknown region. The proposed active exploration with a fallback controller is activated when the autonomous robot is blocked by other obstacles. The safe path (in blue) was planned prior to the robot motion execution.

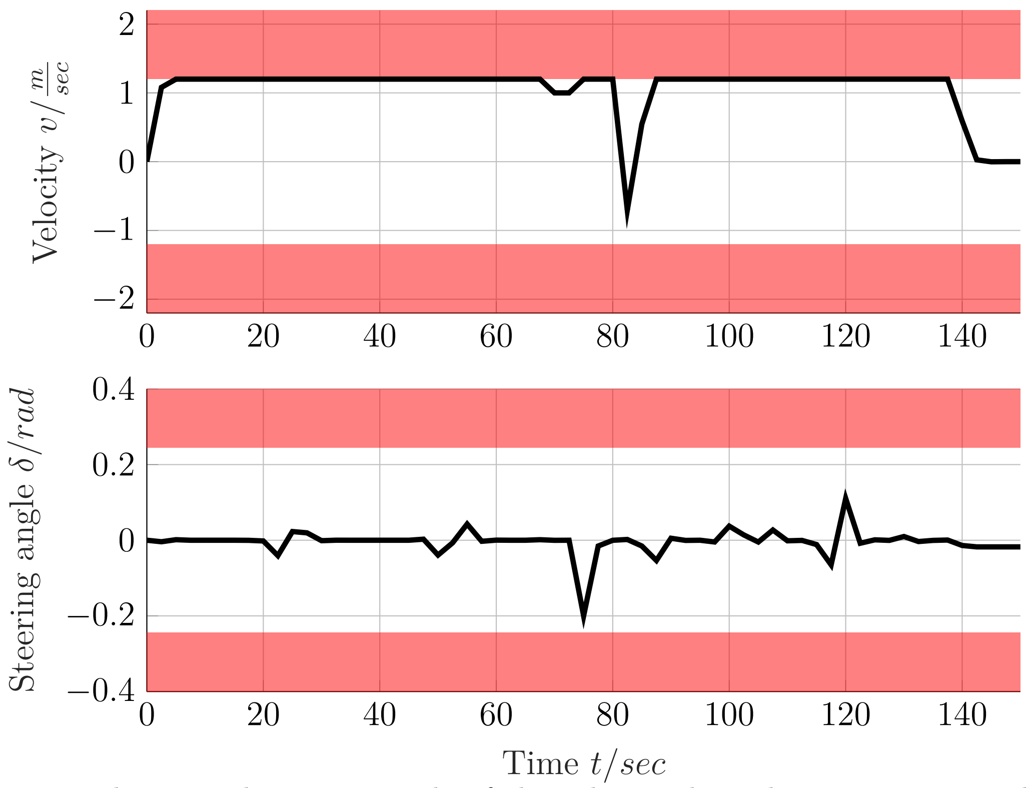


Figure 5.13: Velocity and steering angle of the robot. The red areas represent the input limitations. Note that the hierarchical dual controller steers the robot with maximum velocity. The fallback controller is activated when no new information is added, exploiting the safe preplanned path and explored waypoints.

While exploration enhances control performance and task efficiency, it may also increase deviation from the optimal path and introduce additional risk from unforeseen obstacles. To address these challenges, a safety strategy is introduced. It serves two key functions: (i) terminating exploration if the associated cost becomes excessive and (ii) ensuring task completion by steering the robot back onto a pre-planned safe path when necessary.

The effectiveness of the proposed framework is demonstrated through multiple simulations, showcasing improved performance in terms of adaptability, safety, and efficiency.

This chapter contributes to the field of sensor-based navigation by introducing an efficient hierarchical dual control approach that (1) integrates perception-aware planning, (2) models environmental uncertainty via virtual system dynamics, and (3) balances exploration and task efficiency through an explicit dual MPC framework.

6 Conclusions

This thesis explores how autonomous robots can safely and efficiently navigate unknown environments using real-time planning and control. Robots are increasingly deployed in hospitals, elderly care, disaster response, and logistics, where they must operate in dynamic, unstructured spaces without relying on pre-mapped data. Unlike conventional navigation, where a full map of the environment is available, these robots must make decisions on the fly, using only real-time sensor data from LiDAR and cameras. The key challenge is ensuring that robots move safely and intelligently while adapting to newly detected obstacles and unknown terrain.

To address this, we developed a Hierarchical Receding Horizon Planning and Control framework that allows real-time navigation while guaranteeing safety. A high-level planner generates a safe trajectory based on simplified robot dynamics, using a Mixed-Integer Programming approach to efficiently compute paths that avoid detected obstacles. The low-level controller refines this trajectory execution, but crucially, it operates only within the robot's sensor field of view, ensuring that computation remains manageable even on embedded systems found in mobile robots. This approach enables fast, reactive navigation, making it suitable for robots assisting elderly patients in homes or hospitals, where real-time responsiveness is critical for safety.

While this framework provides safe and computationally efficient navigation, it remains reactive, meaning that robots only respond to obstacles as they appear, without actively trying to learn more about their surroundings. However, in many real-world applications-such as search-and-rescue operations or hospital delivery robots navigating crowded hallways-a robot that can actively gather more information would perform better. For example, a hospital robot delivering medication could anticipate better routes based on newly detected obstacles, rather than simply avoiding them last-minute. This led us to a new, more advanced formulation that actively integrates exploration into the control process.

To achieve this, we introduced a Hierarchical Dual Rolling Horizon Planning and Control scheme, which allows the robot to not just avoid obstacles, but also strategically explore its surroundings to improve future decision-making. The high-level planner defines a safe region where the robot is encouraged to explore, while the low-level controller models uncertain environmental features as virtual system dynamics, meaning the robot can intelligently predict what might be hidden beyond its sensors. Using a mathematical optimization approach, the robot balances following its planned trajectory with actively reducing uncertainty, allowing it to find better paths in complex environments like hospitals, warehouses, or disaster zones.

However, exploration introduces risks-a robot that explores too much might stray from its optimal path or get stuck in unexpected obstacles. To counteract this, we developed an adaptive safety mechanism that continuously evaluates the cost of exploration. If the robot is at risk of losing too much time or getting blocked, it automatically switches back to a pre-planned safe path. This ensures that exploration enhances performance without jeopardizing safety, making it well-suited for robots assisting people in unpredictable environments like elder care facilities, where safety is paramount.

Through extensive simulations, we demonstrated that robots using active exploration navigate more efficiently, avoid obstacles better, and adapt intelligently to changing environments. This approach is a step toward autonomous systems that not only move safely but also think ahead, making better decisions through real-time learning. The combination of hierarchical control, exploration, and adaptive safety strategies creates a more reliable, intelligent, and autonomous robotic system that can operate in hospitals, homes, disaster zones, and industrial settings-anywhere robots need to move safely while continuously adapting to their surroundings.

In conclusion, this thesis provides a foundation for next-generation autonomous robots capable of real-time decision-making, safe navigation, and intelligent exploration. By integrating hierarchical control with learning-based exploration, robots can move efficiently, adapt to new situations, and ensure safety-paving the way for their expanded role in healthcare, logistics, and human-centered environments.

6.1 Outlook and Future Research Directions

While this thesis presents a structured, optimization-based approach for autonomous path planning and control, it does not incorporate learning-based methods, which are currently revolutionizing both robotic planning and control. Recent advances in machine learning, reinforcement learning, and generative models offer promising alternatives that could further enhance adaptability, decision-making, and real-time optimization in autonomous systems. Integrating these approaches with the proposed Hierarchical Dual Rolling Horizon Control framework could yield even more intelligent and adaptive robots, capable of learning from experience and anticipating environmental changes beyond their current sensor data.

One particularly promising direction is the use of diffusion models for trajectory generation. These models, inspired by advances in generative artificial intelligence, have demonstrated remarkable capabilities in generating high-quality motion plans. Instead of solving optimization problems from scratch at each time step, a diffusion-based planner could learn an implicit model of feasible trajectories and generate optimal paths in real-time. This could significantly reduce computational demands, making the proposed framework even more efficient and scalable for real-world deployment in resource-constrained robotic systems.

Another avenue is the integration of reinforcement learning for adaptive control. While our approach relies on mathematically formulated reward structures, reinforcement learning could allow the robot to dynamically learn optimal exploration and safety strategies through trial and error. This would enable robots to adapt to new environments without predefined constraints, making them particularly useful for long-term autonomy in changing conditions-for example, robots assisting in hospitals or navigating dynamic urban environments. Furthermore, techniques such as safe reinforcement learning could help ensure that exploration does not compromise safety, complementing the fallback mechanisms developed in this thesis.

In addition to learning-based planning and control, future work could explore hybrid approaches that combine model-based optimization with learning-based adaptation. For example, meta-learning techniques could allow the system to improve its own optimization models over time, reducing reliance on handcrafted constraints and predefined cost functions. This would create self-improving control frameworks, where the robot continuously refines its ability to balance safety, efficiency, and exploration based on its experiences. Similarly, uncertainty-aware deep learning models could replace manual uncertainty modeling, enabling the system to dynamically infer obstacles, occlusions, and sensor limitations from real-world interactions.

Ultimately, the integration of machine learning and optimization-based control represents the next frontier in autonomous robotics. While this thesis establishes a solid mathematical foundation for safe and adaptive robot navigation, future research should focus on leveraging the power of modern AI techniques to create truly intelligent, learning-enabled autonomous systems. By incorporating generative models, reinforcement learning, and hybrid learning-control approaches, we can move toward robots that not only follow safe paths but also understand and predict their surroundingsâ€”making autonomous systems more efficient, robust, and capable of operating in complex human environments.

Bibliography

[1] Amazon, UPS, domino's & the future of drone delivery services. `https://www.insiderintelligence.com/insights/drone-delivery-services/`.

[2] Back to business: Thanks to the new agricultural robot of continental farmers can refocus on the essentials. `https://www.continental.com/en/press/press-releases/20210922-robotics-its/`.

[3] Drones for search & rescue missions - altigator. `https://altigator.com/en/drones-for-search-rescue-missions/`.

[4] KUKA expands its autonomous mobile robot line-up with KMR iisy. `https://www.linkedin.com/pulse/kuka-expands-its-autonomous-mobile-robot-line-up-kmr-iisy-5m8me/`.

[5] Prepare your house for easy cleaning with a robot cleaner. `https://roboticsandautomationnews.com/`.

[6] This robot isn't going to replace your in-home nurse... yet. `https://www.cnet.com/home/this-robot-isnt-going-to-replace-your-in-home-nurse-yet/`.

[7] M. A. Abbas, R. Milman, and J. M. Eklund. Obstacle Avoidance in Real time with Nonlinear Model Predictive Control of Autonomous Vehicles. *Canadian Journal of Electrical and Computer Engineering*, 40(1):12–22, 2017.

[8] T. Ademoye and A. Davari. Trajectory Planningfor Multiple Autonomous Systems Using Mixed Integer Linear Programming. In *Proceeding of the Thirty-Eighth Southeastern Symposium on System Theory*, pages 175–179, 2006.

[9] O. Adiyatov and H. A. Varol. Rapidly-exploring random tree based memory efficient motion planning. In *IEEE International Conference on Mechatronics and Automation*, pages 354–359, 2013.

[10] O. Adiyatov and H. A. Varol. A novel RRT-based algorithm for motion planning in dynamic environments. In *IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 1416–1421, 2017.

[11] J. Alster and P. R. Bélanger. A technique for dual adaptive control. *Automatica*, 10(6):627–634, 1974.

[12] F. Amigoni and A. Gallo. A multi-objective exploration strategy for mobile robots. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3850–3855, 2005.

[13] F. Aurenhammer. Voronoi diagrams a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

[14] I. Ballesteros-Tolosana, S. Olaru, P. Rodriguez-Ayerbe, G. Pita-Gil, and R. Deborne. Collision-free trajectory planning for overtaking on highways. In *IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2551–2556, 2017.

[15] M. Bangura and R. Mahony. Real-time model predictive control for quadrotors. *IFAC Proceedings Volumes*, 47(3):11773–11780, 2014.

[16] Y. Bar-Shalom and E. Tse. Dual effect, certainty equivalence, and separation in stochastic control. *IEEE Transactions on Automatic Control*, 19(5):494–500, 1974.

[17] L. Barcelos, R. Oliveira, R. Possas, L. Ott, and F. Ramos. Disco: Double likelihood-free inference stochastic control. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 10969–10975, 2020.

[18] J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(2):224–241, 1992.

[19] V. A. Bavdekar and A. Mesbah. Stochastic model predictive control with integrated experiment design for nonlinear systems. *IFAC-PapersOnLine*, 49(7):49–54, 2016.

[20] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear programming: theory and algorithms.* John Wiley & Sons, 2013.

[21] R. Bellman. Dynamic programming. *Science*, 153(3731):34–37, 1966.

[22] K. Bergman, O. Ljungqvist, and D. Axehill. Improved optimization of motion primitives for motion planning in state lattices. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 2307–2314, 2019.

[23] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano. Control architecture design for autonomous vehicles. In *IEEE Conference on Control Technology and Applications (CCTA)*, pages 404–411. IEEE, 2018.

[24] T. Bock and T. Linner. *Robotic industrialization.* Cambridge University Press, 2015.

[25] A. D. Bonzanini, A. Mesbah, and S. Di Cairano. Perception-aware chance-constrained model predictive control for uncertain environments. In *American Control Conference (ACC)*, pages 2082–2087. IEEE, 2021.

[26] J. Borenstein, Y. Koren, et al. The vector field histogram-fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[27] C. L. Bottasso, D. Leonello, and B. Savini. Path planning for autonomous vehicles by trajectory smoothing using motion primitives. *IEEE Transactions on Control Systems Technology*, 16(6):1152–1168, 2008.

[28] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer. Modified artificial potential field method for online path planning applications. In *IEEE Intelligent Vehicles*

Symposium (IV), pages 180–185, 2017.

[29] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 540–545, 2002.

[30] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization.* Cambridge University Press, 2004.

[31] M. S. Branicky, R. A. Knepper, and J. J. Kuffner. Path and trajectory diversity: Theory and algorithms. In *2008 IEEE International Conference on Robotics and Automation*, pages 1359–1364. IEEE, 2008.

[32] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, pages 723–730, 2011.

[33] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–386, 2005.

[34] J. Butzke, K. Sapkota, K. Prasad, B. MacAllister, and M. Likhachev. State lattice with controllers: Augmenting lattice-based path planning with controller-based motion primitives. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 258–265. IEEE, 2014.

[35] C. V. Caldwell, D. D. Dunlap, and E. G. Collins. Motion planning for an autonomous underwater vehicle via sampling based model predictive control. In *OCEANS 2010 MTS/IEEE SEATTLE*, pages 1–6, 2010.

[36] E. F. Camacho and C. B. Alba. *Model predictive control.* Springer science & business media, 2013.

[37] S. Campbell, N. O'Mahony, A. Carvalho, L. Krpalkova, D. Riordan, and J. Walsh. Path planning techniques for mobile robots a review. In *6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, pages 12–16. IEEE, 2020.

[38] V. Cardoso, J. Oliveira, T. Teixeira, C. Badue, F. Mutz, T. Oliveira-Santos, L. Veronese, and A. F. De Souza. A model-predictive motion planner for the iara autonomous car. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 225–230, 2017.

[39] B. Charrow, G. Kahn, S. Patil, S. Liu, K. Goldberg, P. Abbeel, N. Michael, and V. Kumar. Information-Theoretic Planning with Trajectory Optimization for Dense 3D Mapping. In *Robotics: Science and Systems*, volume 11, pages 3–12, 2015.

[40] G. Chasparis and J. Shamma. Linear-programming-based multi-vehicle path planning with adversaries. In *Proceedings of the American Control Conference (ACC)*, pages 1072–1077 vol. 2, 2005.

[41] J. Chen, C. Du, X. Lu, and K. Chen. Multi-region Coverage Path Planning for Heterogeneous Unmanned Aerial Vehicles Systems. In *IEEE International Conference on Service-Oriented System Engineering (SOSE)*, pages 356–3565, 2019.

[42] W. Chen, M. Yan, Q. Wang, and K. Xu. Dynamic path planning and path following control for autonomous vehicle based on the piecewise affine tire model. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 235(2-3):881–893, 2021.

[43] X. Chen, M. Zhao, and L. Yin. Dynamic path planning of the uav avoiding static and moving obstacles. *Journal of Intelligent & Robotic Systems*, 99(3):909–931, 2020.

[44] L. Chengqing, M. H. Ang, H. Krishnan, and L. S. Yong. Virtual obstacle concept for local-minimum-recovery in potential-field based navigation. In *Proceedings ICRA. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 983–988, 2000.

[45] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, and W. Burgard. *Principles of robot motion: theory, algorithms, and implementations.* MIT press, 2005.

[46] S. Coenen, M. M. Steinbuch, M. van de Molengraft, J. Lunenburg, and G. Maus. Motion planning for mobile robots: A guide. *Eindhoven, Eindhoven University of Technology*, 2012.

[47] B. J. Cohen, S. Chitta, and M. Likhachev. Search-based planning for manipulation with motion primitives. In *IEEE International Conference on Robotics and Automation*, pages 2902–2908, 2010.

[48] G. Costante, C. Forster, J. Delmerico, P. Valigi, and D. Scaramuzza. Perception-aware path planning. *arXiv preprint arXiv:1605.04151*, 2016.

[49] K. F. Culligan. *Online trajectory planning for UAVs using Mixed Integer Linear Programming.* PhD thesis, Massachusetts Institute of Technology, 2006.

[50] A. Dai, S. Papatheodorou, N. Funk, D. Tzoumanikas, and S. Leutenegger. Fast Frontier-based Information-driven Autonomous Exploration with a MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9570–9576, 2020.

[51] B. Davis, I. Karamouzas, and S. J. Guy. C-opt: Coverage-aware trajectory optimization under uncertainty. *IEEE Robotics and Automation Letters*, 1(2):1020–1027, 2016.

[52] M. Dharmadhikari, T. Dang, L. Solanka, J. Loje, H. Nguyen, N. Khedekar, and K. Alexis. Motion primitives-based path planning for fast and agile exploration using aerial robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 179–185. IEEE, 2020.

[53] S. Di Cairano and I. V. Kolmanovsky. Real-time optimization and model predictive control for aerospace and automotive applications. In *Annual American Control Conference (ACC)*, pages 2392–2409. IEEE, 2018.

[54] M. Diehl, H. G. Bock, J. P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[55] H. Ding, G. Reißig, D. Groß, and O. Stursberg. Mixed-integer programming for optimal path planning of robotic manipulators. In *IEEE International Conference on Automation Science and Engineering*, pages 133–138, 2011.

[56] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

[57] J. L. Doob and J. L. Doob. *Stochastic processes*, volume 7. Wiley New York, 1953.

[58] M. Elbanhawi, M. Simic, and R. N. Jazar. Continuous path smoothing for car-like robots using b-spline curves. *Journal of Intelligent & Robotic Systems*, 80(1):23–56, 2015.

[59] Z. Elmi and M. O. Efe. Path Planning using Model Predictive Controller based on Potential Field for Autonomous Vehicles. In *IECON - 44th Annual Conference of the IEEE Industrial Electronics Society*, pages 2613–2618, 2018.

[60] L. H. Erickson and S. M. LaValle. Survivability: Measuring and ensuring path diversity. In *IEEE International Conference on Robotics and Automation*, pages 2068–2073. IEEE, 2009.

[61] S. Erke, D. Bin, N. Yiming, Z. Qi, X. Liang, and Z. Dawei. An a-star based path planning algorithm for autonomous land vehicles. *International Journal of Advanced Robotic Systems*, 17(5):1729881420962263, 2020.

[62] T. Faulwasser. *Optimization-based solutions to constrained trajectory-tracking and path-following problems.* PhD thesis, Magdeburg, Universität, Diss., 2012, 2013.

[63] T. Faulwasser and R. Findeisen. Nonlinear model predictive control for constrained output path following. *IEEE Transactions on Automatic Control*, 61(4):1026–1039, 2015.

[64] T. Faulwasser, T. Weber, P. Zometa, and R. Findeisen. Implementation of nonlinear model predictive path-following control for an industrial robot. *IEEE Transactions on Control Systems Technology*, 25(4):1505–1511, 2016.

[65] A. A. Feldbaum. Dual control theory. i. *Avtomatika i Telemekhanika*, 21(9):1240–1249, 1960.

[66] Z. Feng, G. Cao, K. M. Grigoriadis, and Q. Pan. Secure mpc-based path following for uas in adverse network environment. *IEEE Transactions on Industrial*

Informatics, 19(11):11091–11101, 2023.

[67] G. Field and Y. Stepanenko. Iterative dynamic programming: an approach to minimum energy trajectory planning for robotic manipulators. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2755–2760 vol.3, 1996.

[68] R. Findeisen. Nonlinear model predictive control: a sampled data feedback perspective. 2005.

[69] R. Findeisen and F. Allgöwer. An introduction to nonlinear model predictive control. In *21st Benelux Meeting on Systems and Control*, volume 11, pages 119–141. Citeseer, 2002.

[70] R. Findeisen, K. Graichen, and M. Mönnigmann. Eingebettete Optimierung in der Regelungstechnik–Grundlagen und Herausforderungen. *at-Automatisierungstechnik*, 66(11):877–902, 2018.

[71] L. Folsom, M. Ono, K. Otsu, and H. Park. Scalable information-theoretic path planning for a rover-helicopter team in uncertain environments. *International Journal of Advanced Robotic Systems*, 18(2):1729881421999587, 2021.

[72] E. J. Forsmo, E. I. Grotli, T. I. Fossen, and T. A. Johansen. Optimal Search Mission with Unmanned Aerial Vehicles using Mixed Integer Linear Programming. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 253–259, 2013.

[73] E. J. Forsmo. Optimal path planning for unmanned aerial systems. Master's thesis, Institutt for Teknisk Kybernetikk, 2012.

[74] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[75] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[76] S. L. Francis, S. G. Anavatti, and M. Garratt. Real time cooperative path planning for multi-autonomous vehicles. In *International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pages 1053–1057. IEEE, 2013.

[77] E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[78] G. Ganga and M. M. Dharmana. MPC controller for trajectory tracking control of quadcopter. In *International Conference on Circuit, Power and Computing Technologies (ICCPCT)*, pages 1–6. IEEE, 2017.

[79] W. Gao, M. Booker, A. Adiwahono, M. Yuan, J. Wang, and Y. W. Yun. An improved frontier-based approach for autonomous exploration. In *15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*,

pages 292–297. IEEE, 2018.

[80] S. S. Ge and Y. J. Cui. New potential functions for mobile robot path planning. *IEEE Transactions on Robotics and Automation*, 16(5):615–620, 2000.

[81] S. S. Ge, X.-C. Lai, and A. Al Mamun. Sensor-based path planning for nonholonomic mobile robots subject to dynamic constraints. *Robotics and Autonomous Systems*, 55(7):513–526, 2007.

[82] D. Genin, E. Dietrich, Y. Kouskoulas, A. Schmidt, M. Kobilarov, K. Katyal, S. Sefati, S. Mishra, and I. Papusha. A safety fallback controller for improved collision avoidance. In *IEEE International Conference on Assured Autonomy (ICAA)*, pages 129–136, 2023.

[83] A. Ghezzi, F. Messerer, J. Balocco, V. Manzoni, and M. Diehl. Implicit and Explicit Dual Model Predictive Control with an Application to Steel Recycling. *arXiv preprint arXiv:2204.02282*, 2022.

[84] C. Goerzen, Z. Kong, and B. Mettler. A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent and Robotic Systems*, 57(1):65–100, 2010.

[85] W. Gong. Probabilistic model based path planning. *Physica A: Statistical Mechanics and its Applications*, 568:125718, 2021.

[86] A. H. González, A. Ferramosca, G. A. Bustos, J. L. Marchetti, M. Fiacchini, and D. Odloak. Model predictive control suitable for closed-loop re-identification. *Systems & Control Letters*, 69:23–33, 2014.

[87] D. González, J. Pérez, V. Milanés, and F. Nashashibi. A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1135–1145, 2015.

[88] L. Han, H. Yashiro, H. T. N. Nejad, Q. H. Do, and S. Mita. Bezier curve based path planning for autonomous vehicle in urban environment. In *Intelligent Vehicles Symposium*, pages 1036–1042. IEEE, 2010.

[89] M. Hanevold. Path following model predictive control of a differential drive UGV in off-road terrain. Master's thesis, 2022.

[90] K. G. Hanssen and B. Foss. Scenario based implicit dual model predictive control. *IFAC-PapersOnLine*, 48(23):416–421, 2015.

[91] Y. Hao, B. Li, L. Shao, Y. Zhang, and J. Cui. Multi-objective path planning for unmanned aerial vehicle based on mixed integer programming. In *Chinese Automation Congres (CAC)*, pages 7035–7039, 2017.

[92] R. He, S. Prentice, and N. Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *IEEE International Conference on Robotics and Automation*, pages 1814–1820. IEEE, 2008.

[93] T. A. N. Heirung. Dual control: optimal, adaptive decision-making under uncertainty. 2016.

[94] T. A. N. Heirung, B. Foss, and B. E. Ydstie. MPC-based dual control with online experiment design. *Journal of Process Control*, 32:64–76, 2015.

[95] T. A. N. Heirung, B. E. Ydstie, and B. Foss. Towards dual MPC. *IFAC Proceedings Volumes*, 45(17):502–507, 2012.

[96] J. D. Hernàndez, E. Vidal, G. Vallicrosa, E. Galceran, and M. Carreras. Online path planning for autonomous underwater vehicles in unknown environments. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1152–1157, 2015.

[97] D. Holz, N. Basilico, F. Amigoni, S. Behnke, et al. A comparative evaluation of exploration strategies and heuristics to improve them. In *ECMR*, pages 25–30, 2011.

[98] B. Houska, H. Ferreau, and M. Diehl. ACADO Toolkit – An Open Source Framework for Automatic Control and Dynamic Optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[99] M. Hoy, A. S. Matveev, and A. V. Savkin. Algorithms for collision-free navigation of mobile robots in complex cluttered environments: a survey. *Robotica*, 33(3):463–497, 2015.

[100] C. Hu, L. Zhao, L. Cao, P. Tjan, and N. Wang. Steering control based on model predictive control for obstacle avoidance of unmanned ground vehicle. *Measurement and Control*, 53(3-4):501–518, 2020.

[101] X. Hu, L. Chen, B. Tang, D. Cao, and H. He. Dynamic path planning for autonomous driving on various roads with avoidance of static and moving obstacles. *Mechanical Systems and Signal Processing*, 100:482–500, 2018.

[102] V. T. Huynh, M. Dunbabin, and R. N. Smith. Predictive motion planning for auvs subject to strong time-varying currents and forecasting uncertainties. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1144–1151, 2015.

[103] Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys (CSUR)*, 24(3):219–291, 1992.

[104] Y. K. Hwang and P. C. Chen. A heuristic and complete planner for the classical mover's problem. In *Proceedings of the International Conference on Robotics and Automation*, volume 1, pages 729–736. IEEE, 1995.

[105] Y. K. Hwang, N. Ahuja, et al. A potential field approach to path planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992.

[106] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen. Improved Area Covering in Dynamic Environments by Nonlinear Model Predictive Path Following Control. *IFAC-PapersOnLine*, 52(15):418–423, 2019.

[107] M. Ibrahim. *Design and Real-Time Implementation of Model Predictive Control for Flight Systems in Presence of Uncertainties.* PhD thesis, OVGU, 2020.

[108] M. Ibrahim, J. Matschek, B. Morabito, and R. Findeisen. Hierarchical model predictive control for autonomous vehicle area coverage. *IFAC-PapersOnLine*, 52(12):79–84, 2019.

[109] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu. Mixed-integer programming in motion planning. *Annual Reviews in Control*, 51:65–87, 2021.

[110] S. Ishihara, M. Kanai, R. Narikawa, and T. Ohtsuka. A proposal of path planning for robots in warehouses by model predictive control without using global paths. *IFAC-PapersOnLine*, 55(37):573–578, 2022.

[111] R. N. Jazar. *Vehicle Dynamics: Theory and Application.* Springer, 2017.

[112] A. L. Jennings, R. Ordonez, and N. Ceccarelli. Dynamic programming applied to uav way point path planning in wind. In *IEEE International Conference on Computer-Aided Control Systems*, pages 215–220, 2008.

[113] B. J. Julian, S. Karaman, and D. Rus. On mutual information-based control of range sensing robots for mapping applications. *The International Journal of Robotics Research*, 33(10):1375–1392, 2014.

[114] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. In *Robot Operating System (ROS)*, pages 3–39. Springer, 2017.

[115] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt. In *IEEE International Conference on Robotics and Automation*, pages 1478–1483. IEEE, 2011.

[116] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel. A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3):448–468, 2021.

[117] S. Khan and J. Guivant. Nonlinear model predictive path-following controller for a small-scale autonomous bulldozer for accurate placement of materials and debris of masonry in construction contexts. *IEEE Access*, 9:102069–102080, 2021.

[118] O. Khatib. The potential field approach and operational space formulation in robot control. In *Adaptive and Learning Systems*, pages 367–377. Springer, 1986.

[119] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles*, pages 396–404. Springer, 1986.

[120] A. Koubâa, H. Bennaceur, I. Chaari, S. Trigui, A. Ammar, M.-F. Sriti, M. Alajlan, O. Cheikhrouhou, and Y. Javed. *Robot path planning and cooperation*, volume 772. Springer, 2018.

[121] N. Krausch, J. W. Kim, S. Lucia, S. Gross, T. Barz, P. Neubauer, and M. N. C. Bournazou. Optimal operation of parallel mini-bioreactors in bioprocess development using multi-stage MPC. *bioRxiv*, 2021.

[122] K. Kumar, T. A. N. Heirung, S. C. Patwardhan, and B. Foss. Experimental evaluation of a mimo adaptive dual MPC. *IFAC-PapersOnLine*, 48(8):545–550, 2015.

[123] F. Künhe, J. Gomes, and W. Fetter. Mobile robot trajectory tracking using model predictive control. In *II IEEE Latin-American Robotics Symposium*, volume 51. Citeseer, 2005.

[124] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. Motion planning for urban driving using rrt. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1681–1686. IEEE, 2008.

[125] Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[126] H. La, A. Potschka, J. Schlöder, and H. Bock. Dual control and information gain in controlling uncertain processes. *IFAC-PapersOnLine*, 49(7):139–144, 2016.

[127] H. C. La, A. Potschka, J. Schlöder, and H. G. Bock. Dual control and online optimal experimental design. *SIAM Journal on Scientific Computing*, 39(4):B640–B657, 2017.

[128] C. A. Larsson, A. Ebadat, C. R. Rojas, X. Bombois, and H. Hjalmarsson. An application-oriented approach to dual control with excitation for closed-loop identification. *European Journal of Control*, 29:1–16, 2016.

[129] J.-C. Latombe. *Robot motion planning*, volume 124. Springer Science & Business Media, 2012.

[130] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[131] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects: Steven M. LaValle, Iowa State University, A James J. Kuffner, Jr., University of Tokyo, Tokyo, Japan. *Algorithmic and Computational Robotics*, pages 303–307, 2001.

[132] D. Levine, B. Luders, and J. P. How. Information-theoretic motion planning for constrained sensor networks. *Journal of Aerospace Information Systems*, 10(10):476–496, 2013.

[133] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng. Path planning for autonomous vehicles using model predictive control. In *Intelligent Vehicles Symposium (IV)*, pages 174–179, 2017.

[134] C. Liu and W.-H. Chen. Hierarchical path planning and flight control of small autonomous helicopters using MPC techniques. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 417–422, 2013.

[135] L.-s. Liu, J.-f. Lin, J.-x. Yao, D.-w. He, J.-s. Zheng, J. Huang, and P. Shi. Path planning for smart car based on dijkstra algorithm and dynamic window approach. *Wireless Communications and Mobile Computing*, 2021, 2021.

[136] Q. Liu, L. Zhao, Z. Tan, and W. Chen. Global path planning for autonomous vehicles in off-road environment via an a-star algorithm. *International Journal of Vehicle Autonomous Systems*, 13(4):330–339, 2017.

[137] S. Liu, N. Atanasov, K. Mohta, and V. Kumar. Search-based motion planning for quadrotors using linear quadratic minimum time control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879, 2017.

[138] I. Lluvia, E. Lazkano, and A. Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7):2445, 2021.

[139] J. Löfberg. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *In Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[140] S. Lucia, T. Finkler, and S. Engell. Multi-stage nonlinear model predictive control applied to a semi-batch polymerization reactor under uncertainty. *Journal of Process Control*, 23(9):1306–1319, 2013.

[141] S. Lucia, M. Kögel, P. Zometa, D. E. Quevedo, and R. Findeisen. Predictive control, embedded cyberphysical systems and systems of systems–a perspective. *Annual Reviews in Control*, 41:193–207, 2016.

[142] D. G. Luenberger. Linear and nonlinear programming david g. luenberger, yinyu ye, 2008.

[143] C. E. Luis, M. Vukosavljev, and A. P. Schoellig. Online trajectory generation with distributed model predictive control for multi-robot motion planning. *IEEE Robotics and Automation Letters*, 5(2):604–611, 2020.

[144] N. Makariye. Towards shortest path computation using dijkstra algorithm. In *International Conference on IoT and Application (ICIOT)*, pages 1–3. IEEE, 2017.

[145] G. Marafioti, R. R. Bitmead, and M. Hovd. Persistently exciting model predictive control. *International Journal of Adaptive Control and Signal Processing*, 28(6):536–552, 2014.

[146] J. M. Martín, P. Vega, and S. Revollar. Set-point optimization for enhancing the MPC control of the N-removal process in WWTP's. In *World Automation Congress*, pages 1–6. IEEE, 2012.

[147] R. Maruf. LA residents: Your next Shake Shack order could be delivered by a robot | CNN Business — edition.cnn.com. `https://edition.cnn.com/2024/08/14/business/la-residents-your-next-shake-shack-order-could-be-delivered-by-a-robot/index.html`. [Accessed 06-02-2025].

[148] J. Matschek, T. Bäthge, T. Faulwasser, and R. Findeisen. Nonlinear predictive control for trajectory tracking and path following: An introduction and perspective. In *Handbook of Model Predictive Control*, pages 169–198. Springer, 2019.

[149] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 36(6):789–814, 2000.

[150] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. Scokaert. Survey con-

strained model predictive control: Stability and optimality. *Automatica (Journal of IFAC)*, 36(6):789–814, 2000.

[151] D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *International Conference on Robotics and Automation*, pages 477–483, 2012.

[152] A. Mesbah. Stochastic model predictive control: An overview and perspectives for future research. *IEEE Control Systems Magazine*, 36(6):30–44, 2016.

[153] A. Mesbah. Stochastic model predictive control with active uncertainty learning: A survey on dual control. *Annual Reviews in Control*, 45:107–117, 2018.

[154] R. Milito, C. Padilla, R. Padilla, and D. Cadorin. An innovations approach to dual control. *Transactions on Automatic Control*, 27(1):132–137, 1982.

[155] T. Ming, W. Deng, S. Zhang, and B. Zhu. MPC-based trajectory tracking control for intelligent vehicles. Technical report, SAE Technical Paper, 2016.

[156] J. Minguez and L. Montano. Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios. *IEEE Transactions on Robotics and Automation*, 20(1):45–59, 2004.

[157] F. Molinari, N. N. Anh, and L. Del Re. Efficient mixed integer programming for autonomous overtaking. In *American Control Conference (ACC)*, pages 2303–2308. IEEE, 2017.

[158] H. Nawaz, H. M. Ali, and S. Massan. Applications of unmanned aerial vehicles: a review. *3C Tecnología. Glosas de innovación aplicadas a la pyme. Special Issue*, pages 85–105, 2019.

[159] T. Nayl, G. Nikolakopoulos, and T. Gustafsson. On-Line path planning for an articulated vehicle based on Model Predictive Control. In *IEEE International Conference on Control Applications (CCA)*, pages 772–777, 2013.

[160] J. Ng and T. Bräunl. Performance comparison of bug navigation algorithms. *Journal of Intelligent and Robotic Systems*, 50(1):73–84, 2007.

[161] J. Nieto, E. Slawinski, V. Mut, and B. Wagner. Online path planning based on rapidly-exploring random trees. In *IEEE International Conference on Industrial Technology*, pages 1451–1456, 2010.

[162] J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg. Receding horizon maneuver generation for automated highway driving. *Control Engineering Practice*, 41:124–133, 2015.

[163] I. Noreen, A. Khan, and Z. Habib. Optimal path planning using rrt* based approaches: a survey and future directions. *International Journal of Advanced Computer Science and Applications*, 7(11), 2016.

[164] C. Norén. Path planning for autonomous heavy duty vehicles using nonlinear model predictive control, 2013.

[165] E. Palazzolo and C. Stachniss. Effective exploration for mavs based on the

expected information gain. *Drones*, 2(1):9, 2018.

[166] C. Papachristos, S. Khattak, and K. Alexis. Uncertainty-aware receding horizon exploration and mapping using aerial robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4568–4575. IEEE, 2017.

[167] C. Papachristos, F. Mascarich, S. Khattak, T. Dang, and K. Alexis. Localization uncertainty-aware autonomous exploration and mapping with aerial robots using receding horizon path-planning. *Autonomous Robots*, 43(8):2131–2161, 2019.

[168] A. A. Paranjape, K. C. Meier, X. Shi, S.-J. Chung, and S. Hutchinson. Motion primitives and 3d path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015.

[169] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, et al. A review: On path planning strategies for navigation of mobile robot. *Defence Technology*, 15(4):582–606, 2019.

[170] Z. Peng, B. Li, X. Chen, and J. Wu. Online route planning for UAV based on model predictive control and particle swarm optimization algorithm. In *Proceedings of the 10th World Congress on Intelligent Control and Automation*, pages 397–401, 2012.

[171] B. Penin, P. R. Giordano, and F. Chaumette. Minimum-time trajectory planning under intermittent measurements. *IEEE Robotics and Automation Letters*, 4(1):153–160, 2018.

[172] M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, pages 1–7. Munich Germany, 2005.

[173] M. Pivtoraiko and A. Kelly. Kinodynamic motion planning with state lattice motion primitives. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2172–2179, 2011.

[174] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle. Optimal trajectory planning for autonomous driving integrating logical constraints: An miqp perspective. In *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 205–210. IEEE, 2016.

[175] S. J. Qin and T. A. Badgwell. An Overview of Industrial Model Predictive Control Technology. In *AIche symposium series*, volume 93, pages 232–256. New York, NY: American Institute of Chemical Engineers, 1971-c2002., 1997.

[176] S. J. Qin and T. A. Badgwell. A survey of industrial model predictive control technology. *Control Engineering Practice*, 11(7):733–764, 2003.

[177] L. Quan, L. Han, B. Zhou, S. Shen, and F. Gao. Survey of UAV motion planning. *IET Cyber-systems and Robotics*, 2(1):14–21, 2020.

[178] J. Rathouskỳ and V. Havlena. MPC-based approximate dual controller by information matrix maximization. *International Journal of Adaptive Control and*

Signal Processing, 27(11):974–999, 2013.

[179] J. B. Rawlings. Tutorial overview of model predictive control. *IEEE Control Systems Magazine*, 20(3):38–52, 2000.

[180] J. Rawlings, D. Mayne, and M. Diehl. *Model predictive control: theory, computation, and design*, volume 2. Nob Hill Publishing Madison, WI, 2017.

[181] M. Reda, A. Onsy, A. Y. Haikal, and A. Ghanbari. Path planning algorithms in the autonomous driving system: A comprehensive review. *Robotics and Autonomous Systems*, 174:104630, 2024.

[182] A. Richards and J. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the American Control Conference (IEEE Cat. No.CH37301)*, volume 3, pages 1936–1941 vol.3, 2002.

[183] R. T. Rodrigues, M. Basiri, A. P. Aguiar, and P. Miraldo. Low-level active visual navigation: Increasing robustness of vision-based localization using potential fields. *IEEE Robotics and Automation Letters*, 3(3):2079–2086, 2018.

[184] M. Rosen. The poison of Enthusiasm. *Taula: quaderns de pensament*, pages 7–28, 1991.

[185] P. Ru and K. Subbarao. Nonlinear model predictive control for unmanned aerial vehicles. *Aerospace*, 4(2):31, 2017.

[186] F. Rubio, F. Valero, and C. Llopis-Albert. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems*, 16(2):1729881419839596, 2019.

[187] C. S. Sallaberger and G. M. D'Eleuterio. Optimal robotic path planning using dynamic programming and randomization. *Acta Astronautica*, 35(2-3):143–156, 1995.

[188] L. Schmid, M. Pantic, R. Khanna, L. Ott, R. Siegwart, and J. Nieto. An Efficient Sampling-Based Method for Online Informative Path Planning in Unknown Environments. *IEEE Robotics and Automation Letters*, 5(2):1500–1507, 2020.

[189] T. Schouwenaars. *Safe trajectory planning of autonomous vehicles.* PhD thesis, Massachusetts Institute of Technology, 2006.

[190] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference (ECC)*, pages 2603–2608. IEEE, 2001.

[191] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed integer programming for multi-vehicle path planning. In *European Control Conference (ECC)*, pages 2603–2608, 2001.

[192] T. Schouwenaars, É. Féron, and J. How. Safe receding horizon path planning for autonomous vehicles. In *Proceedings of the Annual Allerton Conference on Communication Control and Computing*, volume 40, pages 295–304, 2002.

[193] A. Shamiri, M. A. Hussain, F. S. Mjalli, and A. Arami-Niya. Temperature control

of industrial gas phase propylene polymerization in fluidized bed reactors using model predictive control. 2011.

[194] M. Shanmugavel. Path planning of multiple autonomous vehicles. 2007.

[195] Z. Shareef. *Path planning and trajectory optimization of delta parallel robot.* PhD thesis, Paderborn, Universität Paderborn., 2015.

[196] N. Sharma, J. K. Pandey, and S. Mondal. A review of mobile robots: Applications and future prospect. *International Journal of Precision Engineering and Manufacturing*, 24(9):1695–1706, 2023.

[197] C. Shen, Y. Shi, and B. Buckham. Model predictive control for an AUV with dynamic path planning. In *54th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 475–480, 2015.

[198] C. Shen, Y. Shi, and B. Buckham. Path-following control of an auv: A multiobjective model predictive control approach. *IEEE Transactions on Control Systems Technology*, 27(3):1334–1342, 2018.

[199] R. Sinha, E. Schmerling, and M. Pavone. Closing the loop on runtime monitors with fallback-safe MPC. In *62nd IEEE Conference on Decision and Control (CDC)*, pages 6533–6540, 2023.

[200] V. Sivakumar and P. Sujit. MPC-based Multi-UAV Path Planning for Convoy Protection in 3D. In *IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 1554–1559, 2021.

[201] N. Sleumer and N. Tschichold-Gürmann. Exact cell decomposition of arrangements used for path planning in robotics. *Technical Report/ETH Zurich, Department of Computer Science*, 329, 1999.

[202] M. Soliman and R. Findeisen. Moving horizon planning and control for autonomous vehicles with active exploration and fallback strategies. In *Proceedings of 21th International Conference on Informatics in Control, Automation and Robotics*, pages 359–367, 2024.

[203] M. Soliman, B. Morabito, and R. Findeisen. Towards Safe Exploration for Autonomous Vehicles using Dual Model Predictive Control. *IFAC-PapersOnLine*, 55(27):387–392, 2022.

[204] A. V. Sprang. Path and trajectory planning by model predictive control in autonomous racing. B.S. thesis, University of Twente, 2021.

[205] T. J. Stastny, A. Dash, and R. Siegwart. Nonlinear MPC for fixed-wing UAV trajectory tracking: Implementation and flight experiments. In *AIAA guidance, navigation, and control conference*, page 1512, 2017.

[206] P. Švestka and M. H. Overmars. Probabilistic path planning. *Robot Motion Planning and Control*, pages 255–304, 1998.

[207] S. Tang and V. Kumar. Mixed integer quadratic program trajectory generation for a quadrotor with a cable-suspended payload. In *International Conference on*

Robotics and Automation (ICRA), pages 2216–2222, 2015.

[208] D. Telen, B. Houska, M. Vallerio, F. Logist, and J. Van Impe. A study of integrated experiment design for NMPC applied to the droop model. *Chemical Engineering Science*, 160:370–383, 2017.

[209] S. Thangavel, M. Aboelnour, S. Lucia, R. Paulen, and S. Engell. Robust dual multi-stage NMPC using guaranteed parameter estimation. *IFAC-PapersOnLine*, 51(20):72–77, 2018.

[210] S. Thangavel, S. Lucia, R. Paulen, and S. Engell. Towards dual robust nonlinear model predictive control: A multi-stage approach. In *American Control Conference (ACC)*, pages 428–433. IEEE, 2015.

[211] S. Thangavel, S. Lucia, R. Paulen, and S. Engell. Robust nonlinear model predictive control with reduction of uncertainty via dual control. In *21st International Conference on Process Control (PC)*, pages 48–53. IEEE, 2017.

[212] L. Tian, Z. Zhang, C. Zheng, Y. Tian, Y. Zhao, Z. Wang, and Y. Qin. An improved rapidly-exploring random trees algorithm combining parent point priority determination strategy and real-time optimization strategy for path planning. *Sensors*, 21(20):6907, 2021.

[213] H. Unbehauen. Adaptive dual control systems: a survey. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium*, pages 171–180, 2000.

[214] E. Žáčeková, S. PrÃvara, and J. Komárek. On dual control for buildings using persistent excitation condition. In *IEEE 51st Conference on Decision and Control (CDC)*, pages 2158–2163, 2012.

[215] N. Van Duijkeren, R. Verschueren, G. Pipeleers, M. Diehl, and J. Swevers. Path-following NMPC for serial-link robot manipulators using a path-parametric system reformulation. In *European Control Conference (ECC)*, pages 477–482. IEEE, 2016.

[216] Ã. B. Viana and N. Aouf. Distributed Cooperative Path-Planning for Autonomous Vehicles Integrating Human Driver Trajectories. In *International Conference on Intelligent Systems (IS)*, pages 655–661, 2018.

[217] R. Walambe, N. Agarwal, S. Kale, and V. Joshi. Optimal trajectory generation for car-type mobile robot using spline interpolation. *IFAC-PapersOnLine*, 49(1):601–606, 2016.

[218] C. Wang, W. Chi, Y. Sun, and M. Q.-H. Meng. Autonomous robotic exploration by incremental road map construction. *IEEE Transactions on Automation Science and Engineering*, 16(4):1720–1731, 2019.

[219] J. Wang, S. Wu, H. Li, and J. Zou. Path planning combining improved rapidly-exploring random trees with dynamic window approach in ROS. In *13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pages 1296–

1301, 2018.

[220] D. J. Webb and J. v. d. Berg. Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints. *arXiv preprint arXiv:1205.5088*, 2012.

[221] H. Wei and Y. Shi. MPC-based motion planning and control enables smarter and safer autonomous marine vehicles: Perspectives and a tutorial survey. *IEEE/CAA Journal of Automatica Sinica*, 10(1):8–24, 2022.

[222] D. Wilson, J. H. Davenport, M. England, and R. Bradford. A" piano movers" problem reformulated. In *15th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 53–60. IEEE, 2013.

[223] L. A. Wolsey. Mixed integer programming. *Wiley Encyclopedia of Computer Science and Engineering*, pages 1–10, 2007.

[224] E. Xidias, P. Zacharia, and A. Nearchou. Path planning and scheduling for a fleet of autonomous vehicles. *Robotica*, 34(10):2257–2273, 2016.

[225] W. Xinyu, L. Xiaojuan, G. Yong, S. Jiadong, and W. Rui. Bidirectional potential guided rrt* for motion planning. *IEEE Access*, 7:95046–95057, 2019.

[226] W. Xue, B. Yang, T. Kaizuka, and K. Nakano. A fallback approach for an automated vehicle encountering sensor failure in monitoring environment. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1807–1812, 2018.

[227] W. Xue, R. Zheng, B. Yang, Z. Wang, T. Kaizuka, and K. Nakano. An adaptive model predictive approach for automated vehicle control in fallback procedure based on virtual vehicle scheme. *Journal of Intelligent and Connected Vehicles*, 2(2):67–77, 2019.

[228] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.*, pages 146–151, 1997.

[229] N. K. Yilmaz, C. Evangelinos, P. F. J. Lermusiaux, and N. M. Patrikalakis. Path planning of autonomous underwater vehicles for adaptive sampling using mixed integer linear programming. *IEEE Journal of Oceanic Engineering*, 33(4):522–537, 2008.

[230] C. Yin, Z. Xiao, X. Cao, X. Xi, P. Yang, and D. Wu. Offline and Online Search: UAV Multiobjective Path Planning Under Dynamic Urban Environment. *IEEE Internet of Things Journal*, 5(2):546–558, 2018.

[231] H.-S. Yoon and T.-H. Park. Motion planning of autonomous mobile robots by iterative dynamic programming. *Intelligent Service Robotics*, 8(3):165–174, 2015.

[232] J. Yu and F. Luo. Fallback strategy for level 4+ automated driving system. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 156–162, 2019.

[233] F. Zhang, N. Li, T. Xue, Y. Zhu, R. Yuan, and Y. Fu. An improved dynamic

window approach integrated global path planning. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2873–2878, 2019.

[234] Z. Zhang. *Active Robot Vision: From State Estimation to Motion Planning*. PhD thesis, Universität Zürich, 2020.

[235] X. Zhao, J. Gao, and W. Yan. A receding horizon motion planner for underwater vehicle manipulator systems. In *OCEANS MTS/IEEE Charleston*, pages 1–7, 2018.

[236] Z. Zhao, K. Yuan, J. Du, Y. Wang, Y. Huang, and H. Chen. Planning and control of autonomous driving in lane-change manoeuvre based on MPC: a framework and design principles. *International Journal of Vehicle Design*, 92(2-4):357–381, 2023.

[237] B. Zhou, J. Pan, F. Gao, and S. Shen. Raptor: Robust and perception-aware trajectory replanning for quadrotor fast flight. *IEEE Transactions on Robotics*, 37(6):1992–2009, 2021.

[238] X.-Y. Zou and J. Zhu. Virtual local target method for avoiding local minimum in potential field based robot navigation. *Journal of Zhejiang University-Science A*, 4(3):264–269, 2003.