Motion Planning and Decision Making with applications to Truck-Trailers and Buses

RUI OLIVEIRA

Doctoral Thesis
Stockholm, Sweden 2022

KTH Royal Insitute of Technology
School of Electrical Engineering and Computer Science
Division of Decision and Control Systems
SE-100 44 Stockholm
SWEDEN

Abstract

This thesis focuses on motion planning algorithms for self-driving heavy-duty vehicles. Motion planning is a fundamental part of autonomous vehicles, tasked with finding the correct sequence of actions that take the vehicle towards its goal. This work focuses on the aspects that distinguish heavy-duty vehicles from passenger vehicles and require novel developments within motion planning algorithms. The proposed algorithms are studied in simulation environments and on two Scania prototype autonomous vehicles: a mining truck and a public transport bus.

We start by addressing the problem of finding the shortest paths for a vehicle in obstacle-free environments. This problem has long been studied, but the considered vehicle models have been simplistic. We propose a novel algorithm that plans paths respecting complex vehicle actuator constraints associated with the slow dynamics of heavy vehicles.

Using the previous method, we tackle the motion planning problem in environments with obstacles. Lattice-based motion planners, a popular choice for this type of scenario, come with drawbacks related to the sub-optimality of solution paths and the discretization of the goal state. We propose a novel path optimization method that significantly reduces both drawbacks. The resulting optimized paths contain less oscillatory behavior and arrive precisely at arbitrary non-discretized goal states.

We then study the problem of bus driving in urban environments. In order to successfully maneuver buses, distinct driving objectives must be used in planning algorithms. Moreover, a novel environment classification and collision avoidance scheme must be introduced. The result is a motion planning algorithm that mimics professional bus driver behavior, resulting in safer driving and increased vehicle maneuverability.

One particular challenge of driving in urban environments is common to buses and trucks with trailers, namely, that of centering the whole vehicle body on the road. In the case of buses, the long wheelbase introduces a conflict between centering the rear axle vehicle or centering the front axle. In the case of trucks with trailers, a similar conflict appears, this time between centering the truck body or centering the trailer body. We propose a framework to design motion planners that optimally trade-off between these conflicting objectives, resulting in planned paths that center the whole vehicle body, improving driving behavior.

Finally, we study the challenges of interacting with human-driven vehicles. We propose a motion planning framework that addresses the multi-modality of human behaviors, the interactive nature of traffic, and the impact of the autonomous vehicle on human drivers' decision making. The result is a motion planner that can reason about multiple future outcomes

of a traffic scene, minimizing the expected cost across all outcomes. Furthermore, we show that incorporating neuroscience-based decision making models of human drivers into the motion planner results in the autonomous vehicle taking safe but assertive maneuvers, reducing the conservativeness usually seen in autonomous vehicles.

Sammanfattning

Denna avhandling fokuserar på algoritmer för rörelseplanering av självkörande tunga fordon. Rörelseplanering är en grundläggande del av autonoma fordon, med uppgift att hitta rätt sekvens av åtgärder som tar fordonet mot sitt mål. Detta arbete fokuserar på de aspekter som skiljer tunga fordon från personbilar och kräver ny utveckling av algoritmer för rörelseplanering. De föreslagna algoritmerna studeras i simuleringsmiljöer och på två prototyper av autonoma Scaniafordon: en gruvlastbil och en kollektivtrafikbuss.

Vi börjar med att undersöka problemet med att hitta de kortaste vägarna för ett fordon i miljöer fria från hinder. Detta problem har studerats länge, men de övervägda fordonsmodellerna har varit förenklade. Vi föreslår en ny algoritm som planerar vägar som respekterar komplexa begränsningar av fordonsmanövrar förknippade med tunga fordons långsamma dynamik.

Med den tidigare metoden adresserar vi rörelseplaneringsproblemet i miljöer med hinder. Lattice-baserade rörelseplanerare, ett populärt val för den här typen av scenarier, har problem med suboptimala lösningar och diskretiseringen av måltillståndet. Vi föreslår en ny optimeringsmetod som avsevärt minskar båda dessa problem. De resulterande optimerade vägarna innehåller mindre oscillationer och når godtyckliga icke-diskretiserade måltillstånd exakt.

Vi studerar sedan problemet med busskörning i stadsmiljö. För att framgångsrikt manövrera bussar måste distinkta körmål användas vid planering av algoritmer. Dessutom måste ett nytt system införas för klassificering av omgivningen och kollisionsundvikning. Resultatet är en algoritm för rörelseplanering som efterliknar professionella busschaufförers beteende, vilket resulterar i säkrare körning och ökad manöverbarhet av fordonet.

En speciell utmaning med att köra i stadsmiljö är gemensam för bussar och lastbilar med släp, nämligen att centrera hela fordonskarossen på vägen. När det gäller bussar introducerar den långa hjulbasen en konflikt mellan att centrera bakaxeln och att centrera framaxeln. När det gäller lastbilar med släp uppstår en liknande konflikt, denna gång mellan centrering av lastbilskarossen och centrering av trailerkarossen. Vi föreslår ett ramverk för att utforma rörelseplanerare som optimalt avväger dessa motstridiga mål, vilket resulterar i planerade vägar som centrerar hela fordonskarossen och förbättrar körbeteendet.

Slutligen studerar vi utmaningarna med att interagera med människodrivna fordon. Vi föreslår ett ramverk för rörelseplanering som tar itu med mångfalden av mänskliga beteenden, trafikens interaktiva karaktär och det autonoma fordonets inverkan på mänskliga förares beslutsfattande. Resultatet är en rörelseplanerare som kan resonera om flera framtida utfall av ett

trafikscenario, vilket minimerar den förväntade kostnaden för alla utfall. Dessutom visar vi att inkorporering av psykologibaserade beslutsfattande modeller av mänskliga förare i rörelseplaneraren resulterar i att det autonoma fordonet väljer säkra men beslutsamma manövrar, vilket minskar den konservativitet som ofta ses i autonoma fordon.

To my parents

Acknowledgments

A big thank you to my supervisor Bo Wahlberg for his guidance and helpfulness during my development as a researcher. His ideas and suggestions have played an important role in identifying the challenges of, and in developing solutions for, autonomous heavy-duty vehicles.

I am also grateful for my long-lived connection to my co-supervisor, Jonas Mårtensson. With his help, I have managed to take part in different exciting projects, learn new things, and demonstrate my work in creative and attractive ways.

This endeavor would have been much more challenging without my industrial supervisors, Marcello Cirillo and Pedro F. Lima. Marcello's experience in motion planning helped me get to grips with the field and be exposed to the latest developments in the area. Pedro introduced me to a new framework for motion planning, which eventually became my primary research focus.

My managers throughout the years, Jon Andersson, Assad Alam, Michael Åström, Kalle Fagerberg, and Daniel Fellke, are responsible for creating a secure and balanced work environment at Scania, in which I was able to thrive.

It is always more fun to collaborate with others, and I am fortunate to have closely worked with Gonçalo Collares Pereira, Erik Ward, and Oskar Ljungqvist.

During this time, I had the opportunity to spend six months at the Model Predictive Control Lab at the University of California, Berkeley as a visiting student. I am grateful to Professor Francesco Borrelli for the opportunity and to Tim Brüdigam, Vijay Govindarajan, and Siddharth Nair for the helpful discussions.

Thank you to the people who took their time to review parts of this thesis and contributed to improving its quality: Martin Björk, Manne Held, Holger Banzhaf, Kristoffer Bergman, Ivo Batkovic, and Gonçalo Collares Pereira.

A final thank you to all my colleagues and friends who have positively influenced me.

This work was partially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Rui Oliveira
August 2022

Contents

Chapter 1

Introduction

Autonomous vehicles have been enjoying ever-increasing attention in recent years. The benefits and possibilities enabled by vehicles that can drive themselves have compelled industry and academia to invest massive resources into this technology. These benefits include increased safety, economic gains, and new mobility options.

Heavy-duty vehicles (HDVs), which consist of trucks and buses, can potentially be the first vehicle segment to adopt autonomous driving functionalities. Costs related to the driver account for 35% of the cost of hauling cargo over trucks [1]. The limitations of human drivers present one of the biggest bottlenecks for long-haulage operations. The introduction of autonomous HDVs can double the productivity of long-haul transport. Moreover, the trucking industry is currently facing a driver shortage [2]. Autonomous trucks can reduce the need for truck drivers and make the truck driving profession more attractive.

Urban transportation will likely benefit from automated driving. An expected increase in the population living in urban areas will increase the need for urban transport. At the same time, a relative decrease in the available workforce is expected, due to an aging population, creating a shortage of drivers. Automation has the potential to solve both problems, and multiple automated mobility pilots are currently testing this technology [3].

Motion planning is one of the building blocks of autonomous vehicles. It deals with finding a sequence of actions that make the vehicle progress along the road or arrive at a specific goal. Planned motions need to be safe and avoid obstacles, smooth and comfortable for passengers or cargo inside the vehicle, and efficient in optimizing fuel costs or travel times.

This thesis studies motion planning and its applications to autonomous HDVs. Unlike passenger vehicles, HDVs often require more space to maneuver, longer braking distances, and different driving techniques. We focus

on the fundamental differences that distinguish HDVs from passenger vehicles and how these dissimilarities influence the design of motion planning algorithms.

1.1 The benefits of autonomous vehicles

Every year a distressing amount of 1.35 million deaths occurs in road traffic accidents. Road traffic injuries are currently the leading cause of death for people ages 5-29 and the eighth leading cause of death for people of all ages [4]. A survey analyzing several facets of vehicle crashes concluded that in 94% of the cases, the driver played a crucial role in the sequence of events leading up to the crash. Some common driver-related errors include inattention, distractions, decision errors, performance errors (overcompensation and poor directional control), and sleep [5].

Autonomous vehicle deployment promises to decrease traffic accidents and traffic-related deaths significantly. It is natural to speculate that a significant part of the 94% of accident cases attributed to human error will disappear once autonomous vehicles outperform the average human driver. Accidents stemming from driver inattention or sleepiness will not happen since the implemented algorithms never reduce their focus on the surroundings or reduce their computational capabilities. Accidents due to excessive speeds will disappear since the vehicles will be programmed to abide by traffic rules and to adapt their velocities according to surrounding traffic or incoming turns. In summary, we can expect a reduction in traffic accidents once autonomous vehicles have driving capabilities better than humans.

Connected and autonomous vehicles will have a substantial impact on the overall economy. According to estimates, an additional 1.2 trillion U.S. dollars in value will be created in the USA alone once autonomous vehicles are a large share of the automotive market [6]. The identified economic effects impact 13 different industries, most seeing gains and a minority of them seeing losses.

The automotive industry is likely to expand its market since autonomous vehicles can serve children, people with disabilities and the elderly, unlike regular vehicles. Software and electronic companies will also grow with the arrival of autonomous vehicles. Currently, software represents 10% of vehicle value. However, this percentage will likely grow to 40% in the future [6]. Truck-Freight transport is estimated to have economic gains of 100 to 500 billion U.S. dollars per year by 2025. These gains are primarily due to the replacement of truck drivers, and associated salary costs, by self-driving systems [6].

Autonomous vehicles can open the way for increased mobility of groups previously devoid of independent ways of transportation. Youth, elderly, and physically impaired people, can now access a convenient and flexible way of transport, increasing fairness and inclusion in society.

The previously mentioned benefits are the motivating force behind many demonstrations and testing projects driven by academia and industry. Countries are also engaging in legislation changes to speed up the introduction of autonomous vehicles. Although raking up millions of miles of driving experience, autonomous vehicles still report multiple accidents. Thus autonomous vehicles still have a long way to go until they reach a maturity level deemed acceptable by societal standards.

1.2 Heavy-duty vehicles

Heavy-duty vehicles (HDVs), which encompass trucks and buses, are a large portion of vehicles driving on roads today. Trucks are responsible for 9.2% of all distance driven on roads nowadays [7]. Regarding inland passenger transport, buses account for 9.2% of passenger-kilometers [8]. HDVs find applications within long-haul, regional and urban delivery, public transportation, construction, and industrial settings.

Trucks with one or more trailers are the preferred choice for long-haulage. These vehicles maximize the total cargo carried by a driver, pushing down transportation costs. A significant share of long-haulage takes place over highways, which are, according to some, one of the first potential use cases of autonomous driving. Highways are simpler environments, where the traffic mainly drives straight and with reasonably constant speeds. However, at high speeds, the consequences of traffic accidents are also more severe.

Furthermore, the long-haulage trucking industry is currently facing a shortage of drivers, and estimates indicate that by 2024 an additional 175 thousand drivers will be needed in the USA. The initial stages of autonomous driving technology will likely allow trucks to drive themselves on the highways during good weather conditions. These initial deployments will increase transport efficiency and reduce the monotony and stress of driving long hours, making the truck driver profession more attractive [9].

Urban logistics and deliveries require HDVs to drive inside cities to deliver cargo to different destinations. Urban scenarios introduce a series of challenges, as the traffic is usually chaotic, can consist of complicated maneuvers, and needs to consider pedestrians. Furthermore, the dimensions of the trucks often make them more complicated to maneuver.

Buses and articulated buses are the backbone of public transportation in many cities. When considering buses, one must deal with the large dimensions of the vehicle, which often has to drive into adjacent lanes or

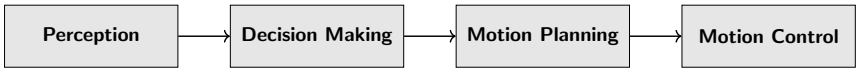


Figure 1.1: Simplified architecture of autonomous vehicle modules.

over curbs to progress along its route. Furthermore, bus driving requires very precise maneuvering, as bus stop approaches often demand the vehicle to stop very close to sidewalks. Articulated buses introduce an additional difficulty as their multiple bodies make the driving task challenging.

Trucks play a central role in many industrial settings, including mines, agricultural lands, construction sites, and freight terminals. Industrial scenarios offer a more controlled environment that minimizes the risk of dealing with pedestrians and other vehicles. The reduced risk makes it a promising application for autonomous driving technologies, as it does not need the same level of technology maturity required in the previously mentioned applications. Even though safer, these environments are often deprived of properly paved roads and can be subject to extreme weather or pollution conditions, introducing novel challenges for the autonomous vehicle.

HDVs differ from passenger vehicles to such an extent that their drivers must partake in specific driving education. The large dimensions, slow dynamics, and multi-body configurations of these vehicles often introduce additional challenges in the driving task. These challenges motivate our work, which focuses on further development and modification of current motion planning approaches, which most often target passenger vehicles [10, 11], to consider the unique characteristics of HDV driving.

1.3 Motion planning for self-driving vehicles

An autonomous vehicle is composed of multiple modules, each implementing different functionalities. This modular approach breaks down the complexity of the driving task into several less complex problems that are easier to tackle. Figure 1.1 shows a simplified system architecture illustrating the modules relevant to this thesis.

Motion planning is an essential capability for an autonomous vehicle. The motion planning module receives information about the environment, corresponding to internal information about the vehicle's state, such as position and speed, and external information, such as lane markings, surrounding vehicles, and obstacles. Furthermore, the module receives a high-level decision, such as *change lane*, *slow down*, or *drive into a given parking spot*. Given this information, the motion planning module is responsible for finding a sequence of actions that achieves the high-level decision.

Figure 1.2: The motion planning task. The vehicle must plan a sequence of states that make it progress along the road while avoiding obstacles.

A motion planning task is often defined by the goal state that the vehicle should arrive at, the obstacles in the environment, the current vehicle position, a performance/objective metric, and the vehicle model. Unlike more traditional robotic approaches where vehicle dynamics are often simplified [12], this thesis considers the underactuated and often complicated dynamics of the system at hand. This thesis is thus concerned with the fields of nonholonomic and kynodynamic planning [13].

The motion planner module is responsible for finding a plan, *i.e.*, a sequence of vehicle states that:

- Arrives at the goal state or makes progress towards it;
- Respects vehicle kinematic and dynamic constraints;
- Avoids collisions with obstacles in the environment;
- Optimizes desirable metrics.

Figure 1.2 illustrates a motion planning situation.

This thesis focuses on HDVs, making the motion planning task significantly hard concerning the second and third requirements listed above. An HDV imposes complex kinematic and dynamic constraints that result from its slow dynamics and sometimes from the complicated vehicle arrangement,

as is the case with tractor-trailers and articulated buses. Collision avoidance also becomes significantly more complicated when considering the large dimensions of HDVs, which often need to drive on narrow roads.

The motion planning task is often impossible to solve optimally, except for particular and limiting scenarios. Thus, current approaches to solving this task often make simplifications and assumptions to keep the problem tractable. These assumptions can consist of simplified vehicle models, a reduced possible solution space, and alternative metrics that roughly approximate the desired objectives. Assumptions about the environment can also help develop algorithms to solve the motion planning task. Different assumptions about the driving environment can be made, depending on whether one considers on-road or off-road scenarios.

A motion control module receives the output of motion planning and is responsible for controlling the vehicle actuators to follow the planned motions as precisely as possible. Since both the vehicle actuators and controllers have performance limitations, the motion planner faces another challenge: planning motions that the underlying control system can follow accurately. The performance of a controller depends on the quality of planned motions. Therefore, improvements in motion planning often result in improvements in the underlying control module.

1.4 Structured and unstructured environments

This thesis makes an important distinction between the type of environments considered for autonomous vehicles. The distinction between structured and unstructured environments allows motion planning algorithms to exploit specific assumptions suitable for each environment.

The most common usages of vehicles that one is familiar with fall within the category of structured scenarios. These scenarios, shown in Figure 1.3, often correspond to on-road driving situations, such as driving on highways and driving in urban environments.

On-road driving environments are usually well structured, *i.e.*, there are clear guidelines on how the vehicle should move. These guidelines are often implicit via lane markings that identify where the vehicle is confined to move and traffic signs that further limit the freedom of the vehicle to make decisions.

This environment drastically reduces the possibilities that a motion planning algorithm needs to consider, limiting it to a very narrow set of possible decisions. However, narrowing down the set of options that the vehicle needs to consider introduces the need for finer granularity within the limited set of possible choices. Furthermore, the interaction with pedestrians and other vehicles presents a far more significant challenge when it comes to decision

(a) Truck used for long-haul of cargo (b) Bus used for public transportation

Figure 1.3: Examples of on-road scenarios (courtesy of Scania CV AB)

making. Thus, self-driving algorithms require complex reasoning capabilities when performing lane changes, driving through pedestrian crossings, or managing intersections without defined priority rules.

A fair share of HDV usage, particularly of truck-trailers, happens in unstructured environments. Unstructured environments often correspond to off-road driving and encompass a large and diverse group of environments, such as mining sites and harbors. Figure 1.4 shows two examples of unstructured environments. These environments are often more controlled and suitable for an earlier deployment of autonomous driving solutions with narrower and less general functionality.

Off-road environments often lack structure and have unclear or not obvious driving patterns. Such environments give the vehicle an extensive range of possible motions, complicating the motion planning task. These environments can sometimes resemble mazes, in which decisions taken at a given step will significantly influence the subsequent performance of the vehicle. Such a decision process has a combinatorial nature, introducing a challenging complexity for motion planning algorithms.

1.5 Research challenges

This thesis addresses motion planning methods for autonomous truck-trailers and buses. We seek computationally efficient methods suitable for online implementation in vehicles with limited computational resources. We also consider the connection between the path planning methods and the underlying control system performance. This thesis highlights the particular aspects that differentiate heavy-duty vehicles (HDVs) from passenger vehicles. Some of these aspects only require slight modifications to state-of-the-art algorithms, but others require significant modifications to existing

(a) Truck carrying ore from a mine

(b) Transporting containers in a harbor

Figure 1.4: Examples of off-road scenarios (courtesy of Scania CV AB)

solutions.

The thesis addresses the following questions:

- How can the slow actuator dynamics of HDVs be dealt with at a planning stage, and what is the impact on controller and vehicle performance?
- Can graph-search methods be improved, concerning oscillation and discretization, using optimal control methods?
- How do common on-road driving behaviors need to be adapted to deal with the specific characteristics of buses and allow maximum maneuverability in urban settings?
- How to ensure that the whole vehicle body drives on the center of the road when considering vehicles with long wheelbases and multiple bodies?
- What practical considerations should be made in the planning module to ensure a successful real vehicle implementation?
- How can a planner take into account the multi-modality, interaction, and decision making aspects of human-driver and autonomous vehicle interaction?

1.6 Thesis outline and contributions

This thesis focuses on both structured (on-road) and unstructured (off-road) application scenarios, with five technical chapters dedicated to solutions developed for them. Additionally, it contains a chapter providing an overview and introduction to motion planning and a chapter with concluding remarks and future work directions.

Introductory material

Chapter 2: Motion Planning

This chapter introduces motion planning, explaining the problem formulation in detail and its challenges. Different approaches to solving the problem are gathered from the literature and explained briefly. The chapter concludes with current state-of-the-art research trends within motion planning.

Unstructured (off-road) environments

Chapter 3: Sharpness Continuous Paths

We introduce sharpness continuous paths as a near-optimal motion planner for obstacle-free environments. Without obstacles, it is possible to develop motion planning methods based on geometric arguments, resulting in computationally fast algorithms. This chapter distinguishes itself from previously developed methods by considering the particularly slow actuator dynamics of HDVs. We show that the method is near length optimal, making it attractive for industrial applications where travel efficiency and fuel consumption are of utmost importance. The contribution of this chapter is the development of a method for the generation of vehicle trajectories that:

- take into account steering actuator magnitude, rate, and acceleration limitations;
- ease the controller task and improve passenger comfort;
- connect arbitrary vehicle configurations;
- have fast computation times.

This chapter is based on the following publication:

> R. Oliveira, P. F. Lima, M. Cirillo, J. Mårtensson and B. Wahlberg, "Trajectory Generation using Sharpness Continuous Dubins-like Paths with Applications in Control of Heavy-Duty Vehicles", 2018 European Control Conference (ECC), Limassol, 2018, pp. 935-940.

Chapter 4: Smooth Path Planning for Unstructured Environments

This chapter presents a planning method that combines a lattice-based planner with the previously presented sharpness continuous paths. The assumption of obstacle-free environments is often too limiting for practical applications, requiring methods such as lattice-based graph search to

be used to take into account obstacles. These methods are powerful when dealing with unstructured scenarios, often characterized by complex, maze-like environments. However, these methods can come with the drawback of oscillatory solution paths. To tackle this problem, we present an extension to lattice graph search that ensures that solution paths are smooth, non-oscillatory, and suitable for HDVs. The contributions of this chapter include:

- A formulation and algorithm for the problem of path optimization that makes use of steering methods and heuristics to achieve real-time performance;
- A novel modification of lattice-based planners that alternates between path planning and path optimization in an interleaved way;
- Simulations and experiments with a heavy-duty truck showing the benefits and applicability of the proposed method.

This chapter is based on the following publication:

R. Oliveira, M. Cirillo, J. Mårtensson and B. Wahlberg, "Combining Lattice-Based Planning and Path Optimization in Autonomous Heavy Duty Vehicle Applications", 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 2090-2097.

Structured (on-road) environments

Chapter 5: Optimization-based On-road Path Planning for Buses

Here, we use numerical optimization to solve the motion planning problem. This type of framework does not rely on the discretization of the planning space, thus not suffering from resolution issues which often affect other approaches. Unlike previous chapters, we consider structured on-road environments, specifically urban driving. We target the challenges of bus driving that arise when considering a large dimension vehicle driving on narrow and sharp roads. The proposed algorithm leverages buses' fundamentally different chassis to maximize their maneuverability. Our contribution is a motion planning algorithm that mimics professional bus driver behavior, resulting in safer driving and increased vehicle maneuverability. The contributions of this chapter include the proposal of a novel path planner that:

- tackles the challenging task of bus driving in urban environments, taking full advantage of the overhangs of buses to sweep over curbs and low height obstacles;

- uses a new approximation technique for the distortions introduced by the road-aligned frame and that affect the vehicle body and obstacles;
- considers the distinct chassis configuration of buses, distinguishing between the overhangs and the wheelbase of the vehicle;
- takes into account three distinct types of surface: obstacle, sweepable, and drivable regions.

This chapter is based on the following publications:

> R. Oliveira, P. F. Lima, G. Collares Pereira, J. Mårtensson and B. Wahlberg, "Path Planning for Autonomous Bus Driving in Highly Constrained Environments", 2019 Intelligent Transportation Systems Conference (ITSC), Auckland, 2019.
>
> P. F. Lima, R. Oliveira, J. Mårtensson and B. Wahlberg, "Minimizing long vehicles overhang exceeding the drivable surface via convex path optimization", 2017 Intelligent Transportation Systems Conference (ITSC), Yokohama, 2017.

Chapter 6: On-road Path Planning for Articulated Vehicles

A significant part of today's goods transportation is done over tractor-trailer vehicles. Tractor-trailer vehicles are composed of two independent, but connected, bodies, the tractor in front and the trailer behind. A common challenge when driving a tractor-trailer is the off-tracking effect, which corresponds to the trailer cutting through turns when the tractor is turning. To successfully drive an autonomous tractor-trailer, one needs to introduce new vehicle models that capture the vehicle dynamics and obstacle constraints that ensure collision avoidance for both bodies. Furthermore, novel optimization criteria must be introduced to properly drive the tractor while at the same time taking into account, and attenuating, the trailer off-tracking effect. The contributions of this chapter are:

- proposal and evaluation of different optimization criteria suitable for on-road path planning of articulated vehicles;
- implementation of a sequential quadratic programming (SQP) solver, which ensures smooth driving while guaranteeing precise obstacle avoidance;
- a sequential method for computing the off-tracking, as well as approximate partial derivatives, of each point of the vehicle bodies suitable for numerical optimization approaches;

- simulation results that show the proposed path planner's ability to solve complicated on-road planning scenarios while considering the most challenging tractor-trailer dimensions.

This chapter is based on the following publication:

> R. Oliveira, O. Ljungqvist, P. F. Lima and B. Wahlberg, "Optimization-Based On-Road Path Planning for Articulated Vehicles", 21st IFAC World Congress, Berlin, 2020.

Chapter 7: On-road Path Planning for Long and Multi-Body Vehicles

The challenges of road driving impact both buses as well as tractor-trailer vehicles. In the case of buses, the long wheelbase introduces a conflict between centering the rear axle vehicle or centering the front axle. The same conflict arises in the tractor-trailer case as a trade-off between centering the tractor or centering the trailer. This chapter presents a framework to design optimization objectives that optimally trade-off between these conflicting objectives. Simulation results show that the proposed design strategy results in planned paths that considerably improve the behavior of both buses and tractor-trailer vehicles by keeping the whole vehicle body in the center of the lane. The contributions of this chapter include:

- geometric derivation of optimal driving objectives, focusing on centering the area swept by the vehicles, suitable for online computation;
- development of a unified framework targeting both long vehicles, such as buses, as well as multi-body vehicles, such as tractor-trailers;
- simulation results showing the proposed planner's ability to solve complicated on-road planning scenarios while considering the most challenging vehicle dimensions.

This chapter is based on the following publication:

> R. Oliveira, O. Ljungqvist, P. F. Lima and B. Wahlberg, "A Geometric Approach to On-road Motion Planning for Long and Multi-Body Heavy-Duty Vehicles", 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, 2020.

Chapter 8: On-road Path Planning Experimental Results

To further validate the algorithms introduced in previous chapters, we implement the proposed on-road motion planner on a Scania prototype autonomous bus. We describe the practical considerations and implementation details required for successful real-life experiments. Extensive experimental

results in a test track show the method's suitability for real-life deployment. The planner runs in real-time on an industrial computing unit, while communicating and sharing computational resources with other modules of the autonomous software stack. The planned maneuvers are smooth and comfortable, even in the presence of noisy and irregular map and road information. Furthermore, the executed paths have low tracking errors, indicating that the planned paths can be feasibly followed by the underlying controllers actuating the vehicle. The experiments also provide insight into a common challenge associated with bus stop maneuvers, which led to developing a novel collision checking method that explicitly considers a bus's front wheels. The contributions of this chapter include:

- extensions and modifications to the previously developed path planner that lead to successful real-life implementation;
- proposal of the novel concept of wheel-aware planning that explicitly considers that the front wheels of the vehicle move relative to the chassis, possibly protruding and colliding with curbs;
- experimental validation of the proposed method by navigating an autonomous bus on urban-like roads;
- demonstration of the benefits of the proposed planner through practical experiments in challenging scenarios.

This chapter is based on the following publication:

> R. Oliveira, P. F. Lima, M. Cirillo and B. Wahlberg, "Autonomous Bus Driving: A Novel Motion-Planning Approach", IEEE Vehicular Technology Magazine, vol. 16, no. 3, pp. 29-37, 2021.

Chapter 9: Decision Making using Branch MPC

One of the toughest challenges affecting autonomous vehicles nowadays is successfully interacting with human drivers on the road. This challenge applies to both heavy-duty and passenger vehicles. However, it becomes more pronounced in the heavy-duty case due to its slow dynamics. The difficulty arises from three challenging aspects of human driving: that human drivers are 1) multi-modal; 2) interacting with the autonomous vehicle; 3) actively making decisions based on the current state of the surrounding environment. This chapter proposes a motion planning framework based on Branch Model Predictive Control that tackles these aspects. Branch Model Predictive Control offers an intuitive way of dealing with the multi-modality using a scenario tree approach. The interaction aspects can be dealt with by

considering an adaption of the Intelligent Driver Model suitable for numerical optimization. Finally, one can use decision making models of humans, as taken from user studies, to model the frequency of different human decisions and, in turn, different probabilities of future outcomes. We present simulation results in various scenarios, showing the advantages of the proposed method. The contributions of this chapter include:

- addressing the multi-modality of human drivers by considering multiple future outcomes associated with different decisions taken by the human driver;
- considering the interactive nature of humans by modeling them as reactive agents impacted by the actions of the autonomous vehicle;
- approximating the decision making process of human drivers by considering a model developed in neuroscience studies with human drivers as subjects.

This chapter is based on recent research results currently being prepared for submission to a scientific journal.

Concluding remarks

Chapter 10: Conclusions and Future Work

This chapter contains conclusions regarding the developed work. Furthermore, it discusses promising directions for further development of the presented work. We also present emerging research topics that we deem relevant in the field of autonomous heavy-duty vehicles.

1.7 Author's contributions

The thesis author is the main contributor to the work in [14]. The co-authors of [14] participated in technical discussions and supervision.

The thesis author is the main contributor to the work presented in [15]. The developed solution builds upon the lattice planner developed by the second author, which contributed significantly to the technical and practical implementation of the solution. The remaining co-authors have been involved in supervision.

The thesis author is a co-author of the work in [16], being involved in technical discussions and in studying motivating examples for the developed solution. The thesis author eventually developed significant improvements to [16], which resulted in being the main contributor to [17]. The second and third authors of [17] participated in technical and implementation discussions. The remaining co-authors have been involved in supervision.

The thesis author is the main contributor to the works [18, 19]. The second author has made significant contributions to the formulation and development of the proposed methods. The remaining co-authors have been involved in technical discussions and supervision.

The thesis author is the main contributor to [20]. The remaining co-authors have been involved in technical discussions and supervision.

Chapter 2

Motion Planning

Motion planning has been the focus of research for many decades, and plays an important role in many autonomous systems that control physical entities, be it self-driving vehicles, autonomous quadcopters, or humanoid robots. In a nutshell, motion planning is responsible for finding a sequence of future actions that an autonomous system must take in order to achieve its goals.

This thesis considers the systems to be self-driving vehicles, and the goals to be short term objectives such as arrive at a goal position, or overtake the vehicle ahead. Due to the complex vehicle dynamics of the vehicles considered in this thesis, the problems hereby addressed fall within the fields of nonholonomic and kinodynamic planning [13].

Section 2.1 presents a definition for the motion planning problem. It enumerates the vehicle constraints and limitations that must be addressed, as well as common formulations for the goal and objectives of motion planning. Section 2.2 illustrates some of the challenges of motion planning for autonomous vehicles, drawing attention to the fact that some of these challenges become even more problematic when HDVs are considered.

We start by introducing a specific class of motion planning methods, called steering methods, in Section 2.3. Steering methods are tasked with finding motions that connect an initial and final vehicle state, with the assumption of an obstacle-free environment. These methods are often used as building blocks of more advanced motion planners.

The concept of lattice-based motion planners is presented in Section 2.4. These type of planners are suitable for environments lacking structure (often off-road scenarios), which often require combinatorial-like methods in order to find a good quality solution. Lattice-based planners have been widely used in numerous self-driving applications.

Another technique that can be used to solve motion planning problems

is continuous optimization methods, presented in Section 2.5. This type of implementation has been gaining popularity due to recent advances in optimization techniques and the ever increasing computation power available on-board autonomous vehicles. These solutions are mostly targeted to structured environments (often on-road scenarios), where the the structure effectively removes the combinatorial nature of the planning problem.

Autonomous vehicle technology is becoming more mature, and self-driving vehicles are being deployed in more complex and traffic heavy environment. This requires the initial motion planning problem to be reformulated so as to take into account the challenges associated with driving in the presence of human traffic participants. Section 2.6 introduces a reformulation of the motion planning problem that takes into account interactions with humans, and presents an overview of the proposed solutions developed to deal with this additional challenging aspect of driving.

2.1 Problem formulation

In this thesis we deal with motion planning for autonomous car-like vehicles. To define the motion planning problem it is necessary to first introduce some of its components, namely the vehicle model, the obstacle space, and the objective function.

Vehicle model

The vehicle model is a mathematical representation of the evolution of vehicle states. We first define a vehicle state vector $q \in \mathbb{R}^n$ containing the relevant n states of the vehicle, which can correspond to vehicle coordinates, velocities, and internal state information.

The simplest, or lowest dimension, vehicle state that is useful for our motion planning purposes, corresponds to the vehicle pose $q = (x, y, \theta)$. Here, x and y correspond to the Cartesian coordinates of the vehicle rear axle center, and θ is the vehicle heading, corresponding to the angle between the forward direction of the vehicle and the X axis, as shown in Figure 2.1.

A natural extension to the vehicle pose corresponds in augmenting the state vector with an extra dimension corresponding to the vehicle curvature κ, so that $q = (x, y, \theta, \kappa)$ The vehicle curvature κ is directly related to its steering wheel angle ϕ.

A vehicle can move from one state to another by applying command inputs u during a defined amount of time. Considering the simplest case, the vector u is 2-dimensional with a component corresponding to the linear velocity of the vehicle v, and an angular rate w. The linear velocity can be

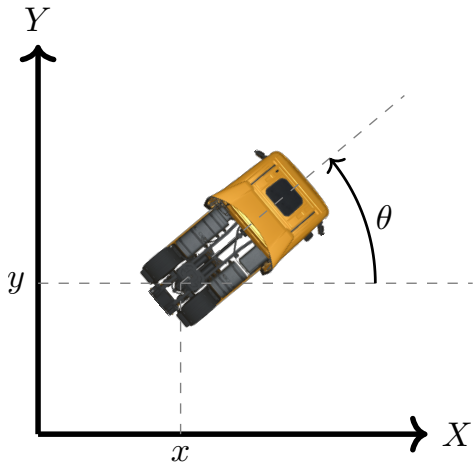


Figure 2.1: The vehicle pose in the Cartesian coordinate system. x and y correspond to the position of the rear axle center, and θ is the heading of the vehicle.

controlled via the vehicle engine, whereas the angular rate is often a result of both the vehicle velocity and the current steering wheel angle.

A vehicle model can now be introduced as:

$$\dot{q}(t) = f(q(t), u(t)). \tag{2.1}$$

This equation defines how the vehicle state evolves over time ($\dot{q} = dq/dt$), given the current vehicle state q, and the applied command inputs u.

Vehicle models can be made arbitrarily complex, in an attempt to more accurately capture the true physical vehicle dynamics. Vehicle models are often divided into two major groups, kinematic models and dynamical models. Kinematic models disregard the forces acting on the vehicle, resulting in simpler equations of movement. When considering vehicles, kinematic models are only accurate when the slip angle is neglectable, *i.e.*, at low speeds or low yaw rates. In order to model more complex phenomena, that can arise from high speed driving or harsh environmental situations dynamic models are introduced. By considering the forces acting on the vehicles, these models can more accurately express the motion of vehicles in the presence of drift, low friction conditions (such as rain or snow), and cargo load.

Previously, when defining the vehicle pose, we made use of the Cartesian frame, however there are several applications when it is beneficial to consider alternative vehicle models using different coordinate systems. An example is the road-aligned model illustrated in Figure 2.2, which makes use of a curvi-

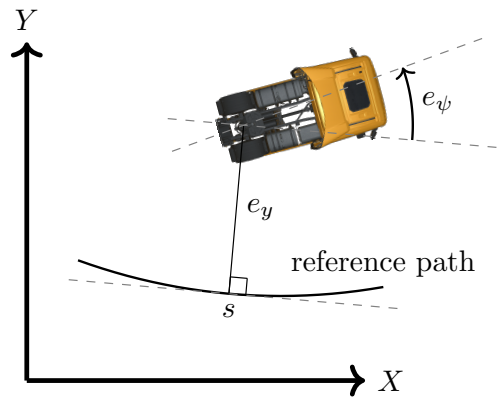


Figure 2.2: Vehicle pose in the road-aligned frame.

linear coordinate system [21]. In this model, the vehicle pose is defined with respect to a frame moving along a reference path. Alternative formulations, such as the road-aligned model, can have many practical motivations, such as faster computational times or avoiding ill-defined behavior of the model. This thesis uses both the Cartesian-based vehicle model (in Chapter 3 and Chapter 4) and the road-aligned vehicle model (in Chapter 5).

Configuration and obstacle space

The configuration space X corresponds to the possible states achievable by the vehicle. Assuming the vehicle state $q \in \mathbb{R}^n$, the configuration space corresponds to the manifold of the possible vehicle states [13]. Often times, the vehicles are affected by constraints of the form

$$g(q, \dot{q}) = 0, \tag{2.2}$$

which originate from the vehicle model Equation (2.1). In car-like systems, these constraints are often affecting the achievable velocities $\dot{q}$, and are referred to as nonholonomic constraints.

Motion planners are often tasked with finding a solution which is collision-free. In the vehicle environment there are often obstacles corresponding to multiple entities existing in the vehicle surroundings. Obstacles can consist of static obstacles, such as parked vehicles by the side of the road, and dynamic obstacles, such as other vehicles and pedestrians.

From the set of obstacles one can create the obstacle space X_{obs}. The obstacle space X_{obs} can be formally defined as the set of all vehicle states q that collide or intersect with the entities that we wish to encompass as

obstacles. For dynamic environments, the obstacle space depends on time: $X_{\text{obs}}(t)$ [10]. We note that the obstacle space X_{obs} is always larger than the obstacle itself, since the vehicle states q have an associated body width and length that must also be outside of the obstacle.

Once the obstacle region is defined, one can define the complementary free space region $X_{\text{free}} = X \backslash X_{\text{obs}}$. The free space region X_{free} corresponds to all the vehicle states q that are collision-free. Similarly, if in the presence of a dynamic environment, one has $X_{\text{free}}(t)$. To guarantee that the vehicle is collision-free, the following condition must be verified:

$$q(t) \in X_{\text{free}}(t). \tag{2.3}$$

Objective function

The output of a motion planner is a sequence of vehicle states:

$$(q(s), u(s)), s \in [0, S], \tag{2.4}$$

where s corresponds to the distance along the path, and S is the path length. If there is a timing law associated to the distance along the path, such that $s(t)$, then the sequence is called a trajectory, otherwise it is called a path.

One is usually interested in optimizing the motion of vehicles with respect to multiple metrics or criteria. Taking the example of comfort, one could define objective $J_{\text{comfort}}(q(s), u(s))$ that measures the comfort of a ride along the planned motion $q(s)$. This measure could correspond to the sum of lateral and longitudinal accelerations of a vehicle driving along the planed motion.

Besides comfort, one could optimize with respect to the time it takes to follow the path, or to the length of the path (both are often correlated), in the form of an objective function $J_{\text{travel time}}(q(s), u(s))$. To increase the efficiency of vehicles, an objective function $J_{\text{fuel}}(q(s), u(s))$ can be designed that measures the consumption of fuel, and consequently minimizes it. To increase safety, one could measure the distance of the vehicle to obstacles, in the form of a function $J_{\text{safe}}(q(s), u(s))$ that would guide the motion planner to prefer to choose paths that have bigger clearance to obstacles, and that are in turn safer.

There is a multitude of objectives that a motion planner can optimize for. Sometimes these criteria can significantly complicate the motion planning problem, as they might be expensive to evaluate, or introduce challenging problem structures. Furthermore, one is usually interested in multiple objectives at the same time. This can be easily achieved by creating a new

criteria corresponding to the sum of other criteria, as follows:

$$J\left(q(s), u(s)\right) = \sum_{i=1}^{M} \omega_i J_{\mathrm{i}}\left(q(s), u(s)\right) \tag{2.5}$$

where $J_{\mathrm{i}}\left(q(s), u(s)\right)$ corresponds to the i-th objective we are interested in minimizing and ω_i is a weight determining the relative importance of this objective against the other $i-1$ objectives.

Problems can arise when different criteria or goals are contradictory in nature. As an example, imagine that we are interested in minimizing both the time it takes to perform a motion, and the fuel consumption. In order for the vehicle to optimize the time it takes to travel the path, it has to accelerate as much as it can, however doing this as a negative effect on fuel consumption. Developers working on motion planning often have to deal with these types of problems, and must decide on appropriate trade-offs that achieve satisfying results in both criteria. This type of trade-off worsens as the number of objectives, and their associated weight ω_i, increases. A high number of weights results in practitioners often having to spend a considerable amount of time tuning the system, *i.e.* finding an acceptable combination of weight values that produces a good performance of the system.

Motion planning formulation

With all components in place, we are now able to define the motion planning problem as follows. We follow a formulation similar to that of [10]:

$$\begin{aligned} \underset{u(t)}{\text{minimize}} \quad & J\left(q(s), u(s)\right) \\ \text{subject to} \quad & \dot{q}(t) = f(q(t), u(t)), \\ & q(0) = q_0, \\ & q(t_G) = q_{\mathrm{G}}, \\ & q(t) \in X_{\mathrm{free}}(t). \end{aligned} \tag{2.6}$$

The first constraint $\dot{q}(t) = f(q(t), u(t))$ corresponds to the vehicle model. It is used to enforce the planned solutions to respect the vehicle kinematic and/or dynamic constraints that are encoded in the vehicle model f. The solution must also start from the initial state q_0, usually corresponding to the current state of the vehicle, furthermore it should end a a goal state q_{G}. It is also required that the planned motion is collision-free in its entirety $q(t) \in X_{\mathrm{free}}(t)$. The objective function $J(q(s), u(s))$ is a user defined metric measuring the quality of a planned motion. Lower values of J correspond to motions with better properties.

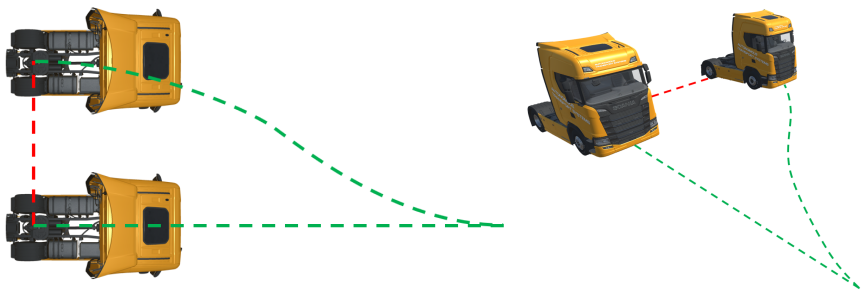


Figure 2.3: A vehicle can only follow specific motion patterns. To move sideways, the red path is unfeasible. A more complex, but feasible to follow path, is shown in green. Bird's-eye (left) and perspective (right) views.

The following section presents some of the aspects that make Equation (2.6) hard to solve.

2.2 Challenges of motion planning

This section details some of the challenges associated with finding a solution to Equation (2.6). They are related to the difficulty in finding paths respecting the vehicle dynamics, the possible complex maze-like nature of the environments, and the narrow and low-clearance roads often present in urban driving.

The vehicle nonholonomic constraints

Consider a vehicle whose goal is to arrive at a state which is laterally displaced from its current position, as shown in Figure 2.3. It is impossible for the vehicle to just move sideways into the goal state, illustrated by the red path, due to the vehicle nonholonomic constraints in Equation (2.2). A possible solution path respecting the vehicle dynamics, and thus the nonholonomic constraints, is shown in green in Figure 2.3.

This example illustrates the difficulty involved in finding a solution path for these vehicles. The constraint $\dot{q}(t) = f(q(t), u(t))$ in Equation (2.6) that encodes the vehicle model, introduces constraints on the possible velocities $\dot{q}$ that the vehicle can achieve. These constraints are usually referred to as nonholonomic constraints, and they make the motion problem in Equation (2.6) challenging to solve.

In Section 2.3 we present some methods that are able to optimally solve the motion planning problem, however under very limiting assumptions.

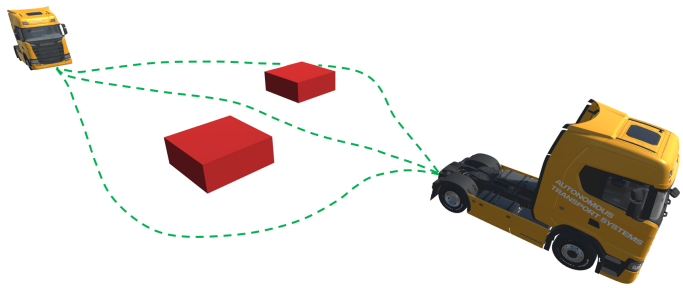


Figure 2.4: The obstacle space is non-convex making motion planning combinatorial in nature.

Furthermore, Chapter 3 presents one such method developed specifically for HDVs. For most practical applications, algorithms that find the optimal solution to Equation (2.6) are not available, and algorithms that find sub-optimal solutions are used instead.

Complex obstacle environments

Many motion planning applications target the task of driving in unstructured environments. These type of applications are often characterized by a large number of obstacles, forming maze like environments. Due to the existence of multiple obstacles, the free-space is in general non-convex, as shown in Figure 2.4. The non-convex nature of the environment often requires solution methods that are combinatorial in nature [22].

In order to keep combinatorial methods feasible to solve, sampling, or discretization, of the configuration space is often used. The resulting motion planning solutions are, most likely, not the true optimal solution, but instead a sub-optimal one, due to the reduced configuration space that is considered. In some cases, the sub-optimality of solutions can result in oscillatory behavior of the vehicle [15], greatly degrading the performance of the autonomous vehicle. Chapter 4 presents in detail one such scenario where this occurs, and proposes a solution to it.

Narrow environments

Certain driving environments might require the vehicle to drive quite close to obstacles. This type of situation is particularly common when large dimension HDVs are driving in urban environments and narrow roads. In Figure 2.5 it is possible to see a bus which is driving through a packed

Figure 2.5: A bus driving in a narrow road is forced to pass very close to other vehicles (courtesy of Scania CV AB).

traffic situation, a task which is hard not only for motion planners but also for drivers.

Two main reasons can be attributed as the cause of struggle for motion planners in narrow environments: discretization and sampling of the state space and conservative collision checking.

Motion planners that discretize the state space, *e.g.*, [23], require a very high resolution to be able to find solution paths that are collision-free. Increasing the resolution however, comes at the cost of increased computational times, which can make the algorithms unsuitable for online implementation. Methods which randomly sample the state space, *e.g.*, [24], are still able to find solutions in narrow environments. However the probability of sampling collision-free paths in narrow environments is very low, which results in excessively high computational times.

Collision checking refers to the process of checking if a tentative vehicle state (often being part of a tentative solution path) is in collision. To keep the collision checking process computationally cheap, approximate collision checking methods are used. These approximations are usually made conservative to guarantee safety of the solution path. Problems arise however, when the conservativeness of the approximation is in the same order of magnitude as the clearance to obstacles, which can be arbitrarily small in some cases.

This challenge is tackled in Chapter 5, where a planner targeted for buses driving in narrow environments is proposed. The planner does not suffer from the discretization of the state space nor does it rely on conservative collision checking methods.

2.3 Steering methods

Steering methods are a subset of the motion planning problem in Equation (2.6) where obstacles are disregarded, *i.e.*, $X_{\text{obs}} = \varnothing \implies X_{\text{free}} = X$. Even without obstacles, the solution to this new problem can be hard to find due to the boundary constraints, defining the start and final states of the motion, and the nonholonomic constraints, which need to be respected when generating the motion. The problem becomes even harder, if one adds an optimization objective to be minimized.

Even though steering methods ignore obstacles, they are of interest for motion planning, as they are often used as building blocks for more complex planners that take into account obstacles. Thus, the performance of complex motion planners often relies on the quality of the underlying steering methods. In this section, we list three different classes of steering methods, Dubins-inspired methods, interpolation methods, and optimization-based methods.

Dubins-inspired methods

It was proven in [25] that the minimum length paths for the simple car-like model, assuming a unit velocity and a minimum turning radius, are composed of straight line segments and arc circles. One such path is shown in Figure 2.6. Finding these paths requires very few computations, making them attractive for practical implementations. The drawback of this method is related to the too simplified car-like model assumed, where the vehicle is limited to moving forward and can either turn at the maximum turning radius or drive straight.

The work in [26] introduces Reeds-Shepp paths, that improve upon Dubins paths by allowing the vehicle to change its direction of motion, *i.e.*, the vehicle can drive both forwards and backwards. Reeds-Shepp paths are proven to be of minimal length for the considered vehicle model. Furthermore, they increase the maneuverability of the vehicle, when compared to its Dubins counterpart. The solution paths are also composed by a set of straight line segments and arc circles, but the vehicle is now able to change direction, which can be seen by the cusps in Figure 2.6.

Following works address more complicated vehicle models. In [27], the vehicle model is made more complex, by now assuming that the steering wheel angle of the vehicle moves continuously. This results in paths that are smoother and easier to follow, but loses optimality guarantees with respect to the length of the path. The work in [28] takes into account that vehicles can turn their steering wheels when stopped, adding extra maneuverability to the vehicle. Futhermore, [29] ensures that the steering wheel angle of the vehicle is continuously differentiable, increasing the smoothness of the path.

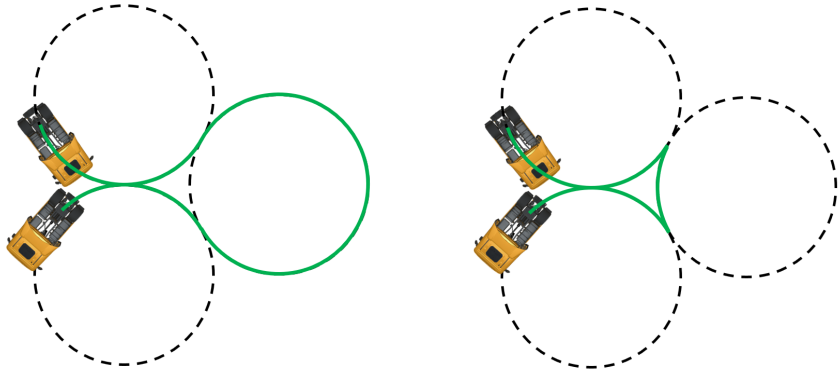


Figure 2.6: A Dubins path [25] on the left and a Reeds-Shepp path [26] on the right. The Reeds-Shepp path includes a reversing motion between the two cusps of the path.

Chapter 5 extends Dubins like paths in order to take into account steering actuator limitations, in the form of maximum magnitude, rate, and acceleration. This extension comes at the cost of losing optimality guarantees with respect to minimum length, similarly to the previously mentioned works [27–29]. However, it is shown experimentally that the path length converges to the optimal Dubins path length, as the steering actuator limitations are relaxed. The loss of optimality guarantees is a small drawback when compared to the benefit of better controller performance and a more comfortable drive.

Interpolation methods

The work in [30] introduces circular arcs and cubic curvature paths to connect two vehicle poses. Circular arcs are shown to be the solution paths when minimizing the integral of the squared curvature of the path, whereas cubic curvature paths are the solution paths which minimize the integral of the squared derivative curvature of the path. These paths can be found using an iterative method based on the Newton-Raphson algorithm. Furthermore, they are result in smoother driving, when compared to methods based on clothoids.

Lane change trajectories can be generated by combining clothoids, circular arcs, and line segments, as presented in [31]. The planned trajectories are designed for emergency maneuver situations and take into account the limited friction between the tires and the road.

The work in [32] uses both cubic and quartic curvature polynomials to

connect sampled endpoints. Quartic polynomials are used when connecting the vehicle pose, since they allow for continuity of the curvature rate, and thus smooth paths, even in the case of frequent re-planning. For other endpoint connections cubic polynomials are used since they result in smaller computation times.

Polynomials of higher order, such as quintic polynomials, can also be used. In [33], the authors make use of quintic polynomials in the Frenet-Serret frame to generate trajectories for high speed driving. The Frenet-Serret frame is a frame that moves along a curve, which in this case, corresponds to the centerline of the road. The trajectories obtained using quintic polynomials minimize the jerk of the vehicle, which translates into ease of driving and a comfortable ride. Furthermore, the optimization objective is defined so as to ensure compliance with Bellman's principle of optimality. This guarantees temporal consistency of the planner, ensuring that subsequent re-planning instances do not deviate from previously found solutions. However, this property only holds for the unconstrained (obstacle-free) case.

Smooth path planning for mining vehicles is targeted in [34]. Using B-Splines of degree 4, the authors plan paths with minimum curvature variation. These paths reduce the jerk of the vehicle, allowing it to run at higher speeds without damaging steering gear and mechanics of the vehicles.

2.4 Lattice-based motion planning

This section introduces one of the most successful approaches to motion planning under differential constraints, as those imposed by the vehicle model. This family of methods is based on the discretization of the state space into a lattice, which then allows for powerful graph search algorithms to be used. Lattice-based motion planners, hereby referred to as lattice planners, have been extensively used in the literature, mostly for off-road scenarios, but also with some applications for on-road scenarios.

Search space

The first step in a lattice planner is the creation of the search space. One of the challenges in creating the search space for autonomous vehicles emerges from the kinodynamic constraints of the considered vehicles. Kinodynamic constraints correspond to both the kinematic, and the dynamic constraints of the vehicle. Early motion planning applications found solution paths by discretizing the Cartesian space into a uniform grid, and using graph search techniques on the resulting discretization. A planned path could then look like the one pictured in Figure 2.7. These types of paths are suitable for early robotic applications, where the considered robots were able to drive

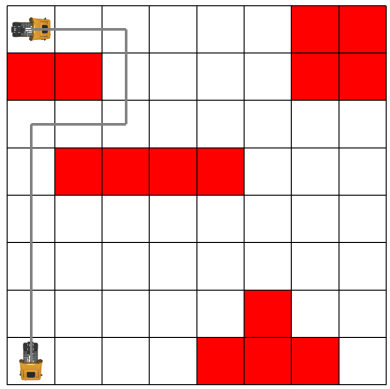


Figure 2.7: Planning on a 4-connected grid. The planned path is impossible to perform by a car-like vehicle, motivating the development of lattice-based motion planners.

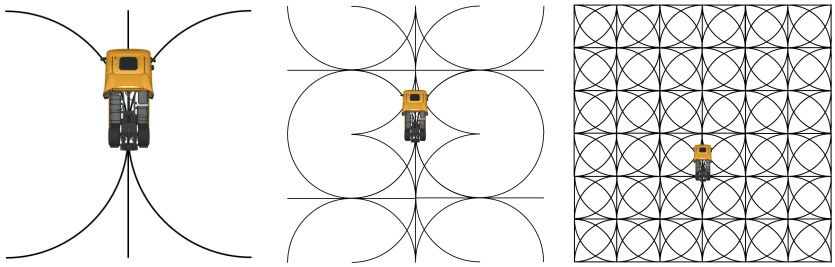
straight and turn on the spot. However with car-like vehicles, and other complex systems, the kinodynamic constraints make it impossible for such paths to be executed, and as such, invalidate these approaches.

Instead of discretizing the space according to a uniformly spaced grid, lattice planners create a special discretization which, by design, follows the kinodynamic constraints of vehicles, *i.e.*, it respects their motion constraints. This discretization is obtained by sampling the state space in a way, that it can form a self-repeating tile, *i.e.*, a lattice, and where connections between sampled states respect the vehicle model.

We start by assuming a vehicle that only has three possible motions, turn left, right and drive straight, all in the forward direction. We refer to these motions as motion primitives, as any solution path consists of a combination of these. Following these motions results in new vehicle states, as shown in Figure 2.8a.

This toy example assumes that the vehicle state is given by $q = (x, y, \theta)$, and that the vehicle can instantaneously change between left turning, right turning and straight driving. This type of vehicle system is invariant to translation and rotation, meaning that the same motion primitives can be applied again to the new vehicle states, as illustrated in Figure 2.8b. If this process is repeated continuously, the motion primitives, together with the intermediate vehicle states, eventually span the whole search space, as shown in Figure 2.8. This process creates a regular and self-repeating pattern of the motion primitives, *i.e.*, a state lattice.

We note that state lattices can also be constructed for more complicated vehicle systems [35, 36]. The approach is usually to define a discretization of



(a) One level of motion primitives.

(b) Two levels of motion primitives.

(c) The resultant state lattice.

Figure 2.8: A lattice can be constructed by successively applying motion primitives of the vehicle. Figure inspired by [23].

the configuration space, and then compute motion primitives that connect these discretized vehicle states. This is the opposite of the toy example presented before, where motion primitives are first defined, and the discretized vehicle states are result from applying them. The computation of motion primitives connecting the discretized states is a motion planning problem in itself. These motion primitives can be computed resorting to steering methods as the ones discussed in Section 2.3, or using numerical optimization approaches which are introduced in Section 2.5.

The lattice can be represented as a graph $\mathcal{G} = \langle V, E \rangle$, where the vehicle states correspond to vertices V, and the motion primitives are edges E between these vertices. In order to plan a path, one is then tasked with finding the shortest path between the graph vertices corresponding to the current and goal vehicle states.

The shortest path in the graph between two nodes is computed by finding the sequence of edges with the lowest cumulative cost connecting them. Each edge has a cost that can reflect different motion planning goals. In the simplest case, the edge cost is equal to the corresponding motion primitive path length, which results in planned motions which are of minimum length. Other costs besides length can be introduced, such as curvature and direction of motion. By including a curvature cost, the planned solutions tend to be smoother. Including the direction of motion results in the vehicle preferring to drive forwards instead of backwards, a common behavior followed by human drivers.

Collision checking

The planner must also take into account obstacles in order to guarantee that the planned solutions are collision free. To achieve this, it is necessary to

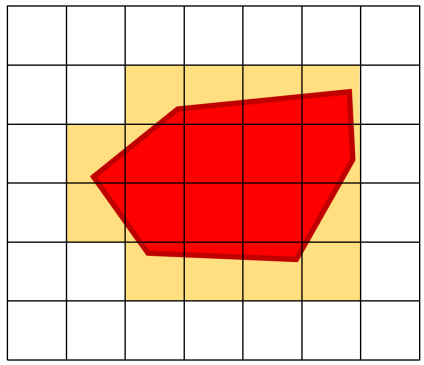


Figure 2.9: The occupancy grid for an environment with an obstacle shown in red. Orange cells are cells classified as obstacles, where free cells are white. The occupancy grid discretization inflates the obstacle.

have collision checking procedures that are able to check if a given vehicle state (vertex) or a motion primitive (edge) are in collision. Vertices and edges that are in collision are then removed from the search graph, ensuring that possible planned paths are collision-free.

Collision checking can be done in multiple ways, and in this section, we focus on occupancy grid-based collision checking. Occupancy grid-based collision checking is used later on in Chapter 4. A different type of collision checking is introduced in Section 2.5 and is used extensively in Chapter 5.

An environment can be represented as an occupancy grid by discretizing the environment into cells. Each cell can then be classified as obstacle, in case a cell contains an obstacle in it, or as free otherwise. An example of an occupancy grid is shown in Figure 2.9. It can be seen that the occupancy grid inflates the obstacles due to the inherent discretization.

To check if a vehicle state is in collision, the occupancy of the vehicle in the occupancy grid must be computed. Then, the vehicle occupancy is checked against the obstacle occupancy, and in case there is an overlap, *i.e.*, at least one cell is classified as being obstacle and also as being vehicle, the the vehicle state is in collision. Figure 2.10 shows the collision checking procedure, for two cases. In the first case, the vehicle is correctly deemed as being collision-free, since no vehicle cells overlap the obstacle cells. However in the second case, the grid discretization causes a vehicle state to be wrongly classified as in collision. This problem can be solved by increasing the grid resolution, however that comes with an increased computational cost. Since collision checking is often one of the most expensive parts of motion planning algorithms, it is important to keep its computational effi-

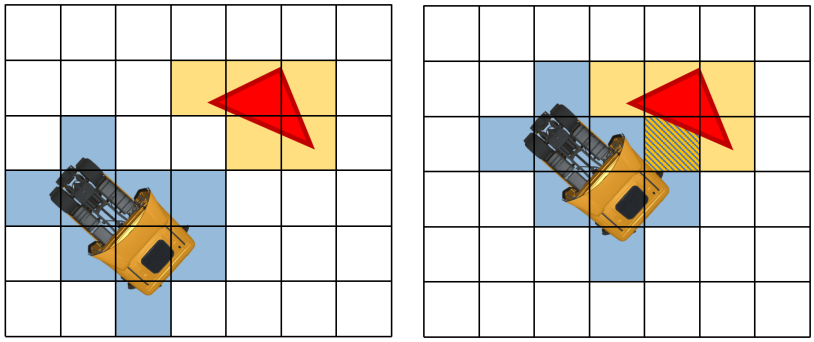


Figure 2.10: Collision checking using occupancy grid. Left: The occupancy grid is successful in identifying that the vehicle state is collision-free. Right: The discretization causes a conservative collision checking and wrongly classifies the vehicle state as in collision.

ciency high.

Lattice planners allow the collision checking computational times to be reduced by allowing the precomputation of path swaths. A path swath is the occupancy grid for a certain motion primitive. Since a motion primitive corresponds to a sequence of infinite vehicle states, computing its occupancy grid is a trade-off between fidelity of the discretization of the path as being a sequence of states, and the computational time required for computing.

In the case of the lattice, path swaths can be computed offline, stored and used online when needed. This is permitted since a limited number of motion primitives is used, and due to translational and rotational invariance principles that applie to certain car-like vehicle models. The invariance property is not trivial to obtain, specially when considering truck and trailer systems, however some specific solutions can be developed for these systems [36].

Graph search techniques

Once the lattice is constructed, the corresponding graph is to be searched in order to find a solution path. Any graph search algorithm can be used to find a solution, however, depending on the intended application, some offer better performance. Here we list some of the graph search algorithms that have been successfully used in autonomous driving applications.

A* [37] is one of the most used graph search algorithms. It evaluates nodes, and decides which parts of the graph to explore by computing an evaluation function $f(n) = g(n) + h(n)$. The cost function $g(n)$ corresponds

to the path cost from the start node to node n, whereas $h(n)$ is an heuristic function that estimates the cost of getting from node n to the goal node.

An heuristic function is defined to be admissible if $h(n) \leq h*(n)$, where $h*(n)$ is the optimal cost from node n to the goal node. In essence admissibility consists in the heuristic never overestimating the cost of reaching the goal. Admissibility of the heuristic plays an important role in the A* search, as it guarantees that the search finds the optimal solution to the goal.

Besides being admissible, an heuristic function $h(n)$ can also be consistent. Consistency indicates that $h(n)$ follows the triangular inequality $h(n) \leq c(n, a, n') + h(n')$, that is, for every successor n' of n obtained by applying action a, the estimated cost to arrive at the goal from n is not greater than the estimated cost to arrive at the goal from n' plus the the cost of getting from n to n'. Consistency of the heuristic plays an important role in the A* search. If A* is provided with a consistent heuristic, then the returned solution is optimal, *i.e.*, it is the lowest cost path. Furthermore, a consistent heuristic also guarantees that A* is optimally efficient, *i.e.*, it does not expand more nodes than other search methods using the same heuristic information [12].

Anytime Repairing A* (ARA*) extends upon A*, and is targeted for planning scenarios in which computational time is scarce [38]. ARA* inflates the heuristic function, using $\epsilon h(n)$ with $\epsilon > 1$, which often results in significantly less node expansions, and therefore faster computational times. The drawback however, is that the heuristic might no longer be consistent, and the solution paths are no longer optimal. The solutions can then be made progressively better, by decreasing the inflation factor ϵ and redoing the search in an efficient way by making use of previous search results. The process is iteratively repeated, as long as time allows, or until $\epsilon = 1$, and the solution becomes optimal, *i.e.*, with the same cost of a solution found by A* for the same problem instance [38].

Time-Bounded A* (TBA*) is a variant of A* that targets the case of real-time environments [39]. In these type of environments, one often has a limited planning time, after which a solution path is expected to be provided. After every planning cycle TBA* is able to provide a temporary path solution, which can be further enhanced in following planning cycles, in an efficient way. In its original implementation, TBA* does not support vehicles with momentum, motivating the development of new extensions targeting these systems [35].

D* Lite [40] targets applications in which the environment is constantly changing, such as robotic applications where new sensor information is obtained as the robot progresses along its path. D* Lite focuses on efficiently recomputing shortest paths, without having to start the whole planning

procedure from scratch.

Relevant works

The concept of lattice planning is introduced in [23], where the authors study the mechanisms of how to best construct the lattices so that optimality and completeness are maximized, while complexity is kept to a minimum. The proposed solution is tested on prototype rovers, and shown to be suitable for implementation using low computational resources.

In [35], a lattice planner is developed for, and tested on, an autonomous truck. The graph search algorithm used is based on TBA* [39] allowing for real-time computation of solution paths. Extensions to the search algorithm are developed so that the slow dynamics of the truck are taken into account. The planner is tested in simulations and in real experiments on an autonomous truck.

A lattice based motion planner for a general truck and trailer system is considered in [36]. These systems cannot be driven backward from arbitrary states, due to its unstable nonlinear behavior, as well as state and input constraints [41]. Due to these characteristics, the lattice is specially designed so that the discretized vehicle states are within circular equilibrium configurations, significantly reducing the dimension of the search space.

The concept of lattices can also be extended to on-road planning by creating a discretization of the state space that adapts itself to the road shape. In [42], a lattice is built online at each planning cycle, by selecting discretized states on the road, and connecting the states with feasible motion primitives. The motion primitives need to be computed online, in order to adapt them to the shape of the current road segment.

The work in [32] uses a similar concept as the previously proposed road lattice, however it simplifies the search by having a coarser discretization of the state space. The found solution is then optimized using a post-processing algorithm, and the results are comparable to those of the road lattice with the original discretization, although they can be obtained at a fraction of the computational time.

2.5 Optimization-based motion planning

Optimization problems such as the one in Equation (2.6) can be solved using continuous numerical algorithms [43]. These methods have the benefit of directly encoding the system dynamics and constraints in its formulation, and are characterized by smooth solutions when compared to other methods. This type of approach is attracting significant attention due to the constantly increasing computational power available, and to the growing

availability of numerical algorithms for solving optimization problems [44–47].

Numerical optimization

In order to understand the numerical optimization approach, we start by introducing the relatively simple optimization problem:

$$\begin{aligned} \underset{x_1,x_2}{\text{minimize}} \quad & f(x_1,x_2) = (x_1-2)^2 + (x_2-1)^2 \\ \text{subject to} \quad & x_1^2 - x_2 \leq 0, \\ & x_1 + x_2 \leq 2. \end{aligned} \tag{2.7}$$

The optimal solution is the point $x^* = (x_1^*, x_2^*)$ inside the feasible set, that minimizes the optimization function f. The feasible set corresponds to the set of points respecting both inequalities. For an introduction to algorithms solving this type of optimization problem the reader is referred to [43].

Depending on the optimization function and on the constraints, the optimization problem belongs to different problem classes. If the optimization function is quadratic and the constraints are linear with respect to optimization variables the problem is referred to as quadratic programming (QP). QP problems benefit from convexity, which allow algorithms to be developed which are guaranteed to find the optimal solutions to the problem.

In case the optimization function or the constraints have nonlinear expressions, then the problem falls within the nonlinear programming class. This type of problems are non-convex, and it is usually hard to develop general algorithms that find the global optimal solution to the problem. Instead general algorithms usually find solutions corresponding to a sub-optimal local minima.

If some of the variables in the optimization problem are restricted to be integers, then the problem belongs to the class of integer programming. Integer constraints typically arise in problems that must make binary decisions (such as deciding if a vehicle should overtake or not). The integer variables introduce a combinatorial complexity, and algorithms for these type of problems ofter suffer from very poor worst-case computational times.

It can be observed that Equation (2.7) is similar to the motion planning problem in Equation (2.6). In fact, the motion planning problem can be seen as numerical optimization in a high-dimensional search space. In the following section, we present in more detail how to use numerical optimization for motion planning.

Formulating the motion planning problem

The vehicle model

A vehicle model $\dot{q} = f(q, u)$ can be enforced by creating suitable constraints on the optimization problem. Typically, a discretized version of the continuous model is used:

$$q_{k+1} = f_k(q_k, u_k), \tag{2.8}$$

where q_k corresponds to the vehicle state at sample k, and f_k is a function that approximates the following state q_{k+1} based on the previous state q_k and input u_k.

Often a linear model is desired, as it is a requirement for many numerical optimization algorithms. The linearized and discretized version of the continuous model then becomes:

$$q_{k+1} = A_k q_k + B_k u_k + b_k. \tag{2.9}$$

Where

$$\begin{aligned} A_k &= \left.\frac{\partial f}{\partial q}\right|_{q_k^{\text{ref}}, u_k^{\text{ref}}}, \\ B_k &= \left.\frac{\partial f}{\partial u}\right|_{q_k^{\text{ref}}, u_k^{\text{ref}}}, \\ b_k &= f(q_k^{\text{ref}}, u_k^{\text{ref}}) - A_k q_k^{\text{ref}} - B_k u_k^{\text{ref}}. \end{aligned} \tag{2.10}$$

A_k, B_k, and b_k can be obtained using numerical differentiation methods. The constants q_k^{ref} and u_k^{ref} correspond to the linearization references.

Equation (2.9) is then incorporated into the numerical optimization problem as constraints on the variables q_k and u_k. We note that usually q_k is not an optimization variable, but instead an auxiliary variable, which is uniquely defined from the initial vehicle state q_0 and the vehicle inputs u_k.

Depending on the intended application, the considered vehicle models can vary significantly in terms of fidelity. In [48], a bicycle kinematic model is used. This type of model does not consider forces, making its usage limited to urban scenarios where driving is characterized by low speed driving with moderate steering angles. This model is extended in [49], by modeling the steering wheel actuator as a second-order system.

Dynamic models take into account forces acting on vehicle, and are often more accurate, although not always [50]. In [51] a model taking into account the longitudinal and lateral load transfers is used. Tire forces are also modeled, based on the Pacejka tire model, and successful experiments are carried on a low friction surface, made of packed snow. The work in [52] is able to adapt to changing road conditions, by adapting to current traction

limitations. This results in enhanced stability during evasive maneuvers and leads to a safer driving.

In case a reference path is known, road-aligned models can be used [53]. In the road-aligned model, the vehicle state is defined with respect to a frame moving along a reference path, instead of being defined in the Cartesian frame. This type of models prove to be useful in on-road situations, as they can simplify the optimization problem by making the optimization objectives convex, or the constraints easier to handle.

Different vehicle models and optimization criteria (discussed later in Section 2.5) are investigated in [54]. The authors study chassis and tire models of varying complexity, and arrive at the conclusion that the choice of model can potentially lead to fundamentally different optimization solutions. This highlights the fact that there is not one vehicle model which is best for all purposes, and that the choice of one is dependent on the application, required performance, and even on the numerical algorithms used.

Optimization objectives

Depending on the particular application that the motion planner is targeting, different goals and target behavior for the vehicle might be expected. These can often be encoded via appropriate optimization objectives.

To reduce actuator stress, the steering wheel and acceleration commands can minimized as part of the optimization objective. This results in a smoother and more comfortable driving for the vehicle passengers [16]. In [55], the minimization objective is chosen to be the norm of the vector of instantaneous yaw accelerations of the vehicle along the path. This objective encourages path solutions to be smooth and to not waste traction unnecessarily.

Progress maximization, or similarly traveling time minimization, can also be an objective to be optimized. When doing so the vehicle plans trajectories resulting in shorter traveling times. This can be interesting when planning trajectories for speeding up track times of a racing car [56], or for increasing the efficiency and productivity of processing relying on heavy-duty vehicle transport [57].

In order to maximize the safety in potentially dangerous maneuvers, the work in [48] considers the maximization of visibility. Certain driving situations might create blind spots for the vehicle sensors. These blind spots can contain unseen obstacles or even other vehicles that will cross the autonomous vehicle path. Thus, the optimization objective is introduced, in order induce the planner to find solution paths that minimize the blind spot areas, thus reducing safety hazards.

Collision checking

Constraints related to collision avoidance must also be incorporated into the optimization algorithm. Unfortunately, formulating such constraints is often challenging within the framework of numerical approaches. Two main reasons for this are the high non-linearity of the constraints, and the combinatorial nature of obstacle avoidance.

Obstacle avoidance constraints can be formulated as ensuring that the vehicle body does not intersect obstacles. This type of constraint is non-convex and non-differentiable in general, making it difficult in numerical approaches. Furthermore, collision avoidance is in general combinatorial, introducing binary decision variables.

As an example, recall Figure 2.4, where the vehicle needs to decide if it avoids the obstacles by passing them by their left or right. This type of decisions is binary, and in the presence of multiple obstacles, it becomes combinatorial, all obstacles can be avoided either by driving through its left or right, and all combinations must be considered. Problems with these type of decision variables fall within the class of integer programming, which has very poor worst-case computational times. In this section, we assume that the planner knows *a priori*, on which side to avoid an obstacle, thus removing the combinatorial aspects of planning.

The work in [58] makes use of distance maps in order to ensure collision avoidance. Distance maps compute for every point in the environment (x, y) the distance to the nearest obstacle. The gradient of a distance map generates a potential field that guides a path that is in collision into a solution path that is collision-free. We note however, that this type of approach, being only locally optimal instead of globally optimal, relies on a good initial guess.

Noticing the sensitivity of numerical approaches to the provided initial guess, [59] combines homotopy methods with numerical optimization. Obstacles in the environment are iteratively introduced during planning iterations using an homotopy method. Thus, the optimization problem is continuously changed from an easy to solve motion planning problem which ignores obstacles, into the original, and hard to solve, motion planning problem considering obstacles. This method allows numerical optimization methods to be used for a wider and more complex class of motion planning problems. However, a new challenge arises, which consists in finding a suitable homotopy for the problem at hand.

Several works have approximated the vehicle body through a combination of simpler shapes that are suitable for usage in numerical optimization methods. In [60], the vehicle shape is described using a set of circles, as shown in Figure 2.11 (Left). The number of circles used is usually a trade-off between conservativeness of the vehicle model (the fewer the circles the

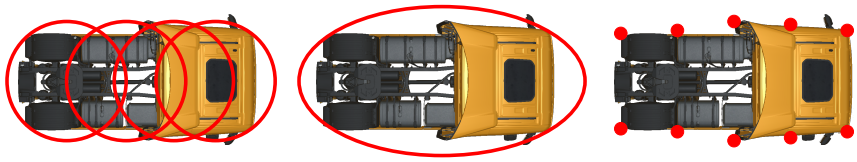


Figure 2.11: Modeling the vehicle body using: a combination of circles [60] (Left); an ellipse [49] (Center); a combination of points [17] (Right).

worse approximation of the shape), and computational times (more circles results in higher computational times). The vehicle shape can also be approximated using a safety ellipse, as proposed in [49] and illustrated in Figure 2.11 (Center). We note that this approach is not suited for long vehicles, such as buses, as the length to width ratio of the vehicle would result in safety ellipses that severely overestimate the vehicle body. A third way to approximate the vehicle body consists in considering multiple points located along its edges as shown in Figure 2.11 (Right). As in the case of the circles-based approximation, the number of points used is a trade-off between fidelity of the approximation, and computational times (more points cause higher computational times).

Based on the concept of approximating the vehicle shape using points, a new set of vehicle body approximations is introduced in [17]. These approximations are specifically targeted for numerical optimization methods that rely on vehicle models using the road-aligned frame. In this frame the vehicles suffer complicated distortions that do not have closed form expressions. The effect is further worsened when considering long vehicles, such as HDVs. This approach is explained in detail in Chapter 5.

Relevant works

The work in [44] introduces a reformulation of the trajectory planning task, that allows the usage of convex optimization methods. The method is applicable to environments where the obstacles can be described as a union of convex set, and ensures that collision avoidance constraints are exact. However, the proposed solution requires an initial good guess, that is obtained via graph-search based motion planning methods. Simulation results are shown for a vehicle parking in very tight environments.

To take full advantage of state-of-the-art numerical optimization methods, [59] makes use of homotopy methods for motion planning. The proposed method targets systems with nonlinear dynamics in complex nonconvex environment, an application where numerical optimization approaches often fail. By introducing an homeomorphic transformation of the environ-

ment, the authors are able to use powerful numerical optimization methods in these challenging applications. One drawback remains however, the need to find a suitable homeomorphic transformation of the environment.

Observing that the combinatorial aspects of motion planning are of minor importance when considering road driving, the work in [60] uses local and continuous optimization for trajectory planning. The approach focuses on the smoothness, dynamics, and optimality, and like other continuous optimization schemes, does not suffer from a complexity which rises exponentially with the dimension of the stat space. The approach is used in a fully autonomous vehicle which drove a route of 103 km and dealt with complex traffic situations.

In emergency scenarios it might be the case that vehicle stabilization needs to be sacrificed in order to avoid collision. In [61] the authors use numerical optimization in order to plan motions which avoid obstacles which suddenly appear on the road. The method is experimented on an autonomous vehicle showing its capabilities of driving at the vehicle's handling limits. Prioritizing collision avoidance over vehicle stabilization can allow for a reduction of accidents when considering extreme emergency situations.

2.6 Interaction-aware motion planning

In the previous sections, we considered the collision avoidance constraint to take the form $q(t) \in X_{\text{free}}(t)$. This type of constraint is suitable when the motion planner is provided with predictions of obstacles, *e.g.*, human-driven vehicles, from a perception system, as illustrated in Figure 1.1. One implicit assumption of Figure 1.1 is that predictions are independent of the decision making and motion planning modules, as they are located downstream in the architecture. However, the traffic scene and the human drivers are affected by and react to the autonomous vehicle's decisions. For example, if the autonomous vehicle is driving in a single lane and slows down, the vehicle closest behind it will adjust its velocity to avoid colliding. Therefore, the evolution of the traffic scene depends on the actions of the autonomous vehicle.

Let us define the input trajectory of the autonomous vehicle for the time interval $[0, t]$ as:

$$u\left([0,t]\right)$$

and the corresponding autonomous vehicle state trajectory as:

$$x\left([0,t]\right).$$

We can now define the free space at time t as:

$$X_{\text{free}}\left(t, u\left([0,t]\right), x\left([0,t]\right)\right). \tag{2.11}$$

Equation (2.11) emphasizes that the free space at time t also depends on the input and state trajectories of the autonomous vehicle. In a traffic scene, Equation (2.11) is used to capture the fact that the maneuvers executed by the autonomous vehicle impact the trajectories of other traffic participants, which in turn leads to the obstacle-free space changing.

The original motion planning problem in Equation (2.6) is then adapted to consider the interaction aspects between the autonomous vehicle and other traffic participants, yielding the new motion planning problem:

$$\begin{aligned} \underset{u(t)}{\text{minimize}} \quad & J\left(q(s), u(s)\right) \\ \text{subject to} \quad & \dot{q}(t) = f(q(t), u(t)), \\ & q(0) = q_0, \\ & q(t_G) = q_{\text{G}}, \\ & q(t) \in X_{\text{free}}\left(t, u\left([0,t]\right), x\left([0,t]\right)\right), \end{aligned} \tag{2.12}$$

where the collision avoidance constraint explicitly considers that the free space, determined by the space not occupied by other traffic participants, is dependent on the autonomous vehicle state and its inputs.

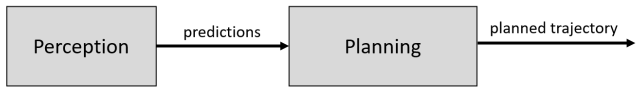
We note that this new formulation is not compatible with the simplified autonomous vehicle diagram presented in Figure 1.1. Figure 1.1 assumes a downstream flow where the predictions of the other traffic participants are provided to the motion planner. Afterwards, the motion planner is tasked with finding a solution that avoids other participants. With the new formulation of the motion planning problem in Equation (2.12), the predictions of other traffic participants are modified in response to the planned maneuver of the autonomous vehicle. The new formulation of the motion planning problem is referred to as *joint prediction and planning*. The modular architectural differences between a framework implementing the original motion planning problem in Equation (2.6) and the new joint prediction and planning problem in Equation (2.12) are shown in Figure 2.12.

Relevant works

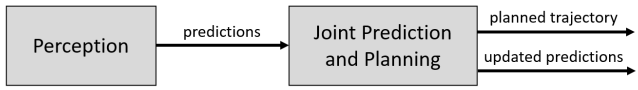
In the following, we present some of the relevant work in the area of joint prediction and planning. The considered relevant work is grouped in the following categories: Numerical optimization, Search and sampling, Partially observable Markov decision process, Game theory, and Learning-based.

Numerical optimization

The work in [62] proposes a Stochastic MPC (SMPC) for autonomous vehicles that can deal with the multi-modality and interaction aspects of the sur-



(a) Predictions as an input to the planner in Equation (2.6).



(b) Predictions and planning are jointly optimized in Equation (2.12).

Figure 2.12: Top: Many planning approaches assume the predictions of other vehicles and plan a motion for the autonomous robot that avoids these predictions. Bottom: To reduce conservatism and increase the performance of autonomous vehicles a joint prediction and planning approach is needed.

rounding drivers. The approach considers that a target vehicle might have multi-modal predictions corresponding to different types of driver behavior. The predictions of the human-driven vehicle depend on the autonomous vehicle decisions, where the human driver will try to keep a constant headway to the vehicle in front. The proposed approach is shown to reduce conservatism and improve the performance of the autonomous vehicle, mainly when interaction plays an important role.

Some works combine learning with MPC [63, 64], extending the capabilities of the underlying MPC from pure reacting behavior to interaction-aware behavior. The intersection case is studied in [63], where an autonomous vehicle is approaching an intersection where other vehicles are also crossing. A deep Reinforcement Learning (RL) approach [65] is used to determine if the autonomous vehicle should stop, cross, or wait for a specific gap to pass, given the positions and velocities of other vehicles. The low-level MPC planner then computes a safe trajectory based on this high-level decision. Similarly, the work in [64] combines a deep RL and Model Predictive Contour Control (MPCC) to tackle the challenging task of merging in dense traffic scenarios. An interaction-aware policy is learned using deep RL. The policy takes as inputs the states of the vehicles ahead and behind the autonomous vehicle and outputs a continuous velocity reference, as opposed to a discrete decision as in [63]. This velocity reference is then used to guide the MPCC, which plans a collision-free trajectory.

Search and sampling

The authors of [66] show the importance of considering social cooperation, *i.e.*, interaction-aware human predictions, when considering merging from entrance ramps. The proposed framework samples different candidate strategies, corresponding to different velocity profiles for the autonomous vehicle. For each candidate strategy, a prediction engine that considers how the human driver adapts to the vehicle ahead of it is used. Afterwards, each traffic scene is evaluated with respect to an expected cost, and the candidate strategy that achieves the lowest cost is selected as the maneuver to execute.

The work in [67] tackles merging scenarios in congested traffic. In these scenarios, merging onto the road might be impossible without the cooperation of human-driven vehicles, highlighting the importance of interaction-aware planning. The proposed solution simulates and evaluates a large number of planned trajectories and chooses the one achieving the highest utility. The predictions of the human-driven vehicles are obtained by considering an Intelligent Driver Model that adjusts to the actions planned by the autonomous vehicle.

Partially observable Markov decision process

A partially observable Markov decision process (POMDP) models the autonomous vehicle decision making by assuming that the traffic scene evolves according to known dynamics, but that the observations of other vehicles can have noise, and equally important, that the intentions of other drivers are unknown, *i.e.*, they are hidden. POMDPs are a powerful tool for planning and decision making under uncertainty. However, finding an optimal solution can often be computationally too complex [68].

The work in [69] formulates the decision making problem of an autonomous vehicle as a POMDP. In their formulation, the intended route of other vehicles is considered a hidden variable. The solution to their POMDP is a policy with the optimal acceleration for an autonomous vehicle following a pre-planned path. A remarkable behavior that appears from this approach is that the autonomous vehicle might postpone making certain decisions. The postponing happens since the autonomous vehicle understands that more information will be present in the future, leading to better predictions of other traffic participants. Therefore, the proposed POMDP can trade-off between exploration, *i.e.*, gathering information, and exploitation, *i.e.*, optimizing the driving behavior. The results show that their approach performs nearly as well as if the actual intentions of other vehicles were known in advance.

The authors of [70] combine the strengths of POMDP and MPC to tackle the joint prediction and planning problem in a T-junction traffic scenario with up to five vehicles. An online POMDP estimator is used to predict the other traffic participants' velocity profiles and their likelihood. Human drivers are assumed to choose between three different intentions: accelerate, brake, and keep speed. Afterwards, a receding horizon optimization plans a trajectory for the autonomous vehicle that guarantees a specified level of safety under all possible intentions of the other vehicles. This is implemented using a chance constrained MPC formulation, which can directly incorporate the likelihood of other traffic driver intention, as estimated by the POMDP. The solution found by the MPC is fed to the POMDP estimator in the next planning cycle, generating a new POMDP traffic scene estimate and, afterwards, a new planned trajectory. The framework effectively performs joint prediction and planning by sequentially predicting and planning over multiple planning cycles.

Game theory

Game theory studies the decision making processes of multiple agents that try to minimize the costs associated with their actions. In the game theory framework, the costs of each agent's actions depend on the actions of the other agents, therefore becoming crucial for an agent to consider how other agents might act [71]. Game theory is a suitable framework for tackling the joint prediction and planning problem since autonomous vehicles and other human-driven vehicles are interacting agents with their own conflicting goals.

The work in [72] models the traffic scene with two agents, an autonomous and a human-driven vehicle, as a partially observable stochastic game. The authors propose approximations for solving this game in real time and show that their proposed method can plan autonomous vehicle maneuvers that are assertive and show intent. In simulation and user study experiments, the autonomous vehicle accelerates or slows down when approaching an intersection to encourage the human-driven vehicle to either give way or proceed first.

The work in [73] proposes a hierarchical trajectory planning algorithm for autonomous vehicles. On a higher level, a dynamic game is solved, which models the long-horizon interaction between the autonomous and a human-driven vehicle. The computations of the higher level are used as guidance for a short horizon trajectory optimization algorithm, effectively allowing the lower short horizon planner to reason about the long-term effects of its actions.

Traffic scenes often contain more than two agents, calling for methods

that are able to solve problems with multiple agents. The work in [74] presents a framework that can solve three-player games efficiently. The proposed approach is used in an intersection scenario, where two cars and a pedestrian traversing a crosswalk make up the traffic scene. Similarly, the work in [75] tackles merging scenarios with four agents. In their work, the authors study the merging maneuver for an autonomous vehicle approaching a highway where three other vehicles are present.

The concept of Social Value Orientation is introduced in [76] to quantify the degree of altruism or selfishness of human drivers. The authors propose a solution that estimates a human driver's Social Value Orientation and then incorporates this information to solve a two-agent game.

Learning-based

Classical prediction approaches tend to produce trajectories, or distributions of trajectories, for the different agents in the scene. This type of representation does not allow the planning and control layers to reason about the interactivity of the scene. Therefore, the planner can only try to avoid the provided trajectories. Realizing this drawback, the authors of [77] propose a prediction framework that outputs Mixtures of Affine Time-varying Systems (MATS). The proposed MATS prediction representation allows a downstream planning layer to reason about the impact of planned maneuvers of the autonomous vehicle on traffic participants, therefore capturing the interaction aspects of driving.

The work in [78] makes use of a transformer, a deep learning model, to enable interaction-aware motion planning. One of the proposed solution's benefits is a unified architecture that allows for different types of prediction for all participants in the traffic scene. Using the same model but different masking strategies, the approach can perform either of the following tasks:

- *Planning*: Propose a goal state that directs a motion planner into a desirable end state, given the current and past history of the whole traffic scene (other vehicles + autonomous vehicle);

- *Behavior prediction*: Predicting the evolution of the whole traffic scene (other vehicles + autonomous vehicle), given the current and past history of the traffic scene;

- *Conditional behavior prediction*: Predicting the evolution of the traffic scene (excluding the autonomous vehicle), given the current and past history of the whole traffic scene and the planned trajectory of the autonomous vehicle.

The first task, planning, can provide a downstream motion planner with a good guess of what is a desirable state to drive towards. This is an indirect way of including interaction in the motion planning framework. The last task, conditional behavior prediction, provides a motion planner with a measure of the impact its action will have on the other traffic participants. This corresponds to a direct way of including interaction in the motion planning framework.

Chapter 3

Sharpness Continuous Paths

This chapter deals with a simplified version of the planning problem in which the obstacles in the environment are ignored. Ignoring obstacles in the environment can, in some cases, allow the development of solutions to the planning problem which are based on analytic or geometric arguments, as opposed to having to resort to numerical optimization or search-based techniques. Analytic or geometric solutions are often computationally inexpensive, and as such, desirable for online implementations.

The simplified formulation might also allow the development of solutions with optimality guarantees. The most frequent metric to be optimized, is the length of the planned solution. In the case of vehicles, this often correlates with the execution time of the solution, thus being an attractive metric to minimize.

In this simplified formulation, the vehicle kinematic and dynamic constraints, which define how the vehicle is allowed to move, are still considered, and are part of the problem formulation. As one might expect, the more complex the constraints considered, the more challenging it is to find optimal solutions to the planning problem. However it will be shown that the most relevant vehicle constraints, associated with the vehicle actuators, can be captured, while still keeping the problem tractable to solve.

Although not obvious, solutions to the obstacle-free planning problem can be directly used as components of algorithms that seek to solve the original planning problem in the presence of obstacles. Thus, improvements in the area of obstacle-free planning, will greatly contribute to the progress of algorithms for the general planning problem. Chapter 4 focuses on solving the planning problem in the presence of obstacles, and makes use of the methods introduced in this chapter.

The contributions of this chapter are the following:

- take into account steering actuator magnitude, rate, and acceleration

limitations;

- ease the controller task and improve passenger comfort;
- connect arbitrary vehicle configurations;
- have fast computation times.

The chapter is organized as follows.

Section 3.1 gives an introduction to the topic of planning in the absence of obstacles. A literature survey on the topic of obstacle-free planning for car-like vehicles is presented, which summarizes relevant works in the area, ranging from the original works already started in the fifties, to very recent publications, which show a reborn interest in the topic over the last couple of years.

Section 3.2 introduces Cubic Curvature paths, an interpolating curve with desirable properties for autonomous vehicles. This curve can be seen as an extension to the well-known clothoid curves, which are commonly used in road design [79].

Section 3.3 introduces Sharpness Continuous paths, a novel proposal, which seeks to solve the obstacle-free path planning problem for the case of heavy-duty vehicles. Inspired by previous works in obstacle-free optimal path planning, we propose a solution which takes into account both the kinematic constraints of the vehicle and the dynamic constraints of steering actuators used in heavy-duty vehicles. This novel proposal considers the rate and torque limitations in the steering actuators of the vehicle, a problem not considered in previous published works.

Section 3.4 presents results of the proposed method and shows its benefits. It is shown that the proposed paths resemble the length optimal Dubins paths [25], and that they come with the benefit of improved controller performance. Furthermore the computational times of the method are measured, and show suitability for online implementation. Concluding remarks are made in Section 3.5.

3.1 Introduction

Planning in the absence of obstacles can be seen as a simplified version of the planning problem, in which the obstacles in the planning space are ignored. The problem can be formulated as:

$$\underset{u}{\text{minimize}} \quad J(q,u) \tag{3.1a}$$
$$\text{subject to} \quad \dot{q} = f(q,u), \tag{3.1b}$$
$$q(0) = q_{\text{S}}, \tag{3.1c}$$
$$q(1) = q_{\text{G}}, \tag{3.1d}$$

where constraint Equation (3.1b) corresponds to the vehicle model. In the following sections we will detail solutions that have been proposed for vehicles with increasingly complex constraints $f(q)$.

Constraints Equation (3.1c) and Equation (3.1d) correspond to the initial and final states of the vehicle. In the planning problem, Equation (3.1c) usually corresponds to the current state of the vehicle, whereas Equation (3.1d) corresponds to the goal state. These two constraints correspond to boundary conditions that the solution to the differential equations must respect, making this problem often be referred to as a two-point boundary value problem.

The objective function Equation (3.1a) defines the metric to be optimized. In this chapter we will consider this function to correspond to the length of the path performed by the vehicle. This is a suitable metric to optimize, as a common goal in driving, is to arrive at the destination as fast as possible, which can usually be achieved by minimizing the length of the driven path. A remark has to be made however, as it is not always the case, that the shortest path is the one that can be performed the fastest. Another shortcoming of this metric is that it does not take into account the comfort of passengers.

This simplified planning problem might seem trivial at first glance, however the presence of the differential constraints Equation (3.1b) complicates the process of finding feasible solutions to it. Figure 2.3 illustrates the difficulty associated with obstacle-free path planning. The vehicle is tasked with moving from its initial state to the goal state. The shortest path between these two configurations would be the straight line shown in red, however, the kinematic constraints prevent the vehicle from executing such a path. A possible solution that the vehicle can perform, is to move forward, switch to reverse, and approach the goal configuration in a backward motion, as shown by the green path in Figure 2.3. This illustrates the complexity of motion planning for vehicles, even in the absence of obstacles.

In the following we review previous works that have managed to provide solutions to variants of the obstacle-free path planning problem Equation (3.1). Some solutions are able to provide optimality guarantees, at the expense of assuming simplified vehicle models. Other solutions consider

more complex vehicle motion constraints, while still showing a performance close to optimal.

Dubins curves

The work in [25] studied the nature of curves of minimal length, with bounded curvature, connecting initial and terminal positions and tangents. The proposed curves have been extensively used in the motion planning and robotics community. This is due to the fact that many mobile robots have system equations that result in motions defined by curves of bounded curvature.

We start by defining the system equations that are considered in [25]:

$$\begin{aligned} \dot{x} &= \cos\theta, \\ \dot{y} &= \sin\theta, \\ \dot{\theta} &= u. \end{aligned} \tag{3.2}$$

This corresponds to a system that moves with unit speed in the xy plane, and that can actuate its angular velocity, through u. The angular velocity is assumed to be bounded, such that $|u| \leq u_{\max}$. The major contribution in [25] was to prove that the shortest path between two vehicle states $(x_{\mathrm{S}}, y_{\mathrm{S}}, \theta_{\mathrm{S}})$ and $(x_{\mathrm{G}}, y_{\mathrm{G}}, \theta_{\mathrm{G}})$ for this kind of system, is always composed by at most three motion patterns. The motion patterns are: drive straight (S), turn left with maximum curvature (L), and turn right with maximum curvature (R). The actuation values for the motion patterns S, L and R are $u = 0$, $u = u_{\max}$ and $u = -u_{\max}$, respectively.

System Equation (3.2) is often referred to as the kinematic car model. The word kinematic originates from the fact that forces actuating on the system are disregarded, and instead only positions, velocities and accelerations are used to describe the system. Kinematic models are fairly simple and accurate for low speeds, making them suitable for many heavy-duty vehicle industrial applications. However when considering high speeds, such as in the case of highway driving, dynamic models which take into account forces are often necessary.

The solutions of minimal length, obtained using the motion patterns S, L, R, are commonly referred to as Dubins paths. Figure 3.1 shows Dubins paths, connecting initial and goal vehicle poses q_{S} and q_{G}, for three different planning instances. In order to find the optimal sequence of motion patterns, one can exhaustively try all possibilities, and chose the one resulting in a shortest path. However there are more efficient ways to find out the minimum length Dubins path between two vehicle configurations [80].

The system presented in Equation (3.2) is equivalent to the kinematic car model if one replaces the actuator command, such that $u = \tan(\phi)/L$,

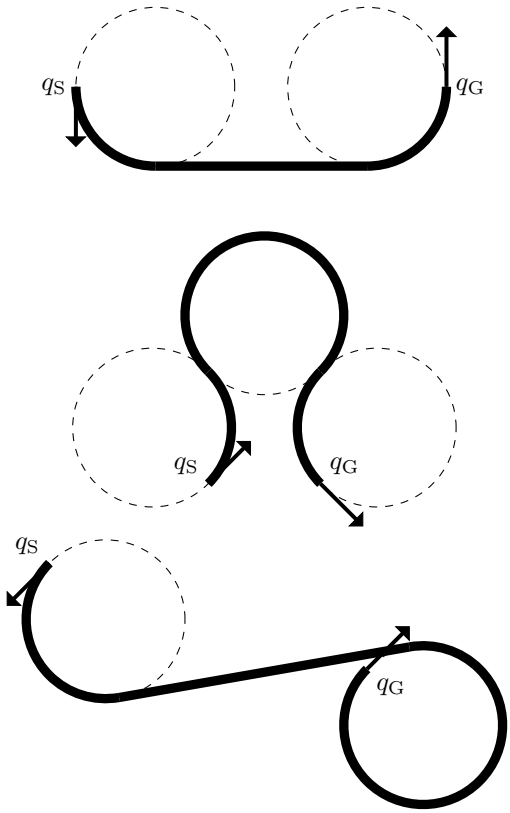


Figure 3.1: Different examples of Dubins paths.

where ϕ is the steering angle of the vehicle, and L is the wheelbase length. The value of $u_{\max}$ corresponds to $\tan(\phi_{\max})/L$, with $\phi_{\max}$ being the maximum steering angle of the vehicle. The inverse of the curvature of the turning motion patterns L and R, corresponds to the minimum turning radius of the vehicle. Therefore, Dubins paths have immediate applicability in motion planning for vehicles, and are widely used in the field of autonomous driving [81–84].

One of the drawbacks of the Dubins paths, is the simplified motion equations that are considered for the vehicle model. Since the actuation u can change instantaneously, the resulting optimal paths have a curvature profile that is discontinuous at the transitions between straight lines and circular arcs. In order for a vehicle to exactly perform these paths, it would have to stop at the junctions of two motion patterns, and turn the steering wheels in place. This is of course time consuming, and results in an increased

wear of the vehicle tires. Typical autonomous driving implementations that make use of Dubins paths, usually ignore this problem, and resort instead to a post-optimization step on the found paths, or simply let the underlying feedback controllers perform the path without stopping between motion patterns. More recent works have tried to tackle the discontinuity problem, as will be seen in the following sections.

Reeds-Shepp curves

A vehicle can move both forward and backward, an aspect that Dubins paths do not capture. Realizing that backward movement is an important part of driving, [26] proposed to solve the minimal path length problem for the more complex vehicle model:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= u. \end{aligned} \tag{3.3}$$

Contrary to the case of Dubins paths, this model allows the vehicle to have a forward or a backward unit velocity, $|v| = 1$. Thus the vehicle can switch directions of movement, allowing it to realize paths that are shorter or equal to the Dubins path.

The work in [26] concludes that the optimal length paths for the car that can move both backward and forward, is also a combination of at most three motion patterns, turn left, turn right and drive straight (L, R and S). However these motion patterns can now be performed either in forward or in backward motion. Similar to the Dubins paths, these path solutions have gained popularity in the motion planning community and are known as Reeds-Shepp paths. Furthermore, it has been shown that there is a total of 46 possible combinations of motion patterns that can result in the shortest path [85].

An example of two planning instances, connecting different initial and goal vehicle poses q_{S} and q_{G}, solved using Reeds-Shepp paths is shown in Figure 3.2. It can be seen that switches in the direction of movement occur, resulting in cusps at intermediate sections of the path.

It should be noted that Reeds-Shepp paths also suffer from the drawback of discontinuous curvature that Dubins paths were shown to have. Thus vehicles performing these paths will have to stop and re-orient their steering wheels at the junction of different motion patterns, in order to follow the path exactly. The next section introduces an extension to Reeds-Shepp paths, which deals with the problem of discontinuous curvature.

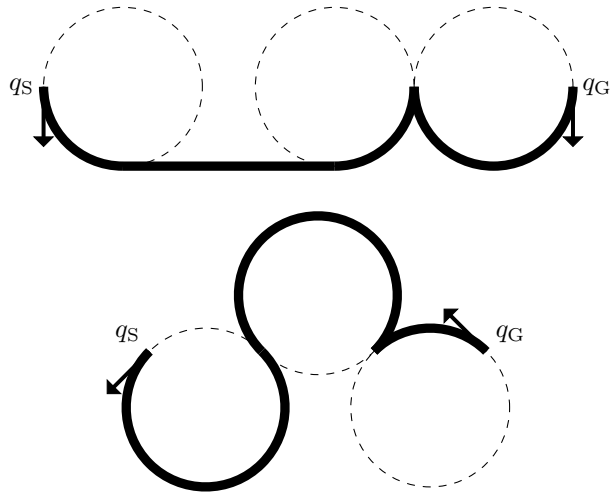


Figure 3.2: Different examples of Reeds-Shepp paths.

Continuous curvature paths

In [27] an extension to Reeds-Shepp paths is proposed, which is able to solve the inherent discontinuity problem. In fact this solution, allows the computation of paths with: 1) continuous curvature; 2) bounded curvature; and 3) upper bounded curvature derivative. This means that the generated paths can be followed by a vehicle of the form:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= v\kappa, \\ \dot{\kappa} &= u. \end{aligned} \tag{3.4}$$

The above system constitutes a more complex version of the kinematic car model Equation (3.3). The dynamics of the real vehicle are more accurately modeled by extending the system with the curvature κ of the vehicle. The longitudinal velocity v can take unit values $|v| = 1$, corresponding to moving forward or backward. The control u is the angular acceleration of the vehicle, and is related to the steering rate of the front wheels.

In order to achieve curvature continuity, the authors in [27] introduce a new motion pattern to the Reeds-Shepp paths. This motion pattern corresponds to a clothoid arc, and is used to ensure a continuous curvature transition between line segments (S) and arc circle segments (L or R). By inserting this extra clothoid arc in between motion patterns of the Reed-

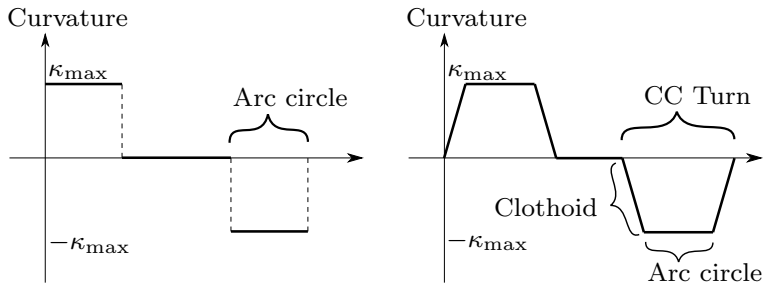


Figure 3.3: The curvature profile of Dubins [25] and Reeds-Shepp [26] paths (left) presents discontinuities. Curvature continuous paths [27] (right) overcome this problem by using clothoid segments between arc circles and line segments. Figure based on [27].

Shepp paths, it is possible to ensure curvature continuity of the path. The junction of motion patterns corresponding to a clothoid arc, an arc circle, and a clothoid arc, is called a continuous curvature (CC) turn. Figure 3.3 shows an example comparing the curvature profile of a Reeds-Shepp path and that of a path as proposed by [27].

Unfortunately, [27] provides no guarantees as to the length optimality of these paths. The problem of finding length optimal paths for the system in Equation (3.4), had already been addressed in [86], which hinted at a complex behavior of the optimal solution, with possibly infinitely many switching points. However [27] successfully shows that the proposed paths converge towards the optimal Reeds-Shepp paths, as $|u| \to \infty$. This in itself is a good indication that the solution paths are of good quality, as it shows that their length is not excessively larger than the optimal paths for the simplified vehicle model.

Recently, some works have tried to further improve Curvature Continuous paths. This can be done by taking into account even more complex vehicle dynamics, which is expected to increase the performance and comfort of autonomous vehicles following it. The following section details some of these works.

Further extensions

The work [87] builds upon the original Dubins paths, adapting them so as to generate smooth curvature profiles. To do so, a Dubins path is computed in a first step, and then, intermediate paths are added at the discontinuous curvature transitions. The additional transitions are characterized by a smooth curvature profile with bounded curvature and curvature derivative. The resulting path is C^∞, meaning that it is infinitely differentiable, and

presents a smooth curvature profile. One of the shortcomings of this work, is that it only focuses on the forward motion of the vehicle, and that it does not consider Dubins paths concatenating three sequential turns.

Targeting the case of tight environments, such as parking lots, the work in [28] proposes Hybrid Curvature steer, an approximation of Reeds-Shepp's paths. The proposed solution has the same benefits as Curvature Continuous paths, however it allows curvature discontinuities where changes in direction movement of the vehicle occur. This is inspired by human behavior when parking, which is characterized by turning the steering wheel while stopped, when changing directions of movement. The authors show that when integrated in a motion planner, Hybrid Curvature steer allows for shorter paths, when compared to Curvature Continuous curves, and smoother paths, when compared to Reeds-Shepp's paths.

The Hybrid Curvature steer approach is further developed into Hybrid Curvature Rate steer in [29]. The work extends the continuous curvature vehicle model into the following model:

$$\begin{aligned} \dot{x} &= v\cos\theta, \\ \dot{y} &= v\sin\theta, \\ \dot{\theta} &= v\kappa, \\ \dot{\kappa} &= \alpha, \\ \dot{\alpha} &= u, \end{aligned} \tag{3.5}$$

where κ, α, and u are all bounded. In order to generate motions that respect this vehicle model, the paths must respect maximum limits on curvature, curvature rate and curvature acceleration. Curvature and curvature rate limits have been tackled by Continuous Curvature paths [27], but curvature acceleration is for the first time considered in this work. By considering limitations on the curvature acceleration, a planner is able to generate paths that take into account actuator limits, and that result in a better tracking performance of the vehicle controller. Tracking performance is very important, specially in tight environments, where poor tracking can lead to deviations from the planned path that might result in collisions with obstacles.

A novel take on steering methods is introduced in [88] where the authors consider planning in the belief state. As opposed to all of the previously mentioned solutions, which assume a perfect following of the planned path, [88] takes into account uncertainties in both the controller and localization modules. The result is a steering method which provides paths with an associated motion execution uncertainty, which is computed by assuming knowledge of the uncertainties in the measurement model and the disturbances affecting the system. By taking into account the uncertainty

associated with the planned path, the proposed motion planner plans motions that have a significantly smaller risk of collision due to control and localization errors.

The work in [89] follows a similar line as that of Continuous Curvature paths. Instead of resorting to clothoid arcs, characterized by C^0 curvature continuity, they make use of a transition segment, which ensures C^1 curvature continuity. Even though C^1 curvature is guaranteed, the produced paths do not have a bounded curvature rate, and can result in curvature rate profiles with noticeable spikes. The transition segment used in this work is derived so as to ensure convergence of a given control law, thus allowing for control guarantees when executing the path. Control guarantees provide a way to know the maximal possible deviation from the planned path, when the vehicle is following the path. If at planning time, this maximal possible deviation is also checked for collision, then it is possible to plan collision-free paths, even in the presence of controller transients/errors. The maximal possible deviation, and collision checking of it, is not pursued in [89], but is instead left as future work.

Summary

The works mentioned previously show a trend of trying to consider increasingly more complex vehicle constraints. The benefit of considering more complex vehicle models at the planning layer, is that this will simplify the control task, which in turn results in more accurate tracking of the planned paths. In most autonomous systems following the sense-plan-act architecture, the collision checking procedures are done only at the planning layer, and for the planned path. A disparity between planned path and performed path is inevitable, meaning that the collision checking procedures at the planning layer might not be sufficient to ensure safety of the system. The previously mentioned works tried to address these issues through three different approaches:

Accurate system dynamics By taking into account more complex vehicle models, which are closer to the real vehicle model, the differences between planned path and performed path are reduced. This results in collision checking procedures which are more accurately portraying the performed path [29, 87];

Controller stability guarantees By studying the properties of the whole system, *i.e.*, considering the whole chain consisting of planner, controller, and vehicle, it is possible to arrive at strong guarantees about its stability and convergence. Furthermore, it is possible to ensure that the system will stay within a region around the planned path. If

the given region is collision-free then the whole system will be safe, even in the presence of controller transients. An initial work into this topic is done in [89];

Uncertainty and disturbance considerations Considering measurement uncertainties and controller disturbances is another possible way to ensure safety of the system. Associated to the planned path is an uncertainty that can be used to measure the probability of collision of a given path. A path is deemed safe if it has a low enough collision probability [88].

In the remainder of this chapter we will present a novel method which follows the first type of approach. This method is motivated by the observation that all of the previously proposed works have been focusing on constraints on the path properties (curvature and curvature rate). However these constraints do not immediately relate to the vehicle actuator limits. Instead they have a complex relation to the path and the velocity at which the path should be followed. The proposed method starts from the vehicle actuator limits, and based on them, and on the desired vehicle velocity, generates constraints on the path curvature and curvature rate. This is a benefit in itself, as curvature rate constraints are not usually known or easy to compute, whereas actuator limits are. Furthermore, this work proposes constraints on the curvature acceleration, as they are a natural consequence of limited actuator torque.

3.2 Cubic curvature paths

Cubic curvature paths are hereby introduced as paths that respect complex steering actuator constraints, namely limitations on steering magnitude, rate and torque. In the following sections we introduce the considered vehicle model, the constraints it puts on a path to be followed, and the suitability of cubic curvature paths for this particular vehicle model.

Vehicle model

We start by considering the already familiar vehicle model used by Dubins [25] and given by Equation (3.2). The curvature κ of a vehicle with wheelbase length L is related to its steering angle ϕ through

$$\kappa = \tan(\phi)/L. \tag{3.6}$$

This allows us to write the angular velocity as a function of the steering angle ϕ, resulting in the vehicle model:

$$\dot{x} = v\cos\theta, \tag{3.7a}$$
$$\dot{y} = v\sin\theta, \tag{3.7b}$$
$$\dot{\theta} = v\frac{\tan(\phi)}{L}, \tag{3.7c}$$

where (x, y) represents the position of the rear wheel axle center of the vehicle, θ its orientation and v is the vehicle velocity. A vehicle pose is defined by the three variables (x, y, θ). If an additional curvature is associated to a pose we obtain a configuration, defined as (x, y, θ, κ).

The steering angle ϕ of the vehicle is physically coupled to an actuator, which like any real system has physical limitations. In this work we consider the following limitations:

- Maximum steering angle amplitude $\phi_{\max}$,
- Maximum steering angle rate of change $\dot{\phi}_{\max}$,
- Maximum steering angle acceleration $\ddot{\phi}_{\max}$.

Thus, in addition to Equation (3.7) the following constraints must be respected:

$$-\phi_{\max} \leq \phi \leq \phi_{\max}, \tag{3.8a}$$
$$-\dot{\phi}_{\max} \leq \dot{\phi} \leq \dot{\phi}_{\max}, \tag{3.8b}$$
$$-\ddot{\phi}_{\max} \leq \ddot{\phi} \leq \ddot{\phi}_{\max}. \tag{3.8c}$$

These constraints directly affect the vehicle motion capabilities and should be dealt with when planning paths.

Path feasibility

Path feasibility depends on the capabilities of the vehicle that executes it and on the path itself. The limited steering angle amplitude $\phi_{\max}$ constraint Equation (3.8a) imposes a maximum allowed curvature on the path $\kappa_{\max}$. This limitation is addressed by generating paths which have a curvature profile $|\kappa| \leq \kappa_{\max}$ [25, 26]. Limited steering angle rate of change $\dot{\phi}_{\max}$ Equation (3.8b), can be tackled by limiting the curvature derivative of the generated paths [27]. We note however that the relation between steering angle rate and curvature derivative is not a trivial one.

In this work we also deal with the third limitation Equation (3.8c), related to the limited steering angle acceleration $\ddot{\phi}_{\max}$. Having a limited

$\ddot{\phi}_{\max}$ results in $\dot{\phi}$ being a continuous function, which in turn indicates that ϕ is a continuously differentiable, C^1 function. Previously mentioned works, with the exception of [29, 87, 89], fail to generate paths with a C^1 curvature profile. Paths that do not have a C^1 curvature profile, require an infinite $\ddot{\phi}_{\max}$, in order to be accurately followed. This is impossible to achieve by an actuator, and motivates the usage of paths with a C^1 curvature profile.

The steering profile is related to the curvature profile through Equation (3.6). The sharpness α is defined as the change of curvature along the path length s:

$$\alpha = \partial\kappa/\partial s. \tag{3.9}$$

By ensuring sharpness continuity in a path, we guarantee that the curvature, and the steering profile of such a path is C^1. A vehicle is thus able to follow the path using a bounded steering acceleration.

In the following section, we detail how to generate paths that respect all three limitations, $\phi_{\max}$, $\dot{\phi}_{\max}$, and $\ddot{\phi}_{\max}$, previously stated.

Cubic curvature paths

Cubic curvature paths are paths where the curvature is given by a polynomial of order three with respect to the length s along the path:

$$\kappa(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0. \tag{3.10}$$

A cubic curvature profile is the minimum degree polynomial that allows us to define arbitrary initial and final curvatures, κ_{i} and κ_{f}, and sharpnesses α_{i} and α_{f}. The sharpness profile of these paths is given by:

$$\alpha(s) = \frac{\partial\kappa(s)}{\partial s} = 3a_3 s^2 + 2a_2 s + a_1. \tag{3.11}$$

In order to find the parameters of the cubic polynomial, we use the initial and final constraints:

$$\kappa(0) = \kappa_{\mathrm{i}}, \tag{3.12a}$$
$$\alpha(0) = \alpha_{\mathrm{i}}, \tag{3.12b}$$
$$\kappa(s_{\mathrm{f}}) = \kappa_{\mathrm{f}}, \tag{3.12c}$$
$$\alpha(s_{\mathrm{f}}) = \alpha_{\mathrm{f}}, \tag{3.12d}$$

where s_{f} is the path length. Assuming the path length s_{f} is known, the terms of the polynomial Equation (3.10) can be determined by solving the

linear equation system:

$$a_0 = \kappa_\mathrm{i}, \tag{3.13a}$$

$$a_1 = \alpha_\mathrm{i}, \tag{3.13b}$$

$$\begin{bmatrix} 3s_\mathrm{f}^2 & 2s_\mathrm{f} \\ s_\mathrm{f}^3 & s_\mathrm{f}^2 \end{bmatrix} \begin{bmatrix} a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \alpha_\mathrm{f} - a_1 \\ \kappa_\mathrm{f} - a_1 s_\mathrm{f} - a_0 \end{bmatrix}. \tag{3.13c}$$

We use cubic curvature paths as transition segments between straight lines and arc circles. In order to ensure that the path formed by stitching together different path segments has a C^1 curvature profile, we need to ensure that both the curvature and the sharpness profiles are continuous. Initial constraint Equation (3.12a) is used to ensure that the initial curvature is equal to the final curvature of the preceding segment path. Final constraint Equation (3.12c) is used to ensure that the final curvature is equal to the initial curvature of the following segment path. Analogously, Equation (3.12b) and Equation (3.12d) are used to ensure that the initial and final sharpness match those of the preceding and following paths. By doing so, we ensure continuity of the curvature and of the sharpness, which leads to paths with C^1 curvature profiles.

It should be noted that so far, we have only considered curvature and sharpness constraints, that ensure continuity of the curvature profile. However it is necessary to also consider the steering limitations Equation (3.8), in order to ensure that the generated paths are feasible to follow. The following section addresses this issue.

Ensuring steering rate and acceleration constraints

In order to have a feasible path, we need to ensure that a vehicle can follow it while complying with its steering constraints Equation (3.8). Each constraint introduces a different requirement on the generated cubic curvature path.

The limitations regarding the steering magnitude can be addressed by selecting appropriate initial and final curvature and sharpness constraints. If we make both κ_i and κ_f smaller in magnitude than κ_max, and α_i and α_f equal to zero, we can ensure that the curvature profile will be contained within the bounds $[\min(\kappa_\mathrm{i}, \kappa_\mathrm{f}), \max(\kappa_\mathrm{i}, \kappa_\mathrm{f})]$. Since the steering angle is related to the curvature according to Equation (3.6), we get that the steering angle will be contained within the bounds $[\min(\phi_i, \phi_f), \max(\phi_i, \phi_f)]$, where $\phi_i = \arctan(\kappa_i L)$ and $\phi_f = \arctan(\kappa_f L)$.

Given κ_i, κ_f, α_i, and α_f, and selecting an arbitrary path length s_f, a cubic curvature profile is generated according to Equation (3.13). Afterwards, the steering angle profile corresponding to the cubic curvature path is computed

from the path curvature using Equation (3.6). Assuming then that the vehicle is following the path at a given fixed velocity v, the steering angle rate and acceleration profiles are computed.

The steering angle rate profile will have a peak rate $\dot{\phi}_{\text{peak}}$, which can be found by numerical methods. In our case we simply perform an exhaustive search over the whole steering rate profile. We note that a numerical procedure is necessary due to the non-linear relation between the curvature profile and the steering rate profile. In case $\dot{\phi}_{\text{peak}}$ is larger than the allowed maximum steering rate $\dot{\phi}_{\max}$ the length s_{f} needs to be increased so that $\dot{\phi}_{\text{peak}} = \dot{\phi}_{\max}$. In case $\dot{\phi}_{\text{peak}}$ is smaller than the allowed maximum steering rate, then the length should be increased, this is done to ensure that paths with excessive length are not allowed. The needed change in path length s_{f} corresponds to scaling by a factor of $\dot{\phi}_{\text{peak}}/\dot{\phi}_{\max}$.

The peak acceleration $\ddot{\phi}_{\text{peak}}$ can also be found by numerically computing the maximum magnitude of the steering acceleration profile. If the maximum magnitude of the steering acceleration profile $\ddot{\phi}_{\text{peak}}$ is exceeding or under the acceleration limitation $\ddot{\phi}_{\max}$, then a scaling of the path length must be done. In this case the needed scaling factor is given by $\sqrt{(\ddot{\phi}_{\text{peak}}/\ddot{\phi}_{\max})}$.

To guarantee that the path respects both steering rate and acceleration limitations, we need to scale its length by the greater of the scaling factors. If both scaling factors are smaller than one, then we are reducing the path length, thus ensuring that the path is as short as possible and making use of the full actuation capabilities of the vehicle. Once the new path length s_{f} is found, the cubic curvature path is recomputed by solving Equation (3.13). This ensures that the vehicle steering limitations are respected, as long as the vehicle follows the given path at a velocity v, or lower. Algorithm 1 summarizes this procedure.

3.3 Sharpness continuous paths

Making use of the previously introduced cubic curvature paths, a path planner for obstacle-free environments can be developed. Our proposed method is of a similar nature to that in [27] (explained in Section 3.1). However, instead of clothoid arcs, we use cubic curvature paths as transition segments between line segments (S) and arc circles (L or R). This results in paths that respect the steering limitations Equation (3.8).

Sharpness continuous turns

We propose Sharpness Continuous (SC) turns, which consist of three segments, an initial cubic curvature path $\Gamma_{1,2}$, an arc circle $\Gamma_{2,3}$, and a final cubic curvature path $\Gamma_{3,4}$. Figure 3.4 shows an example of such an SC

Algorithm 1: Cubic Curvature Path coefficient computation

Input: κ_{i}, κ_{f}, v, $\dot{\phi}_{\max}$, $\ddot{\phi}_{\max}$, $\alpha_{\mathrm{i}} = 0$, $\alpha_{\mathrm{f}} = 0$, $s_{\mathrm{f}} = 10$
Output: a_0, a_1, a_2, a_3

$a_0, a_1, a_2, a_3 \leftarrow$ solve Equation (3.13);
$\dot{\phi}(t) \leftarrow \mathrm{ComputeSteeringRate}(a_0, a_1, a_2, a_3, \mathrm{v})$;
$\dot{\phi}_{\mathrm{peak}} \leftarrow \mathrm{GetPeakSteeringRate}(\dot{\phi}(t))$;
$C \leftarrow \dot{\phi}_{\mathrm{peak}} / \dot{\phi}_{\max}$;
$\ddot{\phi}(t) \leftarrow \mathrm{ComputeSteeringAcceleration}(a_0, a_1, a_2, a_3, \mathrm{v})$;
$\ddot{\phi}_{\mathrm{peak}} \leftarrow \mathrm{GetPeakSteeringAcceleration}(\ddot{\phi}(t))$;
if $\ddot{\phi}_{\mathrm{peak}} / \ddot{\phi}_{\max} > C$ **then**
    $C \leftarrow \ddot{\phi}_{\mathrm{peak}} / \ddot{\phi}_{\max}$;
$s_{\mathrm{f}} \leftarrow C \cdot s_{\mathrm{f}}$;
$a_0, a_1, a_2, a_3 \leftarrow$ solve Equation (3.13);

turn. The initial segment $\Gamma_{1,2}$ starts with null sharpness at a configuration $q_1 = (x_1, y_1, \theta_1, \kappa_1, \alpha_1)$. It then ends with maximum curvature, $\pm\kappa_{\max}$, and null sharpness, at configuration $q_2 = (x_2, y_2, \theta_2, \kappa_2, \alpha_2)$. The second segment is an arc circle $\Gamma_{2,3}$ with radius $\kappa_{\max}^{-1}$ and arbitrary arc length, starting at q_2 and ending at $q_3 = (x_3, y_3, \theta_3, \kappa_3, \alpha_3)$. The SC turn is completed with a path $\Gamma_{3,4}$, starting with curvature $\pm\kappa_{\max}$ and null sharpness, and ending at a configuration $q_4 = (x_4, y_4, \theta_4, \kappa_4, \alpha_4)$, also with null sharpness.

We assume, without loss of generality, that the vehicle, and subsequently the path, starts at a configuration $q_1 = (0, 0, 0, 0, 0)$. From q_1, it then follows the path $\Gamma_{1,2}$ taking it to a configuration $q_2 = (x_2, y_2, \theta_2, \kappa_{\max}, 0)$. The path $\Gamma_{1,2}$ has initial and final curvatures 0 and $\kappa_{\max}$, respectively. The initial and final sharpnesses are both set to zero, in order to ensure that $\Gamma_{1,2}$ can be stitched together with arc circles and straight segments while still having sharpness continuity, *i.e.*, the curvature profile is C^1. We note that line segments and arc circles, are special cases of cubic curvature paths, that share in common a constant curvature profile, and a null sharpness profile. Path $\Gamma_{1,2}$ is computed according to Algorithm 1, in order to ensure that steering actuator limitations of the vehicle are respected. The values x_2, y_2, and θ_2 are those that result from following the curvature profile of $\Gamma_{1,2}$ with a starting vehicle state q_1.

Once the vehicle has a curvature $\kappa_{\max}$, it then follows an arc circle path $\Gamma_{2,3}$ with radius $\kappa_{\max}^{-1}$. The arc circle starts at (x_2, y_2) and has its center at a distance $\kappa_{\max}^{-1}$ perpendicular to the orientation θ_2 at point (x_2, y_2). Its

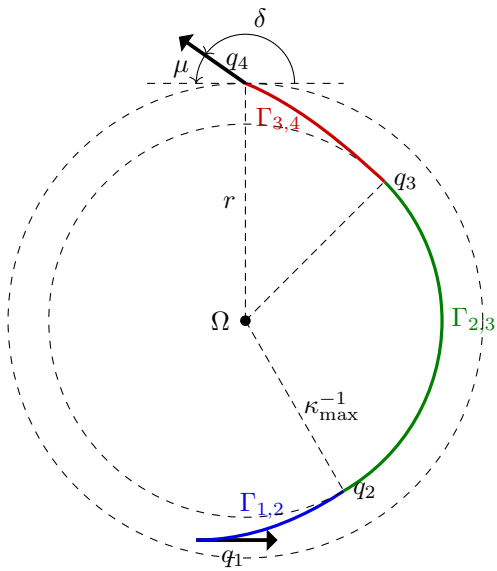


Figure 3.4: Example of a sharpness continuous turn.

center is given by

$$(x_\Omega, y_\Omega) = (x_2 - \kappa_{\max}^{-1} \sin\theta_2, y_2 + \kappa_{\max}^{-1} \cos\theta_2). \tag{3.14}$$

The last path segment $\Gamma_{3,4}$ departs from the arc circle at q_3 and brings the vehicle to a configuration q_4. Configuration q_4 depends on the point of departure from the arc circle q_3. However, it always lies on a circle Ω, which has the same center as the arc circle (x_Ω, y_Ω) in Equation (3.14).

In order to find the radius of circle Ω, we first assume an auxiliary arc circle to be centered at $(x_{\Omega'}, y_{\Omega'}) = (0, \kappa_{\max}^{-1})$. We assume a departure configuration from the circle at $(0, 0, 0, \kappa_{\max}, 0)$. Then, by following the path given by a curvature profile with initial and final curvatures $\kappa_{\max}$ and κ_4, and null initial and final sharpness, computed with Algorithm 1, we will end at a configuration $q' = (x', y', \theta', \kappa', 0)$. Configuration q' is located at an auxiliary Ω' circle (the auxiliary equivalent of the Ω circle), that has the same center as the arc circle. Thus we compute the radius of Ω', which is equal to the radius of Ω, as

$$r = \sqrt{(x' - x_{\Omega'})^2 + (y' - y_{\Omega'})^2} = \sqrt{x'^2 + (y' - \kappa_{\max}^{-1})^2}. \tag{3.15}$$

An additional angle μ is defined as the difference between θ' and the tangential angle to Ω at configuration q'. It is computed using the previous

auxiliary arc circle as

$$\mu = \arctan\left(\frac{y' - \kappa_{\max}^{-1}}{x'}\right) + \frac{\pi}{2} - \theta'. \tag{3.16}$$

Thus, given a certain initial configuration q_1, the possible positions of the ending configuration q_4, resulting from a combination of a cubic curvature path, an arc circle, and another cubic curvature path, *i.e.*, an SC turn, lie on a circle Ω. The possible θ_4 orientations of these configurations are given by the tangential angle at the circle plus μ. Figure 3.4 illustrates circle Ω and the tangential angle μ.

Connecting sharpness continuous turns

An SC path between start and goal configurations q_S and q_G can be found by combining SC turns and line segments. Similarly to the Dubins case, two different types of paths can be found, those composed of two turns connected by a line segment, and those composed of three consecutive turns. Each of these must be handled in different ways, as explained below.

Turn + straight + turn case

Assuming the case when the SC path is composed of two turns connected by a line segment, we have the following path elements:

- an SC turn starting at the start configuration q_S and ending at a configuration q_a with null curvature;
- a line segment starting at q_a and ending at q_b;
- an SC turn starting at configuration q_b with null curvature, and ending at the goal configuration q_G.

Figure 3.5 shows an example of an SC path, with the three elements described above.

We note here that our approach disregards switches of direction of movement, *i.e.*, it assumes that the vehicle only moves forward or backward, but not both. This makes it similar to Dubins paths [25], and can possibly result in paths that are longer than those that would be computed if direction of movement switches were allowed. These shortcomings could be fixed with further development, and consequent generalization, of the proposed approach.

In order to connect two SC turns, we need to find the configurations q_a and q_b that belong to the starting and ending SC turn possible departure configurations, and that can be connected with a line segment. To do so,

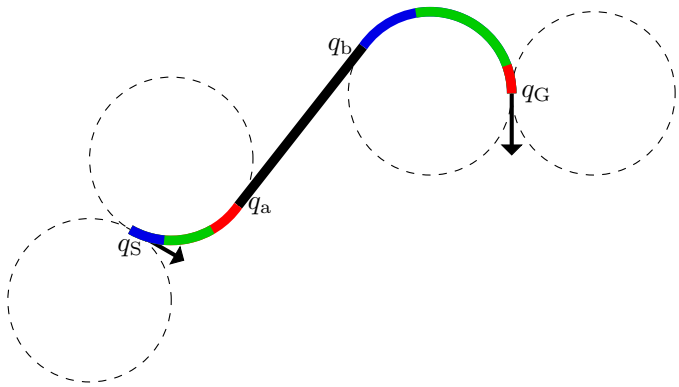


Figure 3.5: Sharpness continuous path example. The path consists of an SC turn between q_{S} and q_{a}, a line segment from q_{a} to q_{b}, and an SC turn between q_{b} and q_{G}. Each SC turn contains a curving cubic curvature path (blue), an arc circle segment (green), and a decurving cubic curvature path (red). The dashed circles correspond to the SC turns that can span from q_{a} and q_{b}.

q_{a} and q_{b} must have the same orientation, *i.e.*, $\theta_{\mathrm{a}} = \theta_{\mathrm{b}}$. Furthermore both must lie on a line segment with an inclination angle θ_{a}.

As seen before, in Figure 3.4, the possible set of departure configurations of an SC turn are located on a circle, and its orientations differ from the circle tangent by μ. Thus, to connect two SC turns, we need a way to connect two circles Ω_{s} and Ω_{f} with arbitrary centers, radii, and μ values.

We first assume two auxiliary circles Ω_{a} and Ω_{b}, as depicted in Figure 3.6 (top). Circles Ω_{a} and Ω_{b} have the same radii and μ values as the original circles Ω_{s} and Ω_{f}. Circle Ω_{a} is centered at $(0, 0)$ and Ω_{b} is located so that q_{a} and q_{b} are collinear. We are interested in finding the center of $\Omega_{\mathrm{b}} = (x_{\Omega_{\mathrm{b}}}, y_{\Omega_{\mathrm{b}}})$. From Figure 3.6 (top) it can be seen that

$$y_{\mathrm{b}} = -r_{\mathrm{a}} \cos \mu_{\mathrm{a}} + r_{\mathrm{b}} \cos \mu_{\mathrm{b}}. \tag{3.17}$$

We assume that the distance $r(\Omega_{\mathrm{a}}, \Omega_{\mathrm{b}})$ between the circle centers is the same as the distance between the original circles $r(\Omega_{\mathrm{s}}, \Omega_{\mathrm{f}})$. We then have

$$x_{\Omega_{\mathrm{b}}} = \sqrt{r(\Omega_{\mathrm{s}}, \Omega_{\mathrm{f}})^2 - y_{\Omega_{\mathrm{b}}}^2}. \tag{3.18}$$

We know that $q_{\mathrm{a}} = (r_{\mathrm{a}} \sin \mu_{\mathrm{a}}, -r_{\mathrm{a}} \cos \mu_{\mathrm{a}}, 0, 0, 0)$ and $q_{\mathrm{b}} = (x_{\Omega_{\mathrm{b}}} - r_{\mathrm{b}} \sin \mu_{\mathrm{b}}, y_{\Omega_{\mathrm{b}}} - r_{\mathrm{b}} \cos \mu_{\mathrm{b}}, 0, 0, 0)$. To find these configurations in the original circles Ω_{s} and Ω_{f}, we need to first apply a rotation $\Delta_\theta = \arctan(y_{\Omega_{\mathrm{f}}} - y_{\Omega_{\mathrm{s}}}, x_{\Omega_{\mathrm{f}}} - x_{\Omega_{\mathrm{s}}})$ to q_{a} and q_{b}. We then translate these configurations by $(\Delta_x, \Delta_y) = (x_{\Omega_{\mathrm{s}}}, y_{\Omega_{\mathrm{s}}})$.

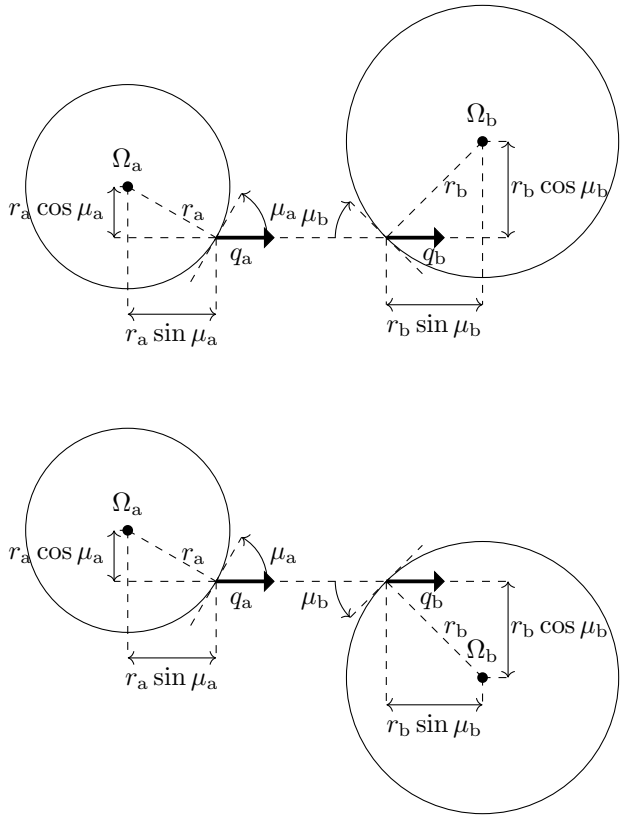


Figure 3.6: Computing the external (top) and internal (bottom) tangents between two circles.

The resulting rotated and translated configurations correspond to the desired tangent configurations between the circles Ω_s and Ω_f.

The above procedure finds the departure configurations between two counterclockwise (left steering) SC turns, shown in Figure 3.6 (top). An analogous procedure can be used to find the possible departure configurations between any combination of clockwise (right steering) and counterclockwise turns, as shown in Figure 3.6 (bottom). These procedures are valid if the found tangent configurations q_a and q_b do not lie inside the circles Ω_b and Ω_a, respectively.

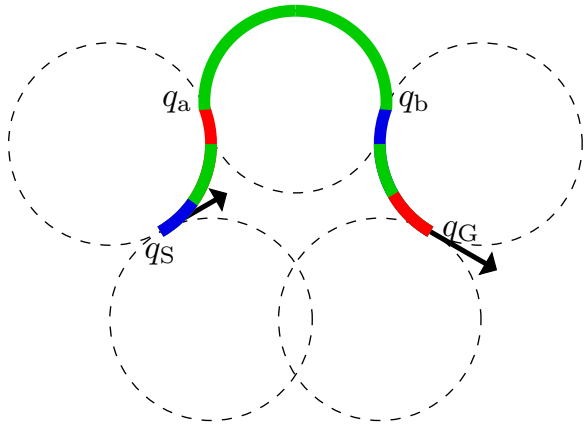


Figure 3.7: Sharpness continuous path example. The path consists of an SC turn between q_S and q_a, an arc circle segment from q_a to q_b, and an SC turn between q_b and q_G. Each SC turn contains a curving cubic curvature path (blue), an arc circle segment (green), and a decurving cubic curvature path (red). The dashed circles correspond to the SC turns that can span from q_a and q_b, but also to the circle that the intermediate arc segment belongs to.

Turn + turn + turn case

An SC path can also be created by combining three consecutive turns. In that case, the path has the following elements:

- an SC turn starting at the start configuration q_S and ending at a configuration q_a with maximum curvature $\pm\kappa_{max}$;
- an arc circle segment with maximum curvature (equivalent to minimum turning radius) connecting q_a and q_b;
- an SC turn starting at configuration q_b with maximum curvature $\pm\kappa_{max}$, and ending at the goal configuration q_G.

Figure 3.7 shows an example of an SC path, with the three elements described above.

We start by defining the auxiliary circles Ω_a and Ω_b, as depicted in Figure 3.8. Circles Ω_a, Ω_b, and the associated μ values are computed as detailed in Section 3.3. The initial circle Ω_a is associated with the start configuration q_S, whereas the final circle Ω_b is associated with the goal configuration q_G.

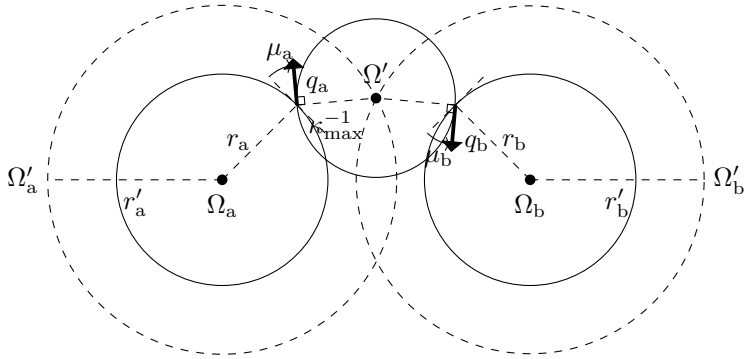


Figure 3.8: Computing the tangency of a turn + turn + turn path.

We assume that from the exit configuration q_a, the vehicle follows an arc circle with minimum turning radius $\kappa_{\max}^{-1}$. The circle to which this arc circle belongs will be referred to as Ω'. Since the exit configuration q_a can lie anywhere on the circle Ω_a, we get also that the center of Ω' can lie anywhere within a circle Ω'_a that is concentric to Ω_a. The radius r'_a of Ω'_a is given by:

$$r'_a = \sqrt{\left(r_a + \kappa_{\max}^{-1} \cos \mu_a\right)^2 + \left(\kappa_{\max}^{-1} \sin \mu_a\right)^2}. \tag{3.19}$$

The equivalent procedure is applied to the auxiliary circle Ω_b and the entry configuration q_b, in order to obtain the circle Ω'_b of possible locations of the center of the circle of maximum turning radius Ω'. Analogously, we compute the radius of Ω'_b as:

$$r'_b = \sqrt{\left(r_b + \kappa_{\max}^{-1} \cos \mu_b\right)^2 + \left(\kappa_{\max}^{-1} \sin \mu_b\right)^2}. \tag{3.20}$$

The remaining task is to compute the intersection between the two circles of possible locations of the center of Ω'. In the general case, two intersection points can be found, however only one of them will allow the concatenation of paths to be continuous in the orientation.

Figure 3.8, shows the procedure to find an SC path composed of a counterclockwise (left steering) SC turn, a clockwise (right steering) arc circle, and a counterclockwise SC turn. An analogous procedure can be made to find the SC path composed by a clockwise SC turn, a counterclockwise arc circle, and a clockwise SC turn.

One important difference in the initial and final SC turns tangent configurations, q_a and q_b, of the turn+turn+turn case, is that they have maximum curvature, instead of null curvature, as in the turn+straight+turn case. This allows the turn+turn+turn case to be composed of two SC turns with an

arc circle in between them, as opposed to two SC turns with another SC turn in between them. By using an arc circle as an intermediate connection between the start and final SC turns, instead of yet another SC turn, we can guarantee a shorter and more natural transition between the maximum curvature limits between turns.

Finding the shortest SC path

In order to find the shortest SC path between two configurations q_S and q_G, we need to compute all the possible SC turns that can be spanned from these configurations. The SC turns are then connected, via line segments or arc circles, in order to generate possible SC paths.

Each of the configurations q_S and q_G can span a total of four SC turns, depending on whether the vehicle is moving forward or backward, or whether it is turning left or right. The possible SC turns that span from q_S and q_G are shown in Figure 3.5 as dashed circles (forward and backward SC turns are equivalent, so they lie on top of each other). Figure 3.4 shows an SC turn which assumes a vehicle moving forward and turning left. The method explained in Section 3.3 can be readily used to obtain SC turns moving forward, independent of the direction they are turning. The procedure to obtain an SC turn moving backward is analogous.

There are a total of 20 possible SC paths between the two sets of 4 SC turns spanned from q_S and q_G. 16 of these possible paths are composed of two SC turns connected via a line segment, while the remaining 4 are a combination of two SC turns connected via an arc circle. All path combinations are computed and checked for feasibility, using the methods detailed in Section 3.3. Each path length is evaluated, and the shortest feasible path is selected as the solution.

3.4 Results

We present here experiments showing the benefits of the proposed SC paths, and indicate its amenability to online implementation.

Convergence of SC paths to Dubins paths

As previously mentioned, Dubins and Reeds-Shepp paths are proven to be optimal in terms of length. The optimality guarantee comes however at the cost of considering a simplified vehicle model. When considering more complex vehicle dynamics, the optimal solution often presents complex behaviors, and proves hard to compute analytically [27, 86]. It is still possible however, to understand how near-optimal paths are, by comparing them

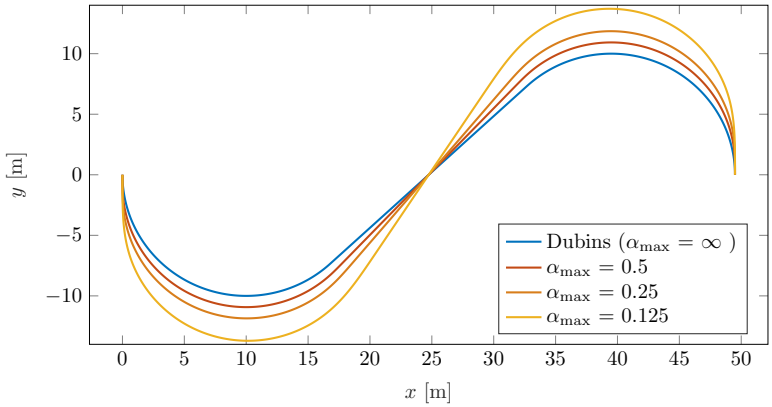


Figure 3.9: SC paths with increasing sharpness α converge to the length optimal Dubins path.

with the optimal paths, for vehicles with simpler, but similar constraints. Thus we analyse our proposed method, that respects actuator dynamic limitations, by comparing it to the Dubins paths, which does not comply with these limitations.

SC paths are, not surprisingly, longer than the Dubins path corresponding to the same initial and final vehicle states. This is a natural consequence of introducing a transition segment between the arc circle turns and the line segments. With the added transition segment, SC turns have a longer turning radii than if the turn was done resorting only to an arc circle with radius $\kappa_{\max}^{-1}$, as is the case with Dubins paths.

Figure 3.9 shows the Dubins path for a given start and goal configuration q_{S} and q_{G}. Overlayed are SC paths with different maximum sharpness $\alpha_{\max}$ for the same configurations q_{S} and q_{G}. It is seen that the greater the maximum sharpness $\alpha_{\max}$ of the SC paths is, *i.e.*, the greater the achievable steering rate and accelerations of the vehicle, the closer it approaches the Dubins path. This is somewhat intuitive, as increasing $\alpha_{\max}$ results in increasing the rate of change of the curvature profile, and consequently in a reduction of the length of the transition segments (cubic curvature paths). If $\alpha_{\max} \to \infty$, then the curvature changes would be immediate, and the transition segments would vanish, making the SC path equivalent to the Dubins path, and as such, length optimal.

To further study the convergence of SC paths to the length optimal Dubins paths, we run several planning instances, in which the steering limitations are relaxed by a tuning factor K_ϕ. Thus, for each planning instance, several SC paths are computed with steering limitations $\dot{\phi}_{\max} = K_\phi \dot{\phi}^0_{\max}$

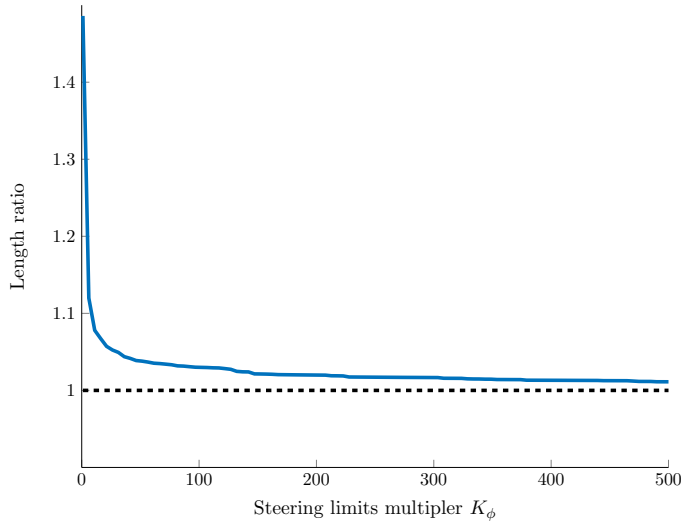


Figure 3.10: Convergence of the length of SC paths to the length of Dubins paths.

and $\ddot{\phi}_{\max} = K_\phi \ddot{\phi}^0_{\max}$. The maximum steering angle magnitude $\phi_{\max}$ is held fixed. The length of the paths are compared to the corresponding Dubins path with maximum steering angle magnitude $\phi_{\max}$. Figure 3.10 shows the ratio between the lengths of the SC paths and the Dubins path for 1000 randomized planning instances. It can be seen that as K_ϕ is increased the length of the SC paths converges to the length of the Dubins path.

Precomputation of cubic curvature paths

As previously mentioned, given two configurations q_S and q_G, the SC method computes all possible 20 SC turns and how they can be connected. The connection process, as detailed in Section 3.3, is computationally cheap. The bulk of processing is due to the generation of the cubic curvature paths corresponding to the transition segments.

As seen in Section 3.3, an SC turn depends on the cubic curvature paths that are part of it. In order to evaluate these paths, one has to generate their curvature profiles from the given initial and final constraints. To comply with the steering constraints, a numerical evaluation of a steering profile must be done, in order to find the path length scaling factors, as detailed in Section 3.2. When one has the desired curvature profile, the orientations θ can be obtained analytically. The x and y positions of the path are found by integrating the vehicle model equations, using an Euler method, which

has a high computational cost. This cost can be greatly reduced using precomputations, under the following assumptions.

If one assumes that the start and end configurations q_S and q_G always have null curvatures, then one can compute, in an initialization procedure, all the possible cubic curvature paths starting and ending at curvatures $\kappa = 0, \pm\kappa_{max}$. Thus it is possible to avoid the expensive generation of cubic curvature paths needed to find out the possible SC turns. In order to generate an SC turn, one just has to use the precomputed paths and apply rotations and translations to them.

The precomputation of paths can still be achieved, without limiting the start and goal configurations q_S and q_G to have null curvature. In fact, one can allow q_S and q_G to have curvature values belonging to finite discrete set. The set of precomputed paths can then be stored in a look-up table and used online during the computation of sharpness continuous paths.

Timing evaluation

We test the steering method, measuring its computational speed for several problem instances. The method is implemented in `C++` and running on a Linux Mint distribution. The computer used is equipped with an Intel Core i7-6820HQ Processor running at 2.70 GHz, and with 16.0 GB of RAM.

We generate 1000 random pairs of start and goal configuration queries. Each query is repeated 100 times to get a better estimate of the average computation time. The start and goal configurations of each query are generated by sampling the x and y coordinates from a uniform random distribution between -50 and 50. The orientations are sampled from the interval $[-\pi, \pi]$, and the curvatures from a discrete equispaced set of 11 curvatures $[-\kappa_{max}, \ldots, 0, \ldots, \kappa_{max}]$.

Without the use of precomputations, the average time for finding a solution path is 16ms, whereas when making use of precomputations the average time is 88µs. The precomputations decrease the computational time by three orders of magnitude, mostly due to the fact that the generation of the cubic curvature paths, which involves the expensive integration of the vehicle model equations, is avoided. These results indicate that the steering method is extremely inexpensive when using precomputations, making it suitable for real-time applications.

Controller performance

A simulation test is run in order to understand how the proposed paths affect the performance of a vehicle tracking them. A kinematic vehicle model coupled with a detailed steering actuator model is used to simulate the vehicle.

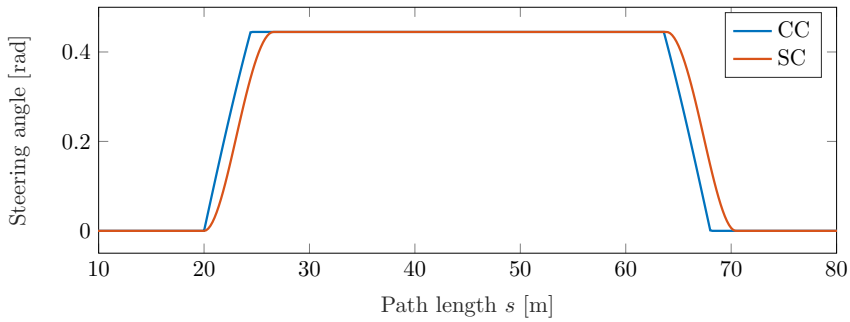


Figure 3.11: Steering reference profiles used in simulation.

In the test, two paths consisting of a straight segment, a turn, and a straight segment are generated. The first path is a CC path [27], while the second is our proposed SC path. Both paths abide by the same maximum steering angle magnitude $\phi_{\max}$ and steering angle rate $\dot{\phi}_{\max}$ constraints. Additionally, the SC path respects also the steering angle acceleration $\ddot{\phi}_{\max}$ constraint, unlike the CC paths.

The simulation assumes a steering actuator that is limited in terms of achievable steering angle magnitude, rate, and acceleration. The steering angle is controlled making use of a PID controller, which receives a steering angle reference, and actuates on the steering angle torque. The PID controller was tuned to achieve a step response with a relatively fast settling time and little overshoot. The steering angle reference is provided from a high-level path tracking controller. The high-level controller consists of a feedforward part and a feedback part. The feedforward part is obtained by finding the closest path point, and getting the corresponding steering angle reference at that point. The feedback part is a proportional controller regulating both lateral and heading errors. Such a controller is a simple implementation commonly used in path tracking applications.

Figure 3.11 shows the steering reference profiles of the paths to be tracked. The difference between them is in the shape of the increasing and decreasing sections of the steering angle. In the CC case, the steering angle, directly related to the curvature, follows a linear profile while in the SC it follows a cubic profile.

Figure 3.12 shows the lateral and heading errors when the vehicle tracks both paths. The vehicle is initially placed at the start of the path, and it follows the first straight segment perfectly. However, when the turning section starts, a deviation from the path begins to arise. The feedback part is then responsible for trying to regulate the errors to zero. Shortly after the turn begins, the CC case becomes unstable. On the other hand, the SC

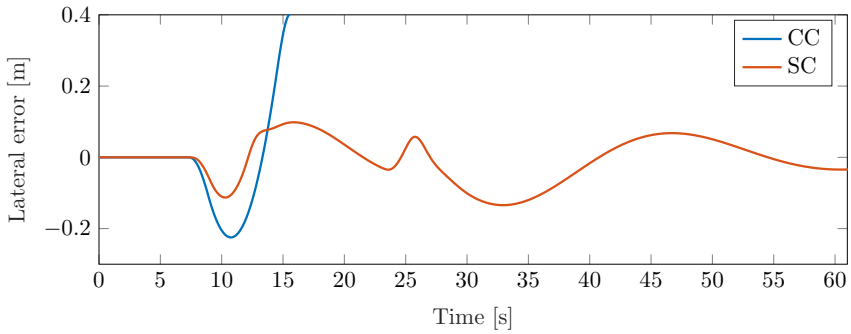


Figure 3.12: Lateral error when tracking a CC and an SC path. When tracking the CC path, the controller becomes unstable, resulting in an error that grows indefinitely, and out of scope of the graph. The SC path tracking is seen to be stable.

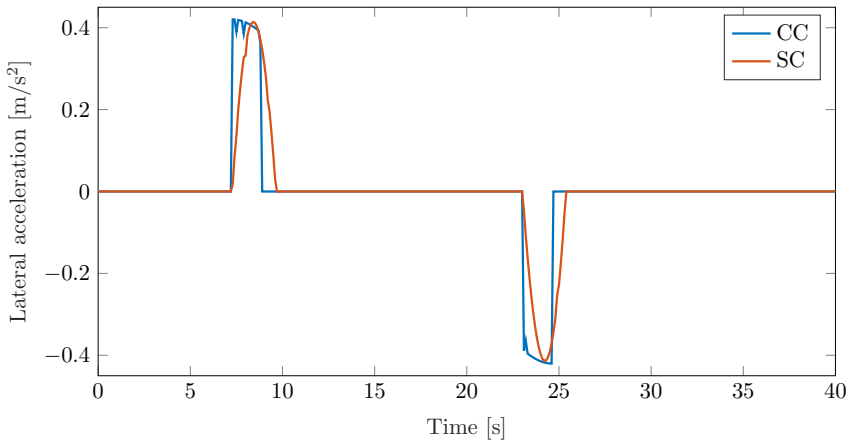


Figure 3.13: Lateral acceleration when tracking a path without feedback, *i.e.*, using only feedforward references.

case is stable, and its error converges to zero. The error profiles show that the controller performance is worse when tracking CC paths.

The lateral acceleration and jerk (acceleration rate) experienced by the vehicle are related to passenger comfort. Figure 3.13 shows the lateral accelerations for a vehicle following the reference steering profiles without feedback actuation. It is seen that the CC path has large jerk values, which result from an aggressive steering actuation. The SC path steering profile achieves smoother lateral acceleration profiles.

3.5 Conclusions

This section provided an introduction to obstacle-free path planning methods, listing relevant contributions in the field for the application case of autonomous vehicles. A trend towards more complex vehicle models can be observed in the related work section, however, a common shortcoming to many approaches is the lack of compliance to explicit steering actuator limitations. The works deal indirectly with path properties, such as curvature rate and curvature acceleration, which have a complex relation with the steering actuator properties. To address this issue we propose cubic curvature paths, which, with additional assumptions on vehicle velocity, can be feasibly followed by a vehicle with known steering actuator constraints. The cubic curvature paths, are then integrated within Sharpness Continuous (SC) turns, which can be stitched together with line segments to generate SC paths. We show experimentally that these paths resemble the length optimal Dubins paths.

We thus introduce a novel steering method, which unlike previous approaches, is able to take into account steering actuator limitations explicitly, instead of requiring cumbersome constraints to be formulated on the path curvature properties. The generated paths respect the maximum steering angle constraints, maximum steering rate and acceleration constraints. These properties ease the low-level controller task and introduce an higher degree of smoothness, improving the driving comfort and reducing actuator effort. This is of importance when dealing with heavy-duty vehicles, which are characterized by slow steering actuator dynamics.

As future work, the method could be extended so as to handle direction switches, as in the case of Reeds-Shepp paths [26], thus allowing for more complex maneuvers and shorter paths. A natural development, is to take into account further limitations on the vehicle, that do not only are related with steering actuator dynamics, but also with safety aspects, such as friction limits, rollover limitations, or comfort, such as lateral acceleration and jerk limitations.

With respect to control design, controllers could also be developed such that they take advantage of the smooth properties of the path. Due to the high quality of the reference paths, simpler controller approaches can be used, which have a lower computational burden when compared to more complex and robust approaches, such as Model Predictive Control. Furthermore the nature of the proposed method, which can be seen as a planner that concatenates together motion patterns computed offline, can allow the computation of attraction funnels of said motion patterns. The attraction funnels allow the planner to take into account the uncertainty and transient errors associated with the controller. By collision-checking the funnels, one

can develop a planner which guarantees safety of the whole system composed of the planner, controller and vehicle [88, 90].

Chapter 4

Smooth Path Planning for Unstructured Environments

Unstructured environments, such as those encountered in agriculture, forestry, mining, and construction industries, are usually characterized by the presence of obstacles and maze-like scenarios. An autonomous truck operating in these environments requires motion planning capabilities, such that it can plan a path to reach a goal destination while avoiding obstacles. Furthermore, in order to achieve the maximum throughput and efficiency, it needs to find the shortest path, out of a multitude of possible ones, to get to its destination.

Unlike the previous chapter, we hereby consider the existence of obstacles that the vehicle must avoid. Obstacles introduce a combinatorial nature to the problem, which motivate the usage of graph search-based planning methods. In order to comply with the kinodynamic constraints of the vehicle, graph search is combined with a state lattice, originating lattice-based motion planners.

A state lattice corresponds to a discretization of the search space, which is based on a regular tiling of motion patterns. These motion patterns are generated taking into account the vehicle constraints, and as such can be feasibly executed. The lattice can then be explored using graph search algorithms, by creating a graph where the edges correspond to the motion patterns and the nodes are the discretized vehicle states.

The discretization of the lattice introduces sub-optimality of the planned paths. In critical cases the discretization can introduce an oscillatory behavior, greatly affecting vehicle performance. Moreover, the state discretization does not allow planned motions to arrive at arbitrary goal locations within the environment. In applications requiring high accuracy, this can hinder the deployment of autonomous heavy-duty vehicles.

To deal with these problems, we introduce in this chapter a new methodology for smooth and accurate motion planning. A lattice-based planner is proposed which interleaves exploration of the graph with smoothing of the solution path. The result is an improved lattice planner that is able to reduce the sub-optimality of solution paths and arrive accurately at desired goal locations.

The contributions of this chapter are the following:

- A formulation and algorithm for the problem of path optimization that makes use of steering methods and heuristics to achieve real-time performance;
- A novel modification of lattice-based planners that alternates between path planning and path optimization in an interleaved way;
- Simulations and experiments with a heavy-duty truck showing the benefits and applicability of the proposed method.

The chapter is organized as follows.

In Section 4.1 we introduce the planning framework, and detail the lattice construction process, as well as the graph search algorithms used. We then detail the challenges that arise with lattice-based planners, and that are the motivation behind the work developed in this chapter.

In Section 4.2 we introduce a path optimization approach that is able to optimize a given solution path, resulting in a smoothened solution with reduced oscillations and length. The optimized solutions are computed using greedy search algorithm, which makes use of the sharpness continuous paths described in Chapter 3.

Section 4.3 details how to use the introduced path optimization step in combination with the lattice planner. Unlike previous approaches, the optimization is performed in between planning cycles, and is used to modify the search graph online. Modifying the graph online improves not only the current solution path, but also the solution paths of future planning iterations.

In Section 4.4 we present simulation and real-work experiments, which quantify the benefits brought forward by the proposed method. The method is shown to reduce the average length of planned paths. However, the biggest advantage comes from the reduction of path oscillations. This allows for smoother rides, and for the vehicle to achieve higher speeds, effectively increasing the efficiency of autonomous HDV operations.

Finally, Section 4.5 presents concluding remarks together with possible avenues of future work.

4.1 Planning framework

In this section we introduce the planning framework used, and present the drawbacks with using a lattice-based motion planner. These drawbacks are the motivation behind the development of a novel path optimization technique.

State lattice planning

Planning in a state lattice consists of a graph search, in which the graph corresponds to a specially discretized state space. Associated to the discretized space, a set of elementary motions is chosen so that it allows neighboring discrete states to be connected over feasible motions [23, 91].

We first define the world space $\mathcal{W}$, as the set of possible states of the system. $\mathcal{W}$ can have an obstacle region $\mathcal{W}_{\text{obs}}$, which denotes the set of all states in $\mathcal{W}$ that lie in one or more obstacles. We then define the obstacle-free space $\mathcal{W}_{\text{free}}$ as the set of states that do not lie in any obstacle, $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{W}_{\text{obs}}$.

Additionally, we define the graph to be searched as $\mathcal{G} = \langle V, E \rangle$. Each vertex $v \in V$ represents a discretized state $q \in \mathcal{W}$. In our case, a state q is defined by a 4-tuple $q = (x, y, \theta, \kappa)$, where x and y represent the position of the vehicle rear axle in a 2D environment, and θ its heading (recall Figure 2.1). κ is the vehicle curvature and is related to its steering angle ϕ as $\kappa = \tan(\phi)/L$, where L is the wheelbase length. Each edge $e \in E$ encodes a feasible motion connecting two states. There is an associated nonnegative cost $l(e)$ to each edge e, corresponding to the cost that our system incurs in, when performing it. The set of possible edges E is defined as to respect the kinematic constraints of the vehicle for which we intend to plan [13] and to enforce curvature continuity, *i.e.*, we require the steering wheel rate to be bounded.

A planning problem is defined by a starting state q_{init}, a goal state q_{goal}, and the obstacle-free space $\mathcal{W}_{\text{free}}$. A valid solution is a sequence of feasible motions, that are collision-free, connecting q_{init} to q_{goal}. The state space can be explored using graph search algorithms that seek to find a sequence of edges connecting the vertex corresponding to q_{init} to the vertex corresponding to q_{goal}. Ideally the search algorithm returns the sequence of edges $\mathcal{E}_S = \{e_1, \ldots, e_N\}$ that does so with the minimum cumulative cost $J(\mathcal{E}_S) = \sum_{i=1}^{N} l(e_i)$. From $\mathcal{E}_S$, a path Π can be created. A path Π corresponds to a sequence of states, that abide by the feasible motions of the vehicle, and can be used to take the vehicle from q_{init} to q_{goal}. One such graph search algorithm is described below.

Lattice Time-Bounded A*

Lattice Time-Bounded A* (LTBA*) [35] is a search algorithm designed to be used in real-time problems. The algorithm runs an A* search in a time-sliced fashion, and it is both real-time and incremental.

LTBA* introduces a mechanism to ensure that planned paths take into account systems that can have considerable momentum. At each planning cycle, the position and speed of the system, together with the previous solution path, are used to predict the future progression of the system. From this, a committed state q_{com} is computed, that corresponds to the state that lies on the solution path before which, the system, at its current speed, is not capable to come to a stop. Once q_{com} has been selected, the algorithm prunes away part of the graph, so that all edges starting from the vertices laying along the current (partial) solution and between the states q_{init} and q_{com} are eliminated. Moreover, the vertices between q_{init} and q_{com} can no longer be used for further exploring the lattice.

In practice, this means that at every cycle the algorithm commits to an initial sub-portion of the provided solution by advancing q_{com} along it. As new information arrives to the planner (*e.g.*, from sensor updates), the current solution can be adapted to the new conditions, but only from the vertex representing q_{com} onwards.

Challenges

Lattice-based motion planners suffer from some shortcomings that can significantly harden the task of finding a high quality solution path. We enumerate here the challenges introduced by the drawbacks of lattice planners.

Excessive steering

One problem with solution paths returned from lattice planners is that they are likely to contain oscillatory behavior. This is a consequence of both the state discretization and the set of elementary motions available. A lattice planner can only produce a resolution optimal solution, where the resolution is limited by the set of elementary motions used. These solutions are most often not optimal if one considers the equivalent problem in the continuous search space.

As an illustrative example consider a simplified lattice as the one shown in Figure 4.1. We want to find the best path taking the vehicle from the start state (arrow q_{init}) to the goal state (arrow q_{goal}). The best path is defined as the one with the lowest cost $J(\mathcal{E}_S)$. A typical choice for the cost of an edge $l(e_i)$ consists in the length of the path that it represents. Figure 4.1 shows two possible solution paths, Π_1 and Π_2 that connect the

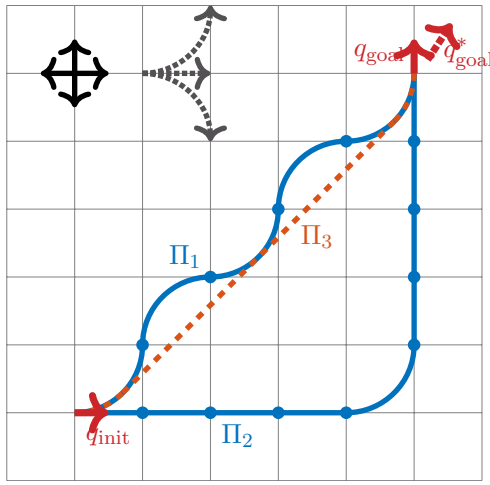


Figure 4.1: A toy example of a simplistic lattice with a positional discretization of one meter and four possible headings $\theta = \{0, \pi/2, \pi, -\pi/2\}$. Three elementary motions (dashed lines at the top left) are allowed, move forward (length 1) and turn left/right (length $\pi/2$). A start state q_{init} and a goal state q_{goal} are shown as filled arrows. The undiscretized goal state q^*_{goal} is shown as a dashed arrow. Two solution paths, Π_1 and Π_2, lying in the lattice, as well as a third path Π_3 that does not follow the lattice are shown.

start and goal states. Path Π_1 has a length $l = 5\pi/2 \approx 7.9$ and path Π_2 has a length $l = 8 + \pi/2 \approx 9.6$.

Since the path Π_1 represents the shortest solution within the set of elementary motions, it is returned as the best solution. Even though it is shorter than path Π_2, it might not be desirable to select path Π_1 as the solution, since it has an oscillatory behavior. Path Π_2 only has one turning motion, at the expense of being longer. This path would have been preferable if one wanted to minimize the oscillations, or maximize the comfort of the ride, two commonly used objectives. Moreover, the straight segments could allow the vehicle to achieve considerable speeds, and arrive at the goal state faster, even though it is following a longer path.

It is possible to prioritize the choice of paths with straight sections by making the edge costs not only a function of the length of the path, but also of the amount of steering required. However, designing such a cost function, *i.e.*, finding the right trade-off between length and steering amount of an edge, is a challenging and time consuming process, and with high subjectivity. Furthermore, such tuning methods are often scenario and ve-

hicle dependent and need to be adjusted to each individual application. These are common problems that arise when dealing with multi-objective optimization where conflicting objectives exist. In our case, the conflicting objectives are the path length and the amount of steering required.

Figure 4.1 also shows a path Π_3 that is arguably better, as it is shorter and with no oscillatory behavior. However, Π_3 cannot be a solution, since it is not contained in the lattice, and therefore cannot be found by the graph search. We note that this path corresponds to a Dubins-like path, more specifically a sharpness continuous path, as introduced in Chapter 3.

Goal state discretization

As previously stated, the search is performed between two states, q_{init} and q_{goal}. Since q_{init} and q_{goal} are limited to the discretized states of the lattice, it becomes impossible to find solution paths to these states, since they are not represented in the lattice.

A common approach when one wants to compute a path to a state q^*_{goal} that lies outside of the lattice, is to simply compute a path to the nearest discretized lattice state. One such example is shown in Figure 4.1, where the desired undiscretized goal state q^*_{goal} is shown as a dashed arrow. Since it does not coincide with any of the discretized states, it is approximated by the closest one, corresponding to q_{goal}. The resulting solution path will then not take the vehicle into the desired state q^*_{goal}, but instead to a nearby state q_{goal}.

This becomes problematic in scenarios which require a high accuracy of the goal state. Driving into a tight parking spot, or trying to stop by a fixed conveyor belt, are applications where it becomes extremely important to achieve an exact desired goal state.

Reducing lattice coarseness

The previous drawbacks can be mitigated by reducing the coarseness of the discretization and increasing the set of available feasible motions. However these strategies will not fully remove these problems, just alleviate them, and they will severely increase the search space the algorithm needs to explore, and consequently its running time.

Due to the limited computing power available in self-driving vehicles, it becomes important to find solutions which are computationally efficient. Thus, reducing the coarseness of discretization, and consequently increasing computational times, is a last resort solution.

In the following sections we propose a solution for these problems that can be implemented while respecting the real-time requirements of online motion planning.

4.2 Path optimization

Here we introduce an algorithm to optimize solution paths computed by a lattice planner. The optimization objective consists in minimizing the path length, which has the indirect effect of reducing path oscillations.

Problem formulation

Given initial and goal states $(q_{\text{init}}, q_{\text{goal}}) \in V \times V$, the lattice planner described in Section 4.1 computes a sequence of vertices and edges

$$\begin{aligned} \mathcal{V}_S &= \{q_{\text{init}}, q_1, \ldots, q_M, q_{\text{goal}}\} \subseteq V, \\ \mathcal{E}_S &= \{e_{\text{init},1}, e_{1,2}, \ldots, e_{M,\text{goal}}\} \subseteq E, \end{aligned} \tag{4.1}$$

that connects q_{init} to q_{goal}. An edge $e_{i,j}$ corresponds to a motion connecting q_i and q_j. Let $J(\mathcal{E}') : \mathcal{E}' \subseteq E \to \mathbb{R}_0^+$ denote the function that maps a sequence of edges $\mathcal{E}'$ to the sum of the cost of each edge. Here we assume that the cost of each edge is the length of the edge, and as such $J(\mathcal{E}')$ becomes the length of the concatenated path formed by them.

We define $\mathcal{V}'_S$ to be an arbitrary subsequence of $\mathcal{V}_S$, and $\mathcal{E}'_S$ to be a sequence of new edges resulting from connecting the vertices in $\mathcal{V}'_S$ over sharpness continuous paths, described in Chapter 3. Given two vertices q'_i and q'_{i+1}, the steering method from Chapter 3 is able to compute a path, *i.e.*, an edge $e'_{i,i+1}$, that connects the vertices with a feasible motion. However, this path computation ignores the existence of obstacles.

The set $\mathcal{W}_{\text{free}}$ represents the obstacle-free space in which vehicle states along the path of an edge must be. An edge is considered collision-free if the vehicle states along its path are all collision-free. We use the notation $e_{i,j} \in \mathcal{W}_{\text{free}}$ to indicate that an edge $e_{i,j}$ is collision-free. Furthermore q^*_{goal} corresponds to the undiscretized goal state.

The discrete optimization problem to solve can be stated as follows:

Problem 1 *Given the states $q_{\text{init}}, q^*_{\text{goal}}$ and a solution to the path planning problem $(\mathcal{V}_S, \mathcal{E}_S)$ find a sequence of states $\mathcal{V}'_S = \{q'_1, q'_2, \ldots, q'_N\}$ and a sequence of paths $\mathcal{E}'_S = \{e'_{1,2}, e'_{2,3}, \ldots, e'_{N-1,N}\}$ connecting the states $\mathcal{V}'_S$, such that:*

$$\begin{aligned} \underset{\mathcal{V}'_S \subseteq \mathcal{V}_S}{\textit{minimize}} \quad & J(\mathcal{E}'_S) \\ \textit{subject to} \quad & e'_{i,i+1} = SteeringMethod(q'_i, q'_{i+1}) \\ & q'_1 = q_{\text{init}} \\ & q'_N = q^*_{\text{goal}} \\ & \mathcal{E}'_S \in \mathcal{W}_{\text{free}} \end{aligned}$$

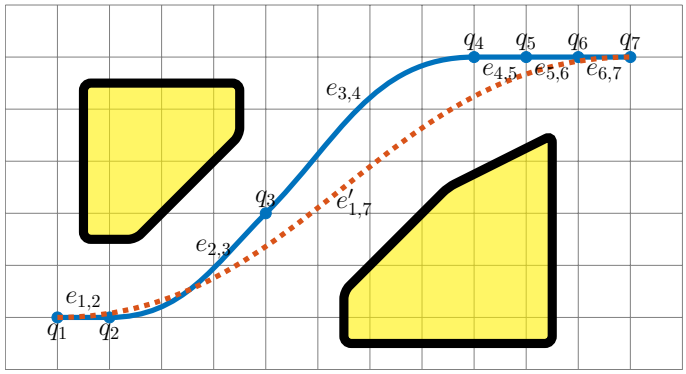


Figure 4.2: An example solution path as the filled line (blue). The path is composed of motion primitives connected over intermediate nodes (filled circles). A possible path resulting from a steering method connecting the initial and final nodes is shown as the dashed line (red). $e_{1,2}, e_{2,3}, \ldots, e_{6,7}$ represent the edges connecting the intermediate nodes, as provided by the lattice planner. $e'_{1,7}$ shows a path generated by the steering method, that connects states q_1 and q_7. The polygons are obstacles that the solution path must avoid.

The objective of solving Problem 1 is to find a subset $\mathcal{V}'_S$ of $\mathcal{V}_S$ that when connected through the paths $\mathcal{E}'_S$, minimizes the length of the edges $J(\mathcal{E}'_S)$. An edge $e'_{i,i+1}$ which is part of the sequence $\mathcal{E}'_S$ corresponds to a sharpness continuous path (Chapter 3) between vertices q'_i and q'_{i+1}. The sequence $\mathcal{E}'_S$ must also be collision-free, *i.e.*, all of its edges must be collision-free, $e'_{i,i+1} \in \mathcal{W}_{\text{free}}$.

The cost function to minimize is selected to be the length of the path $J(\mathcal{E}'_S)$. One possible alternative would be to minimize the steering or the oscillations of the path, as that is the main drawback of lattice planner solutions. However, it is also noticed in practice, that optimizing with respect to the length of the path results in decreased oscillations.

It is worth noting that, even though solutions to Problem 1 are collision-free, it might be of interest to have obstacle clearance as an optimization objective. This would result in paths that maximize the safety margin to obstacles.

Figure 4.2 shows an example of a possible problem instance as described in Problem 1. A path that is computed by the lattice planner (filled line), is composed of several intermediate states (circles), $q_1, q_2, \ldots, q_7$, that are connected over edges $e_{1,2}, e_{2,3}, \ldots, e_{6,7}$. After the optimization, a possible solution path (dashed line) is one that connects states q_1 and q_7 directly. In

this particular case, the optimized path is the result of applying the steering method between states q_1 and q_7, and its corresponding edge is denoted as $e'_{1,7}$.

The optimal solution for Problem 1 can be found using a brute-force method, *i.e.*, by evaluating all possible and valid solutions existing in the search space. However this type of approach is computationally expensive, as the number of possible combinations has dimension 2^n, where n is the number of elements in the set $\mathcal{V}_S$. In fact, for a sequence of states $\mathcal{E}_S$ with n states, a brute-force approach will require at least $O(2^n)$ steering method queries and its respective collision checks. This makes it unsuitable for our application, in which computations must be kept to a minimum, in order to maintain real-time capabilities.

Greedy optimization

In this section, an algorithm that finds a sub-optimal solution to Problem 1 is proposed. One important aspect of the algorithm is its computational time, which needs to be kept low in order to comply with real-time requirements. We thus require the algorithm to be time-bounded. Also, we will have the opportunity to replan in future planning cycles, so it is beneficial to prioritize optimization in the path edges closer to the current position of the vehicle, which will be executed first. Edges that are farther away can be optimized in later planning cycles. Taking into account these aspects we formulate Algorithm 2.

Algorithm 2 takes as input the sequence of states $\mathcal{V}_S$ corresponding to the lattice planner solution, and the obstacle-free space $\mathcal{W}_{\text{free}}$. The algorithm will run for a limited time ΔT_{limit} (line 4), after which the *while* loop is exited. Alternatively it can finish before, if it considers the algorithm complete (line 18). In a first step the algorithm tries to generate a feasible path between the first state q_1 to all the states in $\mathcal{V}_S$ that follow it. The path is generated making use of the steering method (line 7) and for it to be valid it needs to be collision-free (line 8). The farthest state from q_1 that it was possible to connect to, is then selected as part of the solution, and both the state and the edge are added at the end of sequences $\mathcal{V}'_S$ and $\mathcal{E}'_S$ (lines 13-14). The process then restarts, but now it tries to connect from the farthest node, to all of the following nodes. In the worst case, each state tries to connect with all following states, resulting in a complexity of $O(n^2)$ in the number of steering attempts.

Remark 1 *It can happen that $\mathcal{V}'_S$ does not include q_N, this means that the algorithm could not compute a feasible path to q_N, either due to infeasibility or shortage of computation time. If this happens, a path that ends up in q_N can still be obtained, by concatenating the original states and edges contained*

Algorithm 2: Greedy Optimization Algorithm

Input: $\mathcal{V}_S = (q_1, \ldots, q_N)$, $\mathcal{E}_S = (e_{1,2}, \ldots, e_{N-1,N})$, $\mathcal{W}_{\text{free}}$, ΔT_{limit}
Output: $\mathcal{V}'_S$, $\mathcal{E}'_S$

```
1  V'_S ← (q_1);
2  E'_S ← ( );
3  i ← 1;
4  while ΔT_limit do
5      success ← false;
6      foreach q_j ∈ V_S \ (q_1, ..., q_{i-1}, q_i) do
7          e_{i,j} ← SteeringMethod(q_i, q_j);
8          if e_{i,j} ∈ W_free then
9              e_{i,k} ← e_{i,j};
10             k ← j;
11             success = true;
12     if success then
13         V'_S ← V'_S + q_k;
14         E'_S ← E'_S + e_{i,k};
15         i ← k;
16     else
17         if V_S \ (q_1, ..., q_{i-1}, q_i) = {} then
18             break;
19         else
20             V'_S ← V'_S + (q_i, q_{i+1}, ..., q_N);
21             E'_S ← E'_S + (e_{i,i+1}, ..., s_{N-1,N});
```

in the original lattice solution $\mathcal{V}_S, \mathcal{E}_S$ that come after the last state in $\mathcal{V}'_S$ (lines 20-21).

By designing the algorithm in this way, we can incrementally improve the original solution throughout consecutive planning cycles. And we do so prioritizing the nearest states of the path, and complying with time limitations. Furthermore, in the worst case, when the algorithm is not able to generate any improvements, the resulting path will simply be the original lattice planner solution.

Reducing the computational complexity

Problem 1 can have a very large search space ($2^{\mathcal{V}_S}$ possible solutions). Reducing the search space will reduce the problem complexity. The search

space can be reduced by decreasing the number of elements in $\mathcal{V}_S$ over which we optimize. This forms a new of sequence of elements $\mathcal{V}_S^*$ which will correspond to the new decision variables of our discrete optimization problem. However, doing this removes a number of possible valid solutions to Problem 1, and possibly removes the optimal or near-optimal solutions from the search space, which is not desirable.

Taking into account the trade-off between complexity and the possibility of removing desired solutions, Algorithm 3 is proposed, which reduces the original search space $2^{\mathcal{V}_S}$, while trying to keep the expressiveness of the original problem, in the now reduced search space $2^{\mathcal{V}_S^*}$.

Algorithm 3: State Set Reduction

Input: $\mathcal{V}_S = \{q_1, \ldots, q_N\}$, $l_{\min}$
Output: $\mathcal{V}_S^*$

```
1 V_S^* ← q_1;
2 k ← 1;
3 for i = 2 to N − 1 do
4     if Distance(q_k, q_i) > l_min then
5         V_S^* ← V_S^* ∪ q_i;
6         k ← i;
7 V_S^* ← q_N;
```

The function $Distance(q_k, q_i)$ (line 4) is defined as the euclidean distance in the (x, y) position between states q_k and q_i, $||(x_k - x_i, y_k - y_i)||_2$. In essence, Algorithm 3 loops over the original set $\mathcal{V}_S$ and removes states that are near each other, *i.e.*, at a distance smaller than $l_{\min} \in \mathbb{R}_0^+$ (line 4).

Algorithm 3 is designed taking into account a practical observation. Assuming Π_1 to be a path generated by the steering method [14] between states q_a and q_b, and in an obstacle-free environment. And assuming a path Π_2 between states q_a and q_c generated in the same conditions. Then it observed experimentally that the smaller the value of $Distance(q_b, q_c)$, the more similar (*i.e.* smaller Hausdorff distance) the paths Π_1 and Π_2 are.

Using the intuition provided by the previous experimental observation, Algorithm 3 removes states from $\mathcal{V}_S$. The removed states are expected to not alter the search space $2^{\mathcal{V}_S}$ drastically. Assuming that a state q_i is removed from $\mathcal{V}_S$ by Algorithm 3, then there exists another state q_k nearby q_i, that allows the reduced search space $2^{\mathcal{V}_S^*}$ to contain solution paths which present a high similarity (measured as the Hausdorff distance) to those that were removed from the original search space $2^{\mathcal{V}_S}$ when removing q_i. There are however exceptions, as certain states in $\mathcal{V}_S$ might be crucial to guarantee

a collision-free path, even though they might be very close to other states in $\mathcal{V}_S$.

4.3 Interleaving graph search and path optimization

In this section, we propose a novel way to interleave path optimization, as the one described in Section 4.2, with the LTBA* described in Section 4.1. We explain why this is not directly implementable, and propose a modification to LTBA* in order to achieve interleaved planning and optimization.

Problem statement

At the end of each planning cycle, the lattice planner returns a solution path as a sequence of states $\mathcal{V}_S$ and edges $\mathcal{E}_S$. This solution is then fed into the optimization algorithm described in Section 4.2, resulting in a new sequence of states $\mathcal{V}'_S$ and edges $\mathcal{E}'_S$. The structure of the search graph is then altered, such that the original states $\mathcal{V}_S$ and edges $\mathcal{E}_S$ are replaced by the optimized states $\mathcal{V}'_S$ and edges $\mathcal{E}'_S$.

As an illustrative example, assume that a given optimization procedure resulted in a $\mathcal{V}'_S$ which only contains the first and last element of $\mathcal{V}_S$. This means that the solution obtained from the optimization step, is simply the path computed by the steering method between the start state q_1, and end state q_N of $\mathcal{V}_S$. $\mathcal{E}'_S$ corresponds then to a single edge connecting the two states in $\mathcal{V}'_S$. Additionally, assume that states q_1 and q_N are located far apart, meaning that the edge connecting them will have a large length. According to the mechanisms of LTBA* explained in Section 4.1, the committed state q_{com} is at least q_N or a state after it. Since planned path solutions from subsequent planning cycles must include q_{com}, this means that the vehicle is forced to commit to the whole path, and not only to a portion of it. This results in reduced capability for replanning and maneuvering around newly sensed obstacles. To solve this issue, it is necessary to ensure that when changing the structure of the search graph, edges with large lengths are prohibited.

Escape vertices

In order to avoid the creation of edges with large lengths, we introduce a mechanism to split an arbitrary path into several short edges. We introduce the concept of escape vertices, which are vertices that are created in order to split a long optimized path into multiple edges. The mechanism of creation of escape vertices is presented in Algorithm 4.

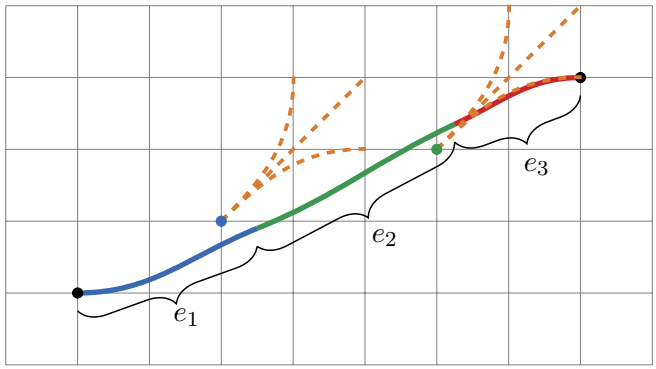


Figure 4.3: Splitting an optimized path in order to create escape vertices. The sub-segments e_1, e_2, e_3 of the optimized path become coupled to the nearest escape vertices. A set of edges resulting from the graph search expanding the escape vertices is shown as dashed orange branches.

Algorithm 4: Finding Escape Vertices

Input: e', d_{min}, d_{Δ}
Output: $\mathcal{V}_{\text{esc}}, \mathcal{E}_{\text{esc}}$

```
1  l_e' ← Length(e');
2  d_last ← 0;
3  for d = 0, d_Δ, 2d_Δ, ..., l_e' do
4      if d − d_last > d_min then
5          q_path ← StateAtPathLength(e', d);
6          q_esc ← UnexpandedVerticeNearby(q_path);
7          if q_esc ≠ null then
8              e ← PathSection(e', d_last, d);
9              d_last ← d;
10             V_esc ← V_esc ∪ q_esc;
11             E_esc ← E_esc ∪ e;
```

Algorithm 4 tries to split an edge e' resulting from the optimization in Section 4.2, into a sequence of shorter edges $\mathcal{E}_{\text{esc}}$. First it computes the length of the path encoded in e' (line 1), and then it iterates over the path states (line 5) at regular distances with increments of d_{Δ} (line 3). For a given path state q_{path}, it finds a nearby lattice vertex q_{esc}, that has not yet been expanded, *i.e.*, it has no child vertices (line 6). In case it has found q_{esc} (line 6), it then proceeds to add it to $\mathcal{V}_{\text{esc}}$ (line 10). It also creates an e_{esc}, corresponding to the section of the original path, evaluated from distance

d_{last} to d (line 8), and adds it to $\mathcal{E}_{\text{esc}}$ (line 11). By updating (line 9) and checking (line 4) the last distance d_{last} at which the path was split, it avoids creating escaping edges that are too short, *i.e.*, with a length smaller than $d_{\min}$.

An arbitrary path might not overlap with any of the discretized lattice vertices, as is the case of the path shown in Figure 4.3. Thus, it is likely that Algorithm 4 will create edges that do not connect discretized lattice vertices with a feasible motion, *i.e.*, one that respects the system model. If the graph search then expands these vertices, the resulting edges will be discontinuous, as illustrated by the orange branches in Figure 4.3. This violates the assumption of the lattice search, and can result in solution paths that are discontinuous. To deal with this it is necessary to introduce a few modifications to the original LTBA*.

The search graph $\mathcal{G}$ is updated by the path optimization step, where the original lattice planner solution states $\mathcal{V}_S$ and edges $\mathcal{E}_S$, are replaced by the optimized states $\mathcal{V}'_S$ and edges $\mathcal{E}'_S$. However the new states $\mathcal{V}'_S$ are marked as being escape vertices. When a vertex is marked as an escape vertex, it is known that the edge associated to it does not respect the lattice continuity. As such, whenever a potential solution path is about to be returned at the end of the planning cycle, it is checked for path continuity. The path continuity can be checked by evaluating its vertices, and in case they are escape vertices, checking if their connecting edges are discontinuous. In case they are, the whole path is invalidated, and an alternative solution path is requested. This is an existing feature of the LTBA* algorithm, implemented in order to deal with the appearance of new obstacles [35]. We note that this is a very rare occurrence in the scenarios used in our experiments.

4.4 Results

This section presents results of the previously proposed methods. Computational times are obtained using the computing unit in a prototype autonomous heavy-duty vehicle for industrial applications. A selected planning instance is also presented and is obtained from real tests making use of the autonomous heavy-duty vehicle shown in Figure 4.4.

Problem setup

We generate 5000 randomized test scenarios corresponding to typical planning problems. A planning problem is characterized by the initial vehicle state, a goal state at which the vehicle should arrive, and a set of obstacles to avoid. One possible random problem instance is shown in Figure 4.5. To generate a random planning problem, the initial and goal states are ob-

Figure 4.4: An autonomous heavy-duty vehicle for which the work presented is developed and tested on.

tained by sampling from uniform distributions x and y coordinates and a heading θ. The coordinates are sampled from a distribution with a width of 100 meters, and the headings from a distribution $[0, 2\pi[$. Three random square obstacles are also generated, and their location follows the same uniform distribution with a width of 100 meters. The lattice planner and path optimization are required to run in a continuous loop, at a frequency of 2 Hz.

Path optimization

We compare the proposed Greedy Optimization Algorithm (*Greedy*) against a brute-force method (*Brute*) as solvers for the discrete optimization problem stated in Problem 1. For each test scenario we get the lattice planner solution and corresponding states $\mathcal{V}_S$, and then solve Problem 1 using both methods. We measure the computation time Δt and solution cost $J(\mathcal{E}'_S)$ of both methods.

Table 4.1 shows the comparison of both methods for 5000 random planning instances, in which the original lattice planner solution has an average length $J(\mathcal{E}_S) \approx 80$ meters. The minimum distance parameter $l_{\min}$ in Algorithm 3 is set to 10 meters. We define $\delta_J(\mathcal{E}_S, \mathcal{E}'_S)$ as the normalized difference between the lattice solution length $J(\mathcal{E}_S)$, and the optimized solution length $J(\mathcal{E}'_S)$:

$$\delta_J(\mathcal{E}_S, \mathcal{E}'_S) = \frac{J(\mathcal{E}'_S) - J(\mathcal{E}_S)}{J(\mathcal{E}_S)}. \tag{4.2}$$

The smaller $\delta_J(\mathcal{E}_S, \mathcal{E}'_S)$, the greater the improvement in terms of path length reduction. The computation time Δt corresponds to the amount of time required for the algorithm to run.

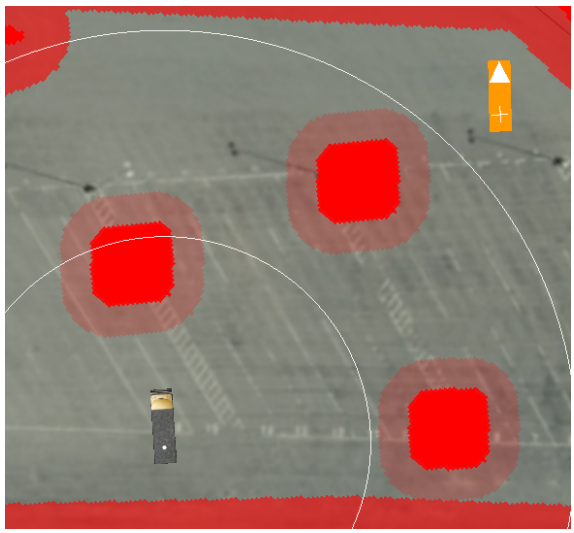


Figure 4.5: An example of a planning problem. From the current state (vehicle location), find a path to a goal state (highlighted in orange), while avoiding obstacles (regions in red).

Table 4.1: Comparison of *Brute* and *Greedy* results for 5000 random planning problem instances. The theoretical complexity O of steering method calls with respect to the number of states n is also presented.

Measure	*Brute*	*Greedy*
Average $\delta_J(\mathcal{E}_S, \mathcal{E}'_S)$ [%]	-4.0886	-3.9749
Standard deviation $\delta_J(\mathcal{E}_S, \mathcal{E}'_S)$ [%]	3.1508	3.1574
Average $\Delta t[s]$	0.1835	0.0058
Standard deviation $\Delta t[s]$	0.3375	0.0034
Theoretical complexity	$O(2^n)$	$O(n^2)$

The results presented in Table 4.1 show that *Brute* achieves on average shorter paths than *Greedy*. Even though the *Greedy* results are worse than the *Brute* ones, they are still fairly similar. Comparing the computational times Δt of both algorithms shows a clear advantage for *Greedy*. It can be seen that *Greedy* requires much less time (two orders of magnitude smaller) than *Brute*.

From the observed results, we conclude that *Greedy* finds solutions with a cost $J(\mathcal{E}'_S)$ comparable to that of *Brute*, however it does so in a fraction of the computational time required by *Brute*.

Interleaved planning and optimization

In this experiment we run the LTBA* and the Greedy Optimization Algorithm (Algorithm 2), in an interleaved fashion, as explained in Section 4.3. For each test scenario, we compare the solution path obtained from the original LTBA* implementation [35] with the solution path obtained from our proposed interleaved planner. The results are presented in Table 4.2.

Table 4.2: Average results from 5000 random planning instances.

Metric	Original path	Optimized path
Average path length [m]	76.54	73.51
Average straight length [m]	29.03	53.65
Average number of steering changes	12.42	4.19

From Table 4.2 it can be observed that the mean length of the original solution paths is around 77 meters. When using the optimization procedure, the mean length decreases to 74 meters. The optimization is thus, on average, successful in shortening the length of the solution paths.

We introduce the straight length metric, which is a measure of the length of the path corresponding to straight driving, *i.e.*, with zero curvature. We see that the original paths have on average 29 meters of straight section. The optimized paths are able to greatly increase this length to 54 meters. This corresponds to paths that have more sections of straight driving, which is desirable, since it relates to more comfort, resembles human-like driving, and allows for increased driving speeds.

A third metric that is measured is the number of steering changes. This metric corresponds to the number of times that a vehicle will have to change between left, right, or null steering when following a path. The original paths present an average of 12 steering changes, while the optimized paths greatly reduce this number to 4 steering changes. Such a drastic reduction shows that the paths have a reduced number of turning maneuvers that the vehicle performs, and by consequence, reduced oscillatory behavior.

Goal state reachability

We also study if the interleaved planner solution paths are able to deal with the problem of goal state discretization mentioned in Section 4.1. We measured that 85% of the interleaved planner paths arrived at the exact goal state q^*_{goal}, instead of at the discretized goal state q_{goal} corresponding to the closest neighbor in the lattice. The other 15% end up in the discretized lattice approximation q_{goal}, like regular lattice-based planners. Ideally the paths would always arrive at the exact goal state q^*_{goal}. However, due to the

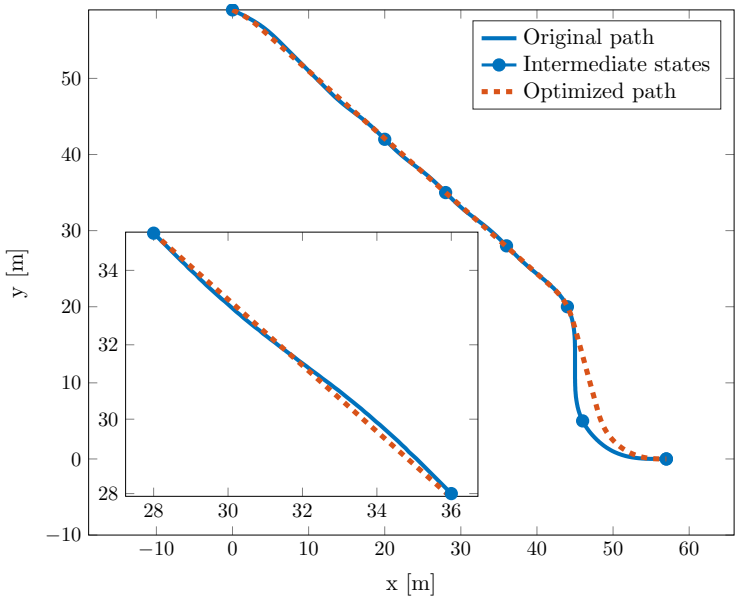


Figure 4.6: An example of a planned path (filled line) and the resulting optimized path (dashed line). The circles correspond to intermediate states. A view of a zoomed section highlights the excessive steering of the original non-optimized path.

greedy nature of the optimization, which prioritizes optimizing edges in the beginning of the path, it is the case that the path is sometimes optimized without successfully connecting to q^*_{goal}. A possible improvement is to also try to optimize from q^*_{goal} backwards, to try to increase the success rate with which optimized path solutions arrive exactly at q^*_{goal}.

Illustrative example

Figure 4.6 shows a selected problem instance. The paths seem quite similar, but there is a drastic reduction of the number of steering changes from 20 in the original path (filled line), to 6 in the optimized path (dashed line). Even though the original path seems quite straight, it has several small oscillations, resulting from the drawbacks of lattice-based motion planners detailed in Section 4.1. Our proposed interleaved planner is particularly good at improving the path quality in this aspect.

4.5 Conclusions

This chapter presented a novel motion planning framework that was evaluated on an autonomous heavy-duty vehicle. The methods introduced in this chapter are motivated by the shortcomings of lattice-based planners, namely sub-optimality of the solution path and goal state discretization. Both problems are a direct consequence of the search space discretization imposed by the lattice.

The proposed solution combines lattice-based motion planning with path optimization using sharpness continuous paths introduced in Chapter 3. Through extensive simulations and real-life experiments we show that the proposed method is successful in addressing both shortcomings of lattice-based planners. The planned paths present a drastic reduction in the number of steering changes, and a significant increase in the amount of straight driving. This results in reduced oscillatory behavior, which leads to more confortable driving, and the possibility of achieving higher driving speeds. Moreover, the optimized solution paths are on average shorter, compared to solutions computed by lattice-based motion planners alone.

The path optimization step is formulated as a discrete optimization problem, and we propose a greedy algorithm to solve it. It is shown that the greedy algorithm finds solutions with comparable quality to brute-force methods, with the advantage of performing the computations in a fraction of the time. The low computational times make it amenable to be implemented in systems which must perform motion planning online. To further reduce computational times, the optimization is done in an interleaved fashion in between the planning cycles, instead of as a final planning step, as is common in other approaches. This allows the optimization results to be reused in subsequent planning cycles, without the need to recompute them again.

As future work, one could refine the algorithms used to find a solution to the discrete optimization problem. To increase the capability of the path optimization to arrive at the undiscretized goal state, it is worth studying a greedy optimization algorithm that not only optimizes from the starting state forwards, but also from the true goal state not restricted to the lattice backwards. Since the process of interleaving the graph search with path optimization changes the graph structure, completeness guarantees of the search algorithm might be lost. It is of interest to study these mechanisms further, to understand how they influence the efficiency and completeness properties of the underlying graph search.

Chapter 5

Optimization-based On-road Path Planning for Buses

This chapter deals with path planning for buses driving in tight on-road environments. We focus the developed solution to the specific case of public transportation via city buses. Here, we introduce a new methodology for path planning, based on numerical optimization techniques. This technique proves to be very suitable for the particular challenges buses face when driving in urban environments.

Driving in cities is often characterized by chaotic and packed traffic, which can stress even experienced drivers. The situation becomes even more complicated when considering buses that must share the road with other vehicles and road users. It is often the case that a bus needs to make its way through very constrained spaces, *i.e.*, sections of the road where the distance to obstacles or other vehicles is very small. Such situations can be caused by narrow lanes, vehicles temporarily parked on the sides of the road, or maneuvering in and out of bus stops.

Motion planning algorithms for this type of application must be able to find solutions in constrained environments. Algorithms such as lattice-based planners and rapidly exploring random trees (RRTs) often have difficulties with environments where the clearance to obstacles is small. This is due to the inherent state space discretization of the lattice, and to the low probability of sampling collision-free states in RRTs.

On the other hand, numerical optimization methods conveniently deal with this problem. Due to the continuous nature of optimization methods, it is possible to find solutions even in highly constrained environments. Furthermore, optimization methods allow for a direct encoding of the vehicle model, and are characterized by solutions with a high degree of smoothness.

Unlike any other vehicle class, buses have a special design, where the

chassis extends quite far way from its own wheels. The part of the chassis extending beyond the wheels is referred to as overhangs, and has a significant height, allowing it to sweep over curbs. This design choice is based on optimizing the vehicle passenger capacity, while maintaining maneuverability and stability of the vehicle body.

Making the most out of this particular vehicle design, professional bus drivers often allow overhangs of the vehicle to go over curbs and low height obstacles. It is essential to mimic this type of behavior if one is to deploy autonomous buses for city driving applications. Our proposed solution is successful in replicating this behavior, and is shown to increase the maneuvering capabilities of the bus.

The contributions of this chapter are the following:

- tackles the challenging task of bus driving in urban environments, taking full advantage of the overhangs of buses to sweep over curbs and low height obstacles;
- uses a new approximation technique for the distortions introduced by the road-aligned frame and that affect the vehicle body and obstacles;
- considers the distinct chassis configuration of buses, distinguishing between the overhangs and the wheelbase of the vehicle;
- takes into account three distinct types of surface: obstacle, sweepable, and drivable regions.

The chapter is organized as follows.

Section 5.1 introduces the problems with planning in constrained environments. We present some motion planning techniques that have been used in the literature to deal with constrained urban environments. Finally, we highlight some motion planning solutions that have resorted to numerical optimization techniques.

Section 5.2 details the spatial-based road aligned vehicle model, which is particularly suited for on-road driving and optimization techniques. We also present the main drawback of this model, related to the distortion of the vehicle body. This drawback is addressed via new vehicle body approximation techniques, which ensure obstacle-free solutions that are not conservative in their assumptions.

Section 5.3 formulates the path planning problem as a numerical optimization. A novel region classification scheme is introduced, which enables motion planning algorithms to take full advantage of the increased maneuverability made possible by overhangs. Furthermore, we propose optimization objectives which encode the professional bus driver behavior that we desire to mimic.

Section 5.4 presents simulation results of the proposed method. The benefits of the newly proposed optimization objectives are shown, and motivated based on noticing that resulting maneuvers are safer. Furthermore, we show a scenario where explicitly taking into account the overhangs proves to be the only way to successfully progress along the road.

Concluding remarks and possible future work directions are given in Section 5.5.

5.1 Introduction

We have seen in Chapter 4 a solution that satisfies the planning needs of an HDV, however, here we introduce a new motion planner that is based on a different framework. The motivation behind the development of a new motion planner relates to the fact that the lattice-based planner used previously is not suited for the considered application. For on-road driving applications, particularly for the case of buses, lattice-based planners as the one described in Chapter 4, become unusable due to two main reasons: solution discretization and collision checking approximations.

Solution discretization

Lattice planners rely on a placing a lattice on top of the environment, and then finding a sequence of edges that arrive at the goal in a collision-free way. This methodology usually works well in environments where the free space is somewhat large in comparison to the lattice discretization, *i.e.*, the positional spacing between the lattice states.

To understand the problem of discretization, we start by overlaying a lattice on top of an on-road environment, as shown in Figure 5.1. It can be seen that this lattice does not allow one to find a obstacle-free path that takes the vehicle along the road. This is caused by the coarseness of the lattice discretization, which, even though suited for the unstructured environments considered in Chapter 4, becomes unusable for on-road scenarios.

A possible solution is to increase the resolution of the lattice by more finely discretizing the state space. However this comes at the cost of a tremendous increase in the state space considered, which significantly increases the computational complexity of the algorithm.

Collision checking

Collision checking is an essential part of motion planning algorithms, and often corresponds to a significant share of the computational times [93]. In essence, collision checking is responsible for evaluating if a vehicle state, or

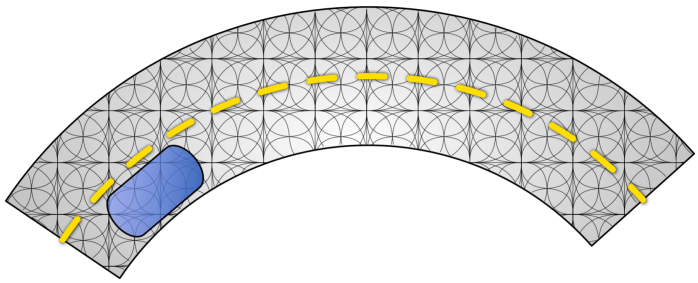


Figure 5.1: A lattice on top of a road (from [92]).

sequence of states along a path, does not collide with obstacles detected in the environment. Collision checking methods are not limited to checking if a vehicle state is colliding. They can also be used to check if vehicle states are contained inside the road. In Chapter 4 the planner makes use of a collision checker based on grid discretization. This type of collision checking is one of the most popular ways of collision checking, due to its simplicity and good performance. However, it comes with the drawback of being conservative, which makes it unsuitable for highly constrained environments, such as when driving a bus in urban scenarios.

In a first step the environment is discretized into a grid, which can be composed of squares, or other shapes that are more convenient. Each cell in this grid is classified according to the contents inside itself. We assume here a simple case where each cell is classified as either inside the road or outside of it. To classify each cell, one looks into its spatial contents. A cell is deemed outside of the road if any region of it exits the road boundaries. A cell is only considered inside if its completely contained inside the road. Using this methodology one arrives at the classification illustrated in Figure 5.2.

Once the environment is discretized and classified, it is then necessary to check if the vehicle state is inside the road. In a similar procedure to the one mentioned before, a vehicle mask is created, which corresponds to the occupancy of the vehicle in the discretized grid, as shown in Figure 5.2. To then understand if the vehicle state is inside the road, one simply checks that none of the cells in the vehicle mask correspond to an cell outside of the road.

Figure 5.2 shows how a vehicle state that is inside the road, can be wrongly classified as outside of it due to the discretization effects of the grid. This problem can be addressed by increasing the resolution of the grid, however that comes with an increased computational cost. If the target application requires finely discretized grids, then this procedure becomes too time consuming. Bus driving in urban scenarios is one such application,

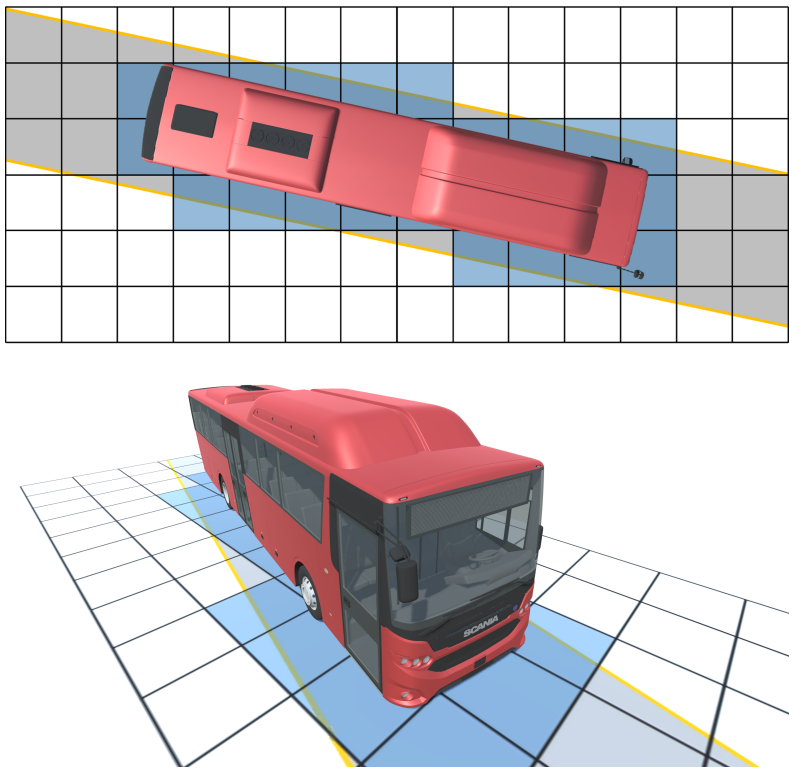


Figure 5.2: Collision checking a bus state using an occupancy grid. The discretization of the grid can cause bus states to be wrongly classified as outside of the road. This effect is worsened when considering buses driving on narrow roads. Bird's-eye (top) and perspective (bottom) views.

where it is of extreme importance to have precise collision checking, since the roads are very narrow for the dimensions of the vehicle considered.

Motion planning in constrained environments

Due to the challenges mentioned before, motion planning in constrained environments has been the subject of research. The approaches found in the literature show that traditional planning algorithms often have to be adjusted in order to deal with this type of scenario.

In [94], the authors make use of RRTs for planning the motion of a vehicle. Realizing that narrow roads introduce a challenging situation for the motion planner, the authors propose several algorithmic changes that

make it suitable for road driving. One of these changes is related to the randomized sampling of new states. Instead of uniformly sampling the new states, the sampling procedure is biased along the center of the lane. This reduces the amount of samples that need to be discarded due to collision, thus saving computational time.

Graph search algorithms can also be used for dealing with constrained environments. In [95] the authors propose extensions to the A* graph search algorithm that allow the planner to deal with challenging urban driving situations. The discretization problem is avoided by making use of a pure-pursuit expansion, which simulates a vehicle that is steered by a path-tracking controller. The controller tries to steer the vehicle towards the center of the lane, thus avoiding the search from straying into invalid collision states. This comes however with the loss of the original completeness guarantees of the A* algorithm. The algorithm is experimented in practice, showing its ability to perform in real time.

The state lattice framework can still be used for on-road driving, however it must be adjusted to the road shape. In [96, 97], a lattice is created by discretizing the state space at regular positions along the road. The vertices of the lattice are created by sampling along the road length and at different lateral offsets from the lane center. This makes the lattice adjust to the shape of the road, however it forces the lattice to be computed online, thus increasing computational times. As opposed to the approach described in Chapter 4, in on-road driving scenarios one cannot pre-compute state lattices. Thus, it becomes necessary to have a fast method that is able to compute the motion primitives of the lattice online.

Dynamic programming is used in [98] to plan vehicle maneuvers in tight environments. The proposed solution is shown to successfully compute complicated parking maneuvers in a computationally efficient way. The algorithm benefits from a varying discretization of the configuration space, where finer discretization is used where more accurate maneuvers are expected. However, the amount and resolution of the finer discretization is decided *a priori*, using human intuition. The approach is implemented on a passenger vehicle, and the planned paths are shown to be easily tracked by a simple linear controller.

Motion planning using numerical optimization

The works mentioned before try to tackle the inherent problem associated with planners that discretize the state space. The inherent discretization forces these algorithms to consider an exponentially larger search space when trying to achieve the granularity needed to plan in constrained environments. On the contrary, numerical optimization frameworks do not require

discretization of the state space, thus being an attractive choice when considering constrained environments. Furthermore, optimization approaches are often characterized by the smoothness of the solutions and benefit from a straightforward encoding of the vehicle model [99].

In [60], trajectory planning is done resorting to nonlinear optimization methods. Collision checking is implemented by modeling the obstacles as polygons. The proposed solution is used on a real vehicle, and driven along a 103 km route. The trajectory planner is shown to be able to deal with situations where there is little free space available.

A combined trajectory planning and control approach is presented in [100]. The problem is formulated as a nonlinear model predictive control method, where a special solver is used that exploits the sparsity of the MPC problem. Simulation results show its effectiveness even in challenging emergency maneuvers.

Trajectory planning can also be formulated as sequential linear programming, as in [101], which targets the case of passenger vehicles. The approach makes use of the road-aligned vehicle model, a model particularly suited for road driving. One of the drawbacks of the model is that it introduces distortions in the vehicle bodies, often complicating the formulation of collision avoidance constraints.

To deal with the distortion of vehicle bodies introduced by the road-aligned vehicle model, [16] introduces approximations seeking to ensure paths that are collision-free. Although safe, the approximations are too conservative, and the planner might fail in highly constrained environments. This work seems to be the first to address the challenges faced by buses when driving in urban environments.

In this chapter, we build upon [16], and introduce a novel approximation method for vehicle body distortions, developing a path planner that guarantees safe collision-free solutions, without conservative approximations. Furthermore, we develop new optimization objectives that seek to minimize the amount of overhang being swept by the vehicle, which is a common concern in bus driving. The result is a motion planner that is able to mimic the behavior of professional bus drivers.

5.2 Modeling

We introduce the road-aligned vehicle model used and propose a new approximation for the vehicle body, which deals with distortions introduced by the road-aligned frame.

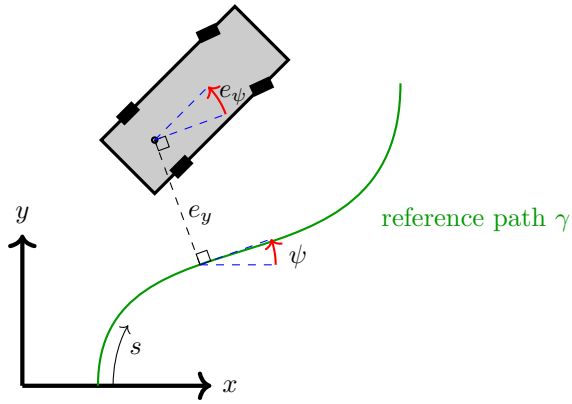


Figure 5.3: Global and road-aligned frames. Vehicle states (s, e_y, e_ψ) on the road-aligned frame, are defined with respect to the reference path γ.

Vehicle model

We describe the vehicle state evolution using the space-based road-aligned vehicle model used in [16]:

$$\begin{aligned} e'_y &= \frac{\rho_s - e_y}{\rho_s} \tan(e_\psi), \\ e'_\psi &= \frac{(\rho_s - e_y)\kappa}{\rho_s \cos(e_\psi)} - \psi'_s. \end{aligned} \tag{5.1}$$

The model describes the vehicle state using a frame that moves along a reference path. This model is chosen since it allows for the convex formulation of common on-road optimization objectives and is independent of time.

As shown in Figure 5.3, the road-aligned vehicle states are given by (s, e_y, e_ψ) corresponding to the distance along the reference path s, the lateral displacement e_y, and the orientation difference e_ψ between the vehicle heading and the path heading ψ. These states are defined w.r.t. a reference path γ. The vehicle is controlled by input u corresponding to the vehicle curvature, which is related to the steering wheel angle ϕ as $u = \tan(\phi)/L$, where L is the wheelbase length.

The reference path is uniformly discretized across its length every Δs, so that $\{s_i\}_{i=0}^N$, where $s_i = i\Delta s$. To obtain a linear system, the vehicle model is linearized around reference states given by $\bar{\mathbf{s}} = \{\bar{s}_i\}_{i=0}^N$, $\bar{\mathbf{e}}_\mathbf{y} = \{\bar{e}_{y,i}\}_{i=0}^N$, $\bar{\mathbf{e}}_\psi = \{\bar{e}_{\psi,i}\}_{i=0}^N$, and $\bar{\mathbf{u}} = \{\bar{u}_i\}_{i=0}^N$. This results in a linear system of the form $q_{i+1} = A_i q_i + B_i u_i + G_i$, where $q_i = [e_{y,i}, e_{\psi,i}]^T$.

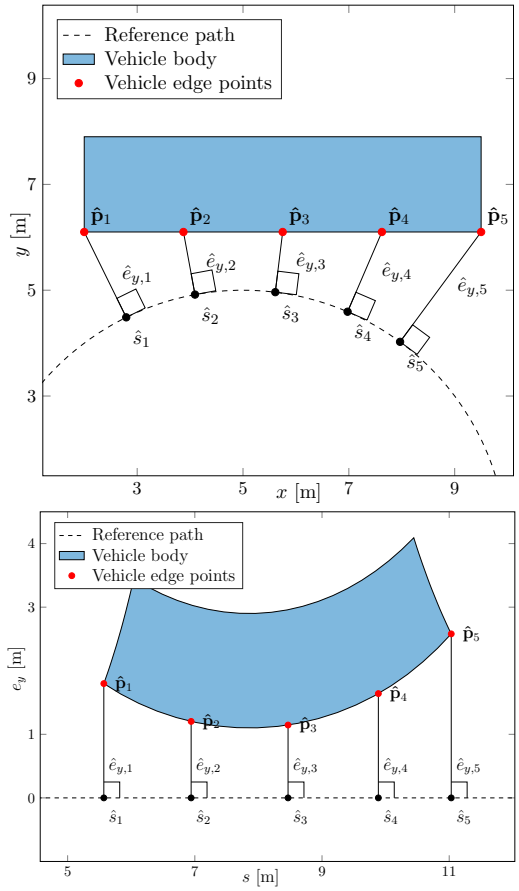


Figure 5.4: The vehicle body in the road aligned frame (bottom), can be converted to the Cartesian frame (top) using geometric method $\mathcal{T}$. $\mathcal{T}$ finds the normal projection of the vehicle body in the reference path, shown in the top figure. The resulting vehicle body in the road aligned frame becomes distorted, as shown in the bottom figure.

Conversion to road-aligned frame

A Cartesian position $\mathbf{p} \in \mathbb{R}^2$, can be converted to the road-aligned frame using a geometric algorithm. We define the reference path γ as the map $\gamma : \bar{\mathbf{s}} \rightarrow (x, y) \in \mathbb{R}^2$, where the domain is the discretized path length. One can convert $\hat{\mathbf{p}}$ to the road-aligned frame, by finding the location $\gamma(\hat{s})$, in which the normal to the path is pointing towards $\hat{\mathbf{p}}$. $\hat{e}_y$ is then $||\gamma(\hat{s}) - \hat{\mathbf{p}}||$. We define this conversion as $\mathcal{T} : \mathbf{p} \in \mathbb{R}^2 \rightarrow (s, e_y) \in \bar{\mathbf{s}} \times \mathbb{R}$. $\mathcal{T}$ is useful when

one needs to evaluate the vehicle body in the road-aligned frame. Figure 5.4 illustrates the results of $\mathcal{T}$, when converting points along the vehicle edge.

Distortions in the road-aligned frame

When planning a path it is necessary to take into account the vehicle body and check it against obstacles. When using the road-aligned model, it is necessary to account for the heavy distortion of objects due to transformation $\mathcal{T}$, shown in Figure 5.4. Since $\mathcal{T}$ is obtained via a geometric algorithm, an analytical approximation of it is of interest, in order to be able to use optimization algorithms. In the following, we derive such an approximation.

Given a road aligned vehicle state (s, e_y, e_ψ) and the corresponding Cartesian state (x, y, ψ), corresponding to position and orientation, one can compute the first order Taylor expansion for a point along the vehicle edge. Assuming a vehicle body edge point located at position $\hat{\mathbf{p}}$, one can get the corresponding road aligned coordinates as $(\hat{s}, \hat{e}_y) = \mathcal{T}(\hat{\mathbf{p}})$ (see Figure 5.4). Assuming a fixed $\hat{s}$, the first order Taylor expansion w.r.t. e_y and e_ψ, around linearization point $(\bar{s}, \bar{e}_y, \bar{e}_\psi)$, is computed:

$$\hat{e}_y = \mathcal{T}_{e_y}(\hat{\mathbf{p}}) + \frac{\partial \mathcal{T}_{e_y}(\hat{\mathbf{p}})}{\partial e_y}(e_y - \bar{e}_y) + \frac{\partial \mathcal{T}_{e_y}(\hat{\mathbf{p}})}{\partial e_\psi}(e_\psi - \bar{e}_\psi), \tag{5.2}$$

where $\mathcal{T}_{e_y}$ is $\mathcal{T}$ with a co-domain corresponding to the lateral displacement e_y only. Note that $\mathbf{p}'$ depends on vehicle states e_y and e_ψ, as they determine the vehicle position and orientation, and by consequence, the location of points on the vehicle body.

The partial derivatives can be approximated via a finite difference formula. However this requires numerous calls to the geometric method $\mathcal{T}_{e_y}$, which is computationally expensive. Instead, we propose an alternative approximation to the partial derivatives which is faster to compute.

Arc-circle approximation

In the road-aligned coordinate frame, the edges of the vehicle body are distorted to curves that resemble arc-circles, as seen in Figure 5.5. We exploit this insight to formulate an approximation to the partial derivatives in Equation (5.2).

During experiments, it was observed that the edges can be approximated by an arc-circle with a radius similar to the inverse of the reference path curvature, κ^{-1}, evaluated at length s. Moreover, the center of the arc-circle is located perpendicularly to the vehicle rear axle (s, e_y, e_ψ). We thus have

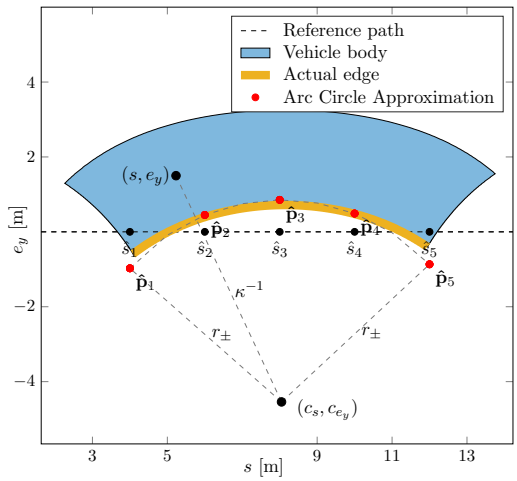


Figure 5.5: Distorted vehicle edges can be approximated by arc-circles.

for the center of the arc-circle (see Figure 5.5):

$$\begin{aligned} c_s &= s + \kappa^{-1} \sin e_\psi, \\ c_{e_y} &= e_y - \kappa^{-1} \cos e_\psi. \end{aligned} \tag{5.3}$$

Depending on the edge to be considered, left or right, the arc radius $r_\pm$ is equal to the inverse of the road curvature, plus or minus half the width ω of the vehicle, that is, $r_\pm = \kappa^{-1} \pm \omega/2$. The expression of the circle to which the arc belongs to is then:

$$\left(\hat{e}_y - c_{e_y}\right)^2 + \left(\hat{s} - c_s\right)^2 = r_\pm^2. \tag{5.4}$$

Assuming a constant $\hat{s}$, and writing in order to $\hat{e}_y$, the previous expression becomes:

$$\hat{e}_y(e_y, e_\psi) = c_{e_y} + \sqrt{r_\pm^2 - (\hat{s} - c_s)^2}, \tag{5.5}$$

where $\hat{e}_y$ corresponds to the lateral offset of the edge, evaluated at length $\hat{s}$. We write $\hat{e}_y(e_y, e_\psi)$ to highlight the dependency on vehicle states e_y and e_ψ.

In essence, this approximation assumes that the different points along the vehicle edge can be thought to belong to an arc-circle that is attached to the vehicle state (e_y, e_ψ). Relying on this dependency, we approximate the partial derivatives in equation Equation (5.2) by the partial derivatives

of $\hat{e}_y$:

$$\begin{aligned} \frac{\partial \mathcal{T}_{e_y}(\hat{\mathbf{p}})}{\partial e_y} &\approx \frac{\partial \hat{e}_y(e_y, e_\psi)}{\partial e_y}, \\ \frac{\partial \mathcal{T}_{e_y}(\hat{\mathbf{p}})}{\partial e_\psi} &\approx \frac{\partial \hat{e}_y(e_y, e_\psi)}{\partial e_\psi}. \end{aligned} \tag{5.6}$$

Doing so, we skip the computationally expensive process of computing the partial derivatives of $\mathcal{T}_{e_y}$ via finite differences.

The previous procedure gives us an expression for Equation (5.2). The positional constraint $p_{e_y} \leq \hat{e}_y$, which forces a vehicle body edge point to be contained in a certain region, can then written, by reorganizing the terms in Equation (5.2), as:

$$p_{e_y} \leq Pq + p. \tag{5.7}$$

Where $p_{e_y} \in \mathbb{R}$ is the position constraint (*e.g.*, corresponding to the boundary of an obstacle), $P \in \mathbb{R}^2$ is a row vector for the terms associated with $q = [e_y, e_\psi]^T$, and $p \in \mathbb{R}$ is a scalar, encompassing all constant terms in Equation (5.2).

5.3 Problem formulation

We introduce the objectives and constraints that path solutions must take into account. Special attention is given to the challenges faced by buses, which must be dealt with by developing special constraints for the overhang parts.

Driving objectives

A goal in on-road driving is to drive as much as possible in the center of the lane. Assuming that the reference path corresponds to the center of the lane, as is often the case in on-road driving, we define the optimization objective J_{center} to be the squared Euclidean norm of the lateral displacement, $\|(e_{y,0}, e_{y,1}, \ldots, e_{y,N})\|_2^2$.

Passenger comfort is also of importance, especially in buses. Thus, we introduce the minimization objective J_{smooth} given by $\sum_{i=1}^{N-1} (u_i - u_{i-1})^2$. Minimizing J_{smooth} results in a smooth control input profile, *i.e.* steering profile, which in turn results in more comfortable driving.

Overhangs and environment classification

Buses have relatively big overhangs (see Figure 5.6), when compared to other vehicles. The overhangs allow the bus to have a large passenger capacity, while keeping the wheelbase small, which increases the turning

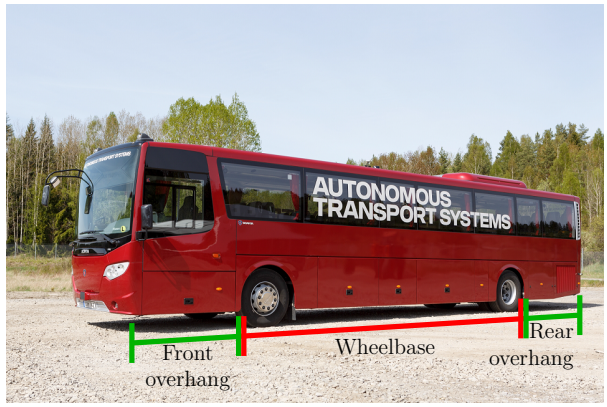


Figure 5.6: A prototype autonomous bus, the dimensions of which are used for the experimental results in this chapter. The distinct vehicle body, with its large and elevated overhangs, allows it to sweep over curbs and low height obstacles (courtesy of Scania CV AB).

radius. Furthermore, the smaller wheelbase allows for a better load balance on the vehicle chassis. Experienced bus drivers take advantage of the height of the overhangs, and use it to better maneuver the vehicle. Often a driver maneuvers the bus in a way that allows the overhangs to sweep over curbs.

Planning approaches typically take into account the dimensions of the vehicle body and use it to compute collision-free paths. It is common to split the planning space using a binary classification into obstacle or obstacle-free regions [13]. Buses suffer from such a classification scheme, as they do not allow sweeping over low height obstacles.

To address this issue, we introduce a three-label approach, classifying the space into three different regions, as shown in Figure 5.7. The obstacle region corresponds to obstacles that the vehicle body cannot collide with. The sweepable region corresponds to obstacles of height lower than the overhangs, such as curbs, that can be swept over by the overhangs. The drivable region corresponds to the road lane, where the wheels are allowed to be.

To formulate the obstacle constraints we make use of the arc-circle approximation introduced in Section 5.2, and repeat it for K equispaced points along the vehicle edge, for both edges. Each point is then constrained, using Equation (5.7), to be inside the left or right obstacle region boundaries, depending on which vehicle edge is considered. The obstacle constraints for all vehicle edge points can then be packed together as:

$$p_{e_y}^{\text{obs},i} \leq P_i q_i + p_i, \; i \in [1, ..., N], \tag{5.8}$$

where $p_{e_y}^{\text{obs},i} \in \mathbb{R}^{2K}$, $P_i \in \mathbb{R}^{2K\times 2}$, and $p_i \in \mathbb{R}^{2K}$. The $2K$ rows of Equation (5.8) correspond each to a positional constraint in the form of Equation (5.7).

Analogously, we limit the wheelbase to be inside the drivable region, by formulating the arc-circle approximation for M equispaced points along the wheelbase edges. The resulting drivable region constraints are:

$$t_{e_y}^{\text{driv},i} \leq T_i q_i + t_i,\ i \in [1, ..., N], \tag{5.9}$$

where $t_{e_y}^{\text{driv},i} \in \mathbb{R}^{2M}$, $T_i \in \mathbb{R}^{2M\times 2}$, and $t_i \in \mathbb{R}^{2M}$.

To minimize overhangs entering the sweepable region, we first introduce optimization variable σ corresponding to the amount of overhang exiting the drivable region. Then, the arc-circle approximation is used for the four vehicle corner points. Combining with the drivable region limits, together with the constraint that σ must be non-negative, we get:

$$\begin{aligned} r_{e_y}^{\text{driv},i} &\leq R_i q_i + r_i - \sigma_i^r,\ i \in [1, ..., N], \\ \sigma_i^r &\geq 0,\ i \in [1, ..., N], \end{aligned} \tag{5.10}$$

where $r_{e_y}^{\text{driv},i} \in \mathbb{R}^4$, $R_i \in \mathbb{R}^{4\times 2}$, $r_i \in \mathbb{R}^4$, and $\sigma_i^r \in \mathbb{R}^4$.

The optimization variable σ is then penalized through objective $J_{\text{overhang}} = \|(\sigma_1^r, \sigma_2^r, \ldots, \sigma_N^r)\|_2^2$. This minimization objective, together with the constraints defined previously, make σ a measurement of the amount of overhang that exits the drivable region. Thus, minimizing J_{overhang} results in reducing the amount of overhang exiting the road.

System constraints

We also define the state evolution constraints, corresponding to the discretized space-based road-aligned vehicle model introduced in Section 5.2:

$$q_{i+1} = A_i q_i + B_i u_i + G_i,\ i \in [0, ..., N\text{-}1]. \tag{5.11}$$

Furthemore, it is necessary for the planned path to start from the current vehicle state, and with the current steering angle. This originates constraints:

$$q_0 = q_{\text{start}}, u_0 = u_{\text{start}}. \tag{5.12}$$

Finally, we introduce constraints related to actuator limits, which are formulated as:

$$\begin{aligned} &u_{\max} \geq u_i \geq -u_{\max}, i \in [1, ..., N\text{-}1], \\ &u'_{\max} \geq u_i - u_{i\text{-}1} \geq -u'_{\max}, i \in [1, ..., N\text{-}1]. \end{aligned} \tag{5.13}$$

$u_{\max}$ and $u'_{\max}$ are space-based limitations of the curvature that reflect magnitude and rate limits of the steering actuator.

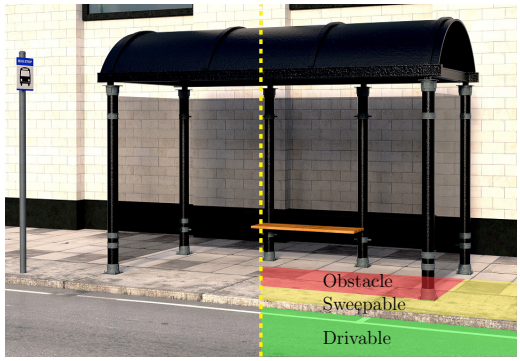


Figure 5.7: Example of a bus stop. On the right half of the image are overlayed the different region types. In red, yellow, and green are the obstacle, sweepable, and drivable regions. The vehicle body cannot enter the obstacle region in order to avoid collisions, however the overhangs are allowed to enter the sweepable region.

Sequential Quadratic Programming formulation

Combining all the optimization objectives and constraints mentioned before, one can formulate the following QP, [102]:

$$
\begin{aligned}
\underset{u}{\text{minimize}} \quad & J_{\text{center}} + J_{\text{smooth}} + J_{\text{overhang}} \\
\text{subject to} \quad & q_{i+1} = A_i q_i + B_i u_i + G_i,\ i \in [0, ..., N\text{-}1], \\
& q_0 = q_{\text{start}}, u_0 = u_{\text{start}}, \\
& p_{e_y}^{\text{obs},i} \leq P_i q_i + p_i,\ i \in [1, ..., N], \\
& t_{e_y}^{\text{driv},i} \leq T_i q_i + t_i,\ i \in [1, ..., N], \\
& r_{e_y}^{\text{driv},i} \leq R_i q_i + r_i - \sigma_i^r,\ i \in [1, ..., N], \\
& \sigma_i^r \geq 0,\ i \in [1, ..., N], \\
& u_{\max} \geq u_i \geq -u_{\max}, i \in [1, ..., N\text{-}1], \\
& u'_{\max} \geq u_i - u_{i\text{-}1} \geq -u'_{\max}, i \in [1, ..., N\text{-}1].
\end{aligned}
\tag{5.14}
$$

With optimization variable u corresponding to control inputs $(u_0, u_1, \ldots, u_{N\text{-}1})$.

The optimal inputs $\mathbf{u}^*$, and vehicle states $\mathbf{e}_y^*$, $\mathbf{e}_\psi^*$, which are the solution to the optimization problem Equation (5.14), can be relatively far from the linearization references $\bar{\mathbf{u}}$, $\bar{\mathbf{e}}_\mathbf{y}$, $\bar{\mathbf{e}}_\psi$. This means that the vehicle body approximations Equation (5.8), Equation (5.9), and Equation (5.10) lose

accuracy, possibly resulting in planned paths that violate the vehicle body constraints.

To overcome this problem, we use Sequential Quadratic Programming (SQP) [101]. In SQP, problem Equation (5.14) is sequentially solved, and at each iteration, the previous solution becomes the linearization reference for the current QP. Thus, we can guarantee that the final QP solution has an arbitrarily small distance to the linearization reference. By setting the allowed distance to a small value, we can enforce the quality of our approximations.

As the successive linearizations of the problem are solved, one gets that $\mathbf{e}_y^* \to \bar{\mathbf{e}}_\mathbf{y}$ and $\mathbf{e}_\psi^* \to \bar{\mathbf{e}}_\psi$. This in turn indicates that the first order Taylor expansion Equation (5.2) converges to the constant term, *i.e.*, $\hat{e}_y \to \mathcal{T}_{e_y}(\hat{\mathbf{p}})$, corresponding to the exact value of the edge location. Thus, as SQP progresses along iterations, so does the approximation become more accurate.

5.4 Results

Here, we present results showing how the novel constraints and optimization objectives are successful in capturing the desired behaviors of professional bus drivers. Furthermore, we present a road driving scenario that can only be solved by allowing the overhangs to exit the lane. Finally, we show the convergence of the proposed approximations into the real vehicle dimensions, illustrating that these approximations are safe and not conservative.

The motion planning framework is implemented in MATLAB, and we make use of CVX [103] as the convex solver of each SQP iteration. The vehicle dimensions are those of the prototype autonomous bus shown in Figure 5.6, and correspond to a wheelbase length of 6 m, a front overhang length of 3.34 m, a rear overhang length of 2.66 m, and a vehicle width of 2.54 m.

Wheelbase constraints and overhang minimization

Figure 5.8 shows the influence of wheelbase constraints in Equation (5.9) and optimization objective J_{overhang} on the planned paths. If both J_{overhang} and Equation (5.9) are disregarded, the vehicle follows the center of the road (Figure 5.8 left). Considering constraint Equation (5.9) forces the vehicle to the inside of the turn in order to keep the wheelbase inside the road lane (Figure 5.8 center). By also minimizing J_{overhang} the planned path is further pushed to the inside of the turn (Figure 5.8 right).

The maximum amount that the vehicle body exited the road is also measured, and it can be seen that it is greatly reduced from 1.29 meters

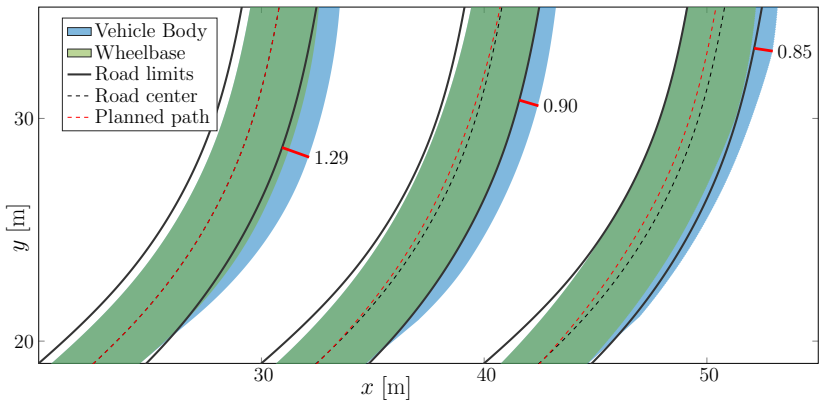


Figure 5.8: The influence of wheelbase constraints and overhang minimization on the planned path. Disregarding wheelbase constraints and overhang minimization (left), considering wheelbase constraints only (center), and considering both wheelbase constraints and overhang minimization (right). The maximum amount (in meters) that the vehicle body exits the road is shown in red, and it decreases from 1.29 (left) to 0.90 (center) and finally to 0.85 (right).

to 0.85 meters once the constraints and optimization objective are added. This results in less invasive maneuvers for vehicles on adjacent lanes.

Highly constrained maneuver

One of the biggest challenges that path planners face are highly constrained scenarios, where the solution must pass through small obstacle-free regions [95]. We set up a scenario where an obstacle on the road forms a passage with low clearance, as illustrated in Figure 5.9. One could imagine this to be a possible representation of a scenario in which there is a temporarily stopped vehicle on the side of the road.

Figure 5.9 shows that the path planner is able to find a collision-free solution that makes the vehicle progress through the low clearance passage. Furthermore, the planned path makes use of the sweepable region, allowing the overhangs to exit the road limits, while keeping the wheelbase contained in the drivable region corresponding to the road. If the overhangs were not allowed to exit the road limits, as is the case with other planners, then it would not be possible to find a solution. This illustrates the importance of allowing the overhangs to go over sweepable regions.

We note that the planned path takes the turn on the inside, except when avoiding the obstacle. This is done to minimize the amount of overhang

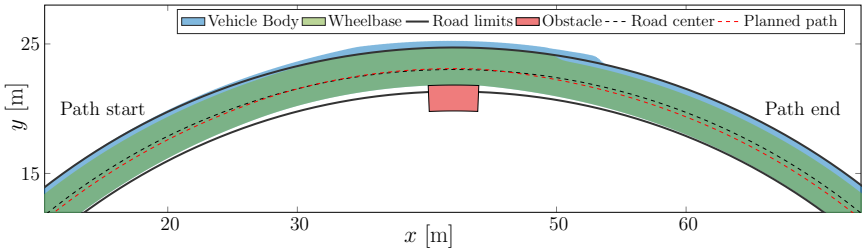


Figure 5.9: Planned path on a road with an obstacle forcing the bus to drive through a passage with small obstacle clearance.

exiting the driving lane.

Remark: The proposed planner assumes that a reference path is already obtained. The reference path usually corresponds to the road center, but can also be obtained by using a simplified path planner that determines if obstacles should be avoided by driving through the left or right of them.

Improvement of distortion approximations

We present in Figure 5.10 a zoomed-in version of a selected planning instance, in which an obstacle is present on the road. The figure shows the position of the vehicle's front right corner, when following the planned path, for different iterations of the SQP. In initial iterations the roughness of approximations makes the vehicle corner intersect the obstacle. However, the SQP iterations improve the accuracy of the approximation, resulting in successful obstacle avoidance.

The results show that the proposed path planner is capable of reducing the amount of overhang exiting the road, resulting in safer driving. Furthermore it is capable of dealing with highly constrained scenarios, where the bus can barely fit. This is achieved making use of approximations which are precise, being both safe and not conservative.

5.5 Conclusions

This chapter presented a path planning framework targeted for buses driving in urban environments. The work is motivated after noticing that there is a lack of literature when considering motion planning for buses. Even though several works are written in planning in constrained environments, none of them seem to be able to deal with the extremely complicated case of buses driving in narrow roads.

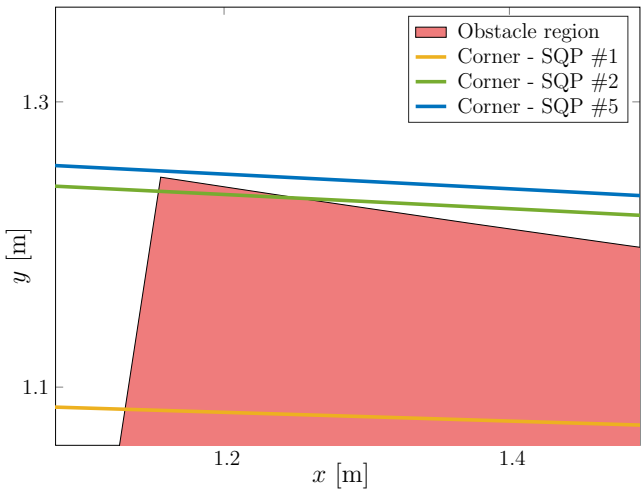


Figure 5.10: Zoomed view of a selected problem instance. The curves represent the location of the front right corner of the bus, according to the planned path, at three different SQP iterations (SQP converged at iteration 5). At each iteration the distortion approximation becomes more accurate, and the planned paths converge to a solution avoiding the obstacle.

The path planning problem makes use of the road-aligned vehicle model, and is solved via sequential quadratic programming. This type of solver, together with a novel formulation for the obstacle body dimensions, is able to guarantee obstacle avoidance, without requiring conservative approximations.

The proposed solution is tailored so as to take advantage of the special body characteristics of buses, namely the overhangs. Using a new labeling approach, which takes into account low height structures that can be swept by the overhangs, the planner is able to plan paths otherwise impossible when considering a binary classification into obstacle or obstacle-free regions. The solutions are thus successful in mimicking the expert behavior observed in bus drivers.

The proposed approach has several interesting future work directions. Firstly one should consider implementation of the method in an online fashion, and perform testing in real vehicles. In addition, the quality of the vehicle body distortion approximations could be studied further. We believe that the approximation error can be bounded and taken into account, so that all solution paths are guaranteed collision-free even during intermediate iterations of the SQP. It is also of interest to tackle the problem of bus stop arrival and departure maneuvers. In this type of maneuvers, high precision

is required so as to stop precisely at the boundary of the curb. Moreover, the framework is suitable to be adapted to other vehicle configurations, such as, articulated buses, and truck and trailer systems.

Chapter 6

On-road Path Planning for Articulated Vehicles

A significant part of vehicles in traffic is articulated, such as tractor-trailer combinations which are accountable for 9.2% of all distance driven in roads nowadays [104], and for 65% of USA's consumable goods transport [105]. Even though articulated vehicles play an essential role in the society, being responsible for a great the trucking industry is currently facing a shortage of drivers [106]. At the same time new consumer trends such as *e-commerce* are quickly growing and expected to significantly increase the needs for goods transport in the near future [105]. It is expected that with full autonomy, the operating costs of trucking would decline by about 45 percent [105]. Together, all these factors motivate the development of self-driving technologies targeting tractor-trailer vehicles.

Similarly to what was observed in Chapter 5, that driving a bus introduces many novel challenges when compared to driving passenger cars, so does driving an articulated vehicle introduce specific challenges to be tackled by motion planning frameworks. A tractor-trailer vehicle is characterized by two bodies of large dimensions, making it very difficult to successfully maneuver such vehicles on narrow road stretches, such as roundabouts or city streets. One difficulty arises from the off-tracking effect that forces the trailer to take a shortcut when the tractor performs a sharp turn. As a consequence, planning paths that take the off-tracking effect into account, while avoiding collision with surrounding obstacles, is both practically relevant and an important subject which requires the development of specialized solutions. This chapter focuses on this particular problem.

The contributions of this chapter are the following:

- proposal and evaluation of different optimization criteria suitable for

on-road path planning of articulated vehicles;

- implementation of a sequential quadratic programming (SQP) solver, which ensures smooth driving while guaranteeing precise obstacle avoidance;
- a sequential method for computing the off-tracking, as well as approximate partial derivatives, of each point of the vehicle bodies suitable for numerical optimization approaches;
- simulation results that show the proposed path planner's ability to solve complicated on-road planning scenarios while considering the most challenging tractor-trailer dimensions.

The chapter is organized as follows.

Section 6.1 presents existing works that tackle the motion planning task for articulated vehicles. The existing works mostly focus on off-road scenarios or roads with low curvature, highlighing a gap in existing research for articulated vehicles in urban environments.

The proposed road-aligned model of the tractor-trailer vehicle is presented in Section 6.2 presents the road-aligned model for the tractor-trailer vehicle. The trailer in conjunction with the road-aligned model introduce a need for numerical approximations that are able to model the trailer axle.

Section 6.3 introduces the formulation of the optimization-based path planner. An important component of the formulation is the optimization objective, which in the case of articulated vehicles can take many different forms. We introduce a set of possible optimization objectives that are specifically tailored for on-road driving of tractor-trailer vehicles.

Section 6.4 presents simulation results where the different proposed optimization objectives are compared against each other and the most suitable one is chosen. With the chosen optimization objective we then test the planning framework in tough maneuvering urban scenarios and in the presence of obstacles.

Section 6.5 provides a summmary of the chapter and lists its contributions. Furthermore, an introduction to follow-up work in Chapter 7 is given.

6.1 Introduction

The existing body of work on path planning for tractor-trailer vehicles has mostly focused on off-road scenarios. We start by presenting some of the works in this area.

In [107] the authors make use of a cascaded motion planning approach. At a first stage they compute a collision-free path that does not consider

the non-holonomic constraints of the system, *i.e.*, that does not respect the vehicle model. Afterwards, the path is approximated by collision-free feasible sub-paths, that are computed using a steering method. A final smoothing step is applied, based on the idea of connecting two configurations along the planner path by a shorter collision-free path, in a way similar to that presented in Section 4.2. The whole motion planning and control framework is evaluated on two experimental platforms, corresponding to miniature versions of both a tractor with on-axle and a tractor with off-axle hitching.

In [108] the authors propose a path planner for a truck-trailer system based on numerical optimization, namely, making use of an SQP solver. The planned paths are able to solve complicated parking scenarios involving direction changes. However, the collision avoidance constraints are not directly, nor exactly, formulated inside the optimization problem. Instead collisions are detected outside of the optimization, and then an additional repelling force optimization objetive is introduced, which pushes the vehicle away from the obstacles. The repelling force is an intuitive, and numerically stable approach to collision avoidance, however it is not able to always guarantee that the vehicle actually avoids the obstacles. The work achieves however impressive real-time performance for complex off-road scenarios.

In [109] the authors propose a numerical optimization based approach to plan trajectories for generic tractor-trailer vehicles. The authors introduce a novel solving strategy that progressively constrains a relaxed version of the planning problem so as to facilitate the numerical optimization process. The authors build upon the work of [110], where homotopy methods are used to relax the original planning problem and thereby more easily solve the underlying numerical optimization problems.

The work in [111] proposes a motion planning framework based on the closed-loop rapidly-exploring random tree (CL-RRT) algorithm. The work considers a general two trailer and tractor vehicle performing complex maneuvers in constrained unstructured environments. The authors present results from tests in a small-scale platform that highlight the capability of the method to perform complicated reversing maneuvers with high success rate.

The work in [112] integrates a path planning and control approach that can maneuver the vehicle in complicated off-road scenarios. A lattice-based motion planner is used to efficiently deal with difficult parking and obstacle avoidance maneuvers. Furthermore, a state estimator is developed that successfully estimates the articulation angles making only use of sensors mounted on the tractor vehicle. The work is extensively tested on a full-scale test vehicle showing the real-life applicability of the method.

Another lattice-based approach is presented in [113], but this time con-

sidering multi-steering trailer vehicles, which correspond to trailers that might have fixed or steerable wheels. The work combines in a first step a lattice-based approach that plans motions in unstructured scenarios, and in a second step optimal control. In the second step, the lattice planner solution is used to warm-start the optimal control problem, which will then compute a locally optimal trajectory in the neighborhood of the initial guess. Simulation results show the capability of the method to deal with a set of different parking scenarios.

We note however, that lattice planners do not lend themselves well for on-road applications, as they require the online computation of motion primitives [42], which can be extremely inefficient for collision checking procedures.

The are also some examples considering more structured on-road scenarios. The work in [114] presents a path planning approach for extremely large vehicles driving through narrow roads. More specifically, the authors study the challenging task of transporting Airbus A380 components through small villages. Special truck and trailer systems are used, that can reach 8 meters in width and 50 meters in length. The work studies the feasibility of driving such oversized vehicles through narrow roads of small villages. This can be seen as a motion planning task where the goal is to find out if a solution path exists. The method used is based on trajectory deformation, which is implemented via potential field computations.

The work in [115] also considers on-road environments. The authors focus on the task of planning paths for a heavy load vehicle with multiple steered axles. The implemented planning solution is based on the A*-algorithm together with some application specific adaptations to the original search algorithm. The authors also take into account the existence of "over drivable obstacles" which would roughly correspond to the sweepable areas we presented in Chapter 5. The motion planner achieves impressive results, however the computational times are in the order of minutes, making the algorithm not suitable for online motion planning.

Trajectory generation for long truck combinations is also considered in [116], where a nonlinear model predictive control approach is used. However, the work is limited to highway scenarios, which are characterized by roads with low curvature, and therefore not suited for urban scenarios. Furthermore, collision avoidance is performed by approximating the vehicle body by two points, which is excessively simplistic and cannot guarantee safety during more complicated maneuvers.

There exists in the literature a body of work on reducing the off-track of articulated vehicles [117–122]. However, common to all these approaches is the fact that the solution is implemented in a path-following/control framework. These approaches have the drawback of disregarding obstacle-

Figure 6.1: A tractor-trailer vehicle whose dimensions are used in this work. A tractor-trailer combination is an example of an articulated vehicle. This work considers such vehicle combinations, which also include articulated buses. (courtesy of Scania CV AB)

avoidance constraints, as well as properly deal with the dimensions of the vehicle bodies. By considering the off-track reduction problem already at the path-planning layer, it is possible to guarantee collision avoidance, reduce controller complexity, and minimize more general optimization objectives than just off-track reduction, as will be seen in the current chapter.

6.2 Modeling

We hereby introduce the model of the tractor-trailer vehicle used in the numerical optimization. The model is formulated in the road-aligned frame introduced in Chapter 5. Modeling the tractor-trailer vehicle in the road-aligned frame comes with challenges associated with deriving the position of the trailer axle. Therefore, we also introduce linear approximations for the position of the trailer axle that are suited for usage in the numerical optimization used in the planning framework.

Road-aligned tractor-trailer model

We consider a tractor-trailer vehicle composed of a car-like tractor and a trailer. The vehicle and respective dimensions are shown in Figure 6.2. The tractor is defined by the wheelbase length L_1, and the front and rear

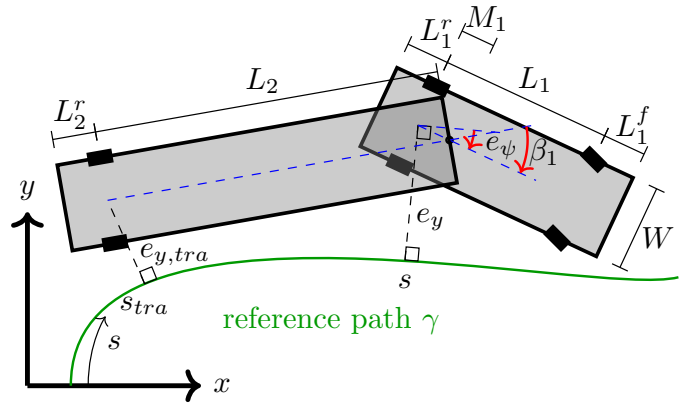


Figure 6.2: An illustration of the tractor-trailer vehicle in the road-aligned frame and definitions of relevant geometric lengths and vehicle states.

overhang lengths, L_1^f and L_1^r, respectively. M_1 corresponds to the signed hitch offset, which in the case of Figure 6.2 corresponds to a negative value. The trailer is defined by the length L_2 corresponding the distance between the trailer's axle and the off-axle hitch connection at the tractor, as well as the rear trailer overhang length L_2^r. Here we assume an equal width W for both the tractor and the trailer, however the approach can just as well consider different widths for the different bodies.

Similarly to the bus case presented in Chapter 5, in the road-aligned frame, the tractor-trailer vehicle is described in terms of deviation from the geometric reference path γ, as shown in Figure 6.2. Variable s is the distance traveled by the position of the tractor's rear axle onto its projection to the reference path γ. The reference path curvature is given by $\kappa_\gamma(s)$. The reference path γ is assumed to be the center of the road, however, depending on the application, it could also be the output solution of a global path planner.

We describe the tractor-trailer vehicle through the configuration vector $q = [s, e_y, e_\psi, \beta_1]^T$. e_y and e_ψ represent the tractor's lateral and orientation error, respectively, with respect to the reference path. $\beta_1 = \theta_{\text{veh}} - \theta_{\text{tra}}$ is the joint angle, defined as the difference between the orientation of the tractor θ_{veh} and the trailer θ_{tra}. The evolution of the tractor states (s, e_y, e_ψ) is equivalent to that of the road-aligned vehicle model used in for the bus case in Chapter 5. The model of the additional state β_1 corresponding to the joint angle is based on [112].

The model of the tractor-trailer vehicle in the road-aligned coordinate

frame is then given by

$$\begin{aligned}
\dot{s} &= v\frac{\cos(e_\psi)}{1-e_y\kappa_\gamma(s)},\\
\dot{e}_y &= v\sin(e_\psi),\\
\dot{e}_\psi &= v\left(\kappa - \frac{\kappa_\gamma(s)\cos(e_\psi)}{1-e_y\kappa_\gamma(s)}\right),\\
\dot{\beta}_1 &= v\left(\kappa - \frac{\sin(\beta_1)}{L_2} + \frac{M_1}{L_2}\cos(\beta_1)\kappa\right),
\end{aligned} \tag{6.1}$$

where $\dot{q} = {}^{\mathrm{d}q}/_{\mathrm{d}t}$, v is the linear velocity of the tractor, and $\kappa = {}^{\tan(\phi)}/_{L_1}$ corresponds to the tractor's curvature. The curvature of the tractor κ is the control input, which is directly related to the actuated steering angle ϕ.

Since we are considering on-road driving, we will only consider forward motion maneuvers. We note that reversing a long articulated vehicle in a road is something that very rarely happens and that should be avoided. By only considering forward motion we get that $v > 0$, and as a result time-scaling can be applied to remove the time-dependency presented in (6.1) and to transform the model into an equivalent spatial model [21]. Using the chain rule, it holds that ${}^{\mathrm{d}q}/_{\mathrm{d}s} = {}^{\mathrm{d}q}/_{\mathrm{d}t}{}^{1}/_{\dot{s}}$, and the resulting spatial model becomes

$$\begin{aligned}
e_y' &= (1-e_y\kappa_\gamma)\tan(e_\psi),\\
e_\psi' &= \frac{1-e_y\kappa_\gamma}{\cos(e_\psi)}\kappa - \kappa_\gamma,\\
\beta_1' &= \frac{1-e_y\kappa_\gamma}{\cos(e_\psi)}\left(\kappa - \frac{\sin(\beta_1)}{L_2} + \frac{M_1}{L_2}\cos(\beta_1)\kappa\right),
\end{aligned} \tag{6.2}$$

where $(\cdot)' = {}^{\mathrm{d}(\cdot)}/_{\mathrm{d}s}$ and $s' = 1$. The new state vector is given by $z = [e_y, e_\psi, \beta_1]^T$. As in Chapter 5 we discretize and linearize the spatial model (6.2) The reference path γ is discretized along its length, $\{s_i\}_{i=0}^N$, with $s_i = i\Delta s$, with a path sampling distance Δs. We then linearize around the reference states $\bar{\mathbf{s}} = \{\bar{s}_i\}_{i=0}^N$, $\bar{\mathbf{e}}_\mathbf{y} = \{\bar{e}_{y,i}\}_{i=0}^N$, $\bar{\mathbf{e}}_\psi = \{\bar{e}_{\psi,i}\}_{i=0}^N$, $\bar{\boldsymbol{\beta}}_1 = \{\bar{\beta}_{1,i}\}_{i=0}^N$ and $\bar{\boldsymbol{\kappa}} = \{\bar{\kappa}_i\}_{i=0}^N$, resorting to a first-order Taylor approximation. After discretization and linearization, we obtain a linear discrete-time model in the form $z_{i+1} = A_i z_i + B_i\kappa_i + G_i$, where $z_i = [e_{y,i}, e_{\psi,i}, \beta_{1,i}]^T$.

Trailer-axle states

The previously presented model (6.2) describes the evolution of the tractor's states as well as of the joint-angle between the tractor and the trailer. However, it lacks direct information about the position and orientation of the trailer's axle. For planning purposes and collision checking it is necessary to also have trailer state variables $(s_{\mathrm{tra}}, e_{y,\mathrm{tra}}, e_{\psi,\mathrm{tra}})$ corresponding to

the distance traveled by the position of the trailer's rear axle, the trailer's lateral and orientation error, respectively, with respect to the reference path γ. The trailer variables are important to define planning objectives, such as minimizing the lateral error of the trailer axle $e_{y,\text{tra}}$, as well as for formulating collision avoidance constraints.

As mentioned in Chapter 5 the road-aligned frame introduces distortions onto the vehicle body. As a result, the road-aligned model does not allow for an analytical expression that expresses the evolution of the trailer states $(s_{\text{tra}}, e_{y,\text{tra}}, e_{\psi,\text{tra}})$ as a function of $q = [s, e_y, e_\psi, \beta_1]^T$ and the tractor's curvature input κ. To tackle this issue we compute an approximate relationship of $(\hat{s}_{\text{tra}}, \hat{e}_{y,\text{tra}}, \hat{e}_{\psi,\text{tra}})$, which depends linearly on the tractor-trailer states of (6.2).

Given a reference vehicle state $\bar{q} = [\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1]^T$, we compute the corresponding position and orientation of the trailer's axle $(\bar{s}_{\text{tra}}, \bar{e}_{y,\text{tra}}, \bar{e}_{\psi,\text{tra}})$ as:

$$(\bar{s}_{\text{tra}}, \bar{e}_{y,\text{tra}}, \bar{e}_{\psi,\text{tra}}) = f(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma).$$

Function f performs three steps to compute $(\bar{s}_{\text{tra}}, \bar{e}_{y,\text{tra}}, \bar{e}_{\psi,\text{tra}})$:

1. compute the equivalent Cartesian x state of $\bar{q} = (\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1)$

2. computes the Cartesian pose $(x_{\text{tra}}, y_{\text{tra}}, \theta_{\text{tra}})$ of the rear axle of trailer for state x,

3. converts pose $(x_{\text{tra}}, y_{\text{tra}}, \theta_{\text{tra}})$ into the road aligned state $(\bar{s}_{\text{tra}}, \bar{e}_{y,\text{tra}}, \bar{e}_{\psi,\text{tra}})$, by projecting it onto the reference path γ.

It is important to note that function f is not analytical since the third step involves the projection of a Cartesian position onto an arbitrary path γ with varying curvature κ_γ. In the particular case of a straight reference path, f can actually be described by a closed-form expression [117], however the assumption of a straight reference path would be too restrictive for the purposes of on-road planning.

To compute the approximation of trailer states $(\hat{s}_{\text{tra}}, \hat{e}_{y,\text{tra}}, \hat{e}_{\psi,\text{tra}})$ we then need to express how $(\bar{s}_{\text{tra}}, \bar{e}_{y,\text{tra}}, \bar{e}_{\psi,\text{tra}})$ changes with respect to the tractor states (e_y, e_ψ, β_1). One possible way is to approximate the partial derivatives $\partial e_{y,\text{tra}}/\partial e_y$, $\partial e_{y,\text{tra}}/\partial e_\psi$, $\partial e_{y,\text{tra}}/\partial \beta_1$, $\partial e_{\psi,\text{tra}}/\partial e_y$, $\partial e_{\psi,\text{tra}}/\partial e_\psi$, and $\partial e_{\psi,\text{tra}}/\partial \beta_1$, using finite differences. This can be captured by approximating

the partial derivatives as follows:

$$\begin{aligned}
\frac{\partial e_{y,\text{tra}}}{\partial e_y} &\approx \frac{f_{e_y}(\bar{s}, \bar{e}_y + \delta e_y, \bar{e}_\psi, \bar{\beta}_1, \gamma) - f_{e_y}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta e_y}, \\
\frac{\partial e_{y,\text{tra}}}{\partial e_\psi} &\approx \frac{f_{e_y}(\bar{s}, \bar{e}_y, \bar{e}_\psi + \delta e_\psi, \bar{\beta}_1, \gamma) - f_{e_y}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta e_\psi}, \\
\frac{\partial e_{y,\text{tra}}}{\partial \beta_1} &\approx \frac{f_{e_y}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1 + \delta\beta_1, \gamma) - f_{e_y}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta\beta_1}, \\
\frac{\partial e_{\psi,\text{tra}}}{\partial e_y} &\approx \frac{f_{e_\psi}(\bar{s}, \bar{e}_y + \delta e_y, \bar{e}_\psi, \bar{\beta}_1, \gamma) - f_{e_\psi}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta e_y}, \\
\frac{\partial e_{\psi,\text{tra}}}{\partial e_\psi} &\approx \frac{f_{e_\psi}(\bar{s}, \bar{e}_y, \bar{e}_\psi + \delta e_\psi, \bar{\beta}_1, \gamma) - f_{e_\psi}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta e_\psi}, \\
\frac{\partial e_{\psi,\text{tra}}}{\partial \beta_1} &\approx \frac{f_{e_\psi}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1 + \delta\beta_1, \gamma) - f_{e_\psi}(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1, \gamma)}{\delta\beta_1},
\end{aligned} \tag{6.3}$$

where functions f_{e_y} and f_{e_ψ} are simply selecting the corresponding e_y or e_ψ co-domain of function f. The linear approximation of the trailer states is then defined as follows:

$$\begin{aligned}
\hat{e}_{y,\text{tra}} = \; & \bar{e}_{y,\text{tra}} + \frac{\partial e_{y,\text{tra}}}{\partial e_y}(e_y - \bar{e}_y) + \\
& \frac{\partial e_{y,\text{tra}}}{\partial e_\psi}(e_\psi - \bar{e}_\psi) + \frac{\partial e_{y,\text{tra}}}{\partial \beta_1}(\beta_1 - \bar{\beta}_1), \\
\hat{e}_{\psi,\text{tra}} = \; & \bar{e}_{\psi,\text{tra}} + \frac{\partial e_{y,\text{tra}}}{\partial e_y}(e_y - \bar{e}_y) + \\
& \frac{\partial e_{y,\text{tra}}}{\partial e_\psi}(e_\psi - \bar{e}_\psi) + \frac{\partial e_{y,\text{tra}}}{\partial \beta_1}(\beta_1 - \bar{\beta}_1).
\end{aligned} \tag{6.4}$$

This model is a linear approximation of the lateral and orientation error of the trailer axle with respect to the reference path γ at a fixed path length $\bar{s}$. These approximations are made around the reference vehicle states $(\bar{s}, \bar{e}_y, \bar{e}_\psi, \bar{\beta}_1)$.

6.3 Problem formulation

We hereby present the Optimal Control Problem (OCP) formulation corresponding to the on-road path planning problem. The OCP is solved using a Sequential Quadratic Programming (SQP) approach. We propose a set of different optimization objectives that target tractor-trailer on-road driving based on principles inspired by human-like driving goals.

Optimal control problem

The on-road path planning problem for the tractor-trailer vehicle is formulated as the following OCP:

$$\begin{aligned}
\underset{\boldsymbol{\kappa}}{\text{minimize}} \quad & J_e^k(\mathbf{e_y}, \mathbf{e_{y,\text{tra}}}) + J_\kappa(\boldsymbol{\kappa}) && \text{(6.5a)}\\
\text{subject to} \quad & z_{i+1} = f(z_i, \kappa_i),\ i \in \{0, ..., N-1\}, && \text{(6.5b)}\\
& z_0 = z_{\text{start}},\ \kappa_0 = \kappa_{\text{start}}, && \text{(6.5c)}\\
& p_{e_y}^{\text{obs},i} \leq g(z_i),\ i \in \{1, ..., N\}, && \text{(6.5d)}\\
& |\kappa_i| \leq \kappa_{\max},\ i \in \{1, ..., N-1\}, && \text{(6.5e)}\\
& |\kappa_i - \kappa_{i-1}| \leq \kappa'_{\max},\ i \in \{1, ..., N-1\}, && \text{(6.5f)}
\end{aligned}$$

where $\mathbf{e_y} = [e_{y,1}\ \ \dots\ \ e_{y,N}]^T \in \mathbb{R}^N$, $\mathbf{e_{y,\text{tra}}} = [e_{y,\text{tra},1}\ \ \dots\ \ e_{y,\text{tra},N}]^T \in \mathbb{R}^N$, and $\boldsymbol{\kappa} = [\kappa_0\ \kappa_1\ \dots\ \kappa_{N-1}]^T \in \mathbb{R}^N$. Vector $\boldsymbol{\kappa}$ is the optimization variable to be optimized, which corresponds to the curvature of the vehicle along the planned path. The vehicle curvature can be directly converted to a steering angle of the vehicle, which is the actuated variable used to steer the vehicle.

The optimization objective (6.5a) is composed of two terms. The term J_e^k penalizes a function related to the vehicle states of the tractor, trailer, or both. In Section 6.3 we will see that different formulations of J_e^k terms can result in very different types of behaviors, with their respective advantages and disadvantages. The term J_κ is used to penalize control inputs and increase ride comfort. To do so, we set $J_\kappa(\boldsymbol{\kappa}) = \sum_{i=1}^{N\text{-}1} (\kappa_i - \kappa_{i-1})^2$, enforcing a smooth curvature profile that is associated with a comfortable driving behavior.

Constraint (6.5b) encondes the vehicle model. (6.5c) defines the initial constraints on the vehicle states and control input, necessary to ensure that the planned paths start from the current vehicle state. Constraint (6.5d) enforces obstacle avoidance by ensuring that the vehicle bodies do not collide with any obstacle. The constraints are formulated in an analogous way to that presented for the bus case in Chapter 5. Finally, constraints (6.5e) and (6.5f) set the tractor's curvature limits including both maximum magnitude $\kappa_{\max}$ as well as maximum rate $\kappa'_{\max}$.

We make use of the same SQP methodology introduced in Chapter 5 to find the solution to OCP (6.5). At each SQP iteration, the vehicle model (6.5b) is re-linearized around the previous solution. The linearization is done using a first-order Taylor series approximation as explained in Section 6.2, resulting in a linear prediction model of the tractor states $\mathbf{e_y}$ and $\mathbf{e}_\psi$ as well as of the joint-angle state $\boldsymbol{\beta}_1$. The remaining trailer states $\mathbf{e_{y,\text{tra}}}$ and $\mathbf{e}_{\psi,\text{tra}}$ are obtained using approximation (6.4).

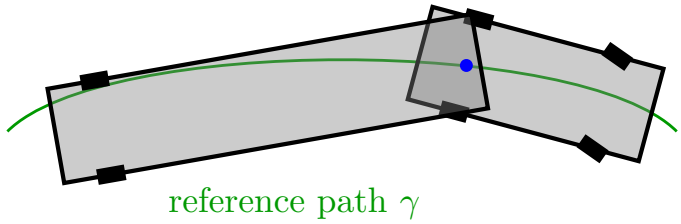


Figure 6.3: Illustration of optimization objective 1, corresponding to minimizing the tractor lateral offset. The optimization objective tries to minimize the blue dot corresponding to the center of the rear axle of the tractor.

Optimization objectives

This section presents a set of different candidate optimization objectives J_e^k that can be used in (6.5). These candidate optimization objectives will be later on compared to understand their performance and suitability for on-road driving of articulated vehicles.

Optimization objective 1 - Tractor centering

In normal conditions, vehicles on the road drive as much as possible in the center of their lanes. As previously mentioned, we assume here that the the reference path γ of the road-aligned frame corresponds to the center of the lane. Therefore, centering the tractor is achieved by minimizing the magnitude of e_y. In the case when the tractor drives precisely on the center of the road, *i.e.*, on top of the reference path γ, we get that $e_y = 0$. We define our first optimization objective J_e^1 to be the square of the euclidean norm of the lateral displacement e_y of the tractor, along the planned path:

$$J_e^1 = \|\mathbf{e_y}\|_2^2 \,.$$

Figure 6.3 illustrates the proposed optimization objective.

Optimization objective 2 - Trailer centering

In the case of a tractor and trailer vehicle, one needs to take into account the presence of the trailer. The trailer can significantly deviate from the center of the road, even when the tractor is centered. This is the case in turns, where the off-tracking effect is clearly noted, causing the trailer to significantly cut through the inside of the curve.

To avoid the trailer from deviating from the center of the road, we define the second optimization objective J_e^2 as a minimization of the trailer lateral

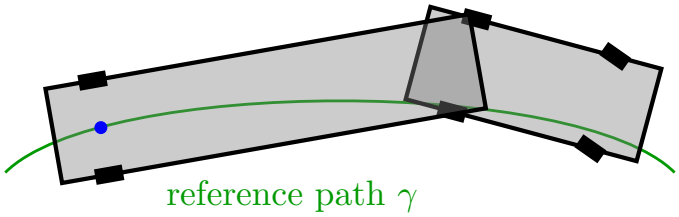


Figure 6.4: Illustration of optimization objective 2, corresponding to minimizing the trailer lateral offset. The optimization objective tries to minimize the blue dot corresponding to the center of the rear axle of the trailer.

offset to the center of the road:

$$J_e^2 = \|\mathbf{e}_{\mathbf{y},\mathrm{tra}}\|_2^2 .$$

Figure 6.4 illustrates the proposed optimization objective.

Optimization objective 3 - Tractor and trailer centering

Ideally one would center both the tractor and trailer, and not just one of them. Therefore the third centers both the tractor and the trailer simultaneously. Instead of considering only on the tractor or the trailer, we formulate an objective that minimizes the lateral error of both. This is achieved with the following optimization objective:

$$J_e^3 = \|(1-K)\mathbf{e}_{\mathbf{y}} + K\mathbf{e}_{\mathbf{y},\mathrm{tra}}\|_2^2 , \tag{6.6}$$

where $K \in [0,1]$ is a tunable parameter. The value of K determines the trade-off between centering the tractor or the trailer around the road center. A method for determining a suitable K is presented in Section 6.4. Figure 6.5 shows an illustration of the lateral offsets to be minimized in the proposed optimization objective. We note that centering the tractor is often in conflict with centering the trailer, and vice-versa. K is therefore crucial for having an appropriate behavior of this optimization objective.

Optimization objective 4 - Tractor and trailer maximum deviation minimization

For the fourth optimization objective, we make use of the $\mathcal{L}_\infty$-norm, that significantly differs from the Euclidian norm considered in the previous objectives. When considering the $\mathcal{L}_\infty$-norm, we minimize the worst lateral deviation of the vehicle states. Intuitively, this should result in planned

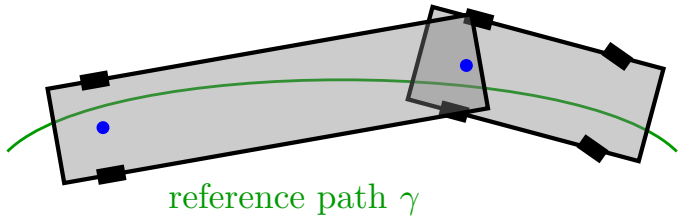


Figure 6.5: Illustration of tractor and trailer lateral offsets involved in the formulation of optimization objectives 3 and 4. Both optimization objectives 3 and 4 tries to keep the blue dots corresponding to the center of the rear axle of the tractor and of the trailer as close as possible to the center of the road.

paths that minimize the maximum lateral deviation of the vehicle axles from the road center. The fourth optimization objective is then defined as:

$$J_e^4 = \|(\mathbf{e_y}, \mathbf{e_{y,\text{tra}}})\|_\infty.$$

Figure 6.5 shows an illustration of the lateral offsets to be minimized in the proposed optimization objective.

Optimization objective 5 - Swept area minimization

The fifth objective minimizes the distance of the vehicle sides to the center of the road. We first define the vector of auxiliary variables

$$\mathbf{q} = [l_1^L, l_2^L, \ldots, l_M^L, l_1^R, l_2^R, \ldots, l_M^R]^T, \tag{6.7}$$

shown in Figure 6.6. The superscripts L and R correspond to the left (L) and right (R) sides of the vehicle bodies. The proposed vector $\mathbf{q}$ measures the displacement of the sides of the vehicle bodies to the reference path, *i.e*, the center of the road. For each vehicle state $(e_{y,i}, e_{\psi,i}, \beta_{1,i})$ we compute the corresponding $\mathbf{q}_i$. Ideally, vector $\mathbf{q}_i$ is kept as small as possible. This in turn, implies that the tractor-trailer vehicle body drives as close as possible to the center of the road.

We can then formulate the optimization objective J_e^5 as:

$$J_e^5 = ||(\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_N)||_\infty.$$

With this formulation, we encourage the optimization problem to find a solution that minimizes the largest displacement of the vehicle sides to the center of the road.

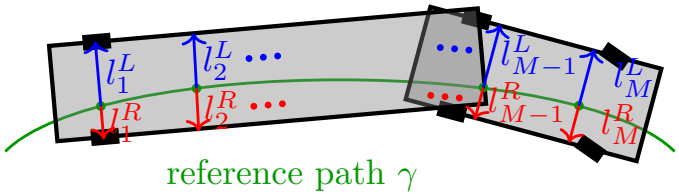


Figure 6.6: Illustration of the auxiliary variables $l_1^L, \ldots, l_M^L, l_1^R, \ldots, l_M^R$ that are sampled uniformly across the vehicle length and used to measure the lateral displacement of the vehicle sides. These variables provide an estimate of how much the vehicle sides deviate from the center of the road.

Note that this objective makes use of the positional information of the vehicle sides in its formulation, and therefore takes into account the vehicle dimensions. Previous candidate objectives only make use of the axle positions to center the tractor-trailer vehicle. Intuitively, one expects that an optimization objective which takes into account the dimensions of the vehicle will achieve better results than an optimization objective which only considers the center positions of the axles.

We remark that it would be interesting as well to formulate this optimization objective with an Euclidean norm instead of the $\mathcal{L}_\infty$-norm. However, during our experiments, we noticed that the Euclidean norm turned out to be computationally too demanding for the optimization problem to be solved.

6.4 Results

To study the performance of the on-road path planner with the different optimization objectives we run a set of simulation experiments and evaluate them using a set of relevant performance metrics. After choosing a suitable optimization objective we simulate two realistic urban driving scenarios that highlight the capabilities of the proposed path planner.

We use the vehicle dimensions of the tractor-trailer vehicle shown in Figure 6.1. We consider a tractor wheelbase $L_1 = 3.78\ m$, a tractor front overhang $L_1^f = 1.46\ m$, a tractor rear overhang $L_1^r = 1.64\ m$ and a hitch length $M_1 = -0.30\ m$ (the hitch connection is located in front of the tractor's rear axle). For the trailer we assume an axle length $L_2 = 13.97\ m$ and a rear overhang $L_2^r = 4.50\ m$. Both the tractor and the trailer have the same width $W = 2.54\ m$. We have chosen a tractor-trailer vehicle with a total length of 24 meters. This vehicle length corresponds to the maximum legally allowed in Sweden, and therefore one of the most challenging tractor-trailer combinations that are allowed to drive on public roads. Furthermore, we

consider a tractor with a maximum curvature magnitude $\kappa_{\max} = 0.1$ m^{-1} and a maximum curvature rate $\kappa'_{\max} = 0.1\Delta s$.

All results shown in this section have been obtained on a computer equipped with an Intel Core i7-6820 HQ@2.7GHz CPU. The numerical optimization problem is implemented in MATLAB, and we make use of CVX [103] as the convex solver of each SQP iteration.

Performance metrics

To measure and compare the performance of the different optimization objectives proposed in Section 6.3, we introduce three different performance metrics.

First, it is desirable to measure the maximum amount the vehicle bodies sweep on the road, *i.e.*, how much the vehicle body deviates at most from the road center. Therefore we introduce performance metrics *max left* and *max right* that measure the maximum offset from any point on the vehicle body to the center of the road. These metrics are illustrated in Figure 6.8.

It is also important to consider the behavior of the vehicle along the whole planned path, and not just at its maximum offset. Therefore we introduce a second metric $a_L - a_R$ that measures the difference between the areas swept by the vehicle bodies to the left and to the right of the road center. In the special case of a straight road, and when the vehicle drives on the center of it, we would get that $a_L - a_R = 0$. Large values of this metric indicate a preference for the vehicle bodies to be off-centered. The case when $a_L - a_R > 0$ indicates a tendency to drive on the left of the road center, whereas $a_L - a_R < 0$ indicates a tendency to drive on the right. The closer this metric is to zero, the more centered the tractor-trailer drives.

The third metric is related to the applicability of the proposed optimization objectives for practical implementation. We introduce metric *CPU time* that measures the amount of time required to solve the OCP in (6.5a). This metric is used to compare the computational effort that the different optimization objectives require from a computing unit. Large values of *CPU time* indicate that the optimization objective is not feasible for real-time implementation, as one expects that path planner to run a high frequency and on limited computational power. Small values of *CPU time* are desirable, as they indicate that the optimization objective is computationally fast and suitable for actual implementation on a vehicle.

Comparison of different optimization objectives

We now compare the results of using the different optimization objectives introduced in Section 6.3. Figure 6.7 shows the envelopes of the areas swept by the vehicle when performing a U-turn. The U-turn road considered

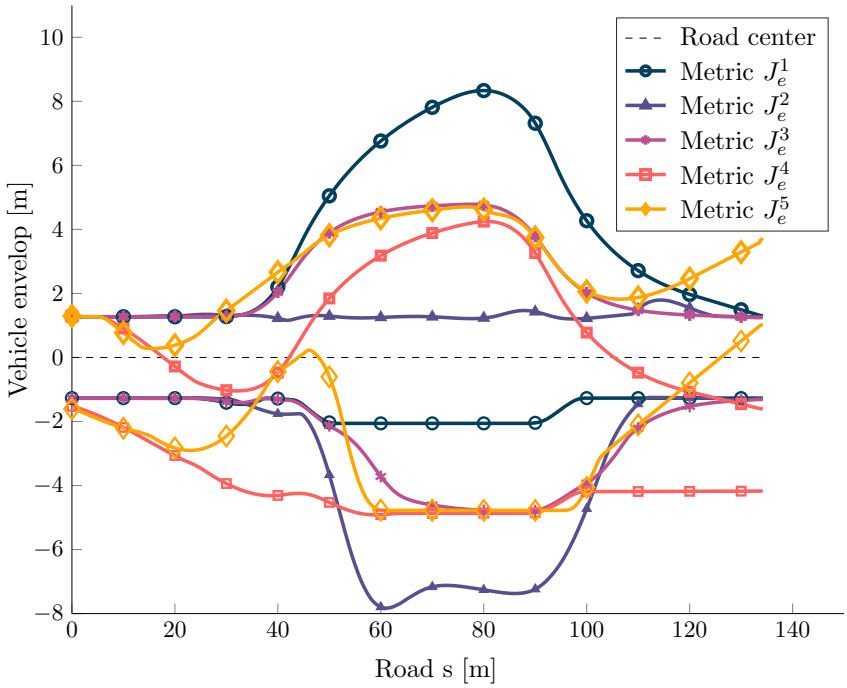


Figure 6.7: Vehicle envelopes of the planned paths for each optimization objective. The road considered is shown in Figure 6.8. Table 6.1 provides detailed information about the maximum envelope offsets, the swept areas, and computational times.

has a curvature of 0.065 m^{-1} (turning radius of 15.38 m) and can be seen in Figure 6.8, resulting in a planning horizon of the optimization problem of 134.2 m, and a discretization of the path of 0.1 m. Table 6.1 presents the different performance metrics for the five candidate optimization objectives. The performance of each optimization objective is studied in detail below.

Optimization objective 1 - Tractor centering

Objective J_e^1 results in the largest positive (left) sweep, corresponding to the trailer taking the turn on the inside up to 8.34 m. Moreover, the vehicle has a swept area of 312 m^2, clearly indicating that it tends to the inside of the turn. This is a direct result of the objective formulation, which tries to keep the tractor on the road center while disregarding the trailer. Considering J_e^1 as an optimization objective results in a large off-tracking effect, where the vehicle clearly cuts in through the inside of the curve. As it is not desirable

that the trailer cuts through the inside of curves, we can determine this optimization objective to not be suited for on-road planning purposes.

Optimization objective 2 - Trailer centering

The second optimization objective J_e^2 has the opposite effect of J_e^1, resulting in the largest negative (right) sweep, corresponding to the vehicle bodies taking the turn on the outside up to 7.83 m from the road center. The swept area is -302 m^2, which clearly shows the tendency of the vehicle to drive on the outside of the turn. This is expected, as the formulation only focuses on keeping the trailer rear axle centered on the road, ignoring the position of the tractor. In order to keep the trailer on the road center, *i.e.*, minimize $\|\mathbf{e}_{\mathbf{y},\text{tra}}\|_2^2$ the tractor needs to drive on the outside of the turn. Intinuitively, it is not desireable that the tractor drives to much on the outside of the turn, and we can therefore determine this optimization objective to be unsuitable for on-road planning purposes.

Optimization objective 3 - Tractor and trailer centering

We note that objective function J_e^3 involves a term K that has not been determined yet. To find an appropriate value for K, we do a discrete search over the interval $K \in [0, 1]$, and measure the performance of the resulting planned paths with respect to the area difference metric $a_L - a_R$. To measure metric $a_L - a_R$ in a relevant way, we run the path planner in several U-turn roads of different curvatures. We then conclude that $K = 0.45$ is consistently performing the best for the majority of roads considered. Therefore, we select $K = 0.45$ in optimization objective J_e^3, and all coming results are obtained considering these value, unless stated otherwise. With this value of K in optimization objective J_e^3, we obtain a balanced trade-off between the cut-in of the trailer (4.79 m), and the cut-out of the tractor (-4.82 m). Furthermore, we obtain a small swept area of -5 m^2, which is expected, as we optimize the value of K with respect to this metric.

Optimization objective 4 - Tractor and trailer maximum deviation minimization

Optimization objective J_e^4 achieves a better trade-off between the cut-in and cut-out than both J_e^1 and J_e^2. However, we note that it is significantly worse than J_e^3. Furthermore, since J_e^4 considers the $\mathcal{L}_\infty$-norm instead of the euclidean norm, the vehicle does not have any incentive to come back to the center of the road after the turn finishes. This effect can be seen in Figure 6.7, where it can be noted that after the turn, *i.e.*, for $s > 100$ m, the vehicle envelope continues with a negative lateral offset, indicating

that the vehicle continues driving on the left side of the road, even in the straight road section. This is a clear disadvantage of the $\mathcal{L}_\infty$-norm which only penalizes the vehicle state that leaves the road center the most. As a result, J_e^4 has an excessively large swept area of -406 m^2.

Optimization objective 5 - Swept area minimization

Optimization objective J_e^5 results in a good trade-off between cutting in and out of the road, resulting in very similar values for the maximum cut-in, *max left*, and maximum cut-out, *max right*. We note that this trade-off is slightly worse than the one achieved by J_e^3, however J_e^5 has the significant advantage of not relying on a tuning of parameter, whereas J_e^3 requires a properly chosen K value. Similarly to J_e^4, objective J_e^5 also is affected by the drawbacks that arise when considering the $\mathcal{L}_\infty$-norm. Figure 6.7 shows that the vehicle does not converge to the center of the road at the final section of the path, instead actually deviating from it.

An even more significant drawback is that objective J_e^5 makes the optimization problem very expensive to solve, as show by the CPU time of 770.71 s that is required to solve the OCP. The high computational time is due to the large number of terms involved in J_e^5. For each vehicle state in the planning horizon, objective J_e^5 must consider several auxiliary variables $\mathbf{q} = [l_1^L, l_2^L, \ldots, l_M^L, l_1^R, l_2^R, \ldots, l_M^R]^T$ corresponding to points along the vehicle body (see Equation (6.7)). This in turn results in large computational times that are not suitable for implementation on a real system with online planning requirements.

Selection of best optimization objective

Based on the above study of the individual optimization objectives, we can conclude that J_e^3 is the most suited for our purposes. Firstly, it achieves a low swept area, as well as a balanced trade-off between maximum cut-in and cut-out. Secondly, since it penalizes all states along the planned path, it takes the vehicle to the center of the road in both turning and straight segments. Thirdly, its CPU time of 22.03 s makes it promising for usage in online planning for autonomous vehicles. A planned path for the tractor-trailer vehicle using optimization objective J_e^3 is shown in Figure 6.8. We observe that the optimal solution achieves a balanced trade-off between the cut-in of the trailer body and the cut-out of the tractor body.

We note that the results presented have focused on a specific U-turn defined by a curvature of 0.065 m^{-1}. However, we performed the same analysis for several other U-turns with different curvatures, as well as for road with curvatures with different sign (turning clockwise, as opposed to counterclockwise). The comparisons and conclusions made previously were

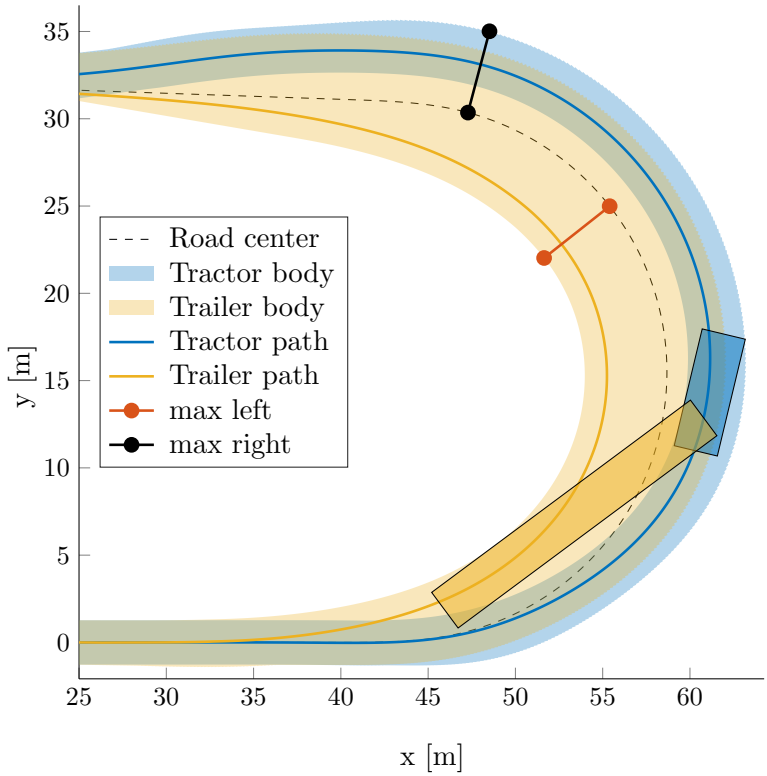


Figure 6.8: Path performed by the tractor-trailer when using optimization objective J_e^3. The planned solution fairly balances the maximum cut-out of the tractor to the right of the road center, and the maximum cut-in of trailer to the left of the road center. *max left* and *max right* measure the maximum amount of sweep of the vehicle body to the left and right of the road center.

also observed in the majority of distinct U-turns that were tested, indicating that the above conclusions can be generalized to a broad set of U-turns.

It should also be noted that all CPU times are quite high in this study. This can be explained by the fact that we have used a MATLAB implementation running on a personal laptop, and that the optimization problem was solved for the whole length of the road, which amounts to 1342 vehicle states (134.2 m long road with a discretization of the path of 0.1 m). In a real implementation, we would expect very significant speed ups in the computational time, due to

Table 6.1: Performance metrics for the proposed optimization objectives. Metrics *max left* and *max right* correspond to the maximum amount swept by the vehicle body to the left and right of the road center, respectively. $a_L - a_R$ is the difference between the area swept to the left (L) and to the right (R) of the road center. *CPU time* is the time taken to solve the OCP (6.5).

Objective	max left	max right	$a_L - a_R$	CPU time
J_e^1	8.34 m	2.06 m	312 m^2	7.92 s
J_e^2	1.79 m	7.83 m	-302 m^2	17.93 s
J_e^3	4.79 m	-4.82 m	-5 m^2	22.03 s
J_e^4	4.25 m	-4.82 m	-406 m^2	13.53 s
J_e^5	4.70 m	4.78 m	4 m^2	770.71 s

1. implementation in a low-level language, such as `C++`, which is expected to achieve computational times when compared to MATLAB;

2. reformulation of the QP problem and usage of a tailored solver;

3. execution in a receding horizon fashion which will significantly reduce the length of the road considered for planning and therefore the number of vehicle states to be optimized;

4. execution in a receding horizon fashion which allows as well for the usage of initial guesses to the optimization problem allowing for efficient warm-starts and faster solving times.

These reasons lead us to believe that the proposed approach can be made real-time given a more time to focus on a more serious implementation of the planning framework.

Driving on a roundabout

After deciding to use optimization objective J_e^3 we now study the performance of the proposed path planner in more complex scenarios. We first consider a 450-degree turn in a roundabou, shown in Figure 6.9. This roundabout is taken from the work in [119], where the authors use this scenario to test for compliance with UK requirements for roundabout maneuvers. The roundabout has a curvature 0.056 m^{-1}, corresponding to a turning radius of 17.88 m. This results in a planning horizon for the optimization problem of 245.8 m. To keep the computational times more tractable we consider a sampling distance is 0.2 m (instead of the previously used 0.1 m). We also adapt the tractor-trailer dimensions to match those considered

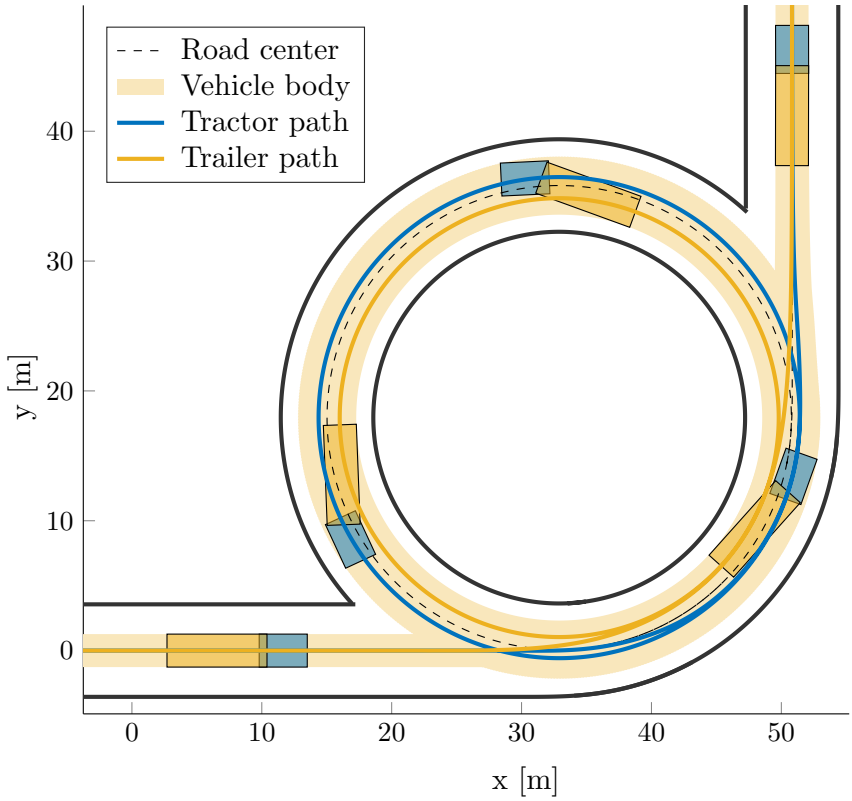


Figure 6.9: The tractor-trailer performing a 450-degree turn in a roundabout. The planned path (using objective J_e^3) balances the tractor and trailer cut-in and cut-out towards the right and left sides of the road center. Both the roundabout and vehicle dimensions are taken from [119].

in [119] considers a smaller tractor-trailer combination. The new vehicle dimensions correspond to a tractor wheelbase $L_1 = 3.47\ m$, a tractor front overhang $L_1^f = 1.16\ m$, a tractor rear overhang $L_1^r = 1.34\ m$ and a hitch length $M_1 = -0.30\ m$ (corresponding to the hitch connection being in front of the rear axle). The new trailer dimensions correspond to an axle length $L_2 = 9.40\ m$ and a rear overhang $L_2^r = 3.03\ m$. The tractor and the trailer have the same width $W = 2.54\ m$. We consider the same curvature magnitude and rate limits as before, $\kappa_{\max} = 0.1\ \mathrm{m}^{-1}$ and $\kappa'_{\max} = 0.1\Delta s$.

We note that since we are considering a vehicle with significantly different dimensions, we need to re-tune parameter K of optimization objective J_e^3. This highlights one significant drawback of this optimization objective,

namely the need to tune K for different vehicle dimensions. Analogously to the tuning process explained in Section 6.4, we run the path planner with different values of K in several U-turn roads. The best value of K is determined to be $K = 0.40$ as it performs best for the majority of roads considered.

Figure 6.9 presents the planned path when considering the newly tuned K value. The planned path properly balances the cut-in and cut-out of the vehicle bodies as it drives along the roundabout. The vehicle smoothly enters and exits the roundabout, while keeping its swept width reasonably small at all times during the maneuver. In this scenario the computational time required to obtained the planned path is 11.88 s.

Collision avoidance

To show the collision avoidance capabilities enforced by constraint (6.5d), we set up a complex scenario with obstacles on the road. Here we consider again the vehicle dimensions of the longest vehicle combination legally allowed in Sweden. In this scenario, the vehicle needs to drive on a sharp U-turn with a curvature of 0.040 m^{-1} and corresponding turning radius of 25 m. Two obstacles are placed on each side of the road creating a complex driving scenario.

Figure 6.10 shows the considered scenario as well as the planned path solution. The vehicle starts by driving towards the outside of the turn in order to avoid colliding with the first obstacle. The collision avoidance constraints of the trailer body are the ones that determine this maneuver, guiding the tractor towards the outside of the road to ensure that the trailer safely avoids the obstacle. Afterwards the second obstacle forces the vehicle into the inside of the turn. In this case, obstacle avoidance constraints of the tractor body force the vehicle into the inside of the road. The trailer follows safely, as it is dragged along through the inside of the turn and away from the obstacle.

In this simulation the planning horizon of the optimization problem is 134.2 m, and we consider a sampling distance of 0.2 m. The resulting computation time required to obtain the planned solution is 108.41 s. This is an excessively large computation time that can be explained by the complexity of the maneuver. The complexity of the maneuver requires the SQP solver to perform several iterations until converging to a feasible solution and collision free solution. Moreover, we have noticed that the dimensions of the vehicle severely impact computation times. When considering the same scenario for a vehicle with smaller dimensions, such as those considered in [119] and used in our roudabout experiments, we get a computation time of 29.89 s, significantly lower than the time required for the case of the

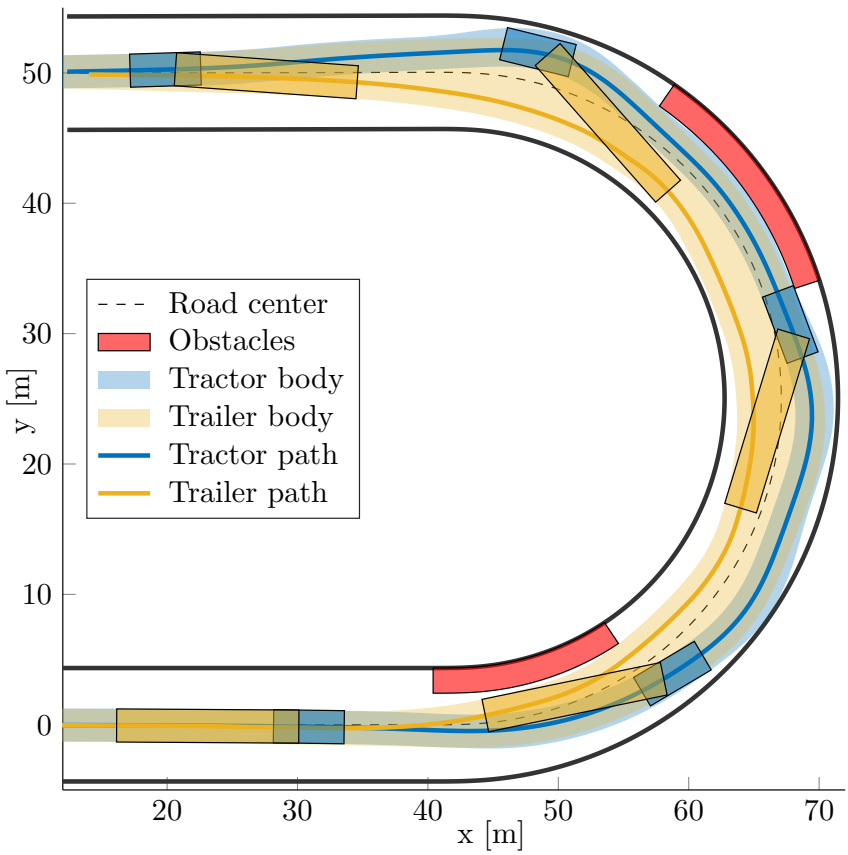


Figure 6.10: The tractor-trailer vehicle avoiding obstacles located in a U-turn. First the tractor drives towards the outside of the turn in order for the the trailer to avoid the obstacle located on the inside of the turn. Then the tractor drives towards the inside of the turn, thereby avoiding a collision with the second obstacle. At the end of the turn, both the tractor and the trailer converge to the road center.

larger vehicle.

Model fidelity

The road-aligned tractor-trailer model described in Section 6.2 is based on the well studied kinematic bicycle model. Several works in the literature have made use of this model in practical experiments, which is a good sign that the model is suitable for describing vehicle movement. We note that most works using this model consider low lateral forces, which corresponds to the use case of the articulated vehicles studied in this work, on-road driving at reasonably low speeds.

Both the linearization and discretization of the vehicle model introduce errors in the vehicle model, however the usage of an SQP strategy partially addresses these issues. When sequentially linearizing the problem and solving until convergence, the SQP algorithm ensures that the planned solution path is arbitrarily close to the linearization reference. Being close to the linearization reference implies in turn that the planned path closely follows the original nonlinear kinematic model.

Furthermore, when considering practical applications, motion planners are implemented in a receding horizon fashion. Doing so, results in the planned solution only being used during the current planning interval. At the next planning interval, the planner re-computes a new path based on the current vehicle state and environment observations. Thefore, the planner works in a closed-loop, minimizing possible errors arising from a mismatch between considered vehicle model and the actual dynamics of the real vehicle.

6.5 Conclusions

This chapter introduced a path planner for tractor-trailer combinations driving in urban environments. Existing works on motion planning for tractor-trailer vehicles mostly focuses on off-road applications, leaving a research gap open for research focuses on on-road and urban scenarios. To deal with such scenarios we have proposed a motion planner tailored for articulated vehicles.

First we formulate the tractor-trailer vehicle model in the road-aligned frame. This model builds upon the well known road-aligned vehicle model, extending it in order to allow for consideration of a trailer attached to the tractor. This model comes with its own challenges, as the road-aligned frame introduces distortions that do not allow for an analytical expression of the trailer states. This lead us to introduce approximations schemes that

allow us to model the trailer rear axle as it moves along the road, as well as to formulate collision avoidance constraints for the vehicle body.

Unlike standard passenger vehicles, articulated vehicles have multiple bodies, and therefore standard passenger vehicle optimization objectives do not suffice to center the tractor-trailer vehicle on the road. Therefore, we introduce a set of candidate optimization objectives for the optimal control problem and study their advantages and disadvantages We then select the optimization objective that results in less intrusive driving caused by the swept path of the tractor-trailer bodies, and that also lends itself suitable for real-time implementation.

The proposed approach is studied in two challenging urban driving scenarios, a roundabout and a U-turn with multiple obstacles. The planner is able to find solution paths that can center the tractor and trailer bodies, and that safely avoid obstacles in a smooth and comfortable way. A drawback of the proposed solution is the need for tuning a parameter K which depends on the vehicle dimensions. In this chapter we have resorted to a brute-force method to find this parameter K. Different vehicle dimensions required a tuning of the parameter K, which can be time consuming. However, we will show in the following chapter that an analytical expression for the K value can actually be used, increasing the generalization of this approach not only to different vehicles, but also to different road curvature profiles.

Chapter 7

On-road Path Planning for Long and Multi-Body Vehicles

In this chapter, we extend the motion planning framework presented in Chapter 6, generalizing it to both long vehicles (buses) and multi-body vehicles (tractor-trailers). This generalization is brought forward by the insight that the tuning parameter K that defines the trade-off between centering different vehicle bodies (see Equation (6.6)) can be derived analytically. This analytical derivation allows the numerical optimization formulation to obtain planned paths that are optimal according to certain performance metrics that measure the centering of the area swept by the vehicle. Furthermore, it provides a simple way to compute the value K, as opposed to having to run offline time-consuming computations that do not generalize for all vehicle dimensions and road combinations.

The work in this chapter also improves upon the planning framework presented in Chapter 5. Using a similar analytical derivation to that of the truck-trailer case, it is possible to derive an optimal trade-off parameter that is able to center the whole bus body. The analytical derivation is done by considering a trade-off parameter that weighs between centering the rear and front axles of the bus. Simulation results show that this extension leads to improvements on the driving behavior, avoiding the problem of the bus driving too close to road boundaries.

The contributions of this chapter are the following:

- geometric derivation of optimal driving objectives, focusing on centering the area swept by the vehicles, suitable for online computation;
- development of a unified framework targeting both long vehicles, such as buses, as well as multi-body vehicles, such as tractor-trailers;

- simulation results showing the proposed planner's ability to solve complicated on-road planning scenarios while considering the most challenging vehicle dimensions.

The chapter is organized as follows.

Section 7.1 introduces a unified formulation of the bus and tractor-trailer vehicle models. Using this model, we then describe the motion planning problem resorting to the already familiar numerical optimization problem formulation.

Section 7.2 first proposes the optimal driving behavior to be achieved by both long and multi-body vehicles. Then, based on this optimal behavior, we derive, through geometric arguments, the expression for the trade-off parameter K that can achieve this behavior. Although similar, the derivation for the long vehicles differs from that for multi-body vehicles, we present therefore the derivation for both types of vehicles.

Section 7.3 studies the benefits of using the proposed geometric derivations for both the bus and the tractor-trailer case. Furthermore, we present results highlighting the advantages brought forward by the capability of being able to compute the K parameter analytically. By adapting K online, the motion planner is able to adapt to the current road curvature profile, further improving the centering of the vehicle body.

Section 7.4 summarizes the chapter and lists its contributions. It also proposes directions for future work and possible extensions of the planning framework.

7.1 Motion Planning Framework

This section presents the vehicle models used for the bus and the tractor-trailer vehicles. The vehicle models are based on the road-aligned frame presented in both Chapter 5 and Chapter 6. The on-road path planning problem is then presented using a generalized formulation that encompasses both the bus and tractor-trailer systems.

Road-aligned bus model

The vehicle system is modeled in the road-aligned frame, where the geometric reference path $\gamma(\cdot)$ corresponds to the road center. The bus in the road-aligned coordinate frame is schematically illustrated in Fig. 7.2. We denote the dimensions of the bus by L_1 corresponding to the wheelbase, W corresponding to the width, and L_1^r and L_1^f are the lengths of the rear and front overhangs.

We define s as the distance traveled along the reference path γ by the rear axle. The lateral displacement of the rear axle to the reference path γ

Figure 7.1: A bus (left), with a considerable vehicle length, and a tractor-trailer (right), consisting of two vehicle bodies, are examples of heavy-duty vehicles studied in this work. The long vehicle dimensions, or the presence of multiple vehicle bodies, introduce novel challenges covered in this work. (courtesy of Scania CV AB)

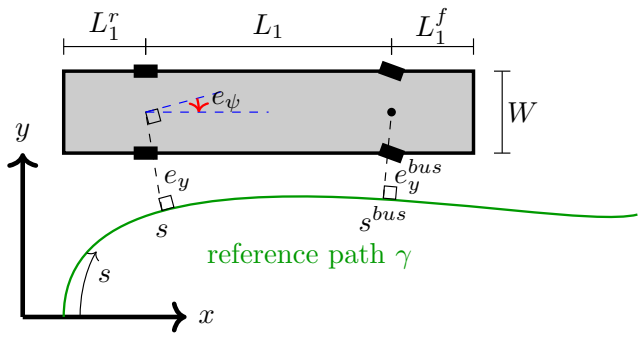


Figure 7.2: Illustration of the bus in the road-aligned frame and definitions of its dimensions and vehicle states.

is defined as e_y and the orientation difference to the reference path tangent as e_ψ. With these three quantities, the bus state is fully described by configuration vector $q = [s, e_y, e_\psi]^T$. These quantities are outlined in Figure 7.2.

The vehicle model for the bus is then given by [21]:

$$\begin{aligned} \dot{s} &= v\frac{\cos(e_\psi)}{1-e_y\kappa_\gamma(s)}, \\ \dot{e}_y &= v\sin(e_\psi), \\ \dot{e}_\psi &= v\left(\kappa - \frac{\kappa_\gamma(s)\cos(e_\psi)}{1-e_y\kappa_\gamma(s)}\right), \end{aligned} \tag{7.1}$$

where $\dot{(\cdot)} = {}^{\mathrm{d}(\cdot)}/_{\mathrm{d}t}$ and κ is the actuated variable corresponding to the curvature of the bus. The curvature of the bus is directly related to its steering angle ϕ through $\kappa = {}^{\tan(\phi)}/_{L_1}$. By restricting the attention to forward motion $v > 0$ and employing time scaling with $\dot{s} > 0$, the temporal model of Equation (7.1) is converted to an equivalent spatial model [21]:

$$\begin{aligned} e_y' &= (1-e_y\kappa_\gamma)\tan(e_\psi), \\ e_\psi' &= \frac{1-e_y\kappa_\gamma}{\cos(e_\psi)}\kappa - \kappa_\gamma, \end{aligned} \tag{7.2}$$

where $(\cdot)' = {}^{\mathrm{d}(\cdot)}/_{\mathrm{d}s}$.

Equation (7.2) describes the evolution of the lateral and orientation error of the bus rear axle, and it is so far, not different from the model introduced in Chapter 5. However, we note that this model does not contain any information related to the lateral error of the bus front axle e_y^{bus} with respect to the reference path $\gamma(\cdot)$. The lateral error of the bus front axle e_y^{bus} is essential in order to ensure a full centering of the vehicle body around the reference path γ. It becomes therefore necessary to represent this auxiliary state e_y^{bus} as a function of the rear axle variables $[e_y \;\; e_\psi]^T$. With the exception of straight reference paths, this relationship cannot be written in a purely algebraic form, as it involves a line integral [118].

To be able to consider the lateral error of the bus front axle e_y^{bus}, we introduce its approximation $\hat{e}_y^{bus}$. Similar to the approximations introduced in Chapter 6 to model the trailer rear axle, it is also possible to numerically compute an approximate relationship of e_y^{bus} which depends linearly on the states $[e_y \;\; e_\psi]^T$. We consider a linearization point $[\bar{s} \;\; \bar{e}_y \;\; \bar{e}_\psi]^T$, and make use of finite differences and iterative projection of the bus front axle in order to compute a linear model for the lateral error of the bus front axle e_y^{bus} as a function of $[e_y \;\; e_\psi]^T$. The approximation is then given by

$$\hat{e}_y^{bus} = \bar{e}_y^{bus} + \frac{\partial e_y^{bus}}{\partial e_y}(e_y - \bar{e}_y) + \frac{\partial e_y^{bus}}{\partial e_\psi}(e_\psi - \bar{e}_\psi), \tag{7.3}$$

where the partial derivatives $\frac{\partial e_y^{bus}}{\partial e_y}$ and $\frac{\partial e_y^{bus}}{\partial e_\psi}$ are computed numerically as described in Section 6.2.

The spatial model is discretized and linearized in order to be suitable for numerical optimization. The reference path is discretized along its length with a given sampling distance Δs, resulting in $\{s_i\}_{i=0}^{N}$ and $\{\kappa_\gamma(s_i)\}_{i=0}^{N}$, where $s_i = i\Delta s$. We define the state vector for the bus as $z_{bus} = [e_y \;\; e_\psi \;\; e_y^{bus}]^T$ and use Euler-forward discretization to obtain the discrete-time nonlinear model of Equation (7.2) and Equation (7.3) that is represented compactly as

$$z_{bus,i+1} = f_{bus}(z_{bus,i}, \kappa_i). \tag{7.4}$$

Road-aligned tractor-trailer model

The tractor-trailer vehicle in the road-aligned coordinate frame is illustrated in Figure 7.3. The geometric lengths for the tractor are defined analogously to the bus case and its kinematics modeled accordingly. The length L_2 is the distance between the trailer's axle and the hitch connection at the tractor, L_2^r is the trailer's rear overhang, and M_1 is the signed hitching offset at the tractor. This hitching offset is negative if the hitch connection is in front of the tractor's rear axle and positive otherwise. To model the tractor-trailer vehicle's kinematics, one needs to additionally consider state β_1, the joint angle between the tractor and the trailer. Its temporal model is given by [112]:

$$\dot{\beta}_1 = v\left(\kappa - \frac{\sin(\beta_1)}{L_2} + \frac{M_1}{L_2}\cos(\beta_1)\kappa\right), \tag{7.5}$$

and as in Equation (7.2), the equivalent spatial model is

$$\beta_1' = \frac{1 - e_y\kappa_\gamma}{\cos(e_\psi)}\left(\kappa - \frac{\sin(\beta_1)}{L_2} + \frac{M_1}{L_2}\cos(\beta_1)\kappa\right). \tag{7.6}$$

Similarly to the bus case,the models in Equation (7.2) and Equation (7.6) only provide information about the axle of the tractor, and as such, there is no explicit information regarding the axle of the trailer's lateral error e_y^{tt} with respect to the reference path $\gamma(\cdot)$. As no closed-form expression exists to express e_y^{tt} as a function of $[e_y \;\; e_\psi \;\; \beta_1]^T$ for paths with nonzero curvature, we compute an approximation $\hat{e}_y^{tt}$ using the techniques presented in [18]. Given a working point $[\bar{s} \;\; \bar{e}_y \;\; \bar{e}_\psi \;\; \bar{\beta}_1]^T$, using finite differences and by iteratively projecting the trailer's axle to the reference path $\gamma(\cdot)$ a linear model of e_y^{tt} as a function of $[e_y \;\; e_\psi \;\; \beta_1]^T$ is obtained

$$\begin{aligned}\hat{e}_y^{tt} = \bar{e}_y^{tt} &+ \frac{\partial e_y^{tt}}{\partial e_y}(e_y - \bar{e}_y) \\ &+ \frac{\partial e_y^{tt}}{\partial e_\psi}(e_\psi - \bar{e}_\psi) + \frac{\partial e_y^{tt}}{\partial \beta_1}(\beta_1 - \bar{\beta}_1),\end{aligned} \tag{7.7}$$

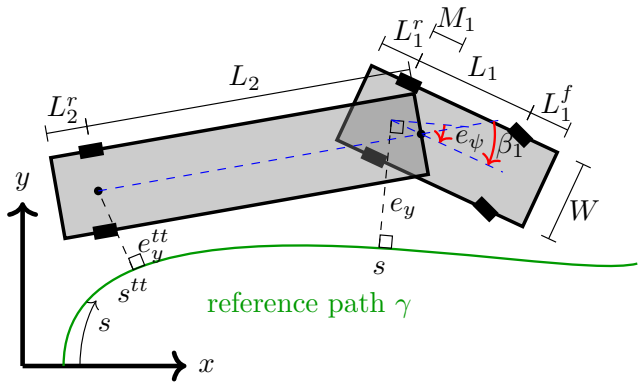


Figure 7.3: Illustration of the tractor-trailer in the road-aligned frame and definitions of its dimensions and vehicle states.

where the partial derivatives $\frac{\partial e_y^{tt}}{\partial e_y}$, $\frac{\partial e_y^{tt}}{\partial e_\psi}$ and $\frac{\partial e_y^{tt}}{\partial \beta_1}$ are computed numerically (see [18] for details).

We define the state vector as $z_{tt} = [e_y \ e_\psi \ \beta_1 \ e_y^{tt}]^T$. As in the bus case, the reference path is discretized and by performing Euler forward discretization, a discrete-time nonlinear model of the tractor-trailer vehicle Equation (7.2), Equation (7.6), and Equation (7.7) is obtained that is represented compactly as

$$z_{tt,i+1} = f_{tt}(z_{tt,i}, \kappa_i). \tag{7.8}$$

Unified numerical optimization formulation

We can now present the on-road path planning problem for the bus ($j = bus$) and tractor-trailer vehicle ($j = tt$), in a unified formulation, as the following nonlinear programming (NLP) problem:

$$\begin{aligned}
\underset{\boldsymbol{\kappa}}{\text{minimize}} \quad & \omega_\kappa J_\kappa(\boldsymbol{\kappa}) + J_j(\boldsymbol{e}_y, \boldsymbol{e}_y^j) && \text{(7.9a)}\\
\text{subject to} \quad & z_{j,i+1} = f_j(z_{j,i}, \kappa_i), \ i \in \{0, ..., N-1\}, && \text{(7.9b)}\\
& z_{j,0} = z_{\text{start}}, \ \kappa_0 = \kappa_{\text{start}}, && \text{(7.9c)}\\
& p_{e_y}^{\text{obst,s}} \leq g_j(z_{j,i}), \ i \in \{1, ..., N\}, && \text{(7.9d)}\\
& |\kappa_i| \leq \kappa_{\max}, \ i \in \{1, ..., N-1\}, && \text{(7.9e)}\\
& |\kappa_i - \kappa_{i-1}| \leq \kappa'_{\max}, \ i \in \{1, ..., N-1\}, && \text{(7.9f)}
\end{aligned}$$

where $\boldsymbol{e}_y = [e_{y,1} \ \dots \ e_{y,N}]^T \in \mathbb{R}^N$ is the vector of lateral displacements along the planned horizon. $\boldsymbol{e}_y^j = [e_{y,0}^j \ \dots \ e_{y,N}^j]^T \in \mathbb{R}^N$ is the vector of

auxiliary lateral displacements, corresponding to the front axle for the bus case, and the rear axle of the trailer for the tractor-trailer case. $\boldsymbol{\kappa} = [\kappa_0 \ \kappa_1 \ \dots \ \kappa_{N-1}]^T \in \mathbb{R}^N$ is the vector of vehicle curvatures corresponding to the actuated control input variables to be optimized. The equality constraint Equation (7.9b) corresponds to the vehicle model, where $j = bus$ implies that the bus model Equation (7.4) is used, or alternatively, in the case when $j = tt$, that tractor-trailer model Equation (7.8) is used. Constraints Equation (7.9c) are the initial constraints on vehicle's state and curvature, needed to ensure that the planned path starts from the actual vehicle state. The collision avoidance constraints as well as constraints that ensure that the vehicle wheels are kept inside of the road boundaries are implemented in Equation (7.9d). The formulation of Equation (7.9d) is done in an analogous way to that presented in Chapters 5 and 6. The vehicle maximum curvature magnitude $\kappa_{\max}$ and maximum curvature rate $\kappa'_{\max}$ limits are ensured via constraints Equation (7.9e) and Equation (7.9f), respectively.

The optimization objective Equation (7.9a) is made up of two distinct terms. The first term J_κ penalizes changes in the curvature control inputs and is defined as

$$J_\kappa(\boldsymbol{\kappa}) = \sum_{i=1}^{N-1} (\kappa_i - \kappa_{i-1})^2.$$

This term promotes a smooth curvature profile that in turn results in comfortable driving behavior for the passengers of the vehicle. The weight ω_κ defines the importance of driving in a smooth and comfortable manner.

The second term J_j penalizes the vehicle's lateral offsets and is defined as

$$J_j(\boldsymbol{e}_y, \boldsymbol{e}_y^j) = \sum_{i=1}^{N} (K_{j,i} e_{y,i} + e_{y,i}^j)^2, \tag{7.10}$$

where $K_{j,i} > 0$ is a design parameter. Since e_y and e_y^j are signed lateral errors, it is possible that $J_j = 0$ even though $\boldsymbol{e}_y$ and $\boldsymbol{e}_y^j$ are nonzero. We exploit this property in the following section, where we make use of geometric techniques to select a value of $K_{j,i}$ that encourages a desired driving behavior. We remark that $K_{j,i} > 0$ is analogous to the K parameter introduced in Equation (6.6). However, in this chapter, we allow this parameter to change along the planning horizon as that allows for an improved centering performance.

7.2 Optimal Driving Behavior

In this section, we discuss and present a desired driving behavior, which is assumed optimal, for long and multi-body vehicles. Based on this desired behavior, term $K_{j,i}$ in optimization objective (7.10) can be found using geometric derivations. We present these geometric derivations first for the bus system, and then for the tractor-trailer system. With the proposed optimization objective and proper value of $K_{j,i}$ obtained from the geometric derivations, we show that the optimal solution to (7.9) yields the desired driving behavior for the particular case of roads with constant curvature.

Desired driving behavior

As already discussed in Chapter 6, the formulation of optimization objectives for multi-body vehicles is non-trivial. In fact, due to the constraints imposed by the need to solve the optimization problem in real time, the optimization objectives used, are often simple mathematical expressions that make the optimal solution behave well according to a certain performance metric. We now proceed to present one such performance metric which will help us define an optimal behavior for a vehicle driving along a road.

When driving vehicles with large dimensions, such as long buses, or multi-body vehicles such as a tractor-trailer, simply centering one of the vehicle's axles on the center of the road does not suffice to center the whole vehicle body. Instead, one needs to pay particular attention to the whole vehicle body (or bodies), and ensure that all of it is kept as close to the center of the road as possible. We note that the same is true for regular sized vehicles, such as passenger vehicles. However, for passenger vehicles it is most often the case that driving with the rear axle centered on the road suffices to achieve an acceptable driving behavior. For long and multi-body vehicles however, this would only work when driving on roads with very low curvatures, *i.e.*, almost straight roads.

The swept area of a vehicle corresponds to the totality of the space that the vehicle body occupied while driving along the road. we will define the desired driving behavior as that which results in a swept area centered along the road. We define that sewpt area is centered if the maximum extent to which the area extends to the left and to right of the center of the road are equal. Figure 7.4 illustrates the desired driving behavior. Figure 7.4 (Top) shows the resulting swept area when considering an optimization objective that only centers the rear axle of the vehicle. As a result, the area swept by the vehicle body, as it progresses along the road, does not have an equal distance to the left and right boundaries of the road. On the contrary, in Figure 7.4 (Bottom), the green vehicle has a more desirable driving behavior. Even though its rear axle is not centered on the road, the spread of

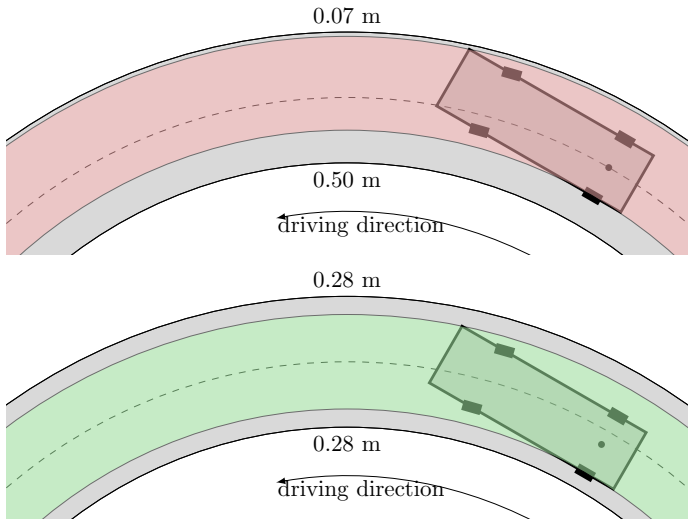


Figure 7.4: We define the desired behavior to be that which achieves the best centering of the whole vehicle swept area. Top: The vehicle rear axle follows the center of the road, shown as a dotted line. The swept area is shown in red. It can be seen that the swept area tends to the right side of the road. Bottom: The vehicle has a swept area, shown in green, that is equally distant to both the left and right limits of the road, corresponding to the desired driving behavior defined earlier. This desired driving behavior is only possible because the rear axle is allowed to deviate from the center of the road.

its swept area is at equal distances to the left and the right boundaries of the road. In the following sections, we formulate an optimization objective expression that can achieve this desired optimal driving behavior for the bus and tractor-trailer cases.

Derivation for the bus case

Figure 7.5 illustrates the scenario of a bus driving along a road with a constant radius R_{road} and achieving the desired driving behavior. We assume that the bus drives with a constant curvature $\kappa > 0$ which in turn results in a constant, but unknown, turning radius $R_1 = 1/\kappa$. Since $|\kappa| \leq \kappa_{max}$, the bus turning radius satisfies $|R_1| \geq 1/\kappa_{max}$. Without loss of generality, we assume as well that the bus is driving in a left turn, resulting in $R_1 > 0$. While making a left turn, the swept area is delimited by the radius $R_{bus,l}$ corresponding to the path traveled by the bus rear left wheel, and by the

radius $R_{bus,r}$ corresponding to the path traveled by the front right corner of the bus body. In order for the area swept by the vehicle body to be equally spread to the right and to the left of the road center, the following must hold:

$$R_{\text{road}} = \frac{R_{bus,l} + R_{bus,r}}{2}. \tag{7.11}$$

Since we consider a constant bus turning radius R_1, basic trigonometry gives that the inner and outer radii $R_{bus,l}$ and $R_{bus,r}$ are given by:

$$\begin{aligned} R_{bus,r}^2 &= \left(R_1 + \frac{W}{2}\right)^2 + \left(L_1 + L_1^f\right)^2, \\ R_{bus,l} &= R_1 - \frac{W}{2}, \end{aligned} \tag{7.12}$$

where it is assumed that $R_1 > W/2$. We remark that this assumption does not pose any practical restrictions on the derivations, as the minimum turning radius of a bus is typically much larger than its body width. Since $R_{bus,r} > 0$ in a left turn, inserting (7.12) in (7.11) yields:

$$R_{\text{road}} = \frac{\sqrt{\left(R_1 + \frac{W}{2}\right)^2 + \left(L_1 + L_1^f\right)^2} + R_1 - \frac{W}{2}}{2}, \tag{7.13}$$

which is a nonlinear equation in the unknown variable R_1. For roads with radius R_{road} such that $R_1 > W/2$, the unique and positive solution to (7.13) is

$$R_1 = \frac{-\left(L_1 + L_1^f\right)^2 + 4R_{\text{road}}^2 + 2WR_{\text{road}}}{4R_{\text{road}} + 2W}. \tag{7.14}$$

Equation (7.14) gives the optimal turning radius of the bus R_1 as a function of the road curvature R_{road}, which is optimal in the sense that the bus left swept width, given by $R_{\text{road}} - R_{bus,l}$, and right swept width, given by $R_{bus,r} - R_{\text{road}}$, are equal. This is what we previously defined as the desired behavior since it perfectly centers the area swept by the vehicle body around the road center.

From the derived turning radius of the bus R_1 we can obtain the constant signed lateral errors of the front axle, e_y^{bus}, and of the rear axle, e_y, as:

$$\begin{aligned} e_y^{bus} &= R_{\text{road}} - \sqrt{L_1^2 + R_1^2}, \\ e_y &= R_{\text{road}} - R_1, \end{aligned} \tag{7.15}$$

where $e_y^{bus} < 0$ and $e_y > 0$. To make the optimization objective $J_{bus} = 0$ at this stationary configuration, we get from (7.10) that $K_{bus}e_y + e_y^{bus} = 0$

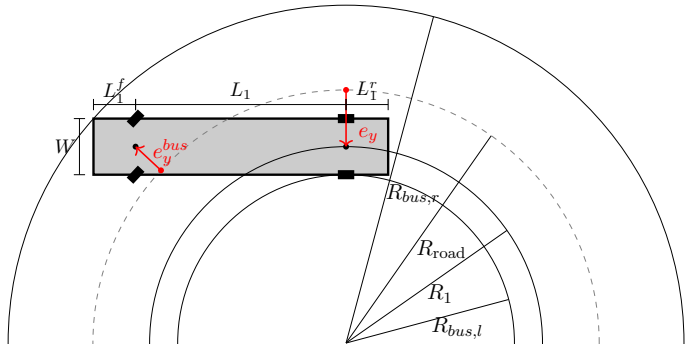


Figure 7.5: Geometric illustration of optimal road centering for a bus on a counterclockwise turn (left turn) with constant radius R_{road}.

must hold. This condition together with (7.15) gives the optimal tuning strategy

$$K_{bus}(R_{\text{road}}) = \frac{\sqrt{L_1^2 + R_1^2} - R_{\text{road}}}{R_{\text{road}} - R_1}. \tag{7.16}$$

The above derivation considered a vehicle driving counter-clockwise (taking a left turn), however for the case of a clockwise turn (right turn) with equal radius, the same geometrically derived tuning of K_{bus} can be used.

With the proposed tuning of K_{bus}, and assuming that no obstacles or other vehicle constraints are present, the optimization objectives J_{bus} and J_κ will obtain their minimum value of zero when the vehicle moves along the road with a constant curvature $\kappa = 1/R_1$, where R_1 is given by (7.14). Therefore, when using this tuning strategy, the optimization-based path planner is encouraged to find a solution with the desired behavior defined in Section 7.2.

Derivation for the tractor-trailer case

We now derive the optimal value K_{tt} for the tractor-trailer case. The derivation is similar to that of the bus case, however with some extra steps related to computing the stationary equilibrium configuration of the articulation angle of the tractor-trailer.

Figure 7.6 shows the tractor-trailer vehicle driving with the desired driving behavior along a road with radius R_{road}. The tractor-trailer vehicle is in a stationary circular equilibrium configuration where the tractor has a constant curvature κ and the articulation angle is static ($\beta_1' = 0$). In these

conditions the articulation angle β_1 can be obtained by [123]:

$$\beta_1 = \left(\arctan\left(\frac{M_1}{R_1}\right) + \arctan\left(\frac{L_2}{R_2}\right)\right), \tag{7.17}$$

where the signed radii $R_1 = 1/\kappa$ and $R_2^2 = R_1^2 + M_1^2 - L_2^2$. The area swept by the vehicle bodies is determined by radius $R_{tt,l}$ corresponding to the path traveled by the rear left wheel of the trailer, and by the radius $R_{tt,r}$ corresponding to the path traveled by the front right corner of the tractor body. As in the bus case, in order to achieve the desired driving behavior, the following must hold:

$$R_{\text{road}} = \frac{R_L + R_R}{2}. \tag{7.18}$$

Since the turning radius of the tractor $R_1 = 1/\kappa$ and the joint angle β_1 are constant, from trigonometry we get that $R_{tt,l}$ and $R_{tt,r}$ are given by:

$$\begin{aligned} R_{tt,r}^2 &= (R_1 + W/2)^2 + \left(L_1 + L_1^f\right)^2, \\ R_{tt,l} &= R_2 - W/2. \end{aligned} \tag{7.19}$$

Here, we have assumed that the turning radius of the trailer axle $R_2 > W/2$, which is typically true when considering on-road driving. Since $R_{tt,r} > 0$, inserting (7.19) in (7.18) gives

$$\begin{aligned} 2R_{\text{road}} =& \sqrt{R_1^2 + M_1^2 - L_2^2} - W/2 \\ &+ \sqrt{(R_1 + W/2)^2 + \left(L_1 + L_1^f\right)^2}, \end{aligned} \tag{7.20}$$

which is a nonlinear equation in the variable R_1.

The positive solution to (7.20) can then be represented as

$$R_1 = g(R_{\text{road}}, W, L_1, L_2, M_1, L_1^f). \tag{7.21}$$

Function g can be found using a symbolic equation solver, in this work we have used MATLAB's Symbolic Math Toolbox. Function g is computed using the amentioned toolbox, however, due to its extensive length, it is not presented in written here.

We can now compute R_2 and the joint angle β_1 using (7.17). From the derived R_1 and R_2, the constant signed lateral errors e_y^{tt} and e_y are given by:

$$\begin{aligned} \boldsymbol{e}_y &= R_{\text{road}} - R_1, \\ \boldsymbol{e}_{\boldsymbol{y}}^{\boldsymbol{tt}} &= R_{\text{road}} - \sqrt{R_1^2 + M_1^2 - L_2^2}, \end{aligned} \tag{7.22}$$

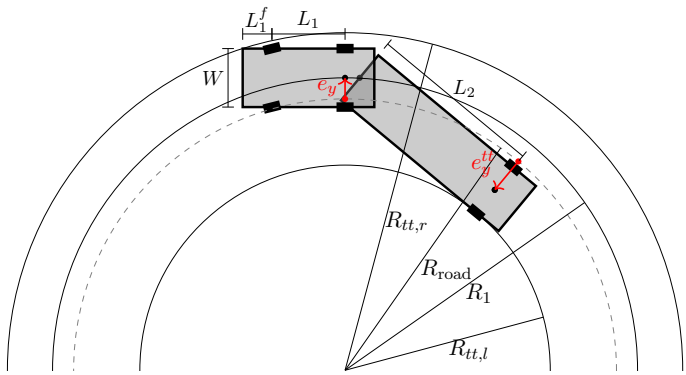


Figure 7.6: Geometric illustration of the stationary and optimal road centering of a tractor-trailer vehicle around a counterclockwise turn with constant radius R_{road}.

where $e_y < 0$ and $e_y^{tt} > 0$. As in the bus case, in order to make the optimization objective $J_{tt} = 0$ at this stationary configuration, we get from (7.10) that $K_{tt}e_y + e_y^{tt} = 0$ must hold. This condition together with (7.22) gives the optimal tuning parameter:

$$K_{tt}(R_{\text{road}}) = \frac{R_1 - R_{\text{road}}}{R_{\text{road}} - \sqrt{R_1^2 + M_1^2 - L_2^2}}, \tag{7.23}$$

where R_1 is obtained from (7.21). We note that in case we consider a clockwise turn with equal radius, the same geometrically derived tuning of K_{tt} can be used.

With the proposed optimal value of K_{tt}, and in the absence of obstacles or other additional vehicle constraints, the optimization objectives J_{tt} and J_κ will be exactly zero when the tractor-trailer vehicle moves along the road with a constant curvature of the tractor $\kappa = 1/R_1$ and a constant joint angle (7.17). As a result, when using this tuning strategy the optimization-based path planner is encouraged to find a solution that achieves the desired behavior of having a balanced swept area of the tractor-trailer bodies to the left and the right of the road center, corresponding to the desired behavior defined in Section 7.2.

Roads with varying curvature

The geometrically derived values of tuning parameters K_{bus} and K_{tt} can now be used in the path planner's optimization objective (7.10). If the road has a constant curvature, one can simply define $K_{j,i}$ in (7.10) to be equal

to the derived K_j in Section 7.2 and Section 7.2. We note however that the geometric derivations in Section 7.2 and Section 7.2 assume a road with constant curvature, which is not true when considering a generic driving situation in which the road has a varying curvature. To deal with this limitation of the geometric derivation we propose to update $K_{j,i}$ along the planning horizon.

We choose the value $K_{j,i}$ at each point along the sampled reference path $\{\gamma(s_i)\}_{i=0}^{N}$ to be computed based on the current curvature $\kappa_\gamma(s_i)$ at that sampled point. Unfortunately we can no longer state that this tuning strategy is optimal for road with varying curvature, but we show in the following results section, that this strategy does have its benefits. First, we show that this strategy results in planned paths that have a behavior that is close to the one expected based on constant curvature assumptions. Second, we show that using this adaptive strategy is provides better centering results than considering a constant value of $K_{j,i}$ along the whole planning horizon.

7.3 Results

Here we present results showing the advantages brought forward by the usage of the optimally tuned trade-off parameter. The results are obtained using a computer with an Intel Core i7-6820 HQ@2.7GHz CPU, and with motion planning framework implemented in MATLAB code. We use the convex quadratic program solver OSQP [124] to solve the intermediate SQP iterations of the motion planning problem.

Bus in a U-turn

The vehicle dimensions are those of the prototype autonomous bus shown in Figure 5.6, and correspond to a wheelbase length of $L_1 = 6$ m, a front overhang length of $L_1^f = 3.34$ m, a rear overhang length of $L_1^r = 2.66$ m, and a vehicle width $W = 2.54$ m.

Figure 7.7 presents the path planning results for a bus driving on a U-turn, when considering the planning framework introduced in Chapter 5 and when considering the planning framework proposed in this chapter. It can be seen that the planning framework of Figure 7.7 results in the bus driving excessively close to the outer road limits. This is a direct result of the optimization objective only considering the lateral displacement of the rear axle e_y. The bus only tries to center its rear axle, and therefore it has no incentive to center the front axle on the road, resulting in the front right corner of the vehicle driving on top of the lane limit. We note that the vehicle body is kept inside of the lane limits due to the overhang

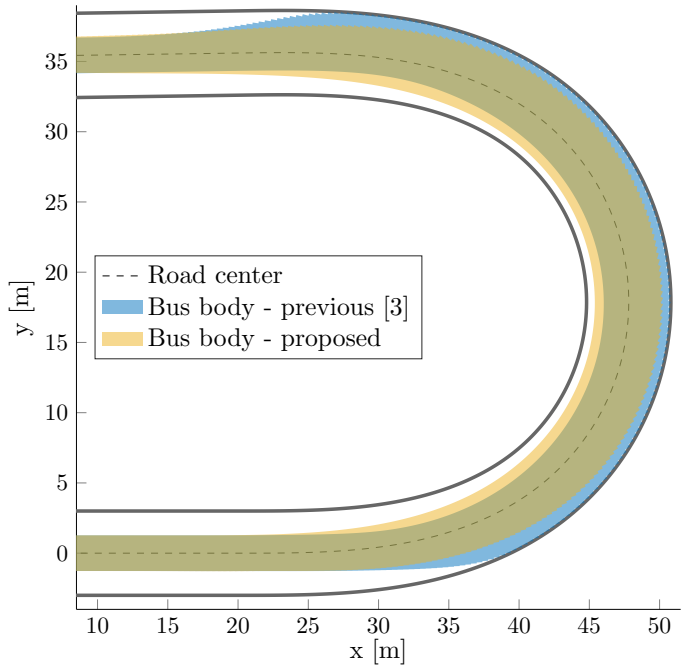


Figure 7.7: The swept area in blue corresponds to the planned path when using the planning framework developed in Chapter 5. It can be seen that the bus drives with the body corner exactly on the lane limit. The swept area in yellow corresponds to the planned path when considering the method proposed in the current chapter. Using the optimal tuning derived earlier results in the bus centering itself on the road and keeping a significant clearance to the lane limit.

minimization objective that penalizes unnecessary exiting of the overhangs from the drivable region.

Figure 7.7 presents as well the planned path when considering the extensions proposed in this chapter. It can be seen that when using the optimization objective that considers both the rear and the front axle results in the vehicle not driving too close to the lane limits. Furthermore, when using the analytical derivation to compute the optimal trade-off parameter K_{bus} results in the bus swept area being centered on the road, keeping an equal distance to the left and right lane limits. This illustrates the benefits of using an optimization objective that considers both the rear and front axle, as well as the importance of computing an appropriate parameter K_{bus}.

Tractor-trailer in a roundabout

We now consider the tractor-trailer driving in a roundabout scenario as previously considered in Section 6.4. The dimensions of the system are given by a tractor wheelbase $L_1 = 3.47$ m, a tractor front overhang $L_1^f = 1.16$ m, a tractor rear overhang $L_1^r = 1.34$ m and a hitch length $M_1 = -0.30$ m. The trailer dimensions correspond to an axle length $L_2 = 9.40$ m and a rear overhang $L_2^r = 3.03$ m. Both the tractor and the trailer have a width $W = 2.54$ m. The considered roundabout has a turning radius of 17.88 m.

Figure 7.8 shows the swept area of the planned path when considering the proposed geometric method to obtain the optimal weighting parameter. It can be seen that the area swept by the tractor-trailer bodies are precisely centered during the roundabout maneuver. We note that unlike the results presented in Section 6.4, here we did not have to find a suitable K parameter through time-consuming offline computations. Instead, the optimal K_{tt} is obtained using the geometrical derivations, that can be computed in an online fashion. This represents a significant advantage over the previous framework considered in Chapter 6.

Figure 7.9 presents in close detail the envelope corresponding to the area swept by the vehicle bodies. Figure 7.9 shows both the envelope corresponding to the planned path obtained by solving the nonlinear optimization problem Equation (7.9) and the geometrically derived envelope. The geometrically derived envelope corresponds to the expected envelope width (according to the derivations in Section 7.2) when considering a road with a constant curvature equal to the curvature at the current road length s. Figure 7.9 shows that the vehicle envelope is well centered along the whole maneuver, as the maximum envelope widths to the left and right of the road center only differ by 0.04 m. Furthermore, it can be seen that the executed envelope width is very close to the geometrically derived envelope width. A transient behavior occurs at the entrance and exit of the roundabout, however, for most of the maneuver, the vehicle drives according to the geometrically derived optimal stationary behavior.

To further validate our geometrical approach, we compare the derived optimal vehicle curvature with the curvature of the planned path. Figure 7.10 (Top) shows the derived optimal vehicle curvature, obtained along the road length s, with the curvature of the planned path obtained as a solution of the nonlinear optimization problem Equation (7.9). Similar to the behavior noticed with the vehicle envelopes (see Figure 7.9), the planned curvature follows the optimal curvature very closely, with the exception of transients at the entrance and exit of the roundabout. Figure 7.10 (Bottom) also compares the planned and optimal articulation angles β. The planned articulation angle also follows its geometrically derived equivalent, however, it is characterized by significantly longer transient behavior due to a slower

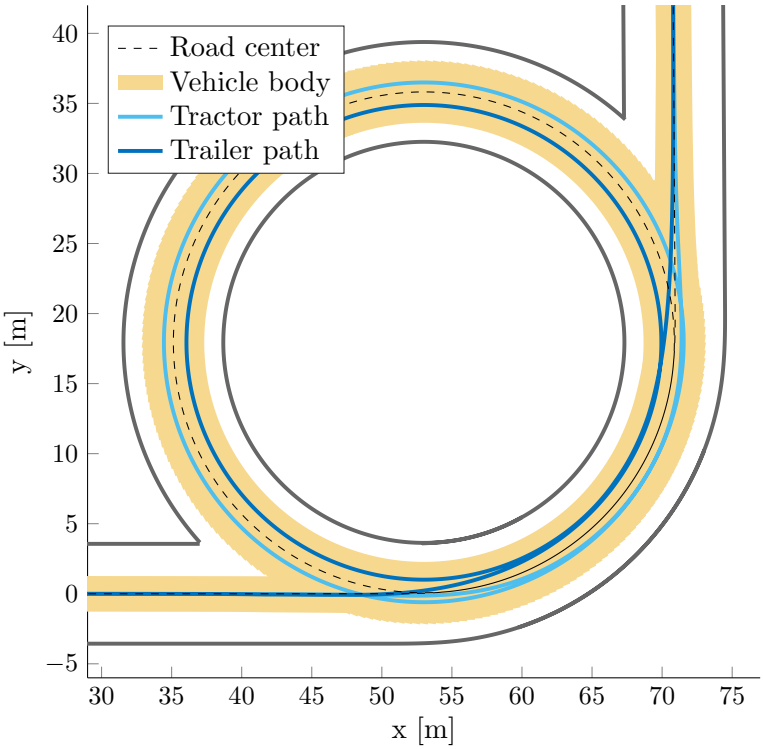


Figure 7.8: The planned path results in the tractor-trailer vehicle centering its whole body as it drives along the roundabout. The swept area keeps a similar clearance distance to both the left and right lane limits, mimicking the desired driving behavior introduced in Section 7.2.

response time of the articulation angle.

Results in S-turns

The geometric derivations in Section 7.2 assume roads with constant curvature, however this assumption does not hold in practice. Section 7.2 proposes that the values of $K_{bus,i}$ and $K_{tt,i}$ are computed online and updated along the planning horizon. For each state in the planning horizon, we compute its respective $K_{bus,i}$ or $K_{tt,i}$ value in accordance with the road curvature at that point. In this way, we can use the derivations from the stationary case that assume a road with constant curvature, in arbitrary roads with varying curvature.

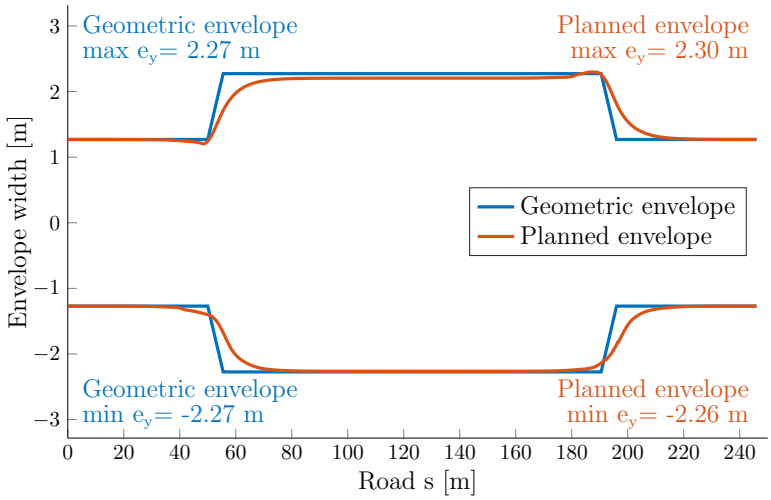


Figure 7.9: Envelopes of the geometrically derived optimal path and of the planned path as the vehicle progresses along the roundabout in Figure 7.8. The proposed motion planner achieves a balanced tractor and trailer centering, where the maximum left and right widths correspond to 2.30 m and 2.26 m respectively. Both the left and right widths are close to the geometrically derived width of 2.27 m.

We compare the results of using a varying $K_{bus,i}$ and $K_{tt,i}$ that adapts to the road curvature against using a fixed K_{bus} and K_{tt}. The fixed K_{bus} and K_{tt} strategy computes the tuning parameters based on the maximum curvature of the road. Both strategies are evaluated by solving the path planning problem Equation (7.9) on a wide range of S-turns. The different S-turns are created by varying the maximum curvature of the turns, as well as the sharpness (rate of curvature change) of the transition segments connecting the straight segments, turns, and counter-turns.

For each sampled path point i along a planned path we measure d_i, corresponding to how much the vehicle envelope extends towards the left side of the road subtracted by the amount that it extends towards the right side. We then define the total envelope displacement along each planned path as $D = \sum_{i=1}^{N} d_i$. For a perfectly centered maneuver, one gets $D = 0$, corresponding to the desired driving behavior where the vehicle is equally distant to both the left and right limits of the road.

We measure D_{fixed} and $D_{varying}$ corresponding to the total envelope displacement for planned paths assuming a fixed or varying $K_{j,i}$ in Equation (7.9), respectively. The overall improvement in percentage is defined as

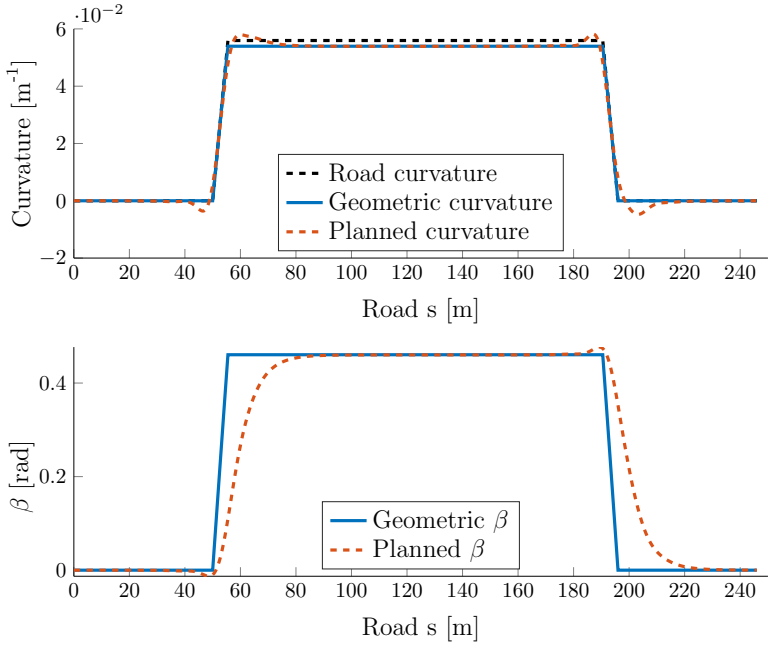


Figure 7.10: Comparison between the geometrically derived and the planned tractor-trailer vehicle states. Top: Curvature of the tractor. Bottom: Tractor-trailer articulation angle β.

$100 \times (D_{varying} - D_{fixed})/D_{fixed}$, and its value is for a large set of possible S-turns and for both the bus, Figure 7.11 (Top), and tractor-trailer, Figure 7.11 (Bottom). For the bus case, using a varying $K_{j,i}$ instead of a fixed one results in planned paths that improve the average displacement D by ip to 10%. This effect is more noticeable in S-turns with high curvature. When considering the tractor-trailer case, we see that the benefits are much more accentuated in S-turns with high curvature and low sharpness, where the improvement of the average displacement D is around 5%. The measured improvements in the displacement D, together with the capability to compute the optimal varying $K_{j,i}$ values online, represent significant improvements over the offline tuning strategy of a fixed K_j value, as previously considered in Chapter 6.

7.4 Conclusions

This chapter has presented an extension to the on-road motion planning frameworks presented in Chapters 5 and 6. The proposed extension tar-

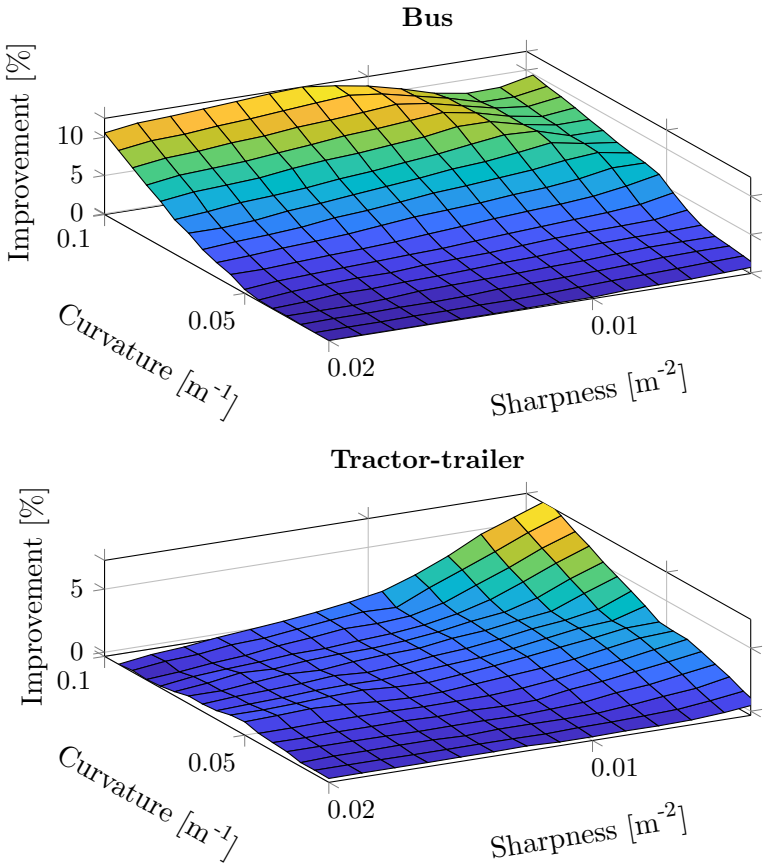


Figure 7.11: Improvement of the average displacement D when using a varying $K_{j,i}$ compared to using a fixed K_j. Top: Improvement for the bus case $j = bus$. Bottom: Improvement for the tractor-trailer case $j = tt$.

gets both buses and tractor-trailer vehicles resulting in a unified framework that can tackle a large number of possible heavy-duty vehicle configurations. Simulation results have shown that the proposed framework is able to improve the driving behavior of both long and multi-body vehicles.

To achieve these improvements we have first defined a desired driving behavior as that resulting in the whole vehicle body driving as centered on the road as possible. This is, intuitively, a desired property, as it indicates the the vehicle body is equally distant to both the left and the right limit, keeping a balanced clearance to both lane limits. Once the desired driving behavior is defined, we derive, via geometric arguments, an optimization objective that can achieve this driving behavior in roads with constant curvature. This optimization objective is suitable for real-time implementation, as it is a computationally friendly optimization objective that lends itself suitable for numerical optimization approaches. The geometric derivations are also computationally modest, allowing for a continuous adaptation of the optimization objective to the current road curvature profile.

Directions of possible future work include the generalization of the developed framework to more complex vehicles, such as vehicles with multiple actuated steering axles, and articulated vehicles composed of a tractor, a dolly, and a trailer. Furthermore, it would be of interest to see the if the benefits of the geometrically derived optimization objective also translate to other on-road motion planning approaches that are not based on numerical optimization, such as RRT or lattice-based planners. It is also of interest to consider different desired driving behaviors besides that of keeping equal distance to both lane limits. As an example, based on the current traffic situation, it might be beneficial to plan paths that maximize the distance between the vehicle swept area and oncoming traffic. In such traffic situations, it might be valuable to reduce the clearance to one of the lane limits, in order to increase clearance, and thefore safety distance, to the opposing traffic lane.

Chapter 8

On-road Path Planning Experimental Results

The motion planning module is tightly integrated with other autonomous vehicle modules. Therefore, it is essential to study the performance of the whole autonomous driving framework in real systems in order to understand the suitability of the proposed motion planning solutions. In this chapter we present several practical experiments perfomed in an prototype autonomous bus. This chapter can be seen partially as an extension of the results presented in Chapters 5 and 7, and partially as further development of the approaches previously presented so as to bring the proposed path planning frameworks from simulation environments and into the real world.

Several aspects that have not been studied in previous chapters can impact the overall performance of the motion planning framework. First, the motion planning module uses information from the vehicle's sensing, perception, and localization modules. These modules give the motion planner the information about the environment surrouding the vehicle, and the current vehicle state. In simulations one often assumes to have the real state of the vehicle and an accurate description of the environment, however in reality, sensor noise and localization uncertainties can affect this information. Practical experiments are therefore necessary to study the feasibility towards these innacuracies. Second, the planned paths are never precisely executed by the vehicle. When a path is planned, it is sent to the vehicle control module, which will try to follow the path as closely as possible. However, it is in practice impossible to follow a planned path exactly, and the path planner must be adapted to deal with this. Finally, when considering a real system one needs to pay extra attention to computational times. It is important to ensure that the implemented algorithms can run in real-time, respecting the computational time allowed for each planning

cycle. This becomes extra challenging, when considering a computational unit that needs to run multiple modules (besides the path planning module) of the autonomous driving software stack.

The contributions of this chapter are the following:

- extensions and modifications to the previously developed path planner that lead to successful real-life implementation;
- proposal of the novel concept of wheel-aware planning that explicitly considers that the front wheels of the vehicle move relative to the chassis, possibly protruding and colliding with curbs;
- experimental validation of the proposed method by navigating an autonomous bus on urban-like roads;
- demonstration of the benefits of the proposed planner through practical experiments in challenging scenarios.

The chapter is organized as follows.

Section 8.1 details the numerical optimization formulation and the solving technique used to find a solution path. The section also describes important design choices that are made in order to tackle issues arising from the implementation on a real vehicle. Namely, we present adaptations to the planned solution paths so as to simplify the task of the downstream controller module.

Section 8.2 presents extensive results of the motion planning framework. These results are obtained by implementing the proposed approach on a Scania prototype bus driving on a test track. Several experiments corresponding to challenging urban driving scenarios are presented, highlighting the benefits of the method. Finally, a study of the computational times is presented.

Section 8.3 presents conclusions related to the performed real life experiments, and lists challenges and future work directions that were identified during tests with the real vehicle. Most importantly, a significant limitation, associated with bus stop maneuvers is presented, motivating the future study and development of *wheel-aware* planners.

8.1 Motion Planning Framework

This section details the numerical optimization formulation considered during the practical experiments. The formulation is based on the formulations already introduced in Chapters 5 and 7. From Chapter 5 we make use of the formulation which tackles the challenges of buses driving on urban scenarios, and from Chapter 7 we use the optimal centering derivations suitable

for long vehicles driving on roads. We present in detail the strategy of receding horizon planning, used to alleviate the computational burden of the motion planner. Finally we introduce the idea of planning paths that are consistent for the motion controller module, something that is crucial for good path tracking performance.

Numerical optimization formulation

The on-road path planning problem is formulated by combining Equation (5.14) and Equation (7.9) into the following nonlinear programming problem:

$$\begin{aligned}
\underset{\boldsymbol{\kappa}}{\text{minimize}} \quad & J_{\text{center}} + J_{\text{smooth}} + J_{\text{overhang}} && \text{(8.1a)}\\
\text{subject to} \quad & z_{bus,i+1} = f_{bus}(z_{bus,i}, \kappa_i),\ i \in \{0, ..., N-1\}, && \text{(8.1b)}\\
& z_{bus,0} = z_{\text{start}},\ \kappa_0 = \kappa_{\text{start}}, && \text{(8.1c)}\\
& p_{e_y}^{\text{obst,s}} \leq g_{bus}(z_{bus,i}),\ i \in \{1, ..., N\}, && \text{(8.1d)}\\
& |\kappa_i| \leq \kappa_{\max},\ i \in \{1, ..., N-1\}, && \text{(8.1e)}\\
& |\kappa_i - \kappa_{i-1}| \leq \kappa'_{\max},\ i \in \{1, ..., N-1\}. && \text{(8.1f)}
\end{aligned}$$

where Equation (8.1b) enforces the vehicle model, in this case the road-aligned vehicle model presented in Equation (5.1). Equation (8.1c) corresponds to the initial vehicle state and input constraints. For implementation in a real vehicle, some adaptations to the initial constraints are required. These adaptations are explained in the coming subsections. Equation (8.1d) enforces the bus body constraints, so as to comply with the environment classification (drivable, sweepable, and obstacle) detailed in Section 5.3. The steering actuator magnitude and rate constraints are enforced via Equations (8.1e) and (8.1f).

The optimization objectives are composed of three terms. J_{smooth} minimizes the squared rate of change of the input,

$$J_{\text{smooth}} = \sum_{i=1}^{N-1} \left(\kappa_i - \kappa_{i-1}\right)^2, \tag{8.2}$$

prioritizing a smooth and comfortable ride. To avoid the vehicle body from excessively exiting the lane limits, J_{overhang} penalizes the amount of overhang that goes into the sweepable areas. Finally, J_{center} tries to center the whole vehicle body, according to the formulation introduced in Chapter 7:

$$J_{\text{center}} = \sum_{i=1}^{N} (K_{bus,i} e_{y,i} + e_{y,i}^{bus})^2, \tag{8.3}$$

where the tuning parameter $K_{bus,i}$ is computed using the geometric derivations in Section 7.2.

To find a solution, not necessarily optimal, to the nonlinear optimization problem Equation (8.1), we make use of Sequential Quadratic Programming (SQP). The SQP implementation is specially adapted to usage in real-world settings, by solving the problem in a receding horizon fashion, as explained below.

SQP implementation in receding horizon fashion

Sequential Quadratic Programming (SQP) is a popular technique for solving optimization problems with nonlinear constraints [125]. In the SQP technique the nonlinear constraints of Equation (8.1) are linearized, resulting in an optimization problem, with a quadratic objective function and linear constraints. We note that all terms J_{center}, J_{smooth}, and J_{overhang} in Equation (8.1a) are quadratic and convex, resulting in a quadratic and convex objective function.

The original nonlinear problem is linearized around linearization references $\bar{\mathbf{u}}$, $\bar{\mathbf{e}}_y$, $\bar{\mathbf{e}}_\psi$. After linearization, the new optimization problem can then be solved using a Quadratic Program solver such as [124]. The resulting solution from the solver gives us the optimal inputs $\mathbf{u}^*$, and vehicle states $\mathbf{e}_y^*$, $\mathbf{e}_\psi^*$. The process is then repeated several times, by making the linearization references $\bar{\mathbf{u}}_k$, $\bar{\mathbf{e}}_{\mathbf{y},k}$, $\bar{\mathbf{e}}_{\psi,k}$ of SQP iteration k, equal to the optimal solution $\mathbf{u}_{k-1}^*$, $\mathbf{e}_{\mathbf{y},k-1}^*$, $\mathbf{e}_{\psi,k-1}^*$ of the previous SQP iteration. This SQP procedure is shown in Algorithm 5.

The SQP loop stops once it fulfills the termination condition. This termination condition can either be a limit on the number of loop iterations to be computed, a limit on the computational time allowed, or requirement for the solution to converge. For the latter, the solution is said to converge if

$$\begin{aligned}\|\bar{\mathbf{u}}_{k-1} - \mathbf{u}_k^*\|_2 &\leq \epsilon_u,\\ \|\bar{\mathbf{e}}_{\mathbf{y},k-1} - \mathbf{e}_{\mathbf{y},k}^*\|_2 &\leq \epsilon_{e_y},\\ \|\bar{\mathbf{e}}_{\psi,k-1} - \mathbf{e}_{\psi,k}^*\|_2 &\leq \epsilon_{e_\psi},\end{aligned}$$

where ϵ_u, ϵ_{e_y}, ϵ_{e_ψ}, are user-defined tolerances. As the SQP progresses along iterations, and as the optimal solutions get closer to the linearization references, the fidelity of the linearized QP improves, leading to an accurate approximation of the original nonlinear optimization problem.

A variant of the SQP method detailed in Algorithm 5 can be developed for the case when the vehicle is moving along the road. In this variant of the SQP method, the motion planning framework will only run one iteration

Algorithm 5: Sequential Quadratic Programming

Input: $\bar{\mathbf{u}}_0$, $\bar{\mathbf{e}}_{\mathbf{y},0}$, $\bar{\mathbf{e}}_{\psi,0}$
Output: $\mathbf{u}^*$, $\mathbf{e}_y^*$, $\mathbf{e}_\psi^*$

```
k ← 1;
repeat
    Obtain QP problem by linearizing the nonlinear optimization
      problem Equation (8.1) using references ū_{k−1}, ē_{y,k−1}, ē_{ψ,k−1};
    Solve QP problem to obtain optimal solution u*_k, e*_{y,k}, e*_{ψ,k};
    ū_k ← u*_k;
    ē_{y,k} ← e*_{y,k};
    ē_{ψ,k} ← e*_{ψ,k};
    k ← k + 1;
until termination condition;
u* ← u*_{k−1};
e*_y ← e*_{y,k−1};
e*_ψ ← e*_{ψ,k−1};
```

of the SQP loop in Algorithm 5 at each planning cycle. Let us consider that each planning cycle occurs at time $t = k\Delta T$, where ΔT is the planning cycle period, and k indicates the k-th planning cycle. At the k-th planning cycle, the vehicle uses linearization references $\bar{\mathbf{u}}_{k\Delta T}$, $\bar{\mathbf{e}}_{\mathbf{y},k\Delta T}$, $\bar{\mathbf{e}}_{\psi,k\Delta T}$, and computes the optimal solution $\mathbf{u}^*_{k\Delta T}$, $\mathbf{e}^*_{\mathbf{y},k\Delta T}$, $\mathbf{e}^*_{\psi,k\Delta T}$.

As the vehicle drives along the road the road references shift, and as such, the linearization references of the previous $k-1$-th planning cycle do not fully cover the planning horizon of the current k-th planning cycle. As an example consider Figure 8.1 that shows a vehicle moving along the road at three time instants, corresponding to three consecutive planning cycles. It can be seen that at each planning cycle the planning horizon is shifted forwards, the so-called *receding horizon*. Since the planning horizon moves forward, and since we are considering only one iteration of the SQP loop per planning cycle, the linearization references of the previous planning cycle $\bar{\mathbf{u}}_{(k-1)\Delta T}$, $\bar{\mathbf{e}}_{\mathbf{y},(k-1)\Delta T}$, $\bar{\mathbf{e}}_{\psi,(k-1)\Delta T}$ cannot directly be used in the k-th planning cyle. To deal with this, a few modifications to the SQP algorithm need to be made.

It can be seen in Figure 8.1 that even though the planning horizon is shifting, the majority of the horizon in between consecutive planning cycles is still overlapping. Let the linearization reference be defined as:

$$\bar{\mathbf{u}}_{k\Delta T} = [\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_M, \bar{u}_{M+1}, \ldots \bar{u}_N]^T \in \mathbb{R}^N,$$

and equivalently for $\bar{\mathbf{e}}_{\mathbf{y},k\Delta T}$, and $\bar{\mathbf{e}}_{\psi,k\Delta T}$. The first M elements of the lin-

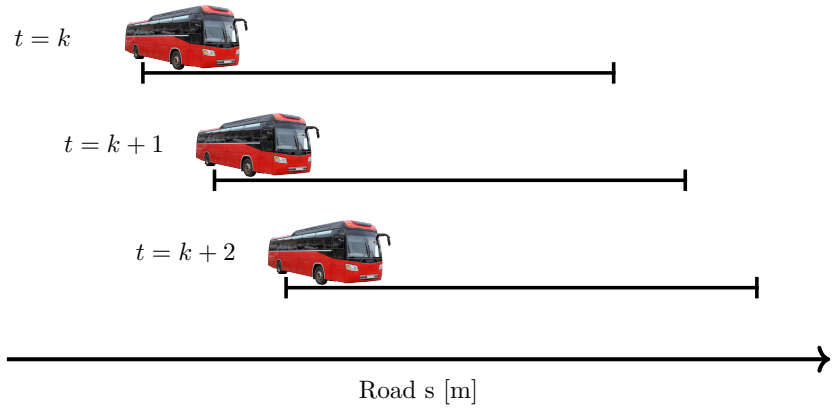


Figure 8.1: The planning horizon during successive planning cycles, as the bus progresses along the road. It can be seen that a planning horizon has a large overlap with the horizon of the previous planning cycle.

earization references correspond to elements which overlap with the previous $k-1$-th planning cycle horizon. The remaining $N-M$ elements are those which are located in a new section of the road that was previously unseen. To reuse the computed results of the previous planning cycle the first M elements are set according to the optimal solution of the previous planning cycle. The remaining $N-M$ elements are set to be equal to the M-th element. By doing so, we create a new linearization reference that make uses of the previously computed solution where possible, resembling the original SQP formulation in Algorithm 5.

There are different ways to set the first M elements of the linearization reference. One possibility is to interpolate the state and input values according to the respective road s coordinate. Each reference and solution vector has an associated set of s coordinates $[s_1, \ \ldots, \ s_N]^T \in \mathbb{R}^N$, that associates each element of the reference vector to a road location. When computing the new reference vector to be used for the sampled states of the current planning horizon, one can use interpolation to infer the reference values at the new road locations.

A second possible way to set the first M elements, and the one that we use in these experiments, is to sample points along the road in a way that is consistent with previous planning cycles. To do this, we ensure that at every planning cycle, the sampled states in the planning horizon section that overlaps with the previous planning horizon, have the same s coordinates as the samples in the previous cycle. An illustration of this is shown in Figure 8.2.

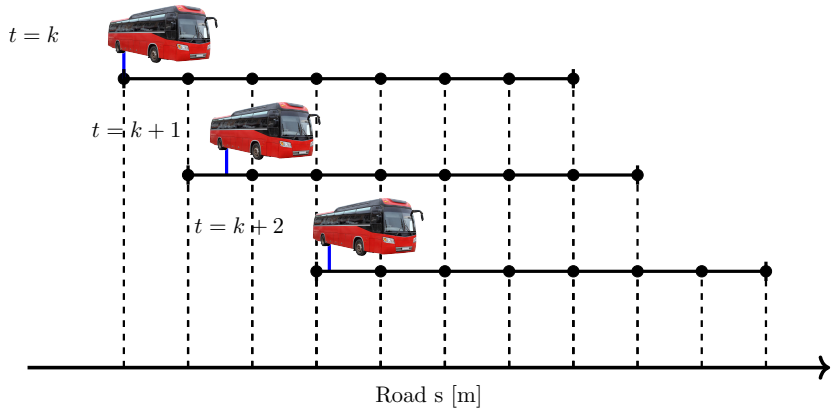


Figure 8.2: The planning horizon during successive planning cycles, as the bus progresses along the road. Depending on the bus velocity the planning horizon might shift a different number of samples. Furthermore, the bus is not always aligned with a planning sample.

As can be by the vehicle position at $t = k+1$ and $t = k+2$ in Figure 8.2, when using this method, is often the case that the current vehicle state does not lie exactly on one of the sampled states in the planning horizon. This then introduces an additional challenge on how to set up the initial vehicle state and input constraints Equation (8.1c), that will be addressed in the coming subsection.

Figure 8.2 illustrates as well that consecutive planning horizons might have shorter or longer overlapping sections depending on the vehicle velocity. From planning cycle $t = k$ to $t = k + 1$, the planning horizon only shifts one sample, whereas from planning cycle $t = k + 1$ to $t = k + 2$, the planning horizon shifts two samples. The required shifting is computed online depending on the current vehicle position at the start of the planning cycle.

We note that when considering a long planning horizon, there is a large overlap between the planning horizons of consecutive planning cycles. Furthermore, with our current planning cycle frequency, sampling distance Δs, and vehicle working speeds, it is often the case that the vehicle moves only one or two samples per planning cycle. Therefore, only one or two new samples in the planning horizon are novel/unseen at each SQP iteration. This allows the receding horizon SQP solution to quickly converge, even as the vehicle moves along the road.

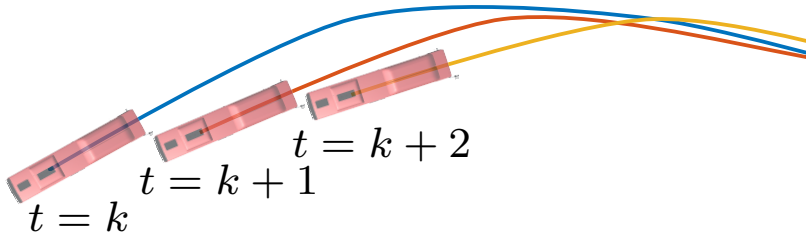


Figure 8.3: Illustration of a naive implementation of path planning initial state constraints. The planned solution paths are always starting at the current position and orientation of the autonomous bus.

Ensuring planned paths consistency

An important aspect to consider when implementing the motion planning framework in the autonomous system is the requirements of the motion control module. The motion control module determines the actuation request to be sent to the vehicle actuators so as to make the vehicle follow the planned path. The paths planned by the planning module serve therefore as a reference for the controller to track. For best performance it is important that the controller receives a consistent path reference, that does not change at every planning cycle.

As an example consider the motion planning implementation where the initial state constraints are always set to be equal to the current vehicle state. This results in the vehicle planning a new path that starts from its current position and orientation at every planning cycle, as illustrated in Figure 8.3. Although intuitive from a planning perspective, implementing the initial state constraints in this way actually makes the controller task very challenging. At every planning cycle, the reference path provided to the controller is updated, and the tracking error that the controller observes becomes zero. This happens because the new reference path always starts from the current vehicle position, leading the controller to have an initial error of zero at each new reference path.

It is important that the error does not get reset to zero, otherwise the controller will not be able to properly determine the actual actuation signals required for the vehicle to properly follow the planned paths. In extreme cases, it might happen that considering this naive implementation of the planned path leads the vehicle to slowly deviate more and more from the desired road center.

To avoid the problem of generating paths that reset the vehicle controller error to zero, the initial state constraints are adapted so that the

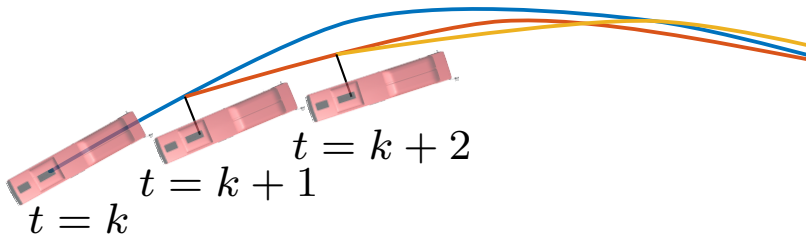


Figure 8.4: To improve controller performance, the planned path is constrained to start from the previous planned path. The initial state constraint is determined by projecting the current vehicle state into the previously planned path.

new planned path starts at a point along the previous planned path. This type of initial state constraint update is illustrated in Figure 8.4. To determine the initial state constraint, one projects the current vehicle position into the previous planned path. In order to comply with the path sampling discretization scheme shown in Figure 8.2, we choose the path sample that is closest, as well as behind, the projected vehicle position. Using this strategy, we now avoid the problem of resetting the control error to zero at each planning cycle.

There is one more aspect to take into account when considering that the planned paths are provided as references to the controller. It can be seen in Figure 8.4 that every new planned path differs slightly from the previously planned path. This happens even in the regions of the horizon that are overlapping with the previous planning horizon. Even though the planned paths are constrained to start along the previously planned path, the new information associated with the new section of the planning horizon, *i.e.*, the non-overlapping section, is enough to change the optimal path to not coincide with the previously computed optimal path.

Many controller approaches make use of path information ahead of the vehicle, such as look-ahead controllers [126] and Model Predictive Control (MPC) [127]. In the considered prototype vehicle, the control module is implemented via an MPC [127] which has an optimization horizon of H_{control} meters. When computing the actuation signals, the MPC simulates the vehicle model H_{control} meters into the future, and tries to make the simulated vehicle follow the path as well as possible. Therefore, the MPC takes into account the reference path from the current vehicle position until H_{control} meters ahead, where H_{control} is usually much shorter than the planning horizon.

In an argument similar to that of resetting the error to zero, it is also im-

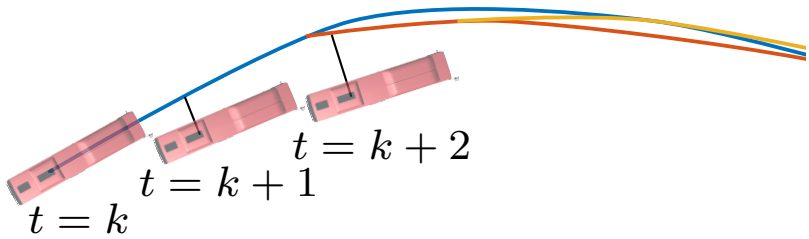


Figure 8.5: The initial state constraints are adapted so that the planned path starts from the previous path, as already shown in Figure 8.4, but now starting H_{control} meters ahead of the current vehicle state projection.

portant for the MPC performance that the reference path is kept consistent along planning cycles. In order to achieve this, we further adapt the initial vehicle constraints in the optimization problem, so that the planner path starts H_{control} meters ahead of the projected vehicle path. This mechanism is illustrated in Figure 8.5. This allows the MPC to receive a reference path that is consistent with the ones previously received, which leads to improved path tracking performance.

It should be noted that setting the initial state constraints to start H_{control} ahead of the vehicle results in the planner not being able to adapt to new obstacles appearing up to H_{control} meters ahead of the vehicle. This can be dealt with by an using an extra safety planner that either stops the vehicle, or quickly reacts to such obstacles. However, this is out of the scope of the considered work.

8.2 Results

We hereby present results obtained using the prototype vehicle shown in Figure 5.6. The vehicle is a Scania bus used for research and development of autonomous transport solutions. The bus is equipped with multiple sensors, computing units, and a full autonomous driving software stack, which our planning framework is part of. In the following, we present experimental results of the autonomous bus when driving through selected scenarios on Scania's test track in Södertälje, Sweden.

Sharp turn

We start by presenting results on a sharp turn as shown in Figure 8.6. This is a turn located on the bus depot of Scania's test track and serves a relevant example of a tight and sharp turning road that requires a driver, or in this

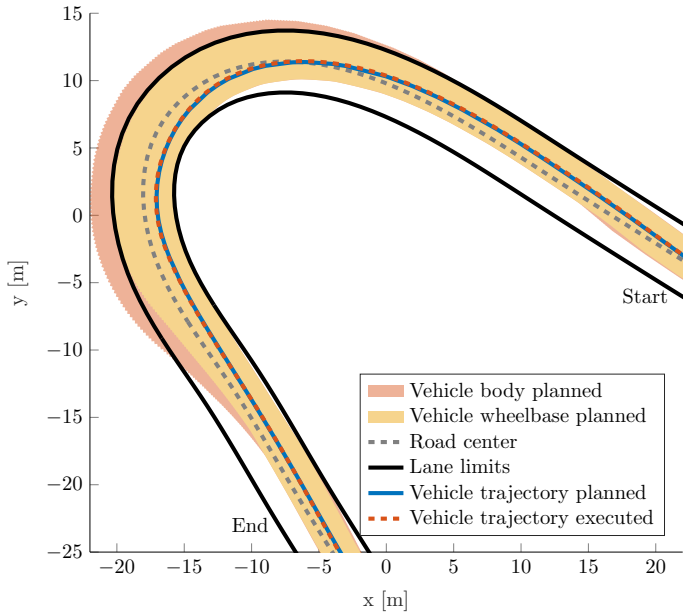


Figure 8.6: Experimental result for the bus driving autonomously in a sharp left turn (counter-clockwise direction).

case the autonomous system, to use the full steering actuation capabilities of the vehicle.

Figure 8.6 shows that the planned path starts by driving the bus towards the right side of the road, even before the turn, so as to better place the bus for the upcoming maneuver. Thereafter, while turning, the bus drives with the left wheels very close to the left lane boundaries, so as to minimize the amount of overhang that exits the lane limits on the opposite side. Once the bus has driven past the turn, it slowly returns to the center of the road, so as to minimize curvature changes and maximize passenger comfort. The vehicle approaches the turn at a speed of 24 km/h, slowing down to a speed of 7 km/h during the turn, and afterwards accelerating back to a speed of 24 km/h.

Figure 8.7 presents the planned curvature magnitude and curvature rate profiles of the maneuver in Figure 8.6. We start by noticing the smoothness of the curvature profile in Figure 8.7 (Top), which results in a comfortable ride for the passengers. It can also be noted that this is a challenging maneuver for the bus, as the motion planner needs to make use of the whole steering actuation capabilities. During a 15 meter-long section starting at around $s = 190$ m, the planned curvature corresponds to the maximum

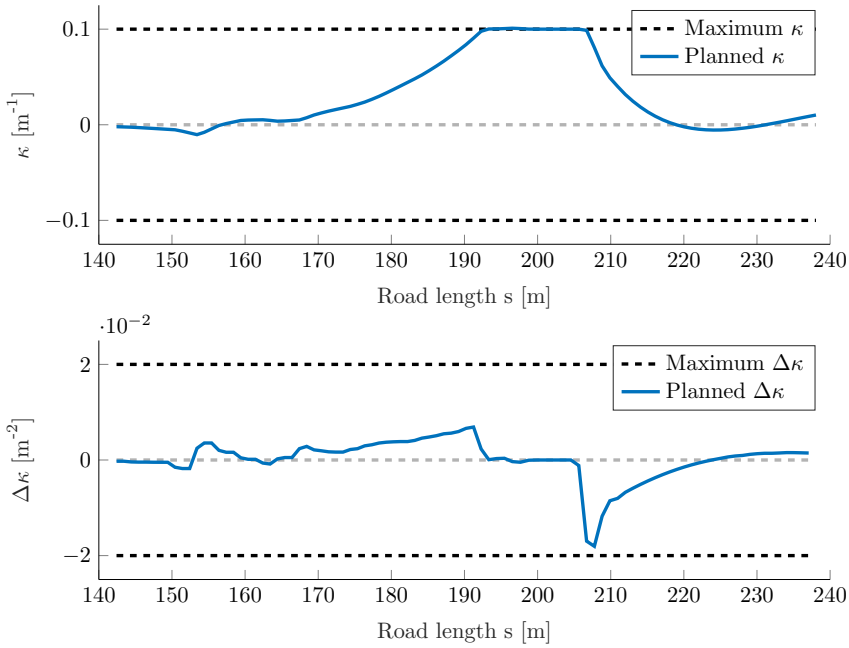


Figure 8.7: Curvature magnitude and rate profiles for the maneuver in Figure 8.6. Top: Curvature magnitude profile. Bottom: Curvature rate profile.

allowed curvature. This shows that the planner is able to make use of the full capabilities of the vehicle steering, and robustly respect the constraints related to the steering actuator limits. Figure 8.7 (Bottom) presents the curvature rate profile, which is characterized by a relatively smooth behavior, and that is contained within the rate limits associated with the steering actuator limits.

Roundabout

Figure 8.8 presents experimental results for the autonomous bus driving on a roundabout. While driving in the roundabout, the bus keeps its rear axle to the left of the road center (and towards the inside of the turn). This is done in order to center the area swept by the vehicle body, and keep a clearance towards the left lane that is approximately equal to the clearance towards the right lane. This is achieved by optimally choosing the tuning parameter $K_{bus,i}$ according to the geometric methods introduced in Section 7.2.

Figure 8.9 presents in more detail the area swept by the vehicle body

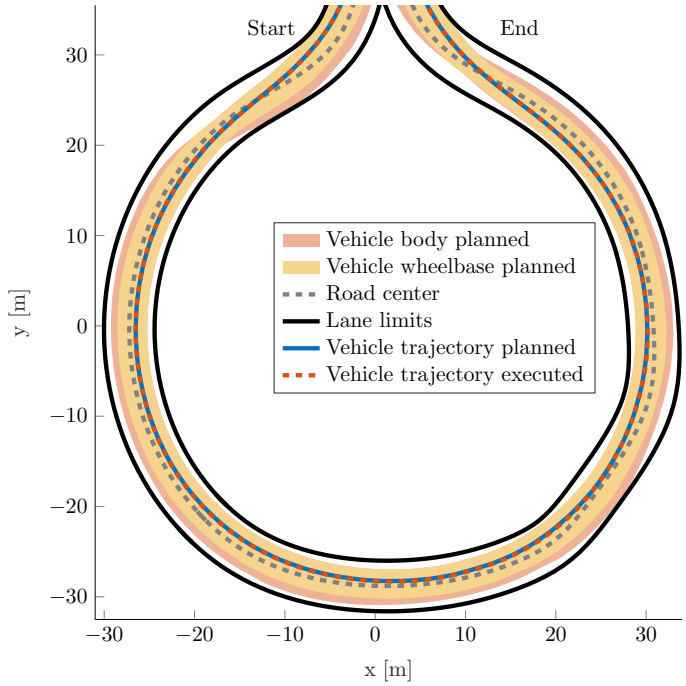


Figure 8.8: Experimental result for the bus driving autonomously in a roundabout (counter-clockwise direction).

while driving along the roundabout. The swept area is presented in the road aligned frame, which more clearly shows the balanced centering of the whole vehicle body. In the beginning and end of the maneuver, at around $s = 10$ m and $s = 190$ m the vehicle is entering and exiting the roundabout, respectively. During the entering and exiting, the bus has a transient behavior where the vehicle body needs to approach the lane limits. However, during most of the roundabout maneuver, the swept area is evenly balanced and fairly contained within the lateral offset values $e_y \in [-2, 2]$, which results in an even clearance to both the left and right lane limits. An exception to this happens around $s = 130$ m, where the road has an irregular and non-smooth shape.

The curvature magnitude and rate profiles have a smooth behavior as can be seen in Figure 8.10. This smooth behavior is achieved even in the presence of a road reference that has an irregular and non-smooth curvature profile, as shown in Figure 8.11. The non-smoothness of the road reference can be caused by low-resolution of the pre-stored maps, uncertainties and noisy measurements occurring in the perception module, or even poor road

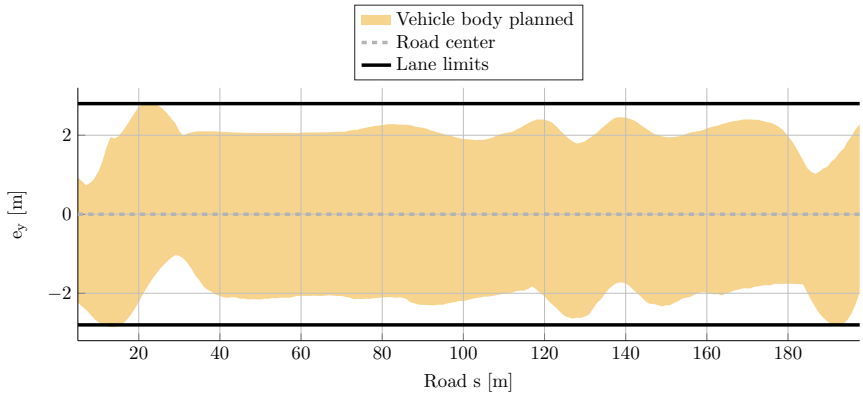


Figure 8.9: The vehicle swept area envelope in the road-aligned frame for the maneuver in Figure 8.8. The vehicle envelope is kept fairly centered during most of the roundabout maneuver, having an even clearance to both the left and right lane limits.

quality, maintenance, or lane markings. We note that even when following a road reference that has a non-smooth behavior, the motion planner is largely unaffected by this, being able to plan a path that is smooth and comfortable for passengers.

Figure 8.12 presents a compilation of frames taken at three different time instants from a video recorded during the roundabout experiments.

Bus stop

We now consider a maneuver that occurs often during daily bus operations, the approach and departure from a bus stop. During the maneuver the vehicle needs to depart from the road center and drive into the designated bus stop area to pick up and drop off passengers. Once passengers have entered and exited the bus stop, the vehicle needs to drive back onto the road center. Even though the bus needs to stop close to the curb in order to facilitate boarding and alighting of passengers, it is important to ensure that the vehicle body does not drive over it.

Figure 8.13 presents the results for a bus stop maneuver performed with the autonomous prototype bus. The bus stop curb is inflated by 30 cm in order to deal with possible control module path following errors and limited accuracy of the perception module. During initial experiments, we have measured a maximum deviation between the planned and executed path of 20 cm, although the deviation often lies significantly below that value. The inflation value of 30 cm was therefore chosen, in a conservative way, to

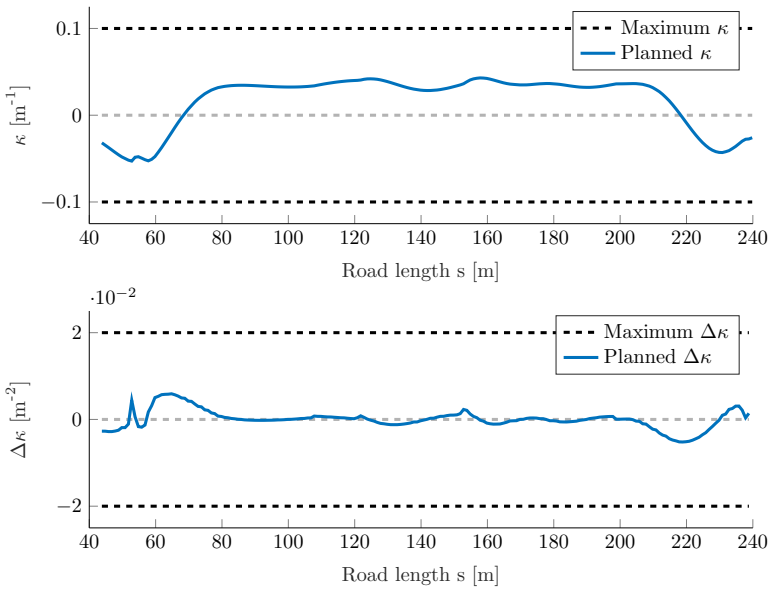


Figure 8.10: Curvature magnitude and rate profiles for the maneuver in Figure 8.8. Top: Curvature magnitude profile. Bottom: Curvature rate profile.

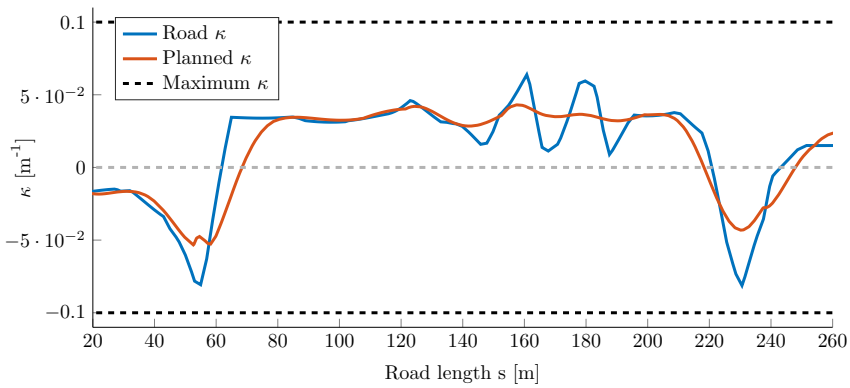


Figure 8.11: Planned path curvature and road curvature profiles for the maneuver in Figure 8.8. Even though the road has a non-smooth and irregular curvature profile the motion planner is able to find a path with a smooth and comfortable curvature profile.

Figure 8.12: The autonomous bus performing the roundabout maneuver in Figure 8.8. This prototype autonomous bus is used in the experimental results presented in this chapter (courtesy of Scania CV AB).

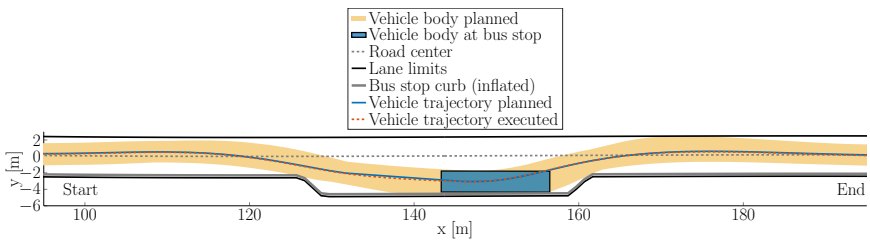


Figure 8.13: Experimental result for the autonomous bus approaching the bus stop, coming to a halt by the curb, and departing back onto the road.

reflect these measured deviations. As seen in Figure 8.13, the bus comes to a halt at the bus stop, being very close to the inflated bus stop curb and well aligned with it. After a short stop, the vehicle starts driving again, and converges back to the road center.

Figure 8.14 presents the curvature magnitude and rate profiles for the bus stop maneuver, including the approach and departure. It can be seen that the curvature magnitude profile is fairly smooth, as expected already from previous experimental results. There is however a small peak in the curvature rate profile, happening at around $s = 100$ m, corresponding to the bus starting the departure maneuver. When starting the departure, the bus quickly turns the wheels, resulting in a temporary high curvature

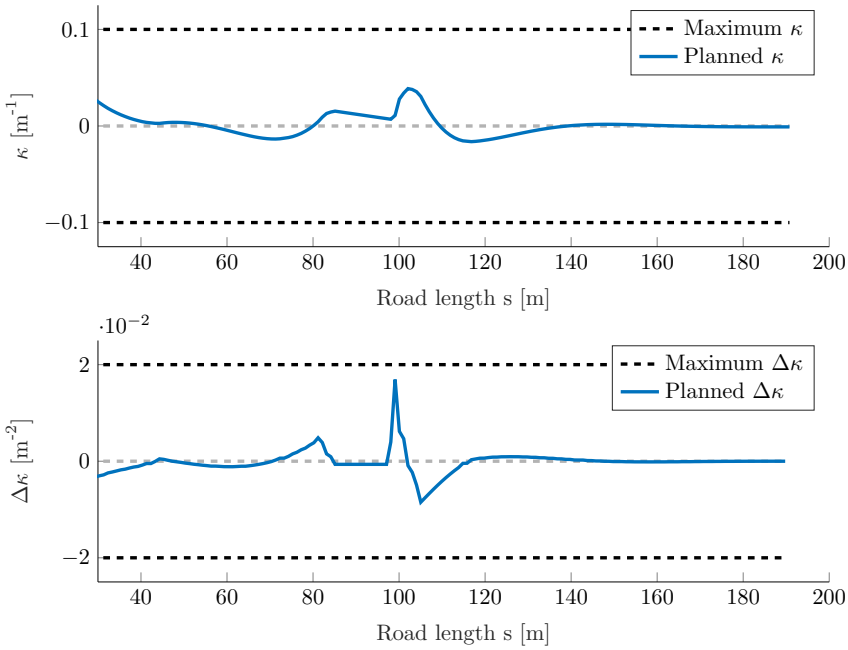


Figure 8.14: Curvature magnitude and rate profiles for the maneuver in Figure 8.13. Top: Curvature magnitude profile. Bottom: Curvature rate profile.

rate. This is done to quickly converge back onto the road. The remaining of the maneuver is characterized by a smooth steering behavior and a low curvature rate profile.

We present as well in Figures 8.15 and 8.16 the disparity between the planned and executed paths during the bus stop approach maneuver. The accuracy with which the vehicle executes the planned path is particularly important during the bus stop approach, as this maneuver defines how close the bus is to the bus stop curb, and therefore, if passengers can easily board and alight the bus. As can be seen in Figure 8.15 the executed path follows closely the planned path, resulting in a final lateral displacement slightly below 10 cm, which we deem to be acceptable for our current experimental settings. We note as well that the bus stop curb inflation of 30 cm allows for appropriate extra safety margins during the bus stop approach, as the lateral displacement is significantly below the inflation value. As for the disparity between vehicle planned and executed orientation, shown in Figure 8.16, it can be noted that it is also quite small. At the end of the maneuver, *i.e.*, when at the bus stop, the bus has an actual orientation that is only 0.25

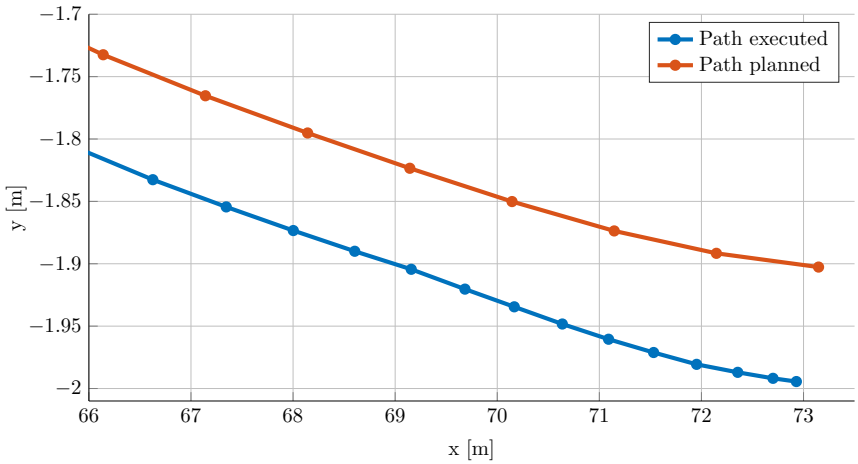


Figure 8.15: Detail of the planned and executed bus path for the bus stop approach maneuver of the experimental results in Figure 8.13.

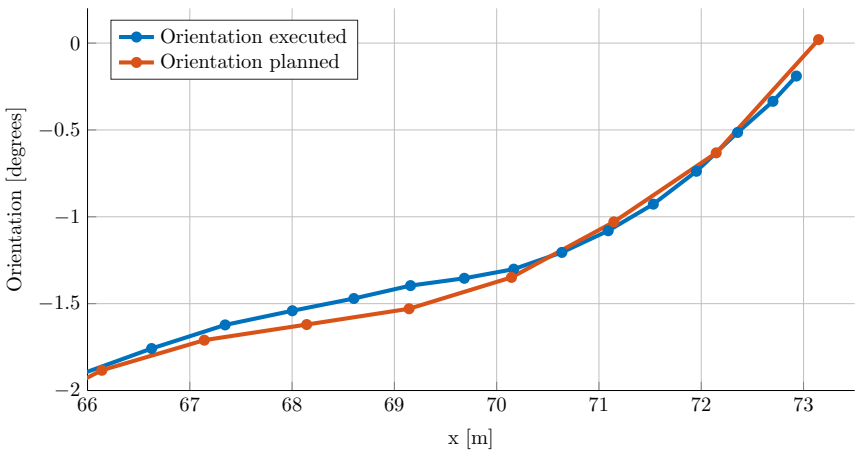


Figure 8.16: Detail of the planned and executed bus orientation for the bus stop approach maneuver of the experimental results in Figure 8.13.

degrees offsetted from the planned final orientation.

During the approach to the bus stop there is a critical moment in which the motion planner formulation is modified by adding the final constraint

$$z_{bus,N} = z_{\text{bus stop}} \tag{8.4}$$

to optimization problem Equation (8.1). Figure 8.17 shows the planned

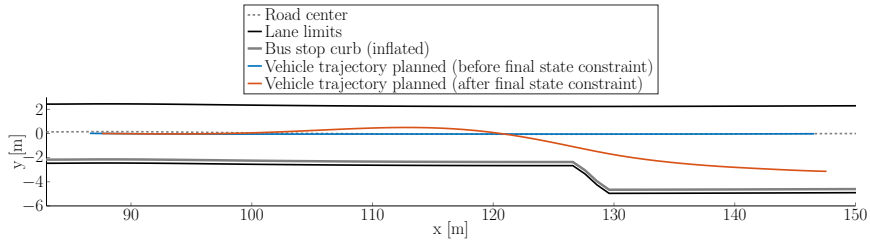


Figure 8.17: Planned paths for the planning cycles before and after the bus stop constraint is formulated. Even though the planned paths are significantly different, the vehicle behavior is smooth and comfortable due to the first H_{control} meters of the latest planned path being consistent with the previous planned path.

paths for the planning cycles immediately before and after final constraint Equation (8.1) is added. The planned paths are strikingly different, as one of them tries to stay in the road center, and the other tries to converge into the newly observed bus stop location. Despite being significantly different, a smooth behavior of the vehicle is guaranteed thanks to considering an initial state constraint that is H_{control} meters ahead. This results in the motion controller receiving consistent planned paths, and in turn the bus having a comfortable behavior for the passengers, even when transitioning between two different motion planning formulations.

Obstacle avoidance maneuver

We consider as well the scenario where an obstacle is located on the side of the road. The capability to deal with unforeseen changes to the environment, such as obstacles, is essential for autonomous vehicles deployed in urban environments. Figure 8.18 presents an experimental scenario where a simulated obstacle is located on the side of the road. This obstacle could correspond to illegally parked vehicles, construction works, or other types of obstructions affecting the drivable area. The obstacle is inflated by 30 cm, similarly to the bus stop curb inflation, in order to reflect the disparity between planned and executed vehicle paths.

The result presented in Figure 8.18 shows that the bus is able to avoid the inflated obstacle, and thus safely avoid the actual obstacle with good clearance. Furthermore, Figure 8.19 shows that the executed maneuver is barely noticeable to passengers, as both the curvature magnitude and rate profiles have a smooth behavior and values close to zero.

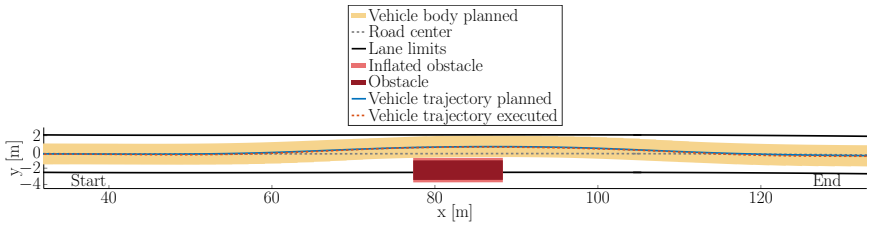


Figure 8.18: Experimental result for the bus autonomously avoiding an unexpected obstacle on the road.

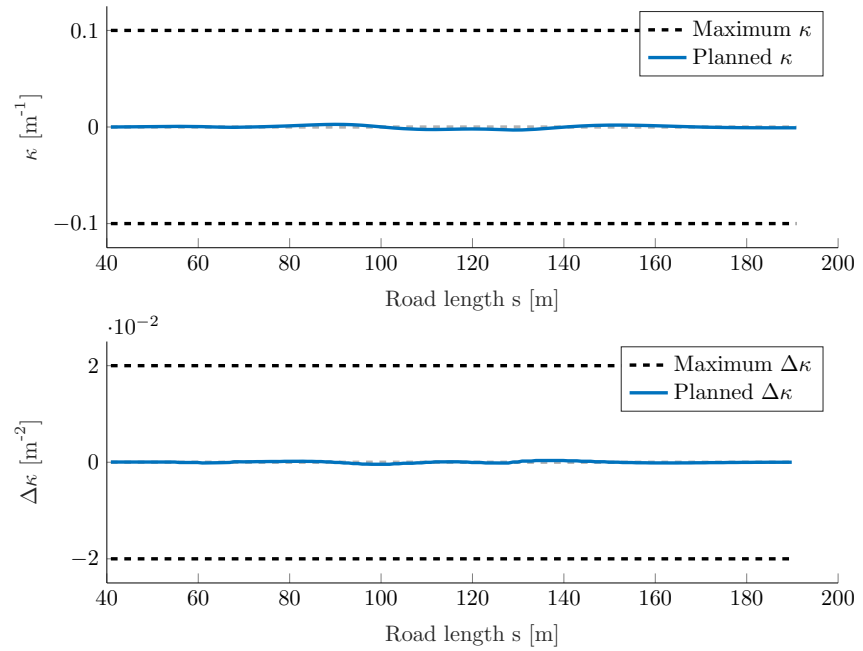


Figure 8.19: Curvature magnitude and rate profiles for the maneuver in Figure 8.18. Top: Curvature magnitude profile. Bottom: Curvature rate profile.

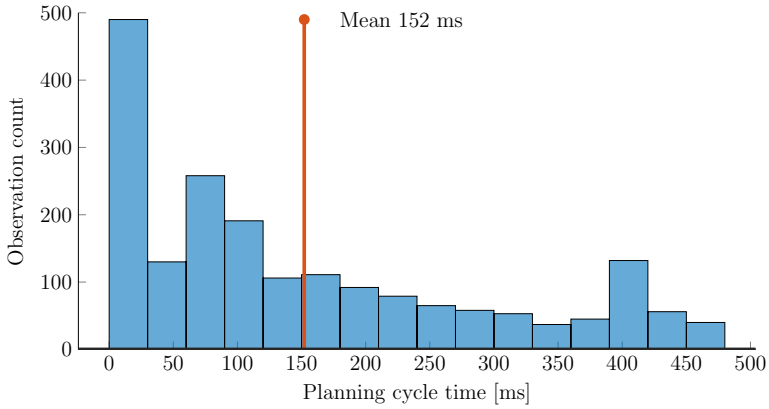


Figure 8.20: Histogram of measured planning cycle computational times for the experiments considered in this chapter.

Computational times

Figure 8.20 presents an histogram of the measured computational times of planning cycles during the experiments presented in this chapter. The computational time of each planning cycle can be divided into two components, the setup time and the solver time. The setup time consists of the computational time required to setup the optimization problem. Setup of the optimization problem involves, linearizing and discretizing the vehicle model, as well as creating the vehicle body constraints, which involves a large number of Cartesian frame to Road-aligned frame conversions. The average setup time in each planning cycle is measured to be 15.49 ms. The solver time corresponds to the time used by the convex solver, in this case OSQP [124], and its average time in each planning cycle is measured to be 136.59 ms. We limit the solver computation time to a maximum of 400 ms, which is enforced by a stopping criterion readily available in the OSQP solver interface [124]. The average time of a full planning cycle is then 152.09 ms.

The vehicle body constraints are responsible for most of the optimization problem complexity, and in turn for most of the computational time in the planning cycle. This can be partially visualized in Figure 8.21 where it can be clearly seen that the computational times are significantly higher just before the bus approaches the turn. When approaching the turn, most of the planning horizon of the bus is within the turn, where the body constraints are active and significantly increase the problem complexity. Here we use the term *active* to indicate that the current optimal solution x^* of the convex problem with inequality constraints $g_i(x) \geq 0$ results in $g_i(x^*) = 0$. The

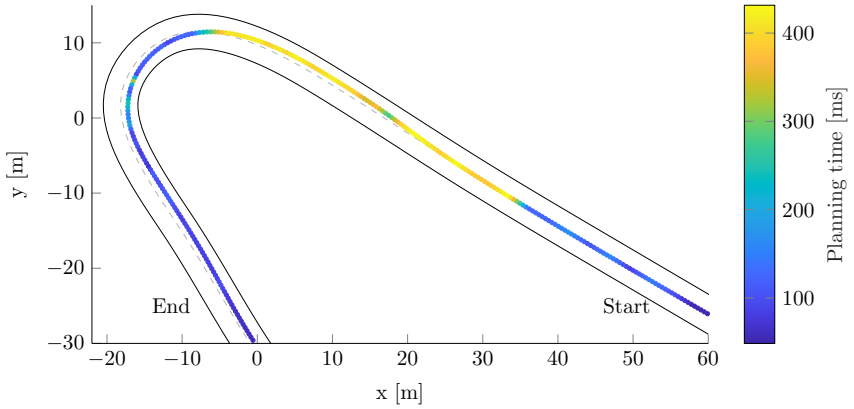


Figure 8.21: Visualization of the computational times during the maneuver in Figure 8.6.

body constraints are always formulated and considered in the optimization problem, however, only in more complex scenarios do they become active. When driving in the straight sections of the road the computational times are much lower, due to the simpler driving case, where body constraints are then not active.

In the presented experimental results the motion planner framework ran at a frequency of 2 Hz, and with a planning horizon length of 40 m. We consider urban driving applications, where vehicle speeds are expected to be under 50 km/h, resulting in a minimum planning horizon time of 2.9 s. This setup allows the vehicle to safely react to new obstacles and events appearing at the end of the planning horizon. A higher frequency of the motion planner would be desirable to quickly react to new information appearing close to the vehicle, however, to achieve that, one would need to either reduce the number of vehicle body constraints, the planning horizon, or both. In these experiments, we consider that a long planning horizon and a high number of vehicle body constraints are more important, as the proposed motion planner is designed to deal with the particular problems associated with buses. An additional safety planner with a high planning frequency could however be implemented, that would act as an extra layer of safety, quickly reacting to unexpected obstacles that might jump into the paths planned by our proposed motion planner. Nevertheless, the computational times measured during these experiments, show the suitability of the proposed motion planner to be implemented and used in autonomous buses driving in urban environments.

8.3 Conclusions

This chapter presented the implementation details and practical experiments of a motion planning framework deployed on an autonomous bus. We presented a series of practical considerations and implementation techniques that allow us to bring the motion planner developed in previous chapters from a simulation environment into the real world. Afterward, we showcase the motion planner performance by running the autonomous bus in several scenarios resembling urban driving.

To solve the complex and computationally-heavy optimization problem, we use an SQP approach. To reduce computational times while maintaining a high solution quality, we use the SQP in a receding horizon fashion, solving only one QP at each planning cycle. As the motion planner is tightly coupled with other modules in the system, some further adaptations are in order. In order to improve the performance of the underlying motion controller module, we propose a path sampling technique that ensures consistency between solution paths at consecutive planning cycles.

An extensive set of experiments is then carried on, studying different aspects of the planning framework. We show that the proposed method can deal with sharp turns and tight maneuvering areas while keeping the vehicle wheelbase inside the road and minimizing the overhangs exiting the driving lane. Further experiments show that despite the large dimensions of the considered vehicle, the planning framework properly centers the whole vehicle body, keeping a balanced clearance to both the left and right sides of the road limits. Furthermore, the approach is robust against noisy road references and poor environment perception. The planning approach is also tested on both a bus stop and an obstacle avoidance maneuver, successfully tackling both. In all considered experiments, the proposed approach guarantees a comfortable ride for the passengers, as evidenced by the smooth curvature profiles planned. Finally, we present a summary of the measured computational times, which confirm that the method is suitable for real-life implementations.

It was noticed during some of the bus stop experiments that, on a few occasions, the vehicle's front wheels would climb over the curb if no curb inflation was considered, see Figure 8.22. During bus stops, the vehicle must stop within centimeters of the curb. In such situations, the wheels might come in contact with the curb, possibly climbing onto the sidewalk or damaging the steering column actuator. When turning, the wheels can protrude beyond the body, which is not captured when only considering wheelbase collision avoidance constraints. Therefore, it becomes essential to explicitly consider the wheels instead of assuming them to be static and fully contained inside the chassis. Therefore, one should consider novel

Figure 8.22: Experimental scenario where the wheel climbed the curb.

collision checking methods that explicitly consider the current steering angle and the wheel dimensions [20]. This new challenge justifies the concept of wheel-aware planning, which we believe to be a promising avenue for future research.

Chapter 9

Decision Making using Branch MPC

In order to achieve full self-driving capabilities, autonomous vehicles have to deal with the presence of other traffic participants, such as human-driven vehicles and pedestrians. Sharing the road with humans is, to date, one of the biggest challenges hindering autonomous vehicle deployment. Although sensor and perception technology allows the autonomous vehicle to obtain a fairly accurate understanding of the current state of the traffic scene (positions and velocities of other traffic participants) the unpredictability of human behavior makes the prediction task, *i.e.*, understanding how the traffic scene evolves, reliable only for a few seconds into the future. Furthermore, the traffic scene evolution is directly impacted by the decisions of the planner, as human-driven vehicles react differently to different maneuvers of the autonomous vehicle. Therefore, this requires that a joint prediction and planning problem is solved, instead of the more common motion planning problem where other traffic participant predictions are assumed fixed, and the robot plans so as to avoid them.

In this chapter, we introduce an MPC framework that jointly predicts the human future states and plans the autonomous vehicle optimal actions. Three challenging aspects arising from interactions with humans are addressed, *multi-modality*, *interactive behavior*, and *decision making*. Human drivers, and driving in general, is characterized by *multi-modality*. Often times, a certain situation can develop into different outcomes that are significantly different than one another, corresponding to different *modes*. As an example, a vehicle approaching an intersection, can either keep its speed, or slow down to a stop, two very distinct outcomes. *Interactive behavior* follows from the fact that driving takes place in an environment where other vehicles have an impact on us, and the other way around. In the case of

packed traffic, a vehicle slowing down causes the vehicle directly behind it to also slow down (in more drastic cases resulting even in phantom traffic jams [128]). Finally, human drivers are inherent *decision makers*. A driver approaching an intersection, will decide to cross depending on factors external to oneself. In the simpler case of a signalized intersection, the (law abiding) human driver will decide to cross the intersection or not, depending on the current state of the traffic light. In the more complicated case of an unsignalized intersection, the human driver will assess the situation, by observing other vehicles approaching the intersection, and make a decision based on its own and other vehicles' distance to the intersection and driving speed.

The contributions of this chapter are the following:

- addressing the multi-modality of human drivers by considering multiple future outcomes associated with different decisions taken by the human driver;
- considering the interactive nature of humans by modeling them as reactive agents impacted by the actions of the autonomous vehicle;
- approximating the decision making process of human drivers by considering a model developed in neuroscience studies with human drivers as subjects.

The chapter is organized as follows.

Section 9.1 introduces some of the challenges associated with driving in the presence of other traffic participants. Toy examples illustrating different driving scenarios present the concepts of *multi-modality*, *interactive behavior*, and *decision making* of human drivers. Finally, we present related work and research currently happening on the topic.

Section 9.2 focuses on the models considered to express the human driving policies. The section present both non-interacting models as well as interacting models that adapt their behavior to the current traffic scene. Finally, a behavioral model of human decision making is presented. The model is later used to predict the probability of future outcomes.

Section 9.3 integrates the concepts of the previous sections into an MPC formulation that constitutes our motion planning framework. The concept of scenario tree is introduced as a suitable way of modeling the multi-modality of human drivers, as well as their decision making. We then present an MPC that uses a scenario tree to optimize over the expected cost of future outcomes.

Section 9.4 presents simulation results of the proposed method. The approach is first compared to other planning frameworks and its performance is quantitatively evaluated. A merging scenario is then consider, which

highlights the benefits of considering the multi-modality aspects of human-drivers. Joint prediction and planning is compared against a prediction then planning approach, showing the importance of considering interaction-aware human models. Finally, the decision making aspect of humans is studied through simulations on an intersection scenario.

Section 9.5 summarizes the chapter, identifying contributions and highlighting directions for further improvement and future work.

9.1 Introduction

We briefly introduce three particular challenges associated with driving in the presence of human-driven vehicles: *multi-modality*, *interaction*, and *decision-making*. We then present related work in the area and compare different approaches according to their capability of dealing with the previous challenges.

Challenges

Multi-modality

When driving, human drivers change constantly between different types of behavior. If a vehicle is arriving at an intersection, as illustrated in the *Initial state* of Figure 9.1, the driver (red car) can choose between multiple significantly different driving behaviors. In the case of a defensive driver, it can slow down and come to a stop, so as to safely check for oncoming cars, shown in the *Possible outcome 1* of Figure 9.1. A more aggressive driver, could instead decide to speed through the intersection, accepting a high risk of collision at the benefit of a shorter traveling time, shown in the *Possible outcome 2* of Figure 9.1.

These distinct behaviors can be seen as different modes that a perception module, as well as a motion planning module, need to consider when predicting the future state evolution of the human driver. The multi-modality of the human introduces a first challenge, as the different outcomes require significantly different driving maneuvers from the autonomous vehicle. Furthermore, since the outcome is not known beforehand, the motion planner has the task of finding a maneuver that is able to deal with both outcomes.

Interaction

Driving is a highly interactive task, where drivers must constantly adapt their actions in response to the actions of other drivers. As an example, consider a human-driven vehicle (red car) and an autonomous vehicle (yellow car) approaching the intersection shown in *Initial state* of Figure 9.2.

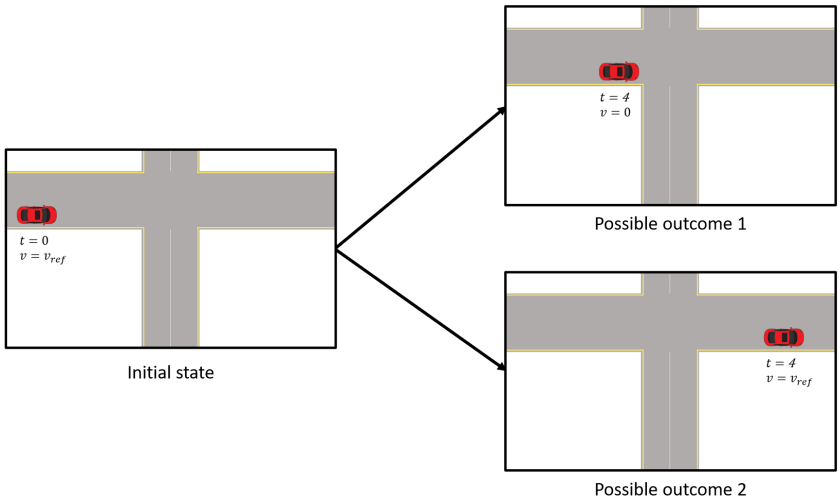


Figure 9.1: Challenge 1: Multi-modality.

Suppose the autonomous vehicle turns right at the intersection, merging into the lane of the human-driven vehicle. In that case, the human driver will slow down to avoid a collision, as shown in *Possible outcome 1* of Figure 9.2. On the other hand, if the autonomous vehicle proceeds through the intersection, the path of the human-driven vehicle will be free, and the human driver will keep its speed and drive through the intersection, as shown in *Possible outcome 2* of Figure 9.2.

Figure 9.2 illustrates a significant challenge that motion planners need to consider: human behavior is affected by the decisions taken by the autonomous vehicle. Interactions occur in numerous traffic situations, such as merges and lane changes. Considering these interactions is crucial to guarantee that autonomous vehicles do not drive too conservatively [67, 72].

Decision making

Human drivers often make decisions during the driving task. These decisions are related and give rise to the multi-modality aspect of human behavior. As an example, consider the scenario shown in *Initial state* of Figure 9.3. The human-driven vehicle (red car) has to decide if it will cross the intersection or not. This decision is affected by its perception of the intended behavior of the autonomous vehicle (yellow car). In case the autonomous vehicle approaches the intersection with a high speed, shown in *Possible outcome 1* of Figure 9.3, the human decides to stop and not cross the inter-

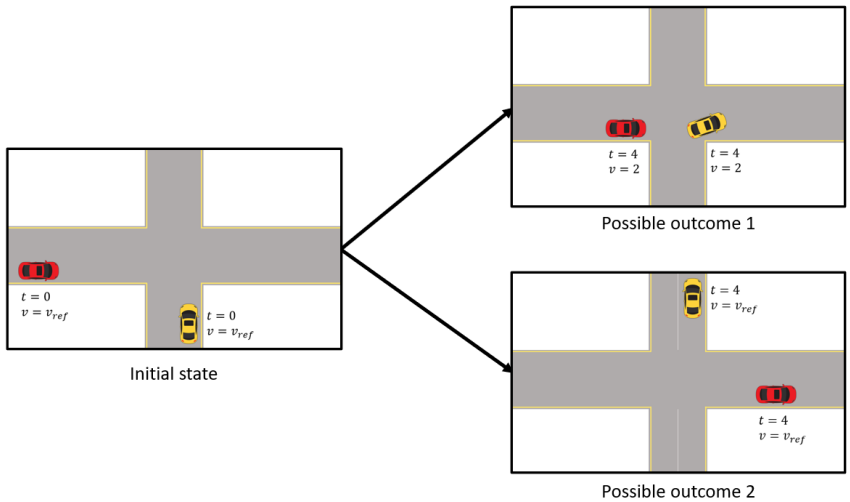


Figure 9.2: Challenge 2: Interaction.

section. However, if the autonomous vehicle would have slowed down when approaching the intersection, shown in *Possible outcome 2* of Figure 9.3, the human driver would have driven through the intersection. Figure 9.3 highlights an interesting aspect of human drivers approaching intersections, that their decision to cross or stop is dependent on the distance and velocity of the oncoming vehicle [129].

This example shows how the traffic scene, in particular the position and velocity of the autonomous vehicle, affects the decision taken by the human driver. We note that this example significantly differs from the interaction aspects shown in Figure 9.2, in this particular case there isn't an imminent collision forcing the human driver to stop. Instead, there is a clear intent shown by the autonomous vehicle to drive through or stop at the intersection that is understood by the human, resulting in the human-driven vehicle stopping at or driving through the intersection, respectively.

Joint Prediction and Planning

A vast majority of planning approaches takes as input the prediction of a traffic scene, namely the prediction of human-driven vehicles, and plans a trajectory for the autonomous vehicle that adapts to the given predictions. The typical architecture for these approaches is shown in Figure 2.12a. This architecture separates the problem of prediction and planning into different modules, which fairly simplifies the development of both modules, and of

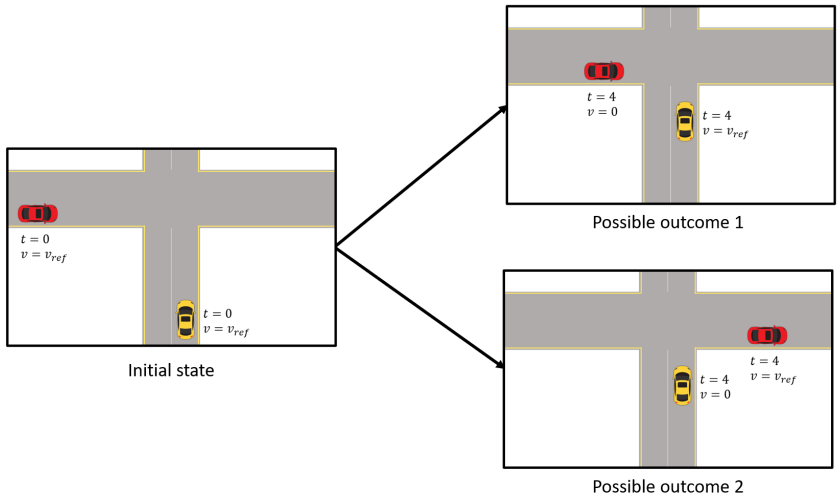


Figure 9.3: Challenge 3: Decision making.

the overall autonomous driving software stack. This type of approach is useful and practical, and works well in non-interactive situations.

However, the architecture is not suited for autonomous vehicles that drive in complex and traffic heavy scenarios. When considering driving scenarios that rely on heavy interaction with other traffic participants, the autonomous vehicle will often get stuck [130], as there is no possible driving maneuver that can comply with the predictions provided by an upstream perception module. In less serious cases, failing to model other drivers' reactions to the autonomous vehicle future maneuvers, leads to sub-optimal self-driving behavior, as the autonomous vehicle will act conservatively instead of assertively. Therefore it becomes necessary to assume that the predictions of other vehicles will adapt to the decisions of the autonomous vehicles.

An architecture that considers how the predictions of human-driven vehicles are affected by the planned maneuver of the autonomous vehicle is shown in Figure 2.12b. In this type of architecture the planning module plans a trajectory for the autonomous vehicle at the same time as it predicts the behavior of other traffic agents in response to that trajectory. A perception module is still providing predictions, therefore it is important that these predictions come in a representation that is suitable for the downstream planning [77, 78]. The architecture in Figure 2.12b, allows for joint prediction and planning, allowing the autonomous vehicle to interact with other traffic participants, reducing conservatism while at the same time

ensuring safe driving behavior.

Related Work

The work in [67] proposes an interaction aware planning framework for merge scenarios. The work is motivated by the fact that in congested traffic it can be impossible to merge without the cooperation of other vehicles. To deal with this challenge, the authors make use of a model for traffic participants that takes into account the effect of the planned autonomous vehicle trajectory, effectively modeling the cooperation of other vehicles. The method is based on simulating the traffic scene evolution for a large set of minimal jerk velocity profiles of the autonomous vehicle, and then choosing the best velocity profile. In our approach we achieve greater planning flexibility by considering a continuous optimization approach that is not limited to a finite number of profiles of minimal jerk. Furthermore we consider the possibility that other vehicles can be multi-modal, instead of following a single Intelligent Driver Model, and also how our actions affect the decision making of the human driver.

An MPC approach for tackling uncertainties stemming from the multi-modality of other road users is introduced in [131]. Combining ideas from tube-based and scenario-based MPC, the authors show that their approach achieves better expected performance than a Robust MPC while still guaranteeing safety through robust constraint satisfaction. The proposed approach reduces conservativeness by planning a feedback policy which considers different scenarios as a function of the mode of the prediction model. However, the approach is only targeted towards uncertainties stemming either from the possible existence of static obstacles, or from the multi-modality of pedestrians.

The works in [132, 133] consider the multi-modality arising from the uncertainty over the different maneuvers types of other drivers. It is assumed that other vehicles might keep their lane or change it to neighboring lanes. An additional uncertainty aspect is considered emerging from uncertain parameters in the prediction model within each possible maneuver. By sampling a small number of future scenarios according to these two uncertainties, the proposed framework then formulates chance constraints that are used to limit the risk of dangerous events. Noticing that Scenario MPC might require an unreasonably large number of samples to accurately predict other vehicle's motions, the authors of [134] combine Scenario MPC with Stochastic MPC. The uncertainties arising from the multiple possible maneuvers of other drivers and the uncertain execution of these maneuvers are handled individually, which allows for less conservative motion planning.

The work in [135] proposes Branch Model Predictive Control as a strat-

egy for tackling the multi-modality arising from other human driver's decision making. To model the human, the authors use a finite set of policies to build a scenario tree, then a feedback policy in the form of a trajectory tree is planned, accounting for every possible scenario. The proposed planner optimizes over a risk measure of the cost function, instead of optimizing for the expected value of the cost function. This is achieved through Conditional Value at Risk (CVaR), a risk measure that encourages risk-averse solutions. Although the CVaR formulation puts more focus on worse outcomes, it can be tuned to tradeoff between performance and robustness of the planner. In [135], the uncontrolled agent policies are propagated independently of other agents, and therefore do not adapt to the autonomous vehicle decisions, this can lead to the freezing robot problem [67, 130]. Furthermore, the decision making models considered for the human are based on Control Barrier Functions and lack a sociological motivation. One could argue as well that the considered tree structure does not accurately reflect the behavior of humans, which do take a decision at a fixed point in time/space, instead of continuously delaying this decision, as is modeled in [135].

Current autonomous vehicles deployed in the real world tend to predict the future trajectories of human-driven vehicles and plan to stay out of their way, resulting in defensive and conservative behaviors. To tackle this excessive conservatism, [72] take into account how the actions of an autonomous car affect other human drivers. They start by modeling human driving maneuvers as the result of an optimization with respect to a given reward function, that depends on the autonomous vehicle planned maneuver, corresponding to a simplification of a partially observable stochastic game. In simulation and user studies the authors successfully show the benefits of their approach. The autonomous vehicle is able to perform complex behavior in interaction with humans, such as accelerating or slowing down at an intersection to show intent to cross or to give way, respectively. The autonomous vehicle is also encouraged to gather information about the human, resulting it nudging the human vehicle, in order to better estimate the parameters behind the human's reward function. The approach is shown to deal with several complex traffic scenarios and result in intricate driving maneuvers, simply as a result of optimizing for driving efficiency, instead of having to rely on hand-coded strategies.

One impressive aspect of the framework proposed in [72] is that the planned trajectories exhibit communicative behavior, *i.e.*, they are legible. A legible maneuver supports other traffic participants in correctly inferring the autonomous vehicle's intentions. The work in [136] proposes a legible MPC to improve the readability of the planned maneuvers. Results show that considering legibility as a goal in the optimization objective increases the readability of the planned maneuvers, improving safety and

performance.

The work in [137] proposes Contingency MPC (CMPC), a framework that tracks a desired path while simultaneously maintaining a contingency plan that can deal with possible emergencies. The MPC considers both a nominal plan and a contingency plan into different horizons, where the first control input is shared between both horizons. In this way the controller never chooses between the nominal and contingency trajectories. It tracks the desired nominal path to its best effort, and in case the contingency event occurs, it returns a safe trajectory that safely deals with it. Experiments in a real vehicle show that the Contingency MPC intuitively deviates from the desired nominal path in order to approach turns more conservatively, and deal with the emergency scenario, in their case, a road covered with ice. By being selective against which disturbances/emergencies to consider, the CMPC achieves responsible but practical conservatism, avoiding the excessive conservatism of Robust MPC approaches. The authors further extend their approach by considering the probability of the contingency event occurring P_c [138], where c stands for contingency. The MPC objective cost then becomes the expect cost from the two possible outcomes, that the contingency occurs, P_c, or that it does not, $1 - P_c$. They show how P_c can be used a tuning parameter that determines the CMPC's focus on nominal performance objectives. As P_c increases to 1, the more conservative the approach, eventually converging to the Robust MPC avoidance at $P_c = 1$.

The challenging task of driving an autonomous vehicle through an intersection is studied in [139]. The authors present a planning framework based on Stochastic MPC that considers multi-modal predictions of surrounding vehicles. The multi-modal predictions of other agents take the form of Gaussian Mixture Models, allowing the proposed SMPC framework to consider probabilistic collision avoidance constraints. The authors introduce a novel parameterized policy class which the SMPC optimizes over. This policy class depends on the predictions of the autonomous and human-driven vehicle states. Optimizing over this policy class, instead of optimizing over an open-loop input sequence allows the autonomous vehicle to safely plan maneuvers while reducing conservative behavior. The authors pose the SMPC optimization problem as a Second-Order Cone Program, which can be solved efficiently, allowing for real-time implementation. Extensive results highlight the benefits of optimizing over parameterized policies in the SMPC formulation and the suitability of the method to deal with multi-modal predictions.

Table 9.1 compares the different approaches mentioned previously regarding the challenges introduced in Section 9.1. It can be seen that no single approach considers all challenges simultaneously. This motivates the approach proposed in this chapter. We build upon the work [135] where a

Branch Model Predictive Control approach is proposed to tackle the multi-modal behavior of humans. We improve upon [135] by considering the reactive behavior of humans, *i.e.*, the fact that humans will adapt their behavior in response to the actions taken by the autonomous vehicle. Furthermore, we consider that humans do take a decision at a particular point on the road, instead of assuming that the human is constantly making or delaying decisions to a moving point ahead of it. Finally, we study a novel way of formulating the human decision making process, by considering a cognitive neuroscience model developed in human studies [129].

Table 9.1: Comparison of different planning approaches in the literature according to the challenges identified in Section 9.1.

Approach	Multi-modal	Interactive	Decision making
Ward et al. [67]	×	✓	×
Batkovic et al. [131]	✓	×	×
Cesari et al. [133]	✓	×	×
Chen et al. [135]	✓	×	✓
Sadigh et al. [72]	×	✓	✓
Alsterda and Gerdes [138]	✓	×	×
Nair et al. [139]	✓	×	×
Our approach	✓	✓	✓

9.2 Modeling

Considered scenarios

In this chapter we consider two scenarios that force the autonomous vehicle to interact with another human-driven vehicle. In the first scenario, the autonomous vehicle is merging onto a road, as shown in Figure 9.4. There are no priority rules in this scenario, and as such no vehicle has the right of way over the other. Therefore the two vehicles approaching the merging point need to negotiate the maneuver between themselves, deciding on which to go first.

The second case considers a non-signalized intersection, *i.e.*, an intersection without traffic lights Figure 9.5. Similarly to the previous scenario, here there are no obvious priority rules. Therefore, to safely traverse the intersection, the two vehicles must interact and decide between themselves which vehicle drives through the intersection first.

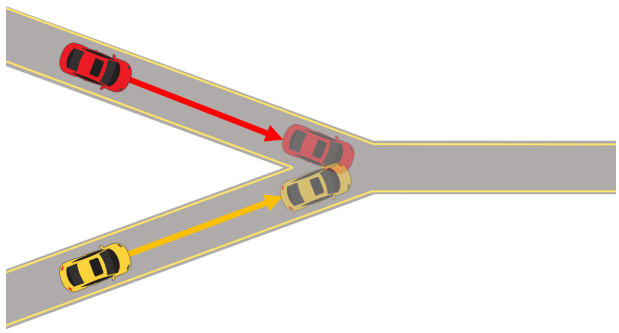


Figure 9.4: The considered merging scenario. The autonomous vehicle (yellow car) needs to merge into a lane where a human-driven vehicle (red car) is also merge into.

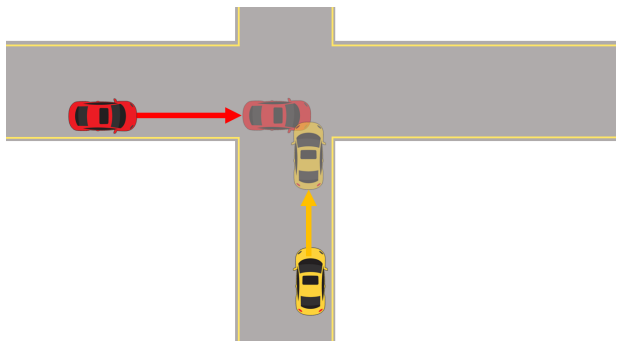


Figure 9.5: The considered intersection scenario. The autonomous vehicle (yellow car) and the human-driven vehicle (red car) need to ensure that they do not cross the intersection simultaneously, which would result in a collision.

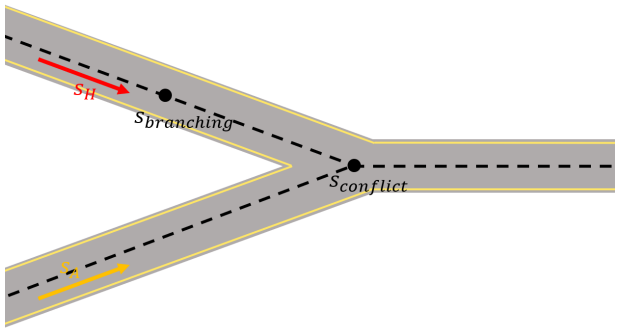


Figure 9.6: Coordinates of the merging scenario shown in Figure 9.4.

Modeling the scenarios

We assume that a vehicle i, $i \in \{H, A\}$, human-driven (H) or autonomous (A), moves along the road centerline along a path length s^i. In the merging scenario, Figure 9.4, the vehicles are on separate lanes until the merging point s_{conflict}. After s_{conflict} the vehicles are effectively on the same lane, and therefore must keep a distance between them in order to avoid a collision. It is assumed that as the human driver approaches the merging point, it will eventually make a decision at $s_{\text{branching}}$, possibly changing its behavior. $s_{\text{branching}}$ is a point located before s_{conflict}, and it is chosen so as to represent a point on the road where the human driver becomes aware of the autonomous vehicle. When the human driver becomes aware of the road it will take a decision, to either continue driving normally, or to adapt its behavior, possibly slowing down or speeding up, depending if the driver is altruistic or egoistic [76]. An illustration of the merging scenario together with the locations of s_{conflict} and $s_{\text{branching}}$ is shown in Figure 9.6.

The coordinates s_{conflict} and $s_{\text{branching}}$ are shown for the intersection scenario in Figure 9.7. Similarly to the merging scenario, the human-driven and autonomous vehicle can collide after a point s_{conflict}. At a point $s_{\text{branching}} < s_{\text{conflict}}$, the human-driven vehicle starts interacting with the autonomous vehicle, changing its behavior according to its preferences, driving style, and even the intention shown by the autonomous vehicle [129].

Vehicle models

In the scenarios considered, we assume the vehicle to drive along the lane center and only plan for its longitudinal motion. This assumption is motivated by the fact that there is little room for lateral movement in these scenarios. Therefore, there is little to no advantage in considering the pos-

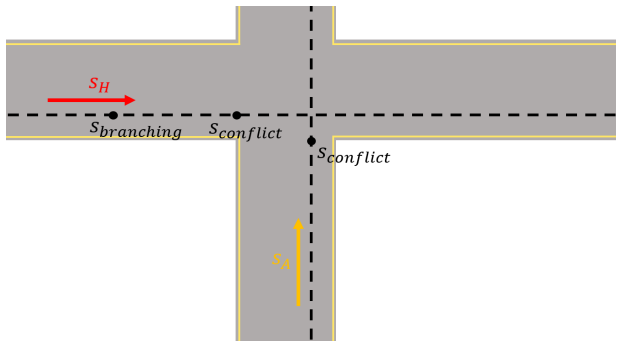


Figure 9.7: Coordinates of the intersection scenario shown in Figure 9.5.

sibility of steering the vehicle laterally and away from the lane center. Furthermore, not considering lateral movement simplifies the planning problem complexity. Both vehicles follow the model

$$x = \begin{bmatrix} s \\ v \end{bmatrix}, \quad \dot{x} = \begin{bmatrix} v \\ u \end{bmatrix}. \tag{9.1}$$

Where s is the current position along the centerline path, v is the vehicle longitudinal velocity, and u is the acceleration, corresponding to the control input. *Remark:* Planning only with respect to longitudinal motion is commonly referred to as velocity planning. We note that this approach could be easily modified into a complete motion planning approach by extending the vehicle states to include lateral motion. However, this would result in higher computational complexity.

The autonomous vehicle input u^A is determined by the solution planned by the planning framework presented in Section 9.3. The human-driven vehicle is assumed to follow a certain driving policy π^{before}, up to $s_{\text{branching}}$, and afterwards π^{after}. The input u^H is then:

$$u^H = \begin{cases} \pi^{\text{before}}(s^H, v^H) & \text{if } s^H < s_{\text{branching}} \\ \pi^{\text{after}}(s^H, v^H, s^A, v^A) & \text{if } s^H \geq s_{\text{branching}} \end{cases} \tag{9.2}$$

Note that π^{after} is a function of s^A and v^A, since the evolution of the human-driven vehicle states is affected by the autonomous vehicle.

Human driving policies

We assume the human-driven vehicle to have two types of policies, one for when there is no leading vehicle ahead of it, *velocity tracking*, and the other for when there is a vehicle ahead, *vehicle following*.

Velocity tracking

When there is no leading vehicle ahead, the human-driven vehicle will attempt to track a desired reference speed v_{ref}. We then have $u^H = \pi^{v_{\text{ref}},\text{track}}$, and the policy is defined as:

$$\pi^{v_{\text{ref}},\text{track}}\left(v^H, v_{\text{ref}}\right) = K_v\left(v_{\text{ref}} - v^H\right). \tag{9.3}$$

Policy $\pi^{v_{\text{ref}},\text{track}}$ takes as inputs the current vehicle velocity v^H and the desired reference velocity v_{ref}, and outputs an acceleration command proportional to the difference between them. This makes the vehicle accelerate or brake until arriving at its desired speed of v_{ref}.

Vehicle following

If there is another vehicle ahead of the human-driven vehicle, it becomes necessary for the human-driven vehicle to adapt its speed to avoid a rear-end collision. In this case $u^H = \pi^{\text{va}}$, where va stands for *vehicle ahead*. Policy π^{va} is defined as:

$$\pi^{\text{va}} = \begin{cases} \pi^{\text{vf}}\left(s^H, v^H, s^A, v^A\right) & \text{if } v^H \geq v_{\text{ref}} \\ \pi^{v_{\text{ref}},\text{track}}\left(v^H, v_{\text{ref}}\right) & \text{if } v^H < v_{\text{ref}} \end{cases} \tag{9.4}$$

with π^{vf}, where vf stands for *vehicle following*, is given by

$$\pi^{\text{vf}}\left(s^H, v^H, s^A, v^A\right) = K_v(v^A - v^H) + K_d\left(d - d_{\text{ref}}\right), \tag{9.5}$$

where $d = s^A - s^H$ corresponds to the distance from the human-driven vehicle to the vehicle ahead.

Policy π^{va} includes two cases. When $v^H \geq v_{\text{ref}}$, the vehicle velocity is greater than its desired velocity, and therefore the current vehicle will simply track its desired velocity v_{ref}, resulting in braking. For the case when $v^H < v_{\text{ref}}$, the vehicle velocity is smaller than its desired velocity, and therefore it should speed up. In order to safely speed up, the vehicle needs to take into account the vehicle ahead of it. The policy is then defined by two terms, $K_v(v^A - v^H)$, which ensures velocity tracking, and $K_d\left(d - d_{\text{ref}}\right)$, which tries to keep a safe distance d_{ref} to the vehicle in front. Policy π^{va} is inspired by the more complex Intelligent Driver Model [140].

Human decision making

In the considered scenarios we assume the human has two different behaviors, one before arriving at $s_{\text{branching}}$, and another after it, as given by Equation (9.2). Before arriving at $s_{\text{branching}}$, the vehicle is following a

velocity tracking policy, and driving at a constant speed, as there is no vehicle ahead of it. After $s_{\text{branching}}$ the human-driven vehicle starts interacting with the autonomous vehicle, and adapts its behavior to a new, possibly equal, policy.

The policy chosen after $s_{\text{branching}}$ depends on the type of driver profile and preferences, and on the traffic scene, and it is often not known in advance. Therefore, a motion planning algorithm should deal with the multiple possible policies that the human driver can decide on. In the merge scenario, Figure 9.4, we consider three types of behavior, corresponding to three types of driver:

- $\pi^{\text{va,fast}}$ - Corresponding to an egoistic driver who speeds up, so that $v_{\text{ref}} = v_{\text{ref}}^{\text{fast}}$,
- $\pi^{\text{va,keep}}$ - Corresponding to a neutral driver who keeps its speed, so that $v_{\text{ref}} = v_{\text{ref}}^{\text{keep}}$ remains unchanged,
- $\pi^{\text{va,slow}}$ - Corresponding to an altruistic driver who slows down, so that $v_{\text{ref}} = v_{\text{ref}}^{\text{slow}}$.

In the intersection scenario, Figure 9.5, we consider two types of behavior:

- π^{cross} - Corresponding to the human driver keeping its speed, so that $v_{\text{ref}} = v_{\text{ref}}^{\text{keep}}$ remains unchanged,
- π^{stop} - Corresponding to the driver slowing down and stopping at the intersection.

At point $s_{\text{branching}}$ the human decides on one of the policies to follow. We make use of research in the field of neuroscience, namely we consider the work in [129], where the authors study the decision making process of human drivers approaching an intersection. The authors propose that drivers' decision making is depends on the degree of safety of the two co-existing possibilities of either crossing or stopping at the intersection. The degree of safety can be quantified by the critical time to cross CT_{cross}, and the critical time to stop CT_{stop}.

The critical time to cross CT_{cross} is defined as the last time to safely cross the intersection by accelerating $A_{\max} = 2m/s^2$. CT_{cross} is given by the positive root of the following second order polynomial

$$\frac{A_{\max}}{2}x^2 + (-\text{TTC}A_{\max})\,x + \left(\frac{\text{TTC}^2 A_{\max}}{2} + \text{TTC}v - d\right)$$

where d is the distance of the of the human vehicle to the intersection. $\text{TTC} = d_o/v_o$ is the time until the oncoming vehicle arrives at the crossing, assuming that the oncoming vehicle drives at a fixed speed.

The critical time to cross CT_{stop} is defined as the amount of time left until the human driver can decide to apply the full braking capabilities of the vehicle and come to a safe stop. It is given by

$$CT_{\text{stop}} = \frac{d}{v} - \frac{v}{2D_{\max}},$$

where d is the distance of the vehicle to the crossing point, v is the vehicle velocity, and $D_{\max}$ is the maximum deceleration of the vehicle.

With CT_{cross} and CT_{stop} now defined, the authors of [129] define the probability of the human driver choosing to cross, *i.e.*, applying policy π^{cross}, as:

$$P_{\pi^{\text{cross}}} = \frac{w}{1 + e^{-aCT_{\text{cross}}}} + \frac{1 - w}{1 + e^{-bCT_{\text{stop}}}}. \tag{9.6}$$

Where the parameters a, b, and w are found by fitting the model Equation (9.6) to experimental data. The experimental data is obtained from thirty experienced drivers whose decision making is studied in a driving simulator. The paper [129] builds a dataset of human decision making at an intersection by exposing the drivers to several different intersection scenarios and analysing their responses. The tuned coefficients resulting from this study are $a = 1.67$, $b = -1.69$, and $w = 0.57$.

Equation (9.6) provides a decision making model that can be used to estimate the probabilities of the human driver taking different decisions. This then allows the autonomous vehicle to understand the likelihood of different future scene outcomes stemming from different decision taken by the human driver. Moreover, the dependency of Equation (9.6) on both the human driver vehicle state as well as the oncoming vehicle, in this case the autonomous vehicle, allows the planner to reason about how different maneuvers can affect the likelihood of different scenarios. This is fundamental in order to achieve assertive behavior from the autonomous vehicle. The following section proposes a framework that can take advantage of model Equation (9.6), providing the planner with a better knowledge of human driver behavior and allowing the autonomous vehicle to drive assertively.

9.3 Motion Planning Framework

Tree formulation - human-driven vehicle

In order to take into account the possible future decisions of the human, and the resulting states of its vehicle, we use a tree structure [135]. Figure 9.8 shows a tree with $J + 1$ branches (branches $[2, \ldots, J - 1]$ not visible). Each branch j in the tree has an associated human driver policy π^j. Within a

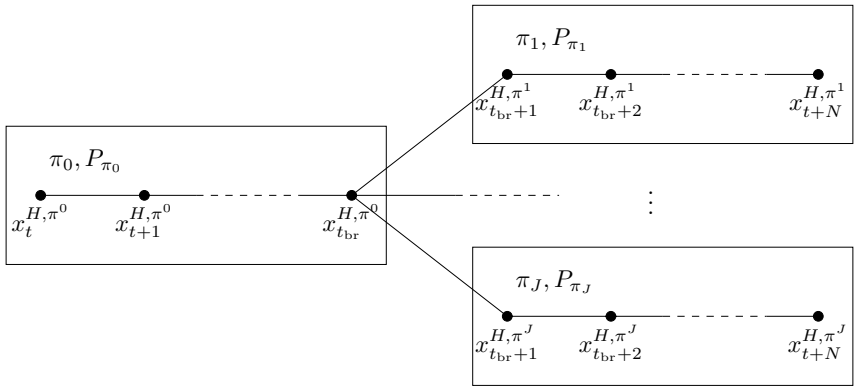


Figure 9.8: Diagram of the tree structure used to model the human-driven vehicle possible future states.

branch j, the human-driven vehicle states are propagated assuming that they follow the associated policy π^j. The evolution of states corresponds to a discretized model of Equation (9.1) so that

$$x^{H,\pi^j}_{t+1} = f^{H,\pi^j}\left(x^{H,\pi^j}_t, x^{A,\pi^j}_t\right), \tag{9.7}$$

where the vehicle input u^{H,π^j}_t is determined according the active policy π^j, and can depend on the autonomous vehicle state, as in Equation (9.5). For the transition states between branches we have

$$x^{H,\pi^j}_{t_{\text{br}}+1} = f^{H,\pi^j}\left(x^{H,\pi^0}_{t_{\text{br}}}, x^{A,\pi^0}_{t_{\text{br}}}\right), \tag{9.8}$$

where $u^{H,\pi^j}_{t_{\text{br}}}$ already follows the policy π^j.

The tree in Figure 9.8 is composed of a root branch and J branches descending from it, corresponding to the set of branches $\mathcal{J} = \{0, 1, 2, \ldots, J\}$. The root branch splits into different branches at time t_{br}, corresponding to the state at which the human-driven vehicle crosses the path length $s_{\text{branching}}$. At this path length $s_{\text{branching}}$, the human changes to a policy that corresponds to interacting with the autonomous vehicle. Since, there are multiple policies that the human can take, $x^{H,\pi^0}_{t_{\text{br}}}$ propagates into J different states $x^{H,\pi^j}_{t_{br}+1}$, corresponding to the different branches with policies π^j that the human can take. Each of the branches has an associated probability P_{π^j} of happening. For the first branch $P_{\pi^0} = 1$, and for the remaining branches $\sum_{j=1}^{J} P_{\pi^j} = 1$.

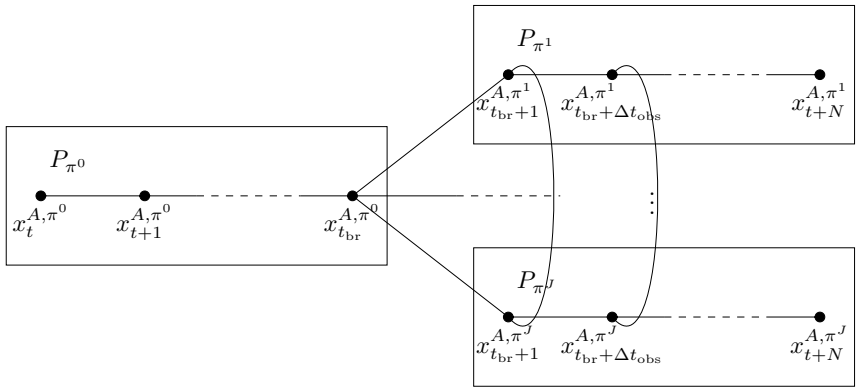


Figure 9.9: Diagram of the tree structure used to model the autonomous vehicle possible future states. The arcs between states in branch 1 and branch J correspond to autonomous vehicle states that are forced to be equal, according to Equation (9.11).

Tree formulation - autonomous vehicle

The autonomous vehicle state evolution follows a similar tree structure as the human-driven vehicle, and is illustrated in Figure 9.9. The evolution of states corresponds to a discretized model of Equation (9.1) so that

$$x^{A,\pi^j}_{t+1} = f^{A,\pi^j}\left(x^{A,\pi^j}_{t}, u^{A,\pi^j}_{t}\right). \tag{9.9}$$

The vehicle input u^{A,π^j}_{i} is determined by the solution to the optimal control problem that is introduced later in Section 9.3. Similarly, for the branching states, we have:

$$x^{A,\pi^j}_{t_{\text{br}}+1} = f^{A,\pi^j}\left(x^{A,\pi^0}_{t_{\text{br}}}, u^{A,\pi^j}_{i}\right). \tag{9.10}$$

In a practical setting, it is not possible to estimate the policy decision made by the human-driven vehicle immediately after it crosses $s_{\text{branching}}$, that is, at time t_{br}. Instead, the sensor and prediction systems of the autonomous vehicle have a delay until accurately estimating the new behavior of the human. To deal with this, we force the autonomous vehicle states to be equal between the different branches $[1, \ldots, J]$ for the first Δt_{obs} seconds of the branch:

$$x^{A,\pi^j}_{t_i} = x^{A,\pi^{j'}}_{t_i}, \forall t_i \in [t_{\text{br}}+1, \ldots, t_{\text{br}}+\Delta t_{\text{obs}}], \{\forall j, j' \in \mathcal{J} | j \neq j'\}. \tag{9.11}$$

Equation (9.11) forces the planned solutions to not assume immediate knowledge of the human-vehicle policy, and instead delay by Δt_{obs} the adaption

to the new behavior. Δt_{obs} is experimentally tuned based on considerations of expected time to perceive a new vehicle behavior and feasibility of the planning problem.

MPC formulation

For each branch $j \in \mathcal{J}$, consider the vectors

$$\begin{aligned}
\boldsymbol{x}_j^H &= [x_{t_i^j}^{H,\pi^j}, x_{t_i^j+1}^{H,\pi^j}, \dots, x_{t_f^j}^{H,\pi^j}], \\
\boldsymbol{x}_j^A &= [x_{t_i^j}^{A,\pi^j}, x_{t_i^j+1}^{A,\pi^j}, \dots, x_{t_f^j}^{A,\pi^j}], \\
\boldsymbol{u}_j^A &= [u_{t_i^j}^{A,\pi^j}, u_{t_i^j+1}^{A,\pi^j}, \dots, u_{t_f^j}^{A,\pi^j}],
\end{aligned}$$

where t_i^j and t_f^j are initial and final times associated with the first and last states in branch j. $\boldsymbol{x}_j^H$ corresponds to the human-driven vehicle states, and $\boldsymbol{x}_j^A$, $\boldsymbol{u}_j^A$ to the autonomous vehicle states, and inputs, respectively.

The MPC is solved at the beginning of every planning cycle at time t, its formulation is as follows:

$$\underset{\{\boldsymbol{u}_j^A\}_{j\in\mathcal{J}}}{\text{minimize}} \quad \sum_{j\in\mathcal{J}} P_{\pi^j} J(\boldsymbol{x}_j^H, \boldsymbol{x}_j^A, \boldsymbol{u}_j^A) \tag{9.12a}$$

$$\text{subject to} \quad \text{Equation (9.7)}, \forall i = t_i^j, t_i^j + 1, \dots, t_f^j, \forall j \in \mathcal{J} \tag{9.12b}$$

$$\text{Equation (9.8)}, \forall j \in \mathcal{J} \setminus \{0\}, \tag{9.12c}$$

$$\text{Equation (9.9)}, \forall i = t_i^j, t_i^j + 1, \dots, t_f^j, \forall j \in \mathcal{J}, \tag{9.12d}$$

$$\text{Equation (9.10)}, \forall j \in \mathcal{J} \setminus \{0\}, \tag{9.12e}$$

$$\text{Equation (9.11)}, \forall j \in \mathcal{J} \setminus \{0\}, \tag{9.12f}$$

$$\begin{cases} P_{\pi^0} = 1 & \text{if } t \le t_{\text{br}} \\ P_{\pi^0} = 0 & \text{otherwise} \end{cases}, \tag{9.12g}$$

$$\begin{cases} \text{Equation (9.6)}, \ j \in \mathcal{J} \setminus \{0\} & \text{if } t \le t_{\text{br}} + \Delta t_{\text{obs}} \\ P_{\pi^{j'}} = 1, P_{\pi^{j'}} = 0, \ j \in \mathcal{J} \setminus \{0, j'\} & \text{otherwise} \end{cases}, \tag{9.12h}$$

$$\begin{cases} x_t^{H,\pi^0} = x_{\text{init}}^H, x_t^{A,\pi^0} = x_{\text{init}}^A & \text{if } t \le t_{\text{br}} \\ x_t^{H,\pi^j} = x_{\text{init}}^H, x_t^{A,\pi^j} = x_{\text{init}}^A, j \in \mathcal{J} \setminus \{0\} & \text{otherwise} \end{cases}. \tag{9.12i}$$

Equations (9.12b) and (9.12d) encode the state evolution constraints within the branches, for the human-driven and autonomous vehicles, respectively. Similarly, Equations (9.12c) and (9.12e) encode the state evolution

constraints between branches. Equation (9.12f) sets the observation constraints forcing the autonomous vehicle states to be equal across branches $j \in \mathcal{J} \backslash \{0\}$, from time t_{br} until time $t_{\text{br}} + \Delta t_{\text{obs}}$. Equation (9.12g) defines the probability of the root branch: in case the current planning cycle happens before the human-vehicle has taken a decision, $t \leq t_{\text{br}}$, the branch exists and is certain, $P_{\pi^0} = 1$. Otherwise, the branch does not exist anymore, and it is not regarded in the optimization objective, $P_{\pi^0} = 0$. Equation (9.12h) defines the probabilities of the leaf branches: in case the current planning cycle happens before the autonomous vehicle can identify the true mode of the human, $t \leq t_{\text{br}} + \Delta t_{\text{obs}}$, its probability is given by the behavioral decision making model in [129]. Otherwise, the probability of the branch corresponding to the true mode is set to 1, and the probability of all other branches is set to 0, and therefore ignored from the optimization objective. Finally, Equation (9.12i) makes the initial human-driven and autonomous vehicle states of the root branch correspond to the current measured states $x^H_{\text{init}}, x^A_{\text{init}}$ of the vehicles if $t \leq t_{\text{br}}$. Otherwise it makes the initial state of all possible branches equal to $x^H_{\text{init}}, x^A_{\text{init}}$.

Solving optimization problem Equation (9.12), we find the optimal autonomous vehicle input vectors $\boldsymbol{u}_j^{A\star}$ for each branch of the tree. At each planning cycle, the optimization problem Equation (9.12) is solved, considering the latest sensor measurements $x^H_{\text{init}}, x^A_{\text{init}}$. The first element of the root branch input vector $\boldsymbol{u}_0^{A\star}$ is used as a vehicle control input if $t \leq t_{\text{br}}$. Otherwise, the first element of an active branch is used as the vehicle control input.

9.4 Results

The following results are obtained in a custom Python simulation environment. The MPC problem presented in Equation (9.12) is formulated using CasADi [141] and the corresponding nonlinear optimization problem is solved using Ipopt [142].

Uncertain Traffic Light

We present the results of a scenario where a traffic light on the road cannot be properly perceived by the vehicle sensors. In this scenario, illustrated in Figure 9.10, the autonomous vehicle is approaching a traffic light. The vehicle knows about the existence of the traffic light, due to information contained in maps or since the perception systems can detect the existence of the traffic light pole. However, the vehicle does not know the state of the traffic light due to poor visibility conditions. As the vehicle approaches the traffic light, it is eventually able to correctly perceive its actual state.

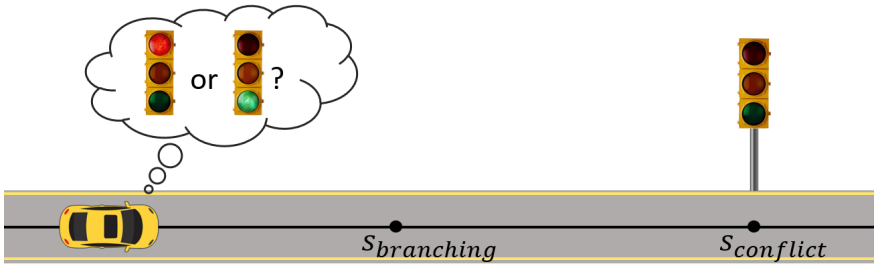


Figure 9.10: Scenario with an uncertain traffic light.

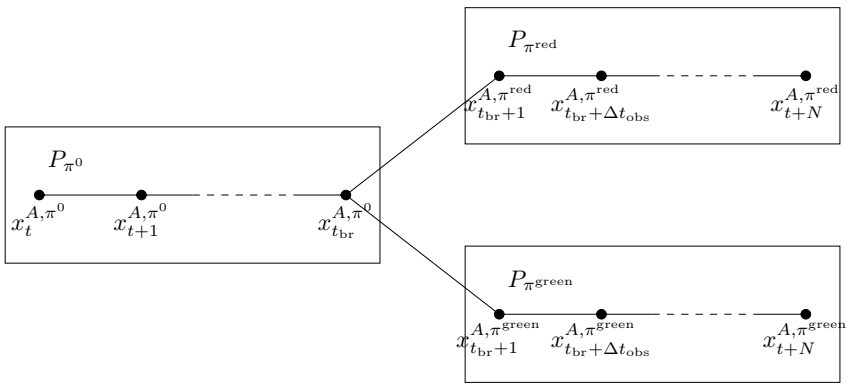


Figure 9.11: Diagram of the tree structure used in the uncertain traffic light scenario illustrated in Figure 9.10.

We assume that at $s_{\text{branching}}$ the vehicle accurately detects the state of the traffic light.

The corresponding tree structure diagram for this scenario is shown in Figure 9.11. After the vehicle crosses $s_{\text{branching}}$, at time t_{br}, it can take one of two policies, either continue driving in case the traffic light is green, π^{green}, or come to a stop in case the traffic light is red, π^{red}.

We study the expected performances for a Robust MPC, a Prescient MPC, a Contingency MPC, and the proposed Branch MPC. The different MPC formulations are as follows:

- Robust MPC - An MPC that assumes that the traffic light is red until $s_{\text{branching}}$, afterwards it knows the actual traffic light state;
- Prescient MPC - An MPC with perfect perception that knows the

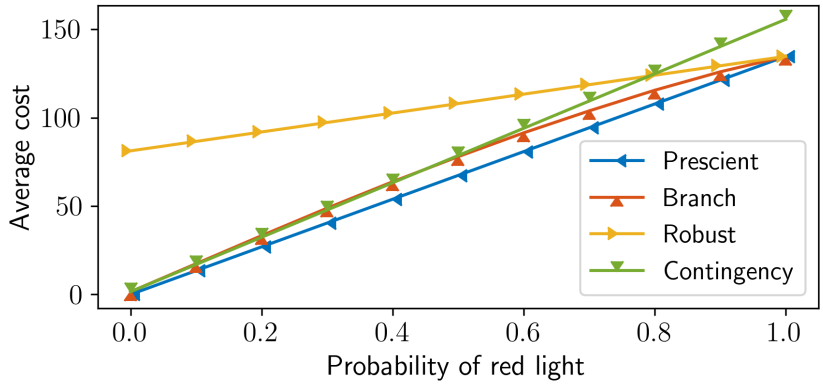


Figure 9.12: Average costs of Robust, Prescient, Contingency, and Branch MPC for different probabilities of the traffic light being red.

state of the traffic light even before $s_{\text{branching}}$;

- Contingency MPC - An MPC optimizing the driving behavior as if the traffic light was green, but always keeping a contingency plan in case the traffic light is red, as in [137];
- Branch MPC - The proposed approach considering both traffic light possibilities as illustrated in Figure 9.11, and assuming $P_{\pi^{\text{green}}} = P_{\pi^{\text{red}}} = 0.5$.

We run closed-loop simulations for different probabilities of the traffic light being red. The closed-loop performance of the different approaches, for different probabilities of the obstacle existing, *i.e.*, of the traffic light being red, is shown in Figure 9.12.

The Robust MPC has an expected cost that is never better than the Prescient or Branch MPCs, no matter the probability of the obstacle existing. Robust MPC also performs worse than Contingency MPC for most situations, however when the probability of the obstacle is high (> 0.8), Robust MPC actually outperforms Contingency MPC on the expected cost. Since Robust MPC always assumes that the obstacle exists, it actually achieves a good performance (compared to the Prescient and Branch alternatives) and a better performance (compared to Contingency MPC) when the probability of the obstacle is high enough. However, for lower probabilities of the obstacle, assuming that an obstacle exists, when it in fact does not, is detrimental for the Robust MPC, which achieves the worst performance of all methods.

Figure 9.12 shows that Prescient MPC achieves the best performance for all possible probabilities of the obstacle. This is expected, as the Prescient MPC knows about the actual existence or not of the obstacle, and therefore is able to optimize its maneuver according to a future observation that is still unknown to all other methods.

As seen in Figure 9.12, the performance of Contingency MPC is comparable to the Branch and Prescient MPCs for lower obstacle probabilities. However, as the obstacle probability increases the performance of Contingency MPC degrades, becoming worse even than Robust MPC. This can be explained by the fact that Contingency MPC is an optimistic planner, assuming that the obstacle does not exist until proven otherwise. Assuming that the obstacle does not exist turns out to be problematic when the obstacle indeed exists. Since Contingency MPC ignores the obstacle until it becomes visible, it must perform a very abrupt braking maneuver, which although being safe, results in a poor performance. On the other hand, in case when the obstacle has low probability of existing, the Contingency MPC actually achieves a performance on part with Prescient and Branch MPC.

Finally, our proposed method, Branch MPC, achieves on average a performance that is always better than Robust MPC, while being equally safe. For lower probabilities of an obstacle existing, the Branch MPC performance is comparable to that of Contingency MPC. However, for higher probabilities of an obstacle, Branch MPC outperforms Contingency MPC, due to not being optimistic about the obstacle existence.

Figure 9.13 shows the solution of a single planning cycle, for the different MPC approaches starting at the same vehicle initial state. The initial vehicle state is chosen so as to reflect the situation where the vehicle is aware that there might be an obstacle, but does not know yet if the obstacle exists with certainty. In this case, the Robust MPC plans a single velocity profile that assumes the existence of the obstacle, and that brings the vehicle to a stop. Contingency MPC plans two velocity profiles to deal with the possibility of existence, or not, of the obstacle. The velocity profile associated with the obstacle existence is not penalized in the cost function, resulting in the contingency maneuver being sudden and uncomfortable. On the other hand, Branch MPC reduces its driving speed (increasing travel time) in order to comfortably deal with the possible existence of the obstacle. Branch MPC is therefore choosing a tradeoff between the performances in both future possible scenarios.

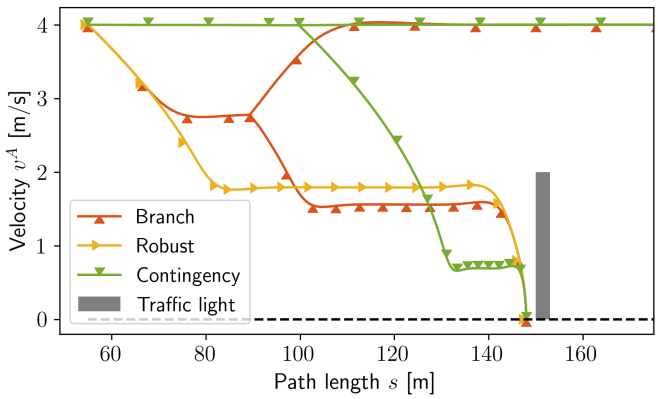


Figure 9.13: Single planning cycle for the autonomous vehicle approaching an uncertain obstacle, under different planning frameworks: Branch, Robust, and Contingency MPCs.

Multi-modality

We now consider the merging scenario shown in Figure 9.4. The human driver is assumed to have three possible future velocity tracking policies $\pi^{\text{va,fast}}$, $\pi^{\text{va,keep}}$, or $\pi^{\text{va,slow}}$. For these policies the resulting tree diagram is shown in Figure 9.14, where the human driver keeps a constant speed policy π_0 until the branching point, where it then chooses one of the three possible policies. We note that although chosen manually in this study, the future tracking policies could be instead the result of a prediction module providing a multi-modal set of outcomes, as in [77, 143].

Figure 9.15 (bottom) shows the planned velocity profiles of a planning cycle occurring during the merge scenario. This planning cycle occurs when both the human-driven vehicle and the autonomous vehicle are heading towards the merging point, but still significantly far from it. It can be seen that the human vehicle is predicted to take three different possible velocity profiles, corresponding to accelerating, keeping its speed, or slowing down. The autonomous vehicle plans a set of velocity profiles that results in it going behind the human driven vehicle in case it decides to keep its speed or accelerate, or in going ahead of the human in case it decides to slow down. The autonomous vehicle plan can be better visualized by looking at the planned path lengths in Figure 9.15 (top). The path lengths (subtracted by the predicted path length of a human vehicle with policy $\pi^{\text{va,keep}}$) show the distinct decisions made by the planner to either go behind or ahead of different possible future policies of the human driver.

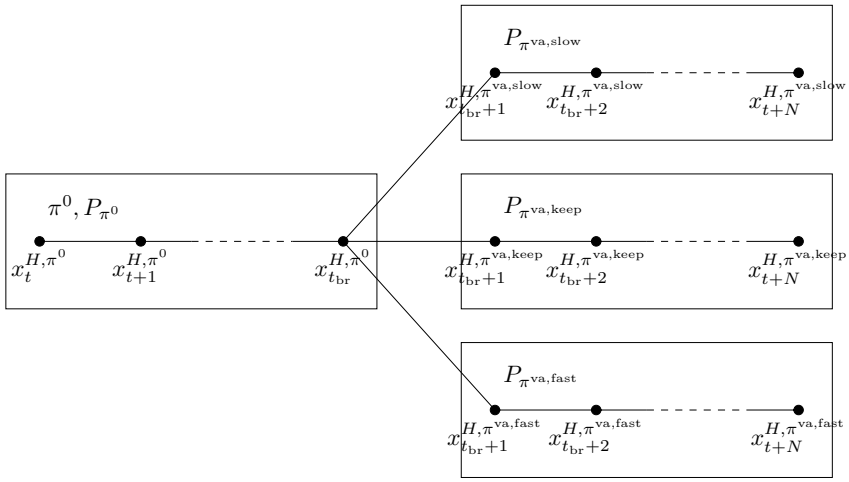


Figure 9.14: Diagram of the tree structure used to model the human-driven vehicle possible future state evolution in the merge scenario.

Figure 9.17 shows the same scenario when solved by a Robust MPC approach. In this case, it can be seen that the planned maneuver is a very conservative one, that decides to go behind all possible realizations of the possible future human policies.

Comparing the Branch MPC with the Robust MPC highlights the benefits of the former, as well as the importance of considering the multi-modality of the human driver. Considering the multi-modality of the human driver, and planning feedback policies dependent on the actual mode that the human eventually decides upon, allow the planned solutions to squeeze in between different possible policies, something that would not be possible with more conservative approaches.

Finally, Figure 9.18 presents the executed velocity profile when using the Branch MPC in closed loop simulations. At a certain point in time, $t = t_{\text{br}}$, the human makes a decision (*human branching*, highlighted in Figure 9.18), and starts following policy $\pi^{\text{va,fast}}$. After a few seconds, at $t = t_{\text{br}} + \Delta t_{\text{obs}}$, the autonomous vehicle is able to accurately estimate the new human policy (*accurate prediction*, highlighted in the Figure 9.18), and the Branch MPC reverts to a single prediction policy MPC.

Non-interaction *vs.* interaction-aware human models

In order to allow for assertive and efficient driving, it is essential to consider that human-driven vehicles react and adapt to the autonomous vehicle. We

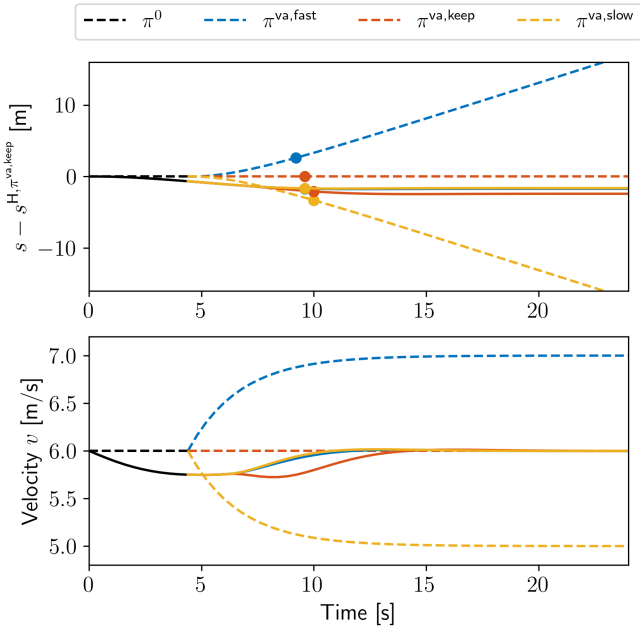


Figure 9.15: Planned autonomous vehicle trajectory (solid line) and predicted human-driven vehicle trajectory (dashed line) for the Branch MPC case. Top: Path lengths (centered around the path executed by $\pi^{\text{va,keep}}$). The rounded markers correspond to the instant when the vehicles cross the conflict point. Bottom: Velocity profiles. A zoomed version is shown in Figure 9.16.

now consider the case of an autonomous vehicle that needs to accelerate from a standstill and merge onto a road with an oncoming vehicle driving at a high speed, similar to the scenario considered in [67]. We consider a Branch MPC which can take a decision of either going ahead of the other vehicle, or waiting for it to pass and then go behind of it.

Figure 9.19 shows the results for a single planning cycle, where the autonomous vehicle is at a standstill, and there is an oncoming vehicle driving at a speed higher than the desired cruising speed of the autonomous vehicle. In the non-interacting vehicle model case, the Branch MPC predicts the human to keep its speed constant, and therefore decides to wait for it to pass, and afterwards, go behind of it. In case of packed traffic, *i.e.*, if there were several oncoming vehicles, the autonomous vehicle could be stuck indefinitely at the junction [67]. However, when considering an interaction

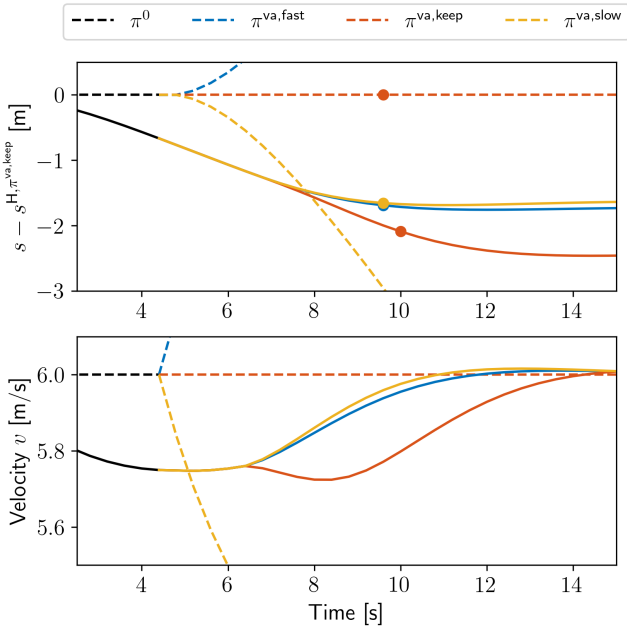


Figure 9.16: Detail of Figure 9.15. Top: Path lengths (centered around the path executed by $\pi^{\text{va,keep}}$). The rounded markers correspond to the instant when the vehicles cross the conflict point. Bottom: Velocity profiles.

aware model, the Branch MPC decides to go ahead of the oncoming vehicle. This is possible as the prediction of the human behavior takes into account that a human driver will slow down and adapt its speed for a vehicle cutting into traffic.

To further highlight the benefits of considering an interaction-aware model, Figure 9.20 shows a comparison of the planning solution costs for different scenarios, where in each scenario the oncoming human-driven vehicle starts at a different distance s_0 from the junction. The planner using the non-interaction model always decides to go behind of the oncoming vehicle. Due to the higher velocity of the oncoming vehicle, the planner predicts that going ahead of it results in a rear-end collision, and therefore it chooses to go behind of the oncoming vehicle.

On the other hand, the planner with the interaction aware model is able to correctly predict that the oncoming vehicle will slow down when the autonomous vehicle merges into traffic, and takes advantage of it to go ahead, reducing the optimal plan cost. However, when the oncoming vehicle

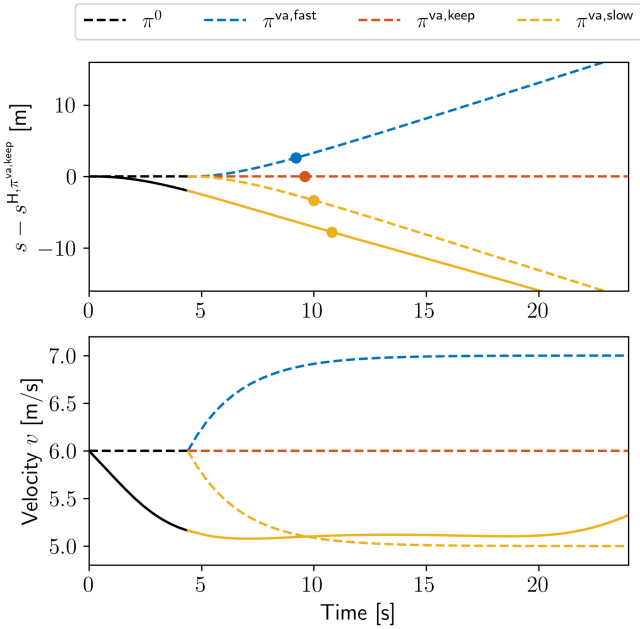


Figure 9.17: Planned autonomous vehicle trajectory (solid line) and predicted human-driven vehicle trajectory (dashed line) for the Robust MPC case. Top: Path lengths (centered around the path executed by $\pi^{\text{va,keep}}$). The rounded markers correspond to the instant when the vehicles cross the conflict point. Bottom: Velocity profiles.

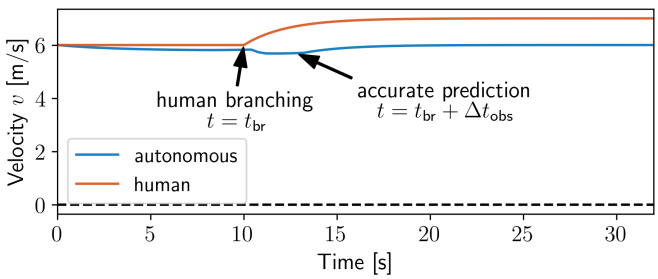


Figure 9.18: Closed loop executed velocity profiles on the merging scenario.

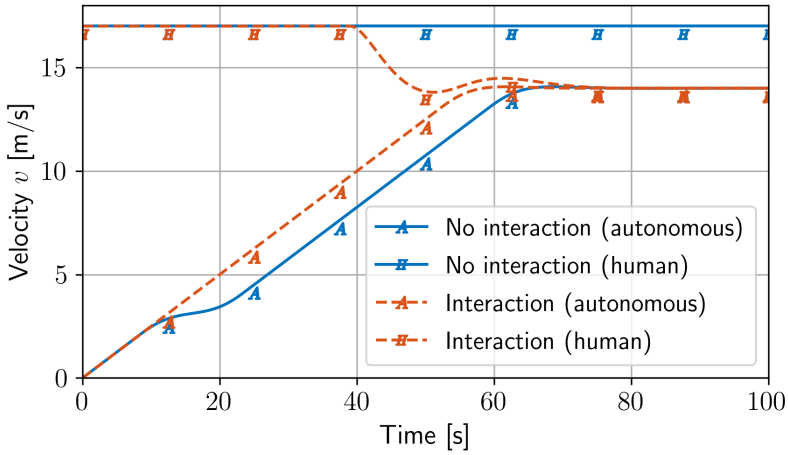


Figure 9.19: Comparison of predicted and planned velocity profiles when considering non-interacting and interaction-aware human-driven vehicle models.

is already too close to the junction, $s_0 > -390$, the planner decides to go behind of it, as going ahead would either cause a collision, or it would cause the human to brake past its limits. The braking limits of the human are set to a reasonable deceleration, to guarantee that the autonomous vehicle does not act in a way that is uncomfortable or discourteous to other traffic participants [144].

Intersection scenario

We now consider the intersection scenario shown in Figure 9.5. In this scenario, both vehicles approach a non-signalized intersection, while having a clear sight of each other. Since there are no traffic lights or priority rules, the vehicles have to negotiate between themselves who goes first. The considered tree structure for this scenario is shown in Figure 9.21.

Figure 9.22a shows the planned velocities when considering that both human-driven vehicle policies have an equal fixed probability $P_{\pi^{\text{cross}}} = P_{\pi^{\text{stop}}} = 0.5$. It can be seen that the autonomous vehicle decides to slow down so as to deal with both possible outcomes of the human decision. In case the human-driven vehicle keeps its speed, that is, it takes policy π^{cross}, the autonomous vehicle further slows down to ensure collision avoidance. Otherwise, if the human-driven vehicle comes to a stop, policy π^{stop}, the autonomous vehicle increases its velocity to its desired reference velocity.

We note that this intersection scenario resembles the one considered

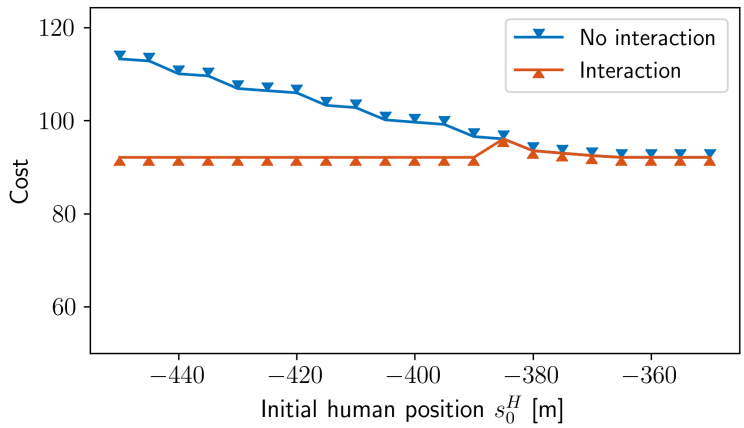


Figure 9.20: Comparison of expected costs for a non-interacting human-driven vehicle model and an interaction-aware model for different initial conditions s_0^H of the human-driven vehicle.

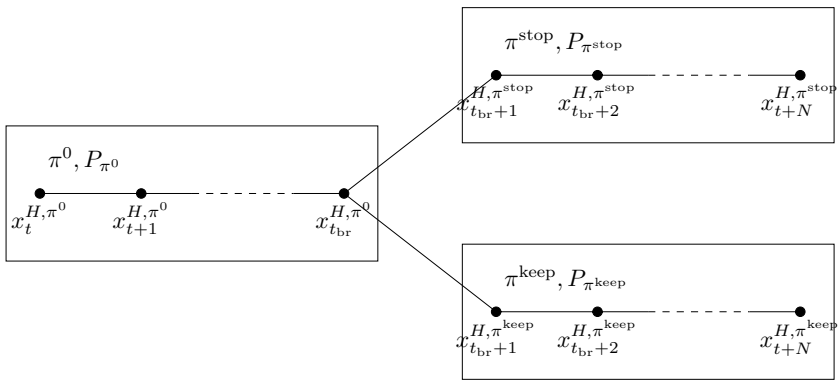
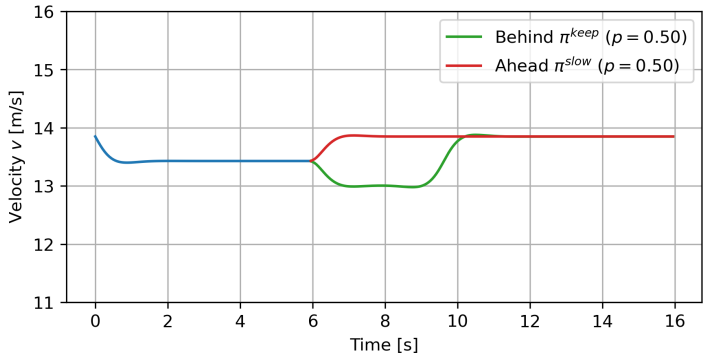
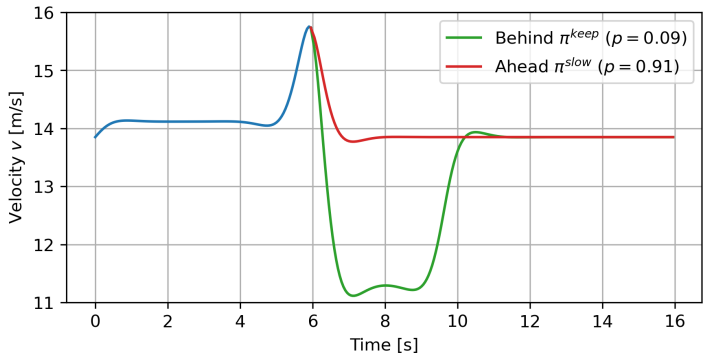


Figure 9.21: Diagram of the tree structure used to model the human-driven vehicle possible future states in the intersection scenario.



(a) Fixed probabilities.



(b) Adapting probabilities.

Figure 9.22: Intersection scenario for different configurations of human-vehicle policy probabilities.

in [129]. Therefore we use the human decision making model Equation (9.6) to determine the probability $P_{\pi^{\text{stop}}}$, and its counterpart $P_{\pi^{\text{cross}}} = 1 - P_{\pi^{\text{stop}}}$. Figure 9.22b shows the results for this scenario when considering that the human policy probabilities follow the human decision model in Equation (9.6). In this case the planner decides to speed up slightly so as to approach the intersection earlier. Furthermore, at around $t = 5$ s, the planner further speeds up, so as to clearly indicate to the human driver that it intends to pass. This results in a higher predicted probability of the human deciding to slow down and give way to the autonomous vehicle, $P_{\pi^{\text{stop}}} = 0.91$. With a lower probability of the human keeping its speed, $P_{\pi^{\text{cross}}} = 0.09$, the autonomous vehicle plans a more aggressive maneuver in the more unlikely event that this branch occurs.

These results show that considering the human decision making model

to determine the probabilities of the possible future branches allows the planner to make assertive driving maneuvers. The autonomous vehicle maneuvers in a way that shows intent to other human vehicles, in order to improve its expected driving outcomes. This comes naturally as a result of minimizing the objective cost of Equation (9.12a), and therefore does not require manually tuned driving strategies for different traffic situations.

9.5 Conclusions

This chapter presented a motion planning framework for tackling the challenges of autonomous vehicles interacting with human drivers. Three challenges associated with human drivers, multi-modality, interactive behavior, and decision-making, were introduced. In order to tackle these challenges, a planning solution must perform joint prediction (of human drivers) and planning (of the autonomous vehicle).

We introduce scenario trees as a suitable model to express human drivers and the multiple outcomes of their decisions. We then present recent research in the field of neuroscience that attempts to model the driver decision-making process at intersections. An MPC problem is then formulated that combines these two concepts.

Through a toy example with an uncertain traffic light, we show how the Branch MPC achieves a better performance than a Robust MPC while still being safe. Furthermore, we show that the Branch MPC is not as optimistic as a Contingency MPC, leading to better performance on average. We also show how the Branch MPC can consider the multi-modal predictions of the human driver. We present a scenario where the autonomous vehicle decides to squeeze between two policies of the human driver's future outcomes, something that would not be possible with a Robust MPC. The interaction-aware models show their advantage when considering a merging scenario where the autonomous vehicle needs to merge ahead of another vehicle. By considering interactions, the human vehicle merges into the lane earlier, with only little disturbance to the following incoming vehicle. Finally, we study the effects of using a decision-making model for the human and show how this leads the planner to find assertive maneuvers and display intent to the human vehicle.

Considering the three challenges mentioned earlier, *multi-modality*, *interaction*, and *decision-making*, we developed a motion planner that interacts with human drivers, planning assertive driving maneuvers and can influence human drivers into making decisions that have favorable outcomes for the autonomous vehicle. The proposed framework is a promising solution to enable self-driving technology in urban traffic, where interactions and assertiveness are needed to drive efficiently.

As future work, it is interesting to evaluate the some of the assumptions when modeling the problem. The assumption that a human-driver makes a single decision at a point $s_{\text{branching}}$ along the road might not capture the true nature of human decision making. Moreover, it would be interesting to consider behavioral decision making for more scenarios than the intersection one, and study if assertive behavior of autonomous vehicles can be achieved in those as well. Finally, a practical implementation on an autonomous vehicle is needed to validate the suitability of the proposed approach to accurately model and solve the joint prediction and planning problem.

Chapter 10

Conclusions and Future Work

This thesis focused on motion planning algorithms for autonomous heavy-duty vehicles. The trending field of autonomous driving promises to disrupt current transportation systems, and fundamentally shape the mobility of the future [145]. Heavy-duty vehicles represent a significant share of vehicles driving on- and off-roads today. Due to its fundamental differences when compared to passenger vehicles, they require special education for drivers [146], and a special adaptation of autonomous driving algorithms, as highlighted throughout this thesis.

In Section 10.1 we summarize the conclusions made so far in our research. These conclusions provide some insight into the key problems that arise when motion planning algorithms move from the passenger vehicle domain into the heavy-duty vehicle one. Future work and research directions are provided in Section 10.2. Based on our previously developed work, we suggest promising research avenues that can be further investigated. We also enumerate a few parallel topics within motion planning, that were not addressed in this thesis. These topics are, in our opinion, expected to play an important role in the development, and eventual deployment of safe and reliable autonomous vehicles.

10.1 Conclusions

A recurrent insight that can be seen across the whole thesis, is that motion planning for autonomous heavy-duty vehicles cannot be reliably achieved using algorithms developed for passenger vehicles. An extensive body of literature exists when it comes to motion planning for passenger vehicles, however, there is relatively little work published that explicitly considers trucks and buses. This thesis, and contributions herein, tries to address

this gap, by identifying and proposing solutions to different challenges that hinder the arrival of autonomous heavy-duty vehicles.

Slow actuator dynamics

When developing motion planning algorithms, it is important to consider as realistic as possible vehicle models, in order to reduce resulting errors between planned and executed motions. Most motion planning approaches ensure quality of the planned paths, by setting constraints on the curvature profile of the paths. However, these constraints lack a direct connection to the vehicle steering actuator limits.

We have developed, in Chapter 3 a motion planner that explicitly takes into account the steering actuator limits. From the steering actuator limits, the motion planner is able to compute the necessary curvature profile constraints required for the planned paths. Steering actuator limits are often easier to measure and understand than its curvature constraints counterpart, and as such are a better input to a motion planning algorithm. The proposed motion planner is generalizable to any car-like vehicle, requiring only the knowledge on the limitations of its steering actuator.

Due to their large weight, heavy-duty vehicles are often characterized by slow actuator dynamics [147]. In order to plan paths that respect these dynamics, we take into account limits not only in the maximum steering angle magnitude and rate, but also in the applicable torque. This results in smoother planned motions that effectively respect all three actuator limitations. The controller effort is thus minimized, and path tracking performance is improved, resulting in more comfortable and safer driving.

Complex obstacle environments

A significant share of heavy-duty vehicles is targeted for use in unstructured off-road environments. These types of environments are often populated with numerous obstacles that are part of a changing landscape. Thus, vehicles need to have motion planning capabilities that allow them to navigate safely, but also efficiently, in order to achieve their intended tasks.

Motion planning in these kinds of environments has been studied for long in the field of robotics. Unlike traditional robotic systems, heavy-duty vehicles are subject to severe kynodynamic constraints, which have motivated the development of new motion planning algorithms. These algorithms usually rely on sampling of the search space, which results in sub-optimality and in some cases, in a drastic reduction of vehicle efficiency.

Chapter 4 tackles the issues resulting from sub-optimality, namely, oscillations of the solution path, and discretization of the goal state. The proposed solution, targeted for lattice-based planners, is able to significantly

remove the oscillations present in solution paths, and increase the length of straight driving sections. This allows for increased vehicle performance, as the vehicles can achieve higher velocities due to the reduced turning motions. With regard to the discretization of the goal state, we show that the proposed solution is often able to successfully plan paths arriving at specific target destinations. This allows heavy-duty vehicles to successfully be used in applications requiring precision driving, such as when it is necessary to precisely stop beside other heavy machinery or fixed infrastructure.

Bus driving behavior

Buses are characterized by a particular chassis design significantly differing from that of other vehicles. In order to be able to increase passenger capacity, the vehicle body is quite large, however, to maintain vehicle structural robustness and increase maneuverability, the wheelbase is kept short. The sections of the chassis extending beyond the wheelbase are known as overhangs.

Overhangs are quite tall and can sweep over curbs and low height obstacles, an attribute that professional bus drivers take advantage of. By allowing the overhangs to sweep over curbs, bus drivers increase the maneuverability of the vehicle. This is of crucial importance when driving in urban environments, and is often observed during sharp turn maneuvers or bus stop approaches.

This type of driving is not observed in any other type of vehicle, and when it comes to motion planning algorithms it is an unaddressed issue. To deal with this, Chapter 5 proposes a new environment classification scheme, which explicitly takes into account that curbs and other low height obstacles can be swept by the bus overhangs. Furthermore, new driving objective functions are defined, which seek to mimic professional bus driver behavior. The result is a motion planning algorithm that is able to tackle the challenges of bus driving, and does so while increasing the safety and maneuverability of buses.

Wide vehicle dimensions

Trucks and buses are distinguished from other vehicles by their wide dimensions. When driving on roads or in constrained environments, it is often the case that drivers struggle in order to successfully drive their vehicles without colliding. The same is true for most motion planning algorithms, which are often plagued with the inability to plan motions that require the vehicle to pass through narrow environments.

In Chapter 5, we develop a motion planning algorithm that can find solutions in arbitrarily narrow roads. The algorithm takes advantage of

the convexity of the environment, which can be assumed for the case of on-road driving. The motion planning problem is solved using a numerical optimization method. We introduce new vehicle body approximations to ensure safe but non-conservative collision checking, which is of the utmost importance when considering vehicles with large dimensions or on narrow roads. The proposed vehicle body approximations, together with the numerical optimization framework used, result in a motion planner that successfully tackles the driving task for buses in urban environments.

Optimal driving behavior for long and multi-body vehicles

Driving in the center of the road is essential to good road driving behavior. To formulate this objective, most motion planners focus on minimizing the distance of a point in the vehicle, usually the rear axle center or the center point of the vehicle, to the road's centerline. In practice, this works well, as long as the vehicle has a short wheelbase, as in passenger vehicles. However, for truck-trailers and buses, formulating the objective of driving in the road center is not trivial, as is observed in Chapter 6.

Chapter 7 starts by defining the optimal driving behavior as that which keeps the area swept by a vehicle equally distant to both lane limits. Using geometric arguments, Chapter 7 proposes a framework that designs cost objectives that, when minimized, force the vehicle to have this optimal behavior. The geometric derivation assumes a road with constant curvature. However, both simulations in Chapter 7 and experiments with a prototype bus in Chapter 8 show that the proposed approach is suitable for roads with varying curvature profiles. The proposed framework for designing objectives is algorithm-agnostic, only requiring that the motion planning algorithm relies on optimizing towards a cost function, which most approaches do.

Interactions with human-driven vehicles

Interacting with human-driven vehicles is a challenge that is highly relevant to all autonomous vehicles nowadays, not only heavy-duty vehicles. When driving with human-driven vehicles, particularly in dense traffic scenarios, autonomous vehicles must consider how humans adapt and possibly cooperate with the autonomous vehicle. Such reasoning is essential to be able to drive in the presence of other humans while not being overly conservative. This challenge is aggravated when considering heavy-duty vehicles due to their slower dynamics and sheer size. During a merge maneuver or lane change in dense traffic, the heavy-duty vehicle requires significantly larger gaps between the other vehicles to perform the maneuver successfully. Understanding the intentions of human drivers becomes essential to predict if other participants will cooperate in the desired maneuver or not.

Chapter 9 proposes a decision making and motion planning approach to tackle the challenge of interacting with human-driven vehicles. The proposed approach jointly predicts and plans maneuvers for the autonomous vehicle, reasoning about other drivers' cooperativeness or lack of it. To more accurately predict the actions of humans, a behavioral model of decision making for human drivers approaching intersections is used. The proposed planning approach tackles several challenges of human-autonomous vehicle interaction, namely, multi-modality, interactions, and decision making of the human. The autonomous vehicle is shown to have a safe but not too conservative behavior and an assertive driving style, conveying its intentions to humans.

10.2 Future work

The work developed in this thesis has identified and helped solve some of the issues currently impairing motion planning for autonomous heavy-duty vehicles. However, during this research process, new problems previously overlooked have emerged, and with them, promising directions for future work. We hereby list these directions, some of them correspond to natural improvements to the work already presented, while others correspond to new frameworks and approaches to motion planning.

Controller performance guarantees

The trend seen in motion planning to consider ever more accurate vehicle models and kinodynamic constraints, can be seen as an attempt to reduce the disparity between planned motions and executed motions. In extreme scenarios, this disparity can result in unsafe systems, since collision checking is performed for the planned path, not for the executed path. Thus, if the vehicle deviates from the planned path, as it will always happen as long as there are modeling errors, collision avoidance is not guaranteed. This can be partially addressed by improving the vehicle models used, however at the cost of requiring system identification procedures. Furthermore, more complex vehicle models will also increase computational times for motion planning algorithms. Alternatively, conservative collision checking procedures can be used, which inflate the vehicle states or the obstacle regions. However, the conservativeness can render planning algorithms useless in narrow environments.

Formal verification techniques such as those presented in [90, 148] are able to deal with parametric model uncertainty and bounded disturbances. Using these techniques, it is possible to compute the set of reachable vehicle states, *i.e.*, an occupancy funnel, in which the vehicle is guaranteed to be

contained when executing a certain maneuver. These occupancy funnels are often expensive to compute and are not suitable for online computations. However, this challenge can be addressed by remembering that some motion planning methods, such as those presented in Chapter 3 and Chapter 4 make use of precomputed motions primitives. Motion primitives and their associated funnels can be computed offline, and then used in an online fashion.

These formal verification techniques are a promising avenue of future work. They provide motion planners with the ability to explicitly minimize the vulnerability of planned paths to disturbances and uncertainties in a systematic way. They also provide a measurement of risk for different planned actions, which could, in the future, help certify and legislate autonomous vehicles.

Wheel-aware planning

The practical experiments with a prototype autonomous bus in Chapter 8 revealed that precise maneuvering near curbs requires the motion planner to consider the vehicle's wheels. As autonomous vehicle technology matures and its deployment broadens, the complexity of maneuvers performed increases. In the case of buses, approaching and departing from a bus stop is a common maneuver. The distance from the vehicle to the bust stop curb should be minimized to allow for easier boarding of passengers. To minimize this distance while guaranteeing the safety of the maneuver, one needs to consider how the wheels protrude outwards from the vehicle body and how they depend on the current steering of the vehicle. One can imagine similar scenarios with autonomous passenger light vehicles, particularly those used in Mobility as a Service businesses, which frequently pick up and drop off passengers.

From CPU to GPU

The solutions presented in this thesis all assumed a sequential implementation in a central processing unit (CPU). However, the performance of graphics processing units (GPUs) has dramatically increased in recent years. The usage of GPUs can allow for a significant reduction in the computational time of motion planning algorithms [92]. Furthermore, new motion planning frameworks can be used, which are based on parallel computing, instead of traditional sequential computing.

From model-based to learning-based

Machine learning approaches have made their way into many areas of engineering, and in autonomous driving, they excel at computer vision and perception tasks. Motion planning can benefit from machine learning approaches. These approaches already permeate the decision making process of autonomous vehicles in many implementations. Machine learning can be incorporated to improve specific procedures of planning algorithms, such as predicting traffic participants' behavior, or they can provide high-level decision making guidance to a simplified planner. Some authors even propose end-to-end approaches where a neural network completely replaces the motion planner [149]. Learning-based approaches are promising solutions to tackle some of the not-yet-solved autonomous driving challenges. However, research efforts are still needed to increase their explainability and validate their safety.

Bibliography

[1] Scania. Annual report 2012. https://www.scania.com/group/en/scania-annual-report-2012-2/, 2013.

[2] B. Costello and R. Suarez. Truck driver shortage analysis 2015. https://www.trucking.org/ATA%20Docs/News%20and%20Information/Reports%20Trends%20and%20Statistics/10%206%2015%20ATAs%20Driver%20Shortage%20Report%202015.pdf, 2015.

[3] J. Ainsalu, V. Arffman, M. Bellone, M. Ellner, T. Haapamäki, N. Haavisto, E. Josefson, A. Ismailogullari, B. Lee, O. Madland, R. Madžulis, J. Müür, S. Mäkinen, V. Nousiainen, E. Pilli-Sihvola, E. Rutanen, S. Sahala, B. Schønfeldt, P. M. Smolnicki, R.-M. Soe, J. Sääski, M. Szymańska, I. Vaskinn, and M. Åman. State of the art of automated buses. https://www.mdpi.com/2071-1050/10/9/3118/, 2018.

[4] World Health Organization. Global status report on road safety 2018, 2018.

[5] National Highway Traffic Safety Administration's National Center for Statistics and Analysis. Critical reasons for crashes investigated in the national motor vehicle crash causation survey, 2015.

[6] L. Clements and K. Kockelman. Economic effects of automated vehicles. *Transportation Research Record*, 2606(1):106–114, 2017.

[7] Bureau of Transportation Statistics. U.S. Vehicle-Miles — Bureau of Transportation Statistics. https://www.bts.gov/content/us-vehicle-miles, 2019.

[8] eurostat. Energy, transport and environment SBK. https://ec.europa.eu/eurostat/documents/3217494/9433240/KS-DK-18-001-EN-N.pdf/, 2018.

[9] American Transportation Research Institute. Identifying Autonomous Vehicle Technology Impacts on the Trucking Industry - American Transportation Research Institute. https://atri-online.org/2016/11/15/identifying-autonomous-vehicle-technology-impacts-on-the-trucking-industry/, 2016.

[10] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, March 2016. ISSN 2379-8858. doi: 10.1109/TIV.2016.2578706.

[11] C. Katrakazas, M. Quddus, W. Chen, and L. Deka. Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions. *Transportation Research Part C: Emerging Technologies*, 60:416 – 442, 2015. ISSN 0968-090X.

[12] S. Russell and P. Norvig. Artificial intelligence - a modern approach. *Artificial Intelligence. Prentice-Hall, Englewood Cliffs*, 25:27, 1995.

[13] S. M. LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[14] R. Oliveira, P. F. Lima, M. Cirillo, J. Mårtensson, and B. Wahlberg. Trajectory generation using sharpness continuous dubins-like paths with applications in control of heavy-duty vehicles. In *2018 European Control Conference (ECC)*, pages 935–940, 2018. doi: 10.23919/ECC.2018.8550279.

[15] R. Oliveira, M. Cirillo, J. Mårtensson, and B. Wahlberg. Combining lattice-based planning and path optimization in autonomous heavy duty vehicle applications. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 2090–2097, June 2018. doi: 10.1109/IVS.2018.8500616.

[16] P. F. Lima, R. Oliveira, J. Mårtensson, and B. Wahlberg. Minimizing long vehicles overhang exceeding the drivable surface via convex path optimization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017. doi: 10.1109/ITSC.2017.8317754.

[17] R. Oliveira, P. F. Lima, G. Collares Pereira, J. Mårtensson, and B. Wahlberg. Path planning for autonomous bus driving in highly constrained environments. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2743–2749, Oct 2019. doi: 10.1109/ITSC.2019.8916773.

[18] R. Oliveira, O. Ljungqvist, P. F. Lima, and B. Wahlberg. Optimization-based on-road path planning for articulated vehicles. In *21st IFAC World Congress*, pages 15572–15579, 2020. doi: 10.1016/j.ifacol.2020.12.2402.

[19] R. Oliveira, O. Ljungqvist, P. F. Lima, and B. Wahlberg. A geometric approach to on-road motion planning for long and multi-body heavy-duty vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 999–1006, 2020. doi: 10.1109/IV47402.2020.9304767.

[20] R. Oliveira, P. F. Lima, M. Cirillo, and B. Wahlberg. Autonomous bus driving: A novel motion-planning approach. *IEEE Vehicular Technology Magazine*, 16(3):29–37, 2021. doi: 10.1109/MVT.2021.3086438.

[21] Y. Gao, A. Gray, J. V. Frasch, T. Lin, E. Tseng, J. K. Hedrick, and F. Borrelli. Spatial predictive control for agile semi-autonomous ground vehicles. In *Proceedings of the 11th international symposium on advanced vehicle control*, 2012.

[22] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb. Making bertha drive - an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2), Summer 2014. ISSN 1939-1390. doi: 10.1109/MITS.2014.2306552.

[23] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[24] S. M. LaValle and J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

[25] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957. ISSN 00029327, 10806377. URL `http://www.jstor.org/stable/2372560`.

[26] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990. URL `https://projecteuclid.org:443/euclid.pjm/1102645450`.

[27] T. Fraichard and A. Scheuer. From reeds and shepp's to continuous-curvature paths. *IEEE Transactions on Robotics*, 20(6):1025–1035, Dec 2004. ISSN 1552-3098. doi: 10.1109/TRO.2004.833789.

[28] H. Banzhaf, L. Palmieri, D. Nienhüser, T. Schamm, S. Knoop, and J. M. Zöllner. Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017. doi: 10.1109/ITSC.2017.8317757.

[29] H. Banzhaf, N. Berinpanathan, D. Nienhüser, and J. M. Zöllner. From g2 to g3 continuity: Continuous curvature rate steering functions for sampling-based nonholonomic motion planning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 326–333, June 2018. doi: 10.1109/IVS.2018.8500653.

[30] Y. Kanayama and B. I. Hartman. Smooth local path planning for autonomous vehicles. In *Proceedings, 1989 International Conference on Robotics and Automation*, pages 1265–1270 vol.3, May 1989. doi: 10.1109/ROBOT.1989.100154.

[31] J. Funke and C. Gerdes. Simple clothoid lane change trajectories for automated vehicles incorporating friction constraints. *Journal of Dynamic Systems, Measurement, and Control*, 138(2):021002, 2016.

[32] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha. A real-time motion planner with trajectory optimization for autonomous vehicles. In *2012 IEEE International Conference on Robotics and Automation*, pages 2061–2067. IEEE, 2012.

[33] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. Optimal trajectory generation for dynamic street scenarios in a frenét frame. In *2010 IEEE International Conference on Robotics and Automation*, pages 987–993, May 2010. doi: 10.1109/ROBOT.2010.5509799.

[34] T. Berglund, A. Brodnik, H. Jonsson, M. Staffanson, and I. Soderkvist. Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles. *IEEE Transactions on Automation Science and Engineering*, 7(1):167–172, Jan 2010. ISSN 1545-5955. doi: 10.1109/TASE.2009.2015886.

[35] M. Cirillo. From videogames to autonomous trucks: A new algorithm for lattice-based motion planning. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 148–153, June 2017. doi: 10.1109/IVS.2017.7995712.

[36] O. Ljungqvist, N. Evestedt, M. Cirillo, D. Axehill, and O. Holmer. Lattice-based motion planning for a general 2-trailer system. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 819–824, June 2017. doi: 10.1109/IVS.2017.7995817.

[37] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, July 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136.

[38] M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 767–774, Cambridge, MA, USA, 2003. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2981345.2981441`.

[39] Y. Björnsson, V. Bulitko, and N. R Sturtevant. TBA*: Time-bounded A*. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 431–436, 2009.

[40] S. Koenig and M. Likhachev. D* Lite. In *Eighteenth National Conference on Artificial Intelligence*, pages 476–483, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence. ISBN 0-262-51129-0. URL `http://dl.acm.org/citation.cfm?id=777092.777167`.

[41] C. Altafini, A. Speranzon, and B. Wahlberg. A feedback control scheme for reversing a truck and trailer vehicle. *IEEE Transactions on Robotics and Automation*, 17(6):915–922, Dec 2001. ISSN 1042-296X. doi: 10.1109/70.976025.

[42] M. McNaughton, C. Urmson, J. M. Dolan, and J. W. Lee. Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *International Conference on Robotics and Automation*, pages 4889–4895, May 2011. doi: 10.1109/ICRA.2011.5980223.

[43] J. Nocedal and S. Wright. *Sequential Quadratic Programming*, pages 529–562. Springer New York, New York, NY, 2006. ISBN 978-0-387-40065-5. doi: 10.1007/978-0-387-40065-5_18. URL `https://doi.org/10.1007/978-0-387-40065-5_18`.

[44] Xiaojing Zhang, Alexander Liniger, and Francesco Borrelli. Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology*, 29(3):972–983, 2021. doi: 10.1109/TCST.2019.2949540.

[45] Victor Fors, Pavel Anistratov, Björn Olofsson, and Lars Nielsen. Predictive Force-Centric Emergency Collision Avoidance. *Journal of Dynamic Systems, Measurement, and Control*, 143(8), 04 2021. ISSN 0022-0434. doi: 10.1115/1.4050403. URL `https://doi.org/10.1115/1.4050403`.

[46] Pavel Anistratov, Björn Olofsson, Oleg Burdakov, and Lars Nielsen. Autonomous-vehicle maneuver planning using segmentation and the alternating augmented lagrangian method. In *21th IFAC World Congress Proceedings :*, volume 53 of *IFAC PapersOnline*, pages 15558–15565, 2020. doi: 10.1016/j.ifacol.2020.12.2400.

[47] Johan Karlsson, Nikolce Murgovski, and Jonas Sjöberg. Computationally efficient autonomous overtaking on highways. *IEEE Transactions on Intelligent Transportation Systems*, 21(8):3169–3183, 2020. doi: 10.1109/TITS.2019.2929963.

[48] H. Andersen, W. Schwarting, F. Naser, Y. H. Eng, M. H. Ang, D. Rus, and J. Alonso-Mora. Trajectory optimization for autonomous overtaking with visibility maximization. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017. doi: 10.1109/ITSC.2017.8317853.

[49] Ivo Batkovic, Mario Zanon, Mohammad Ali, and Paolo Falcone. Real-time constrained trajectory planning and vehicle control for proactive autonomous driving with road users. In *2019 18th European Control Conference (ECC)*, pages 256–262, 2019. doi: 10.23919/ECC.2019.8796099.

[50] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099, June 2015. doi: 10.1109/IVS.2015.7225830.

[51] V. Turri, A. Carvalho, H. E. Tseng, K. H. Johansson, and F. Borrelli. Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 378–383, Oct 2013. doi: 10.1109/ITSC.2013.6728261.

[52] Lars Svensson, Monimoy Bujarbaruah, Nitin R. Kapania, and Martin Törngren. Adaptive trajectory planning and optimization at limits of handling. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3942–3948, 2019. doi: 10.1109/IROS40897.2019.8967679.

[53] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli. Predictive control of an autonomous ground vehicle using an iterative linearization approach. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 2335–2340, Oct 2013. doi: 10.1109/ITSC.2013.6728576.

[54] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen. Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres. *Vehicle System Dynamics*, 52(10):1304–1332, 2014. doi: 10.1080/00423114.2014.939094. URL `https://doi.org/10.1080/00423114.2014.939094`.

[55] G. P. Bevan, H. Gollee, and J. O'Reilly. Trajectory generation for road vehicle obstacle avoidance using convex optimization. *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, 224(4):455–473, 2010. doi: 10.1243/09544070JAUTO1204. URL `https://doi.org/10.1243/09544070JAUTO1204`.

[56] J. Timings and D. Cole. Minimum maneuver time calculation using convex optimization. *Journal of Dynamic Systems, Measurement, and Control*, 135(3):031015, 2013.

[57] P. F. Lima, G. Collares Pereira, J. Mårtensson, and B. Wahlberg. Progress maximization model predictive controller. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1075–1082, Nov 2018. doi: 10.1109/ITSC.2018.8569647.

[58] J. Choi and K. Huhtala. Constrained global path optimization for articulated steering vehicles. *IEEE Transactions on Vehicular Technology*, 65(4):1868–1879, April 2016. ISSN 0018-9545. doi: 10.1109/TVT.2015.2424933.

[59] K. Bergman and D. Axehill. Combining homotopy methods and numerical optimal control to solve motion planning problems. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 347–354, June 2018. doi: 10.1109/IVS.2018.8500644.

[60] J. Ziegler, P. Bender, T. Dang, and C. Stiller. Trajectory planning for bertha — a local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 450–457, June 2014. doi: 10.1109/IVS.2014.6856581.

[61] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes. Collision avoidance and stabilization for autonomous vehicles in emergency scenarios.

IEEE Transactions on Control Systems Technology, 25(4):1204–1216, July 2017. ISSN 1063-6536. doi: 10.1109/TCST.2016.2599783.

[62] Siddharth H Nair, Vijay Govindarajan, Theresa Lin, Yan Wang, Eric H Tseng, and Francesco Borrelli. Stochastic mpc with dual control for autonomous driving with multi-modal interaction-aware predictions. *arXiv preprint arXiv:2208.03525*, 2022.

[63] T. Tram, I. Batkovic, M. Ali, and J. Sjöberg. Learning when to drive in intersections by combining reinforcement learning and model predictive control. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 3263–3268, Oct 2019. doi: 10.1109/ITSC.2019.8916922.

[64] Bruno Brito, Achin Agarwal, and Javier Alonso-Mora. Learning interaction-aware guidance for trajectory optimization in dense traffic scenarios. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2022. doi: 10.1109/TITS.2022.3160936.

[65] Tommy Tram, Anton Jansson, Robin Grönberg, Mohammad Ali, and Jonas Sjöberg. Learning negotiating behavior between cars in intersections using deep q-learning. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3169–3174, 2018. doi: 10.1109/ITSC.2018.8569316.

[66] Junqing Wei, John M. Dolan, and Bakhtiar Litkouhi. Autonomous vehicle social behavior for highway entrance ramp management. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 201–207, 2013. doi: 10.1109/IVS.2013.6629471.

[67] Erik Ward, Niclas Evestedt, Daniel Axehill, and John Folkesson. Probabilistic model for interaction aware planning in merge scenarios. *IEEE Transactions on Intelligent Vehicles*, 2(2):133–146, 2017.

[68] Vikram Krishnamurthy. *Partially Observed Markov Decision Processes: From Filtering to Controlled Sensing*. Cambridge University Press, 2016. doi: 10.1017/CBO9781316471104.

[69] Constantin Hubmann, Jens Schulz, Marvin Becker, Daniel Althoff, and Christoph Stiller. Automated driving in uncertain environments: Planning with interaction and uncertain maneuver prediction. *IEEE Transactions on Intelligent Vehicles*, 3(1):5–17, 2018. doi: 10.1109/TIV.2017.2788208.

[70] Bingyu Zhou, Wilko Schwarting, Daniela Rus, and Javier Alonso-Mora. Joint multi-policy behavior estimation and receding-horizon

trajectory planning for automated urban driving. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2388–2394, 2018. doi: 10.1109/ICRA.2018.8461138.

[71] Alessandro Zanardi, Saverio Bolognani, Andrea Censi, and Emilio Frazzoli. *Game Theoretical Motion Planning. Tutorial ICRA 2021.* ETH Zurich, Zurich, 2021-08. doi: 10.3929/ethz-b-000507914.

[72] Dorsa Sadigh, Nick Landolfi, Shankar S Sastry, Sanjit A Seshia, and Anca D Dragan. Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state. *Autonomous Robots*, 42(7):1405–1426, 2018.

[73] Jaime F. Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S. Shankar Sastry, and Anca D. Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596, 2019. doi: 10.1109/ICRA.2019.8794007.

[74] David Fridovich-Keil, Ellis Ratner, Lasse Peters, Anca D Dragan, and Claire J Tomlin. Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In *2020 IEEE international conference on robotics and automation (ICRA)*, pages 1475–1481. IEEE, 2020.

[75] Simon Le Cleac'h, Mac Schwager, and Zachary Manchester. ALGAMES: A Fast Solver for Constrained Dynamic Games. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi: 10.15607/RSS.2020.XVI.091.

[76] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019. doi: 10.1073/pnas.1820676116. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1820676116`.

[77] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. MATS: An interpretable trajectory forecasting representation for planning and control. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 2243–2256. PMLR, 16–18 Nov 2021.

[78] Jiquan Ngiam, Vijay Vasudevan, Benjamin Caine, Zhengdong Zhang, Hao-Tien Lewis Chiang, Jeffrey Ling, Rebecca Roelofs, Alex Bewley,

Chenxi Liu, Ashish Venugopal, David J Weiss, Ben Sapp, Zhifeng Chen, and Jonathon Shlens. Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=Wm3EA5OlHsG.

[79] S. K. Rao. Unit curve for design of highway transitions. *Journal of Transportation Engineering*, 121(2):169–175, 1995. doi: 10.1061/(ASCE)0733-947X(1995)121:2(169).

[80] A. M. Shkel and V. Lumelsky. Classification of the dubins set. *Robotics and Autonomous Systems*, 34(4):179 – 202, 2001. ISSN 0921-8890. doi: https://doi.org/10.1016/S0921-8890(00)00127-5. URL http://www.sciencedirect.com/science/article/pii/S0921889000001275.

[81] 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, Sep. 2009. ISSN 1063-6536. doi: 10.1109/TCST.2008.2012116.

[82] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The International Journal of Robotics Research*, 29(5):485–501, 2010. doi: 10.1177/0278364909359210. URL https://doi.org/10.1177/0278364909359210.

[83] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt*. In *2011 IEEE International Conference on Robotics and Automation*, pages 1478–1483, May 2011. doi: 10.1109/ICRA.2011.5980479.

[84] T. Shima, S. J. Rasmussen, A. G. Sparks, and K. M. Passino. Multiple task assignments for cooperating uninhabited aerial vehicles using genetic algorithms. *Computers & Operations Research*, 33(11):3252 – 3269, 2006. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2005.02.039. URL http://www.sciencedirect.com/science/article/pii/S030505480500095X. Part Special Issue: Operations Research and Data Mining.

[85] H. J. Sussmann and G. Tang. Shortest paths for the reeds-shepp car: a worked out example of the use of geometric techniques in nonlinear optimal control. *Rutgers Center for Systems and Control Technical Report*, 10:1–71, 1991.

[86] P. Souères and J. D. Boissonnat. *Optimal trajectories for non-holonomic mobile robots*, pages 93–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 1998. ISBN 978-3-540-40917-5. doi: 10.1007/BFb0036072. URL https://doi.org/10.1007/BFb0036072.

[87] G. Parlangeli and G. Indiveri. Dubins inspired 2d smooth paths with bounded curvature and curvature derivative. *IFAC Proceedings Volumes*, 43(16):252 – 257, 2010. ISSN 1474-6670. doi: https://doi.org/10.3182/20100906-3-IT-2019.00045. URL http://www.sciencedirect.com/science/article/pii/S1474667016350650. 7th IFAC Symposium on Intelligent Autonomous Vehicles.

[88] H. Banzhaf, M. Dolgov, J. E. Stellet, and J. M. Zöllner. From footprints to beliefprints: Motion planning under uncertainty for maneuvering automated vehicles in dense scenarios. In *21st International Conference on Intelligent Transportation Systems, ITSC 2018, Maui, HI, USA, November 4-7, 2018*, pages 1680–1687. IEEE, 2018. doi: 10.1109/ITSC.2018.8569897. URL https://doi.org/10.1109/ITSC.2018.8569897.

[89] T. Gawron and M. Michałek. A g3-continuous extend procedure for path planning of mobile robots with limited motion curvature and state constraints. *Applied Sciences*, 8:2127, 11 2018. doi: 10.3390/app8112127.

[90] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017. doi: 10.1177/0278364917712421. URL https://doi.org/10.1177/0278364917712421.

[91] M. Cirillo, T. Uras, and S. Koenig. A lattice-based approach to multi-robot motion planning for non-holonomic vehicles. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 232–239, Sept 2014. doi: 10.1109/IROS.2014.6942566.

[92] M. McNaughton. *Parallel Algorithms for Real-time Motion Planning*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, July 2011.

[93] F. Schwarzer, M. Saha, and J.-C. Latombe. *Exact Collision Checking of Robot Paths*, pages 25–41. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-45058-0. doi: 10.1007/978-3-540-45058-0_3. URL https://doi.org/10.1007/978-3-540-45058-0_3.

[94] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How. Motion planning for urban driving using RRT. In *International Conference on Intelligent Robots and Systems*, pages 1681–1686, Sept 2008. doi: 10.1109/IROS.2008.4651075.

[95] D. Fassbender, B. C. Heinrich, and H. J. Wuensche. Motion planning for autonomous vehicles in highly constrained urban environments. In *International Conference on Intelligent Robots and Systems*, pages 4708–4713, Oct 2016. doi: 10.1109/IROS.2016.7759692.

[96] J. Ziegler and C. Stiller. Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios. In *International Conference on Intelligent Robots and Systems*, pages 1879–1884, Oct 2009. doi: 10.1109/IROS.2009.5354448.

[97] S. Wang. *State lattice-based motion planning for autonomous on-road driving*. PhD thesis, Freie Universität Berlin, 2015.

[98] G. Schildbach and F. Borrelli. A dynamic programming approach for nonholonomic vehicle maneuvering in tight environments. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 151–156, June 2016. doi: 10.1109/IVS.2016.7535379.

[99] W. Schwarting, J. Alonso-Mora, and D. Rus. Planning and decision-making for autonomous vehicles. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):187–210, 2018. doi: 10.1146/annurev-control-060117-105157.

[100] C. Götte, M. Keller, C. Rösmann, T. Nattermann, C. Haß, K. H. Glander, A. Seewald, and T. Bertram. A real-time capable model predictive approach to lateral vehicle guidance. In *International Conference on Intelligent Transportation Systems*, Nov 2016. doi: 10.1109/ITSC.2016.7795865.

[101] M. G. Plessen, P. F. Lima, J. Mårtensson, A. Bemporad, and B. Wahlberg. Trajectory planning under vehicle dimension constraints using sequential linear programming. In *International Conference on Intelligent Transportation Systems*, Oct 2017. doi: 10.1109/ITSC.2017.8317665.

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

[103] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[104] Bureau of Transportation Statistics. U.S. Vehicle-Miles — Bureau of Transportation Statistics, 2019.

[105] A. Chottani, Hastings G., Murnane J., and Neuhaus F. Distraction or disruption? autonomous trucks gain ground in us logistics. `https://www.mckinsey.com/industries/travel-logistics-and-infrastructure/our-insights/distraction-or-disruption-autonomous-trucks-gain-ground-in-us-logistics`, 2018.

[106] American Transportation Research Institute. Identifying Autonomous Vehicle Technology Impacts on the Trucking Industry - American Transportation Research Institute, 2016.

[107] F. Lamiraux et al. Motion planning and control for Hilare pulling a trailer. *IEEE Transactions on Robotics and Automation*, 15(4):640–652, Aug 1999.

[108] P. Zips, M. Böck, and A. Kugi. An optimisation-based path planner for truck-trailer systems with driving direction changes. In *Proc. IEEE Int. Conf. on Robotics and Automation*, pages 630–636, Seattle, WA, USA, May 2015. doi: 10.1109/ICRA.2015.7139245.

[109] B. Li, T. Acarman, Y. Zhang, L. Zhang, C. Yaman, and Q. Kong. Tractor-trailer vehicle trajectory planning in narrow environments with a progressively constrained optimal control approach. *IEEE Transactions on Intelligent Vehicles*, 5(3):414–425, 2020. doi: 10.1109/TIV.2019.2960943.

[110] K. Bergman and D. Axehill. Combining homotopy methods and numerical optimal control to solve motion planning problems. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 347–354, 2018. doi: 10.1109/IVS.2018.8500644.

[111] N. Evestedt, O. Ljungqvist, and D. Axehill. Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT. In *Proceedings of the 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3690–3697, 2016.

[112] O. Ljungqvist, N. Evestedt, D. Axehill, M. Cirillo, and H. Pettersson. A path planning and path-following control framework for a general 2-trailer with a car-like tractor. *Journal of Field Robotics*, 36(8):1345–1377, 2019.

[113] O. Ljungqvist, K. Bergman, and D. Axehill. Optimization-based motion planning for multi-steered articulated vehicles. *IFAC-PapersOnLine*, 53(2):15580–15587, 2020.

[114] F. Lamiraux, J. Laumond, C. Van Geem, D. Boutonnet, and G. Raust. Trailer truck trajectory optimization: the transportation of components for the airbus a380. *IEEE Robotics Automation Magazine*, 12(1):14–21, March 2005. ISSN 1070-9932. doi: 10.1109/MRA.2005.1411414.

[115] S. Beyersdorfer and S. Wagner. Novel model based path planning for multi-axle steered heavy load vehicles. In *Proceedings of the 16th International Conference on Intelligent Transportation Systems*, pages 424–429, Oct 2013.

[116] N. van Duijkeren, T. Keviczky, P. Nilsson, and L. Laine. Real-time NMPC for semi-automated highway driving of long heavy vehicle combinations. *IFAC-PapersOnLine*, 48(23):39–46, 2015.

[117] C. Altafini. Following a path of varying curvature as an output regulation problem. *IEEE Transactions on Automatic Control*, 47(9): 1551–1556, 2002.

[118] C. Altafini. Path following with reduced off-tracking for multibody wheeled vehicles. *IEEE Transactions on Control Systems Technology*, 11(4):598–605, 2003.

[119] B. A. Jujnovich and D. Cebon. Path-Following Steering Control for Articulated Vehicles. *Journal of Dynamic Systems, Measurement, and Control*, 135(3), 03 2013. ISSN 0022-0434.

[120] M. Michałek. Motion control with minimization of a boundary off-track for non-standard n-trailers along forward-followed paths. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1564–1569. IEEE, 2015.

[121] X. Liu and D. Cebon. A minimum swept path control strategy for reversing articulated vehicles. In *Proceedings of the 2018 IEEE Intelligent Vehicles Symposium*, pages 1962–1967, June 2018.

[122] Xuanzuo Liu, Anil K. Madhusudhanan, and David Cebon. Minimum swept-path control for autonomous reversing of a tractor semi-trailer. *IEEE Transactions on Vehicular Technology*, 68(5):4367–4376, 2019. doi: 10.1109/TVT.2019.2895513.

[123] C. Altafini, A. Speranzon, and B. Wahlberg. A feedback control scheme for reversing a truck and trailer vehicle. *IEEE Transactions on Robotics and Automation*, 17(6):915–922, 2001. doi: 10.1109/70.976025.

[124] Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. Osqp: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12(4):637–672, 2020.

[125] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.

[126] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. Springer, 2009.

[127] G. Collares Pereira, P. F. Lima, B. Wahlberg, H. Pettersson, and J. Mårtensson. Reference aware model predictive control for autonomous vehicles. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 376–383, 2020. doi: 10.1109/IV47402.2020.9304670.

[128] Fangyu Wu, Raphael E. Stern, Shumo Cui, Maria Laura Delle Monache, Rahul Bhadani, Matt Bunting, Miles Churchill, Nathaniel Hamilton, R'mani Haulcy, Benedetto Piccoli, Benjamin Seibold, Jonathan Sprinkle, and Daniel B. Work. Tracking vehicle trajectories and fuel rates in phantom traffic jams: Methodology and data. *Transportation Research Part C: Emerging Technologies*, 99: 82–109, 2019. ISSN 0968-090X. doi: https://doi.org/10.1016/j.trc.2018.12.012. URL `https://www.sciencedirect.com/science/article/pii/S0968090X18318345`.

[129] Geoffrey Marti, Antoine HP Morice, and Gilles Montagne. Drivers' decision-making when attempting to cross an intersection results from choice between affordances. *Frontiers in human neuroscience*, 8:1026, 2015.

[130] Peter Trautman and Andreas Krause. Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 797–803, 2010. doi: 10.1109/IROS.2010.5654369.

[131] Ivo Batkovic, Ugo Rosolia, Mario Zanon, and Paolo Falcone. A robust scenario mpc approach for uncertain multi-modal obstacles. *IEEE Control Systems Letters*, 5(3):947–952, 2020.

[132] Georg Schildbach and Francesco Borrelli. Scenario model predictive control for lane change assistance on highways. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 611–616, 2015. doi: 10.1109/IVS.2015.7225752.

[133] Gianluca Cesari, Georg Schildbach, Ashwin Carvalho, and Francesco Borrelli. Scenario model predictive control for lane change assistance and autonomous driving on highways. *IEEE Intelligent Transportation Systems Magazine*, 9(3):23–35, 2017. doi: 10.1109/MITS.2017.2709782.

[134] Tim Brüdigam, Michael Olbrich, Marion Leibold, and Dirk Wollherr. Combining stochastic and scenario model predictive control to handle target vehicle uncertainty in an autonomous driving highway scenario. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1317–1324, 2018. doi: 10.1109/ITSC.2018.8569909.

[135] Yuxiao Chen, Ugo Rosolia, Wyatt Ubellacker, Noel Csomay-Shanklin, and Aaron D. Ames. Interactive multi-modal motion planning with branch model predictive control. *IEEE Robotics and Automation Letters*, 7(2):5365–5372, 2022. doi: 10.1109/LRA.2022.3156648.

[136] Tim Brüdigam, Kenan Ahmic, Marion Leibold, and Dirk Wollherr. Legible model predictive control for autonomous driving on highways. *IFAC-PapersOnLine*, 51(20):215–221, 2018. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2018.11.016. URL `https://www.sciencedirect.com/science/article/pii/S2405896318326703`. 6th IFAC Conference on Nonlinear Model Predictive Control NMPC 2018.

[137] John P. Alsterda, Matthew Brown, and J. Christian Gerdes. Contingency model predictive control for automated vehicles. In *2019 American Control Conference (ACC)*, pages 717–722, 2019. doi: 10.23919/ACC.2019.8815260.

[138] John P Alsterda and J Christian Gerdes. Contingency model predictive control for linear time-varying systems. *arXiv preprint arXiv:2102.12045*, 2021.

[139] Siddharth H Nair, Vijay Govindarajan, Theresa Lin, Chris Meissen, H Eric Tseng, and Francesco Borrelli. Stochastic mpc with multi-modal predictions for traffic intersections. *arXiv preprint arXiv:2109.09792*, 2021.

[140] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[141] Joel AE Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1): 1–36, 2019.

[142] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106(1):25–57, 2006.

[143] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 86–99. PMLR, 30 Oct–01 Nov 2020.

[144] Liting Sun, Wei Zhan, Masayoshi Tomizuka, and Anca D Dragan. Courteous autonomous cars. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 663–670. IEEE, 2018.

[145] T. Baltic, R. Hensley, and J. Salazar. The trends transforming mobility's future. *The McKinsey Quarterly*, 2019.

[146] A. Schwarzenegger. *California Driver Handbook*. Department of Motor Vehicles, 2007.

[147] Gonçalo Collares Pereira, Pedro F. Lima, Bo Wahlberg, Henrik Pettersson, and Jonas Mårtensson. Nonlinear curvature modeling for mpc of autonomous vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–7, 2020. doi: 10.1109/ITSC45102.2020.9294692.

[148] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff. Ensuring drivability of planned motions using formal methods. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–8, Oct 2017. doi: 10.1109/ITSC.2017.8317647.

[149] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019.