Simultaneous Local Motion Planning and Control, Adjustable Driving Behavior, and Obstacle Representation for Autonomous Driving

by

Mohamed Ashraf Gamaleldin Daoud

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical and Mechatronics Engineering

Waterloo, Ontario, Canada, 2020

Authors Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

Chapter 1, 2, 3, 5 and 7 were authored solely by the author of the thesis. The co-authors offered supervision and revised some of the contents included in Chapter 6.

The model predictive control proposed in Chapter 4 was implemented by the author of the thesis. The vehicle and energy consumption modeling as well as the problem formulation of the model predictive control were conceived by the author. Furthermore, the author designed different simulation scenarios used for validating the proposed controller. Finally, the author of the thesis wrote the first draft of the chapter, which was then revised by his co-authors.

Abstract

The evolving autonomous driving technology has been attracting significant research efforts in both academia and industry because of its promising potentials. Eliminating the human intervention in driving will drastically improve the road safety and will create more mobility freedom for the humankind. Decision-making system in autonomy pipeline is the last module that interacts directly with the surrounding environment. Typical decision-making systems perform a variety of tasks including local motion planning, obstacle avoidance, and path-following in a sequential manner. An alternative approach is to perform these tasks simultaneously to obtain faster decision-making actions. This thesis focuses on designing an optimization-based simultaneous controller that performs obstacle avoidance, local motion planning, and vehicle control on roads regardless of their orientation while following a target path, and also incorporates adjustable driving behavior.

Firstly, a decision-making scheme that enables autonomous driving for long trips while expanding the usage of the available computational resources and ensuring obstacle avoidance functionality is proposed. The proposed scheme utilizes a parallel architecture for local motion planning and control layers to increase time efficiency. In addition, a novel feasibility-guaranteed lane change and double lane change planners are introduced for path planning and obstacle avoidance. Finally, an online parameterized curve generator is proposed and integrated with a recently developed path-following controller.

Next, a nonlinear model predictive control (NMPC) scheme is developed for the path-following control of autonomous vehicles. In addition, a dual-objective cost function which is composed of a regulation part and an economic part is introduced. By tuning the weights of this cost, a driving behavior can be implemented; two different driving behaviors are designed, namely, energy-efficient and sport driving modes. Finally, a kinematic bicycle model is used for predicting the vehicle motion while a longitudinal motion dynamic model is used for estimating the energy consumption.

Finally, a novel representation framework for static maps and obstacles based on Fourier Series is proposed. The framework relies on Complex Fourier Series analysis to reduce the computation time and outputs mathematical equations to describe the shape of the considered object. Furthermore, two methods were proposed, namely; offline method and online method. The offline method is used to create accurate representations for static maps and most common obstacles an ego vehicle may encounter. The online method is used to model the free space around the vehicle when dealing with uncertain environments.

The proposed contributions fill significant gaps in the autonomous driving problem. All the proposed work is tested and validated using numerical simulations and some experiments. The results show the effectiveness of the proposed contributions.

Acknowledgements

In the name of Allah, the most gracious and the most merciful. First and foremost praise is to Allah, the Almighty, the greatest of all, on whom ultimately we depend for sustenance and guidance. I thank the Almighty Allah for giving me the determination and the strength to conduct my research. His continuous grace and mercy was with me throughout my life from birth to date and ever more during the journey of my research.

I would like to express my sincere gratitude to Prof. William Melek and Prof. Derek Rayside, for the promising opportunity they provided me with, for their continuous support of my M.A.Sc. study and related research, for their patience, motivation, and immense knowledge. Their guidance and feedback helped me all the time of research and writing of this thesis. Many thanks to Prof. Christopher Nielsen and Prof. Baris Fidan for their thoughtful comments and recommendations to make this thesis better.

I would also like to thank WATonomous design team for the stimulating discussions, and their help throughout my research. In this team, I have worked with many motivated members who are always keen to learn.

I would also like to thank my friends who have supported me during my program. I would like to specially thank Ahmed Eid, Mohamed Alaa, Mahmoud Hanafy, Islam Moheb, Mostafa Osman, Mohamed Alshorman, Dr. Mohamed Mehrez, Dr. Gebreel Abdulrahman, and Dr. Ahmed El-Awady. Their existence in my life made me always feel as if I am home with my family.

Finally, I would like to thank my parents Ashraf and Sahar, my sisters Radwa and Sarah, and my wife Hanan.

Dedication

This is dedicated to my caring sisters Radwa and Sarah, my lovely supporting parents Ashraf and Sahar, my wife and my backbone Hanan.

Table of Contents

List of Figures x

List of Tables xiii

Abbreviations xiv

List of Symbols xv

1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem Statement . . . 2
1.3 Contributions . . . 3
1.3.1 Simultaneous Local Motion Planning and Control . . . 3
1.3.2 Adjustable Driving Behavior using Dual-Objective NMPC . . . 3
1.3.3 Obstacle Representation using Fourier Series . . . 4
1.3.4 Benchmarking Proposed Schemes against Existing Controllers . . . 4
1.3.5 Validation of the Proposed Methodologies . . . 4
1.4 Thesis Structure and Organization . . . 5

2 Background, Preliminaries and Related Work 6

2.1 Automated Driving Standards . . . 6
2.2 Optimal Control and MPC . . . 7
2.3 Vehicle Modeling . . . 8
2.3.1 Vehicle Kinematic Model . . . 8
2.3.2 Vehicle Dynamic Model . . . 8
2.3.3 Vehicle Models Comparison . . . 10
2.4 Decision-Making System . . . 10
2.4.1 Global Motion Planners . . . 10

2.4.2 Behavioral Planners . . . 11
2.4.3 Local Motion Planners . . . 11
2.4.4 Feedback Control . . . 13
2.5 Recent Related Work . . . 14
2.5.1 Obstacle Avoidance using Collision-Free Trajectories . . . 14
2.5.2 Obstacle Avoidance using Hard and Soft Constraints . . . 15
2.5.3 Model Predictive Path-Following Control . . . 16
2.5.4 Case Study: Nonlinear MPC controller on Chevy Bolt . . . 16
2.5.5 Static Map and Obstacle Representation . . . 17
2.5.6 Conclusion . . . 18

3 Simultaneous Local Motion Planning and Control 19
3.1 Introduction . . . 19
3.2 Global Planning and Behavioral Planning . . . 19
3.3 Offline Route Feasibility Checker . . . 21
3.3.1 Waypoints Refinement . . . 21
3.3.2 Feasibility-Guaranteed Lane Change Planner . . . 21
3.3.3 Curvature Calculation and Velocity Profile Refinement . . . 26
3.4 Online Parameterized Curves Generator . . . 27
3.4.1 Vehicle Positioning Relative to Global Route and Dynamic Lookahead Distance . . . 27
3.4.2 Curve Fitting Problem Formulation . . . 28
3.4.3 Feasibility-Guaranteed Double Lane Change Planner . . . 30
3.5 Model Predictive Path-Following Control . . . 33
3.5.1 MPFC Scheme . . . 33
3.5.2 Vehicle Model and OCP Formulation and Discretization . . . 34

4 Path-following and Adjustable Driving Behavior of Autonomous Vehicles using Dual-Objective Nonlinear MPC 37
4.1 Introduction . . . 37
4.2 Bicycle and Energy Consumption Models . . . 38
4.2.1 Bicycle Model . . . 38
4.2.2 Energy Consumption Model . . . 39
4.3 Proposed Dual-Objective NMPC . . . 39
4.3.1 Path-following Control Problem . . . 39
4.3.2 NMPC Formulation . . . 40
4.3.3 Dual-Objective Cost Function . . . 41

5 Obstacle Representation using Fourier Series **42**
5.1 Introduction . . . 42
5.2 Complex Fourier Series Expansion . . . 42
5.3 Obstacle Representation . . . 44
5.3.1 Offline Method . . . 45
5.3.2 Online Method . . . 45

6 Simulation Results and Benchmarking **48**
6.1 Introduction . . . 48
6.2 Simultaneous Local Motion Planning and Control Results . . . 48
6.2.1 Offline Route Feasibility Checker Results . . . 49
6.2.2 Online Curve Generator Results . . . 50
6.2.3 Model Predictive Path-following Control Results . . . 52
6.3 Adjustable Driving Behavior Control Results . . . 55
6.3.1 Dynamic Simulation Scenarios . . . 55
6.3.2 Real-time Simulation Results and Discussion . . . 56
6.4 Obstacle Representation using FS Results . . . 62
6.4.1 Obstacle Representation . . . 62
6.4.2 Static Map Representation . . . 64

7 Conclusion and Future Work **66**
7.1 Simultaneous Local Motion Planning and Control . . . 66
7.2 Adjustable Driving Behavior . . . 66
7.3 Obstacle Representation using Fourier Series . . . 67
7.4 Possible Future Work . . . 67

References **68**

List of Figures

2.1 An illustration of how MPC finds control actions over a predefined prediction horizon and predicts the future state of the system. . . . 7
2.2 Kinematic bicycle model schematic. . . . 8
2.3 Dynamic bicycle model schematic. . . . 9
2.4 A simple graph showing vertices and edges representing a road network. . . . 10
2.5 Left: An example of FSM. Right: An example of RL. . . . 12
2.6 Tire friction ellipse and comfort rectangle constraints. . . . 13
2.7 Combined feedback and feedforward controller block diagram. . . . 14
2.8 An illustration of control actions with and without interpolation. . . . 17
2.9 Experimental results from test track 17

3.1 Architecture of the proposed approach. . . . 20
3.2 The adopted FSM behavioral planner. . . . 21
3.3 Three different sigmoid functions. . . . 23
3.4 Effect of B and a on $\kappa_{max,LC}$. . . . 24
3.5 Left: first method, values of a and $\kappa_{max,LC}$ for $B = B_{lat,LC}$ and the fitted equation. Right: second method, values of $\kappa_{max,LC}$ for different values of a and B and the fitted two dimensional equation. . . . 25
3.6 Vehicle positioning relative to the global route. . . . 27
3.7 Overlapping in parametric curves generation. . . . 30
3.8 Three different bell-shaped functions. . . . 30
3.9 Left: first method, values of m and $\kappa_{max,DLC}$ for $H = H_{lat,DLC}$ and the fitted equation. Right: second method, values of $\kappa_{max,DLC}$ for different values of m and H and the fitted two dimensional equation. . . . 31
3.10 An illustration of finding the optimal H value. Note that the optimal H is not at the center of the left lane. . . . 32

4.1 Schematic diagram of the bicycle model, where I is the inertial frame and V is the vehicle local frame. . . . 38
4.2 A schematic of the path-following problem. . . . 40

4.3 Effect of changing E in the dual-objective function. 41

5.1 The Argand diagram for a complex function $f(t)$. 43
5.2 An obstacle and its sampled boundary. 44
5.3 An example of the offline method used to generate obstacle representation. 46
5.4 An example of the online method used to generate free space representation. 46

6.1 Left: Snapshots for the three LC maneuvers from CARLA, where $\mathcal{R}_{global}$ is in blue and $\mathcal{R}_{global,ref}$ is in red. Right: Plots showing longitudinal and lateral distances for each maneuver. 49
6.2 Road curvature information for each scenario. 50
6.3 Sample of curve fitting results and waypoints for the three scenarios. Right: Case 1. Middle: Case 2. Left: Case 3. 51
6.4 Original parametric curve $\mathcal{P}$ and the traversed parametric curve $\mathcal{P}_{clear}$. 53
6.5 Reference route $\mathcal{R}_{global,ref}$ and the traversed route for each case. 54
6.6 Relation between desired speed, MPC speed, and actual speed. 54
6.7 Left: Path-following error for each case. Right: velocity tracking output for each case. . 55
6.8 The developed ROS package block diagram. 56
6.9 Ackermann vehicle model 56
6.10 Different test cases used for path-following. 57
6.11 Path-following simulation results for the straight line reference path. Left: Energy-efficient mode. Right: Sport mode. 58
6.12 The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode. 58
6.13 Consumed energy (Joules) by the vehicle simulation results. Left: Energy-efficient mode. Right: Sport mode. 59
6.14 Path-following simulation results for the spline reference path. Left: Energy-efficient mode. Right: Sport mode. 60
6.15 The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode. 60
6.16 Path-following simulation results for the circular reference path. Left: Energy-efficient mode. Right: Sport mode. 60
6.17 The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode. 61
6.18 Steering angle control action. Red lines represent the upper and lower limits. Left: Energy-efficient mode. Right: Sport mode. 61
6.19 Effect of the number of frequencies on the accuracy of the obtained representation. . . 62
6.20 Magnitude of every frequency for three different objects. 63

6.21 Obtained representation after filtering out small magnitude frequencies. 63
6.22 Obtained representation for CARLA Town01. 64
6.23 Obtained representation for five-way intersection from CARLA Town03. Left: representation without reverse sampling. Right representation with reverse sampling 65

List of Tables

3.1 A sample of maximum curvature values for different values of a and B. 23

6.1 Cases performed to test the proposed scheme. 49
6.2 LC maneuvers performed to test the proposed scheme. 50
6.3 Curve fitting computation time and error. 51
6.4 Constraints set on the optimization variables. 53
6.5 NMPC computation time and path-following error. 54
6.6 Simulated vehicle parameters 56
6.7 Parameters and weights used for each test case 57
6.8 Comparison between driving modes for straight line path 59
6.9 Comparison between driving modes for the spline path 59
6.10 Comparison between driving modes for circular path 61

Abbreviations

CFS Complex Fourier Series

DLC Double Lane Change

DoF Degree of Freedom

FS Fourier Series

FSM Finitie State Machine

LC Lane Change

LS Least Squares

MDP Markov Decision Process

MIMO Multiple Input Multiple Output

MPC Model Predictive Control

MPFC Model Predictive Path-Following Control

NHTSA National Highway Traffic Safety Administration

NLP Nonlinear Programming

NMPC Nonlinear Model Predictive Control

OCP Optimal Control Problem

ODE Ordinary Differential Equation

PID Proportional-Integral-Derivative

RL Reinforcement Learning

RMSE Root Mean Squared Error

SAE Society for Automotive Engineers

SISO Single Input Single Output

SLAM Simultaneous Localization and Mapping

List of Symbols

ICR Instantaneous center of rotation.

$\kappa_{max,dyn}$ Maximum curvature a vehicle can handle based on its dynamic constraints.

$\kappa_{max,kin}$ Maximum curvature a vehicle can handle based on its kinematic constraints.

$\kappa_{max,v}$ Maximum curvature a vehicle can handle.

$\kappa_{max,DLC}$ Maximum curvature of a double lane change maneuver.

$\kappa_{max,LC}$ Maximum curvature of a lane change maneuver.

κ Curvature of the road at a certain point.

L Wheel base: distance between front axis and rear axis of the vehicle.

M_z Yaw moment acting on the vehicle.

R Turning radius of the vehicle.

a_n Fourier Series cosine coefficient.

ρ_a Air density.

α_f Front-tire slip angle.

α_r Rear-tire slip angle.

a_1 Distance between front axle and C.G.

b_n Fourier Series sine coefficient.

β_s Vehicle side-slip angle.

a_2 Distance between rear axle and C.G.

c_n Fourier series complex coefficient.

$C_{\alpha f}$ Front-tire cornering stiffness.

$C.G$ Center of gravity of the vehicle.

$C_{\alpha r}$ Rear-tire cornering stiffness.

κ_{route} Curvature information of the route.

δ_f Front wheel steering angle

$d_{long,DLC}$ Longitudinal distance needed for a double lane change maneuver.

C_d Drag coefficient of the vehicle.

$d_{LA,dyn}$ Dynamic lookahead distance.

A_f Frontal area of the vehicle.

F_x Longitudinal force acting on the vehicle.

F_y Lateral force acting on the vehicle.

$\mathcal{R}_{global,ref}$ The global route from a starting point to a given target destination after refinement.

$\mathcal{R}_{global}$ The global route from a starting point to a given target destination.

I_z Mass moment of inertia around z-axis.

a_{lat} Lateral acceleration of the vehicle.

δ Double lane change lateral shift.

$d_{long,LC}$ Longitudinal distance needed for a lane change maneuver.

a_{long} Longitudinal acceleration of the vehicle.

m_v Mass of the vehicle.

Ω_z Yaw rate of the vehicle.

$\mathcal{P}_{clear}$ Collision-free reference parametric curve.

$\mathcal{P}$ Reference parametric curve.

κ_{step} Path evolution step-size parameter.

ξ Parametric curve parameter.

ρ Relative longitudinal distance between ego vehicle and an obstacle.

C_R Rolling resistance of the vehicle.

θ Yaw angle of the vehicle.

v_x Velocity of the vehicle in x direction relative to the vehicle's frame.

v_y Velocity of the vehicle in y direction relative to the vehicle's frame.

v Velocity of the vehicle.

d_{wp} Linear distance between two successive waypoints.

w Track width: distance between the center line of each of the two wheels on the same axle of the vehicle.

Chapter 1

Introduction

1.1 Motivation

An autonomous vehicle, a.k.a. self-driving car, is a vehicle that can sense its environment and interpret it correctly in order to drive safely with little or no human input. Autonomous driving problem have enthused hundreds of researchers around the world in both academia and industry. The motivation behind building autonomous vehicles technology is to create a mode of mobility which can enhance safety and reduce accidents resulting from human error and traffic congestion.

In 2018, there were 37,473 recorded fatalities in motor vehicle traffic crashes in the United States alone, and 94% of them were assigned to driver error [1, 2]. Autonomous vehicles technology can potentially eliminate the driver error and, consequently, reduce the number of collisions and fatalities. In addition, autonomous vehicles are supposed to follow driving rules of the road, which will result in safer traffic environments for all road users. Moreover, autonomous driving will provide mobility freedom and support for people with varying degrees of disabilities by making them more independent and productive. From economic aspect, autonomous driving will allow passengers to make the most of their daily commute time. In 2016, there were 12.6 million Canadians who commuted to work by car, spending an average of 24 minutes on the trip, and almost 7% of them spent at least 60 minutes traveling to work [3]. Therefore, autonomous driving can help our workforce make better use of their commute time, and increase their productivity.

Although autonomous driving concept has been around since 1920s, the first autonomous vehicle was developed in 1980s. In 1986, Carnegie Mellon University developed the first autonomous vehicle with people riding onboard. Despite its low speed, which was limited because of low computational power available at that time, it was revolutionary, and inspired many researchers to pay closer attention to the technology. Later on, in 1995, a notable achievement was driving for 5000 km across the United states with 98% of the distance driven autonomously [4]. In 2004 and 2005, DARPA held the DARPA Grand Challenges, which pushed autonomous vehicles technology into new territories. In 2005, four vehicles were able to drive autonomously without human intervention for 212 km on off-road course [5]. This milestone, accompanied by developments in sensing and computing technology, paved the way for more and more progress in autonomous vehicles technology.

Generally, perception, localization and decision-making systems are the basis of autonomous driving research. In perception, the aim is to utilize different sensors in capturing and interpreting data from the environment to extract useful information about the surrounding objects [6]. Localization focuses on estimating the current vehicle state by fusing different sensory information. Finally, the decision-making system interacts directly with the environment to safely drive the vehicle from the starting point to the target destination. Decision-making system can be divided into four subsystems: route planning, behavioral planning, local motion planning, and feedback control [7].

In a decision-making system, a global route planning algorithm tries to find the best route through the road network from the vehicle current position to the destination. Behavioral planning is responsible for the qualitative driving actions that are dependent on the surroundings and other road users. For instance, when the car is approaching a stop line, the behavioral planning subsystem shall command the vehicle to stop. Given the global route and any required qualitative driving actions, the local motion planning layer translates the required maneuvers into feasible trajectories or paths that can be followed by the vehicle. Finally, feedback control layer is used to interface with the actuators to closely follow the desired motion profile [7].

Although many partially autonomous vehicles platforms have been introduced for consumer use, the autonomous driving problem still needs more attention and investigation to make fully autonomous vehicles feasible in the near future. This is basically because of the amount of computational power required to drive the vehicle as well as deploy localization and decision-making algorithms which can allow the vehicle to operate safely and effectively in unstructured environments. Control and decision-making is the last layer in any pipeline of autonomous driving that interacts directly with the surrounding environment. Most of the current decision-making in the literature and associated control schemes are sequential in a way that every layer depends on the previous one until a new control action is generated. Although this sequential hierarchy may make the design of each layer easier, it may introduce time delay limiting the speed of the decision-making scheme. Therefore, a more time-efficient and parallel architecture scheme may reduce the decision-making time while achieving efficient maneuvering performance in real time.

1.2 Problem Statement

Autonomous vehicles typically perform a variety of tasks as part of their decision-making, including local motion planning, obstacle avoidance, and path-following. These tasks are typically implemented as separate modules running sequentially in the decision-making pipeline of the vehicle. An alternative design approach is to integrate the computations of some of these tasks together, while running others in parallel. Such a design has the potential to make faster decisions for three reasons: first, it reduces the number of sequential layers and the volume of data that needs to be passed through the processing pipeline of the vehicle, thus reducing the time latency for updating the control actions; second, parallelism expands the usage of the available computational resources compared to the modular sequential pipeline design; and third, such a design might minimize the computation power needed since some of these tasks are closely interrelated, so there is potential for one algorithm to solve several of them simultaneously.

A typical drawback of parallel-integrated design is that it is often more difficult to design as compared to the sequential technique. That is because of the complexity and uncertainty of each task. Prior work on integrated controllers has been limited to lateral control on straight roads while avoiding obstacles [8]. This work advances the state of the art in parallel-integrated controller design by performing both longitudinal and lateral vehicle control on roads regardless of their curvature and orientation while following a target path, and also incorporates adjustable driving behavior. For instance, an autonomous vehicle might be operated in a more energy-efficient manner or a sportier manner.

1.3 Contributions

1.3.1 Simultaneous Local Motion Planning and Control

Most of the existing global planners tackle the problem of finding the best route from the starting point to a given target destination while abstracting away some of the lower details of the roads such as dynamic obstacles. This approach is acceptable as it helps in reducing the complexity of finding the optimal route.

Therefore, to execute a global route safely, we divide it into smaller sections, incorporate behavioral planning qualitative decisions, and apply any modifications needed to make route calculation within the confines of the vehicle's constraints. These modifications include smoothing Lane Change (LC) and Double Lane Change (DLC) maneuvers and making them feasibility-guaranteed. After that, the generated local motion profile is represented by parametric curves. Such a representation can be executed safely by most feedback controllers. Thus, our novel proposed methodology stands out versus others existing in the literature as it:

- relies on pre-processing the global route to minimize online computation time.
- dynamically changes lookahead distance based on road shape.
- efficiently incorporates feasibility-guaranteed DLC maneuvering without the need of re-planning.
- is parameterized and can be tuned based on the operating conditions.

We finally integrate the proposed methodology with path-following NMPC scheme proposed in [9] with the required adjustments to make it suitable for autonomous vehicles. The proposed scheme relies on integrated-parallel design and we call it Simultaneous Location Motion Planning and Control.

1.3.2 Adjustable Driving Behavior using Dual-Objective NMPC

Driving behavior of vehicles is a main concern in automotive industry and road traffic safety because of its effects on the environment in terms of energy consumption as well as the safety of other road users [10, 11]. Therefore, driving behavior heavily affects energy

consumption of cars. However, there is a trade off between energy saving and performance. Although autonomous vehicles are expected to provide more efficient rides, passengers may opt for faster rides. We propose using a dual-objective NMPC scheme that changes the driving behavior of autonomous vehicles based on the passengers' preference.

1.3.3 Obstacle Representation using Fourier Series

Modeling the static map and the obstacles within the ego vehicle environment is crucial to ensure that the obtained motion plans are safe and collision-free. Since collision checking is performed online, it should be reliable, accurate and computationally efficient. The majority of the current approaches for obstacle representation use conservative approximations to ensure the safety of the obtained plans. However, since obstacles usually have irregular shapes, the approximate representation introduced in the literature may not be accurate enough. Instead, we propose using Fourier Series (FS) to represent static obstacles in the scene, as FS is capable of translating any closed shape into one mathematical equation. Similarly, road area can be modeled as constraints, which can be incorporated into the controller design as soft or hard constraints. Such a technique may help in guiding the vehicle through an optimal collision-free path. Generating static obstacles representation using FS can be done offline, hence, the computational cost is eliminated.

1.3.4 Benchmarking Proposed Schemes against Existing Controllers

To evaluate the performance of different proposed methodologies, we benchmarked their performance against different existing algorithms available in the literature [12–17]. Such comparison validates the efficacy of the proposed work and help identify areas for potential improvement in future investigations.

1.3.5 Validation of the Proposed Methodologies

All of the proposed methodologies have been tested using real-time simulations using CARLA Open-source simulator [18] and Gazebo Dynamic Simulator [19]. The simulations performed cover various road shapes and driving scenarios to show the effectiveness of the proposed schemes.

Some validation with an autonomous Chevy Bolt, belonging to the Watonomous team, has been done. An earlier version of the controller presented in Chapter 3 has been in use on the vehicle for almost a year. This earlier version is described in more details in Chapter 2. A more recent version of the controller has been integrated with the overall pipeline of the vehicle, and this integration has been tested in simulation but not on the road. Due to Covid-19 pandemic, experimental validation had to be suspended in order to abide to the emergency orders by Ontario public health which required campus closure.

1.4 Thesis Structure and Organization

This thesis is organized as follows:

- **Chapter 1:** In this chapter, we provide the motivation for this research thesis. Next, we define our problem statement precisely. Finally, we present the proposed contributions of our research.
- **Chapter 2:** A background, preliminaries and most recent related work are reviewed in this chapter.
- **Chapter 3:** Demonstrates and formulates our simultaneous local motion planning and control scheme.
- **Chapter 4:** Introduces the proposed dual-objective NMPC scheme for driving behavior change.
- **Chapter 5:** Presents the proposed Fourier Series methodology for obstacle representation.
- **Chapter 6:** Details the validation of our approach using numerical simulations, benchmarks the proposed scheme against other algorithms in the literature, and discusses the results.
- **Chapter 7:** Finally, this chapter concludes our work and outlines possible future work.

Chapter 2

Background, Preliminaries and Related Work

In this chapter, we start by discussing the most common automated driving standards. Next we give a brief on optimal control and Model Predictive Control (MPC). Then we discuss different used vehicle models. After that, we give more details on decision-making systems. Finally, we outline some of the most recent related work on path-following using MPC in the literature.

2.1 Automated Driving Standards

Different institutions around the world started to create standards to categorize levels of automated driving, such as German Federal Highway Institute (BASt) [20], the National Highway Traffic Safety Administration (NHTSA) [21], and the Society for Automotive Engineers (SAE) [22]. The most widely used one is the SAE J3016 standard, which categorizes levels of autonomy into six different levels ranging from no driving automation (level 0) to full driving automation (level 5). In level 0, the driver is responsible for all vehicle motion control even when active safety systems are engaged. In level 1 (Driver Assistance), driving automation system is responsible for either the lateral or the longitudinal vehicle motion control (e.g. lane change or cruise control). In level 2 (Partial Driving Automation), both lateral and longitudinal vehicle motion are executed by the driving automation system, but objects detection and event response task is performed by the driver as well as supervising the driving automation system. In level 3 (Conditional Driving Automation), the automated driving system is responsible for the entire dynamic tasks but the driver must be ready to intervene when necessary. In level 4 (High Driving Automation), in addition to all dynamic driving tasks, the system is responsible for bringing the vehicle to safe status in case of fallback until the user intervenes. In level 5 (Full Driving Automation), the system will be able to do all tasks with zero human intervention. Current developed autonomous driving systems lay between level 3 and level 4.

2.2 Optimal Control and MPC

In order to represent dynamic systems and processes mathematically, we use differential equations or difference equations to create such models. These models can then help in designing the best control law either using classical control or modern control. Classical control theory focuses on Single Input Single Output (SISO) systems and is based on Laplace representation of the systems. In addition, the control action, which is the input to the system, is determined based on the error signal as the controller tries to drive it to zero. On the other hand, modern control theory deals with Multiple Input Multiple Output (MIMO) systems. In addition, modern control theory uses state space representation such that the whole system is described using a set of first order differential equation in case of continuous time domain, and difference equations in case of discrete time domain.

Optimal control lies under modern control techniques, and its main target is to minimize or maximize certain performance index while satisfying system constraints. This is done by finding the optimal control actions that will drive the system to a final target state. Therefore, an Optimal Control Problem (OCP) formulation requires a mathematical model of the system, a performance index that reflects the desired behavior of the system, and a set of constraints on states and control actions. For more information on optimal control theory, we refer the reader to [23, 24].

MPC is a finite-horizon optimal control method at which the control actions are calculated by iteratively solving an open loop OCP over a predefined horizon in future. The OCP is formulated given a performance index, which characterizes the control objective. The main challenge in MPC is its computational cost, which usually limits its running frequency. In MPC the target is to usually minimize a performance index, and hence, we call it *cost function* instead. Applied control actions are obtained by taking only the first element of an optimal control sequence, which results from solving the OCP [25]. Solving the OCP is done repetitively on a finite prediction horizon using the actual state of the system as a starting point, and hence incorporates a feedback mechanism. This feedback is essential in handling different disturbances acting on the system as well as inaccuracy of system model used in prediction [26]. MPC has been used in autonomous vehicles [27], holonomic mobile robots [28] and non-holonomic mobile robots [29]. Fig. 2.1 shows an illustration of MPC controller and how it drives dynamic systems to the desired state.

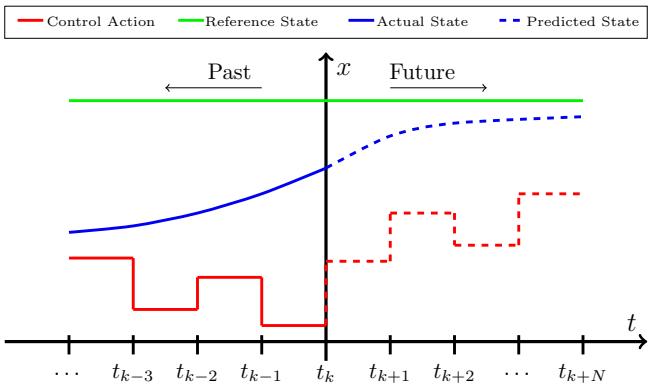


Figure 2.1: An illustration of how MPC finds control actions over a predefined prediction horizon and predicts the future state of the system.

2.3 Vehicle Modeling

Generally, two techniques are used for vehicle modeling. The first one, called kinematic modeling, uses geometric constraints to define the motion of the vehicle. The second one, called dynamic modeling, uses all of the forces and moments acting on the vehicle to define its motion. We will outline both ways in the next subsections.

2.3.1 Vehicle Kinematic Model

For any mobile robot, a kinematic model can be derived by enforcing the constraints of the robot's wheels. In case of vehicles, the front wheels can be combined into one virtual central front wheel. Similarly, rear wheels can be combined into a wheel at the middle. This approximation is valid since $R >> L$ and $R >> w$, where R is the vehicle turning radius, L is the wheelbase, and w is the track width. Now, the motion of the model is governed by the front steerable wheel and the rear fixed wheel. In the kinematic model, we assume that the wheel plane remains vertical and there is only one contact point between the wheel and the ground. In addition, we assume that there is no sliding at the contact patch, and the wheel motion is only because of its rolling. These assumptions enable us to set two constraints on each wheel of the vehicle. Firstly, the wheel must roll to generate motion in longitudinal direction. Secondly, the wheel must not slip laterally. These constraints are also known as vehicle nonholonomic constraints. For more information on wheel constraints, we refer the reader to [30]. Fig. 2.2 shows a schematic of the kinematic bicycle model, where ICR is the instantaneous center of rotation, v is the vehicle speed, δ_f is the front wheel steering angle, θ is the heading angle of the vehicle and β_s is the side slip angle of the vehicle.

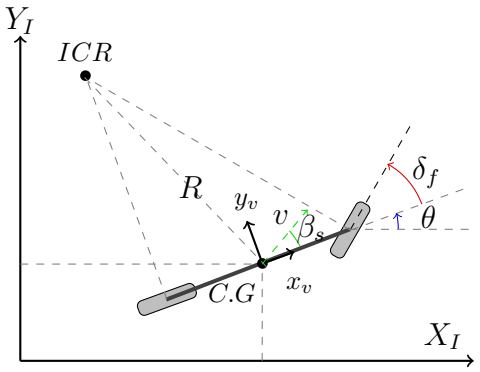


Figure 2.2: Kinematic bicycle model schematic.

Two-wheeled vehicle kinematic model can be derived relative to several reference points. Here, we use the vehicle's $C.G$ as a reference and the vehicle frame is attached to it as shown in Fig. 2.2. Therefore the vehicle kinematic model becomes:

$$\begin{bmatrix} \dot{x}_I \\ \dot{y}_I \\ \dot{\theta}_I \end{bmatrix} = \begin{bmatrix} v\cos(\theta+\beta_s) \\ v\sin(\theta+\beta_s) \\ \frac{v\cos(\beta_s)\tan(\theta)}{L} \end{bmatrix} \tag{2.1}$$

and

$$\beta_s = \tan^{-1}\left(\frac{l_r \tan\delta_f}{L}\right) \tag{2.2}$$

where $\dot{x}_I$ and $\dot{y}_I$ are vehicle velocity in the inertial frame and $\dot{\theta}_I$ is the yaw rate.

2.3.2 Vehicle Dynamic Model

As the vehicle speed increases, effect of dynamics starts to become more significant on the vehicle behavior, and a more accurate vehicle model is thus needed. Since the purpose

here of vehicle modeling is to find the optimal control inputs, there is a trade-off between model accuracy and optimization complexity.

Generally, a high-fidelity model is not required for optimization. The most common dynamic vehicle model used for model-based control combines wheels on each axle into one central wheel similar to the kinematic model introduced above. In addition, it neglects the Degree of Freedom (DoF) of the car suspension system, and hence, the dynamic model has only three DoFs. However, as it incorporates vehicle dynamic effect, the model violates the kinematic constraints, resulting in a more realistic behavior. The relaxation of kinematic constraints means that each wheel may have a lateral velocity component, e.g. $v_{wheel,y} \neq 0$, and the longitudinal motion is due to both rolling and slipping of the tire. Fig. 2.3 shows a schematic of the dynamic bicycle model and forces acting on each wheel, where a_1 is the distance between the front axle and the C.G., a_2 is the distance between the rear axle and the C.G., α_f is the front-tire slip angle, α_r is the rear-tire slip angle and Ω_z is the yaw rate.

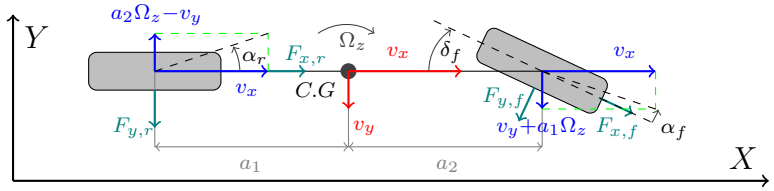


Figure 2.3: Dynamic bicycle model schematic.

Interaction between the road plane and tires generates a 3D force system that includes three forces and three moments. However, we focus only on longitudinal force F_x, lateral force F_y, and yaw moment M_z to obtain the dynamic model:

$$\dot{v}_x = \frac{1}{m_v}(F_{xf} + F_{xr}) + \Omega v_y \tag{2.3a}$$

$$\dot{v}_y = -\frac{C_{\alpha f} + C_{\alpha r}}{m_v v_x} v_y + \left[\frac{-a_1 C_{\alpha f} + a_2 C_{\alpha r}}{m_v v_x} - v_x\right]\Omega_z + \frac{C_{\alpha r}}{m}\delta_f \tag{2.3b}$$

$$\dot{\Omega}_z = \left[\frac{-{a_1}^2 C_{\alpha f} + {a_2}^2 C_{\alpha r}}{I_z v_x}\right]\Omega_z + \left[\frac{-a_1 C_{\alpha f} - a_2 C_{\alpha r}}{I_z v_x}\right] v_y + \frac{a_1 C_{\alpha f}}{I_z}\delta_f \tag{2.3c}$$

where $C_{\alpha f}$ and $C_{\alpha r}$ are the front-tire and the rear-tire cornering stiffness respectively, m_v is the mass of the vehicle and I_z is the moment of inertia around the Z-axis of the vehicle. The inputs for the dynamic model are the steering angle δ_f and driving force F_x coming from the engine torque. By setting the longitudinal speed v_x to a constant, the dynamic model becomes linear, and therefore reduces the computational power needed for optimization. Consequently, assuming constant speed of the vehicle lets us control vehicle lateral motion using only the steering angle. For more information on tire forces and the complete derivation of the model, we refer the reader to [31] or any other vehicle dynamics textbook.

2.3.3 Vehicle Models Comparison

While dynamic vehicle model is able to capture the vehicle actual performance more accurately, using it for optimization in Nonlinear Model Predictive Control (NMPC) is computationally expensive as it is highly nonlinear [32], and requires double integration to get position predictions. Therefore, one way to incorporate it is by decoupling longitudinal and lateral dynamics from each other and designing a separate control for each of them. However, their behavior is tightly coupled.

On the other hand, kinematic model has less nonlinearity, and can be integrated once to get position prediction. The kinematic model can be used for wide range of speeds with relatively high accuracy [32]. However, it fails in predicting the vehicle states accurately in case of high speeds.

Another major drawback of the dynamic model is that its parameters change based on several factors such as tire pressure and ambient temperature [33]. This issue may drift the prediction away from the actual behavior of the vehicle in case of huge errors between model parameters and actual parameters.

In conclusion, kinematic model shows good performance for normal urban driving cases. Meanwhile, dynamic model gives more accurate results in highway driving, but it needs iterative estimation of tire parameters.

2.4 Decision-Making System

2.4.1 Global Motion Planners

With high level of abstraction, a decision-making system shall find the optimal route from a starting location to a given destination by navigating the road network. Usually optimal route refers to the route with the shortest amount of time or distance it takes for the vehicle to reach its given destination [34]. Since the spatial planning scale is in kilometers, neglecting low-level planning constraints is acceptable. Therefore, the main constraints induced on the planning task are speed limits, traffic flow, and road lengths.

Taking the advantage of the road network being highly structured, it can be represented using a mathematical structure known as a graph [35]. A graph is a discrete structure composed of vertices and edges where each vertex represents a point on the road network, and each edge represents a road segment connecting any two points on the road network as shown in Fig. 2.4. The cost of traversing each edge can be modeled by assigning weights to this edge. Therefore, finding the optimal route, or sets of edges that connect the starting location to a given destination, can be done by searching the graph for the minimum cost route.

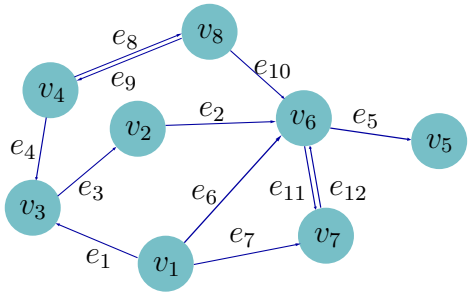


Figure 2.4: A simple graph showing vertices and edges representing a road network.

Various algorithms can be used to find the optimal path such as the well-known Dijkstra [36] or A* [37] algorithms, but as the search space grows and covers millions of edge,

it becomes impractical to use them [7]. Global planning problem has attracted significant interest in the recent years, and many innovative solutions have been introduced to find the optimal route in milliseconds on continent scale road network [38] [39]. For more information, we refer the reader to [34] for a comprehensive survey of practical global planning algorithms.

2.4.2 Behavioral Planners

Behavioral planning facilitates the vehicle to navigate through the global route and to interact safely with other road users. This is done by selecting the proper driving action or maneuver considering the rules of the road, signals from infrastructure, and both static and dynamic obstacles around the vehicle [40]. For example, when the vehicle is approaching a signed intersection, if the traffic light is red, the behavioral planning layer will command the vehicle to stop until the traffic light becomes green and the intersection area is safe to proceed. Therefore, the output of the behavioral planning layer is qualitative actions that decide the driving action that the vehicle shall execute.

Since most of the scenarios that a vehicle may encounter are repetitive, e.g. all-way stop intersections, roundabouts, approaching a leading vehicle, etc, these scenarios can be modeled as a Finitie State Machine (FSM). The transitions between states are governed based on the perceived environment of the vehicle. For instance, most of the teams in the DARPA Urban Challenge have used FSM coupled with different heuristics to solve behavioral planning problem [41]. However, one drawback of using FSM is that it may suffer from rules explosion as the number of modeled scenarios increases [42]. Consequently, this approach is only sufficient for limited operational design domains.

Another way to tackle behavioral planning problem is by Reinforcement Learning (RL) techniques such as Markov Decision Process (MDP). In RL, an agent learns how to interact properly with a given environment by taking actions and receiving a continuous reward. The target for the agent is to maximize the reward by learning the optimal policy that generates the most optimal actions given a certain state of the environment. For example, the work in [43] uses MDP to formulate the behavioral planning problem. However, as RL techniques require intensive training, simple simulation environments are being used for training, which cannot ensure robustness in real-time utilization. Fig. 2.5 shows examples of behavioral planning systems. For more information on RL we refer the reader to [44].

2.4.3 Local Motion Planners

Given the global route to a given destination and the qualitative decisions obtained from the behavioral planning, local motion planning is responsible for generating a feasible, collision-free, and comfortable motion profile to be executed by the control layer.

If the planned motion consists of coupled information of location with respect to time, e.g. when to be where, the motion profile is called trajectory. However, the majority of local planners decompose motion planning task into path generation and velocity profile generation. Local planners can be divided based on how they generate the motion profile into three major categories: variational methods, graph-search methods, and incremental

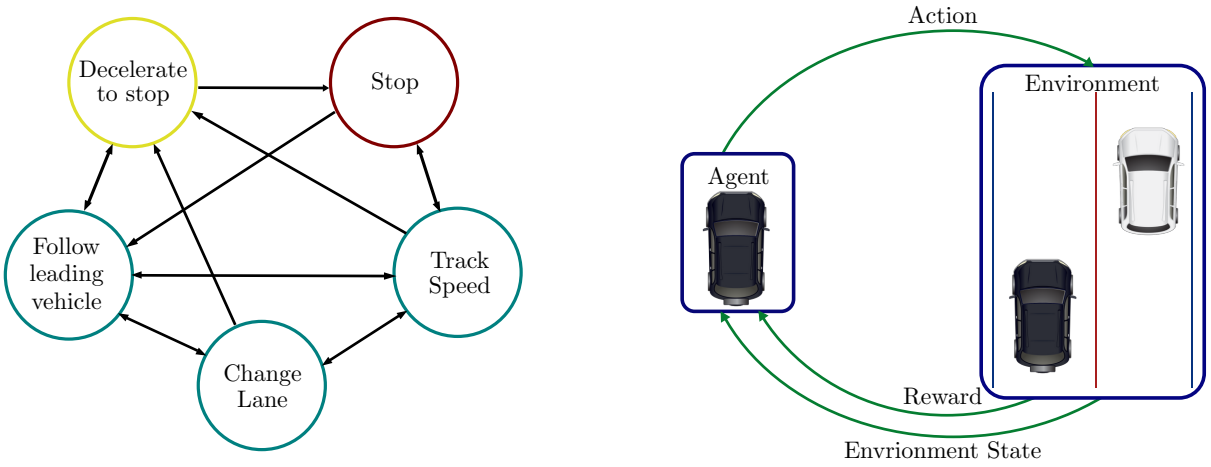


Figure 2.5: Left: An example of FSM. Right: An example of RL.

search methods. Here, we give a brief on each group of methods. For more details on these categories, we refer the reader to [7].

Variational methods

These methods use calculus of variations techniques to find the optimal motion profile. This can be done by representing the search space, obstacles, and vehicle kinematic constraints as an OCP. Although these methods can converge quickly to locally optimal solutions, their convergence to a feasible path is sensitive to the provided initial guess. For a dedicated survey on this topic we refer the reader to [45].

Graph-search methods

Graph-search methods start by discretizing the workspace of the vehicle as a graph, similar to the one shown in Fig. 2.4, where the vertices model a finite set of vehicle states and the edges represent needed control actions to achieve these states. Next, a graph-search algorithm can be used to find the minimum cost path. Although this approach is fast, the discretization of the search space limits the possible maneuvers that the vehicle can perform. An example of these techniques is PRM [46], and state lattice planning [47].

Incremental search methods

These methods randomly sample the control actions, and incrementally generate reachable paths (known as tree) for the vehicle to traverse. Once the tree reaches the target goal region, the search ends and the desired path is then traced and returned to the controller. This category is extremely fast compared to others. However, the generated paths generally have poorer quality. Some examples of these techniques are RRT [48], and SST [49].

The resulting motion profile can take generally two forms; parameterized curves or non-parametric paths. While non-parametric paths such as straight line segments and waypoints may be suitable for representing the global route, parameterized curves are more

preferred for local motion planning. This is because parameterized curves incorporate vehicle kinematic constraints that ensure path feasibility, and dynamic constraints that ensure passengers comfort.

A parametric curve can be described as a set of parameterized equations, e.g. two or more equations parameterized by the same arbitrary variable. In autonomous driving, two of the most used parametric curves are quintic splines [50] and cubic spirals [51]. This is because, they can guarantee that the path is continuous and differentiable.

Velocity profile generation

Velocity profile generation is often constrained to ensure its smoothness, safety and comfort. This is can be done by minimizing the jerk of the velocity profile, and by constraining the longitudinal and lateral accelerations to cope with maximum tire forces. In addition, tire limits are usually higher than the acceleration tolerable by passengers. Therefore, unless in urgent maneuvers, a comfort rectangle that represents comfortable accelerations limit can be imposed instead [52]. Fig. 2.6 shows tire friction ellipse and the comfort window limits.

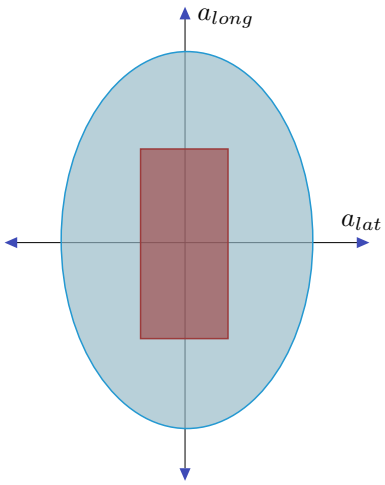


Figure 2.6: Tire friction ellipse and comfort rectangle constraints.

2.4.4 Feedback Control

To drive the vehicle on the desired reference motion profile, a feedback control layer is needed to interface with the actuators and regulate any tracking error. In general, it is likely for errors to be generated while execution because of disturbances acting on the vehicle as well as inaccuracies of the vehicle model used for prediction. Therefore, feedback control layer generates a control law that should be able to minimize any tracking error.

Some control techniques decouple longitudinal motion control and lateral motion control from each other to ease the design of a proper control law. For example, the dynamic system model described in equation (2.3) is often used for lateral control with the assumption of constant longitudinal speed such as in [53]. Longitudinal control, e.g. cruise control, usually use linear feedback control laws such as Proportional-Integral-Derivative (PID) control. Other techniques combine feedforward control with PID to improve tracking performance. In feedforward control law, system models are used to generate control actions. Combining both feedforward and feedback control helps in improving the overall performance [54]. Fig. 2.7 shows a block diagram for such a controller.

On the other hand, lateral control laws are usually nonlinear and can be divided into two main categories; geometric controllers and dynamic controllers. Geometric controllers are based on the geometry of the reference path and vehicle kinematic models. However, they do not consider time evolution of vehicle state, which make their performance limited to low speed driving. The most two well known geometric controllers are Stanley controller [55] and Pure Pursuit controller [56]. On the other side, dynamic controllers can work on minimizing current error as well as future predicted error. The most used dynamic

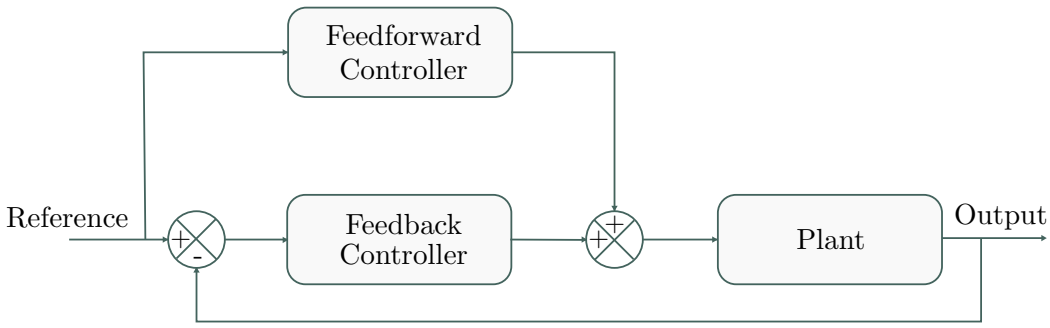


Figure 2.7: Combined feedback and feedforward controller block diagram.

controllers are MPC and sliding mode control [57]. One more point to add is that MPC can handle longitudinal and lateral control at the same time.

2.5 Recent Related Work

In this section we study recent work done related to local motion planning and control.

2.5.1 Obstacle Avoidance using Collision-Free Trajectories

In [15] a combined trajectory planning and tracking system is introduced. The system uses state lattice algorithm for trajectory planning. State lattice algorithm lies under graph-search methods group, and relies on discretizing the workspace into a set of target points. Next, feasible trajectories are generated to each target point, and the optimal trajectory is selected based on a certain performance index. The selected trajectory is then executed by a trajectory tracking MPC. This system works in a sequential way, and generates a new trajectory at every iteration making the tracking part fully dependent on the planning part. Moreover, to increase the speed of trajectory planning, the available target points are limited because of the discretization of the search space as mentioned before. Although the controller design is based on continuously linearizing a vehicle kinematic model similar to (2.1) to solve for optimal control actions, the average computation time was 0.04 s, which is relatively high. In addition, the system uses a constant, relatively short, lookahead distance of 20 m. Finally, as the proposed scheme works in a sequential way, synchronization with the rest of the pipeline cannot be made as time needed for solving trajectory planning and tracking problems may vary.

A vehicle obstacle avoidance trajectory planning and tracking method using optimal control techniques is introduced in [58]. To plan an LC maneuver, a safety constraint representing a safe longitudinal distance between the ego vehicle and the obstacle is set. For lateral acceleration profile, a three segment sinusoidal curve is used. The trajectory planning tries to find fastest LC maneuver that follows the aforementioned profile while maintaining certain constraints such as maximum lateral velocity and acceleration and the safe longitudinal distance. However, as the method does an LC maneuver when it counters an obstacle without returning back to the original lane, a global rerouting may be needed as the path of the vehicle is changed. In addition, this method does not consider a

global route while planning the maneuver. Finally, the proposed method assumes constant longitudinal speed, which is not necessary in case of LC maneuver.

An NMPC controller is presented in [13] to track a sigmoid reference path. The controller approximates the reference path by tracking the yaw angle and lateral displacement while adopting tire dynamics. Tire dynamics are captured by using Pacejka tire model [59], which is a semi-empirical model based on experimental data. The proposed system generates the reference path needed for LC maneuver based on the desired longitudinal and lateral distances. Although using a sigmoid function for LC maneuver gives the user the ability to tune longitudinal and lateral distance needed, the proposed approach does not incorporate having a global route and relies on updating the desired longitudinal distance. In addition, it assumes only straight roads and does not consider road orientation. Finally, although Pacejka formula is extremely useful in modeling tires nonlinear characteristics, its nonlinearity increases the computational burden of NMPC especially that the ego vehicle should maintain operation within the linear region of the tire to ensure stability on the road.

2.5.2 Obstacle Avoidance using Hard and Soft Constraints

A local path planning framework is introduced in [14] that can handle multiple obstacles using an adaptive MPC. The framework focuses on DLC maneuver by defining hard constraints representing any obstacle vehicles that may exist in front of the ego vehicle. While hard constraints approach forces MPC to avoid obstacles, the maneuver is achievable in ideal case where the exact location of every obstacle is known. However, when dealing with uncertainty in real life, MPC may enter the danger region and act in an unexpected way to enforce the hard constraints. In addition, the proposed framework does not address the lateral acceleration of DLC maneuver, which may exceed tire maximum lateral acceleration, or passengers comfort rectangle. Finally, the proposed scheme does not mention path tracking while no obstacles are present. In other words, global route is not incorporated in the framework design.

In [16], a potential field method is used to represent road area and obstacles for path planning task. This representation is then used to generate collision-free path using gradient descent. To track the obtained path, a multi-constrained MPC scheme is used to calculate the front wheel steering angle. While potential fields can help in generating collision-free trajectories in real-time, the feasibility of the generated path is not guaranteed as the optimization problem does not consider vehicles constraints. In addition, to ease potential fields generation, only straight roads are considered. Lastly, the proposed method works in a sequential way, which may result in lowering the frequency of updating the control actions. Similarly, in [8] the same methodology is used for road and obstacle representation. However, instead of using gradient descent, potential field representation is added to the objective function, allowing the obtained trajectory to be feasible for the vehicle. However, these two proposed schemes do not consider the longitudinal motion of the vehicle and assume constant longitudinal velocity. Moreover, simulation tests did not cover long trips with different maneuvers and different road shapes.

2.5.3 Model Predictive Path-Following Control

A novel generic Model Predictive Path-Following Control (MPFC) approach for nonlinear systems is proposed in [9]. The approach relies on parametric curves representation for the desired path. Next, the path parameter that determines path evolution is augmented as an extra state with system states. In addition, a virtual control input is also added to the OCP. The virtual control input determines the time dynamics of the path parameter. This extra virtual control input adds an extra DoF to the controller giving it the chance to optimally follow a given path while meeting the used system model constraints.

Similarly, in [17], the authors used the same path-following technique for longitudinal and lateral vehicle control. The proposed nonlinear MPC scheme shows the effectiveness of the approach in running in real time on an actual vehicle. The authors also introduce a switchable single track model for optimization. The model consists of a kinematic model and a dynamic model, and the switching between these two models is done according to the current vehicle speed. However, as the proposed controller switches between two different models during operation, the stability of the switching is a major issue. Moreover, the proposed scheme does not consider obstacle avoidance and LC maneuvers. In addition, it requires the path to be set before operation.

2.5.4 Case Study: Nonlinear MPC controller on Chevy Bolt

A case study was conducted using an actual autonomous vehicle platform to validate the efficacy of using NMPC to solve path-following problem. In the case study, several experiments have been performed using a basic go-to-goal, a.k.a point stabilization, NMPC formulation. The used platform is an autonomous Chevy Bolt, belonging to the Watonomous team, and it is equipped by a NovAtel GNSS sensor to provide highly accurate location and orientation information of the current state of the vehicle.

The implemented and tested NMPC on Bolt uses the same kinematic model (2.1). Despite of using parameterized curves to describe the desired path of the vehicle, the reference state of the vehicle was being updated iteratively. The new reference states are found using a basic local-planner layer and behavioral-planning layer to allow the vehicle navigate safely through the environment. In addition, the interface with Bolt's drive-train was done using torque commands and not speed. Therefore, longitudinal dynamics of the vehicle is used, equation (2.4), to find the proper torque input.

$$T_m = \frac{r}{G}\left(m_v a_{long} + m_v g C_R + \frac{1}{2}\rho_a C_d A_f v^2\right) \tag{2.4}$$

where T_m is the electric motor torque, r is the wheel radius, G is the total gear reduction ratio, m_v is the mass of the vehicle, a_{long} is the acceleration of the vehicle, g is the gravitational acceleration, C_R is the rolling resistance, ρ_a is the air density, C_d is the drag coefficient, A_f is the frontal area of the vehicle, and v is the longitudinal speed of the vehicle.

The NMPC was running at a frequency of 5 [Hz]. As a result, the vehicle steering angle and the vehicle torque were changing suddenly in uncomfortable way. Decreasing the upper and lower bounds on the acceleration and the steering angle rate did not solve

the problem completely. Thus, a better approach was implemented using interpolation to find new control inputs between every two successive iterations. This increased the smoothness of vehicle motion, and hence, improving passengers experience. As an optimal control sequence for the whole prediction horizon is obtained at every iteration in NMPC and only the first obtained control action is applied, the second one along with the first one are used to interpolate new control actions until the OCP is solved in the next iteration. Figure 2.8 shows an example of control actions with and without interpolation.

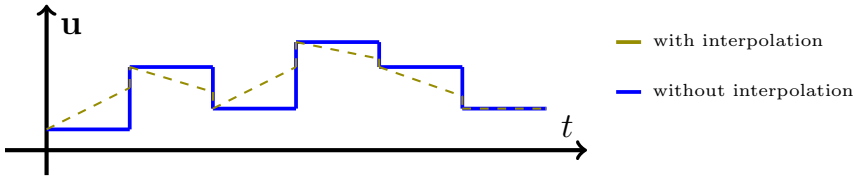
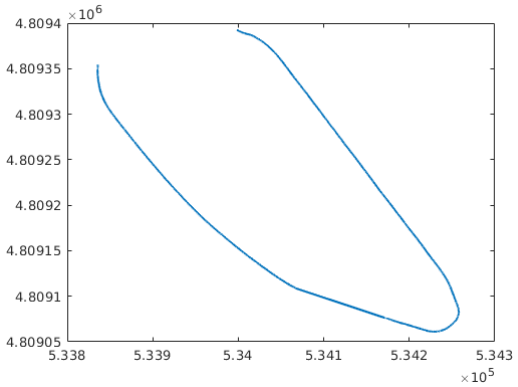


Figure 2.8: An illustration of control actions with and without interpolation.

This NMPC scheme was able to drive Bolt around the test track on an average speed of 18 [km/h]. This speed was set as the maximum possible speed and can be easily increased if needed. Figure 2.9 shows the obtained path of the vehicle on Waterloo test track and its satellite view.



(a) Vehicle path measured using NovAtel GNSS

(b) Google Maps Satellite view

Figure 2.9: Experimental results from test track

Although the implemented NMPC showed promising results, the error between the reference path and the ego vehicle was not satisfactory. In addition, velocity tracking was not verified for different speeds. These limitations are addressed and improved in the subsequent chapters.

2.5.5 Static Map and Obstacle Representation

Different mathematical representations have been proposed and used to represent obstacles. In [60] unions of overlapping spheres technique is discussed to approximate obstacles in 3D. Since the autonomous driving problem can be represented in 2D, this approximation is relaxed to circles. To model obstacles with relatively high aspect ratio, elliptical shapes

can be used [61,62]. However, as these approximations must be conservative; objects must be covered entirely. Consequently, these approximations may result in false representation of free space as obstacles.

2.5.6 Conclusion

To conclude, recent related work addresses various solutions for local motion planning and control. However, most of these solutions focus solely on one particular point. Some approaches try to address both problems, but as explained above, a more generic approach is needed. Finally, most of the frameworks presented above focus on certain maneuvers and do not consider road orientation, long trips, and the preceding layers of the decision-making system. Consequently, our aim is to introduce a novel methodology that incorporate global routing and behavioral planning, while simultaneously generating and tracking local motion profiles.

Chapter 3

Simultaneous Local Motion Planning and Control

3.1 Introduction

In this chapter, we present our proposed simultaneous local motion planning and control scheme. The main focus of our approach is to enable autonomous driving for long trips while reducing the burden of computational complexity and ensuring obstacle avoidance functionality. The main contributions introduced in this chapter are:

- A parallel architecture for local motion planning and control layers.
- Feasibility-guaranteed LC and DLC maneuvers planners.
- An online parameterized curve generator.

We start by planning a route from the current location of the vehicle to a given destination. Next, we execute a route feasibility checker that examines the feasibility of the obtained path in terms of vehicle kinematic and dynamic constraints and makes any necessary modifications. After that, while driving the vehicle autonomously, we run four modules continuously; a behavioral planner that makes any necessary qualitative actions; explained in the next section, an online parameterized curve generator to the target destination, a feasibility-guaranteed DLC planner, and an MPFC to track the path with minimum error. Fig. 3.1 shows the architecture of the proposed approach.

3.2 Global Planning and Behavioral Planning

Nowadays with the recent developments in highly detailed maps creation, a.k.a HD maps, global planners can find routes with significantly high precision to a centimeter level [63]. Therefore, a global route can now describe LC maneuvers with high accuracy [64]. As a result, given a target destination, a global route can include information about LC maneuvers, lane markings, road boundaries, intersections, waypoints at the center of the

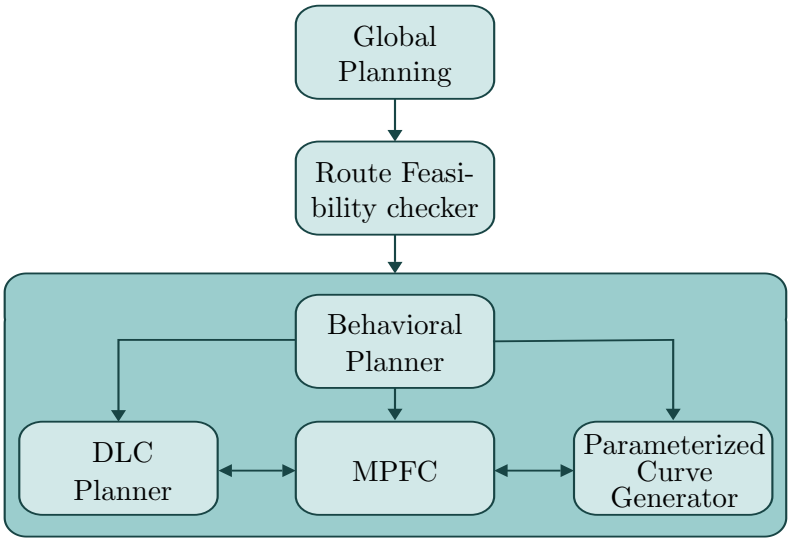


Figure 3.1: Architecture of the proposed approach.

lanes and speed limits. However, although HD maps push the limits of global planning, the obtained route generally suffers from high discretization of waypoints and may not fully incorporate vehicle dynamic constraints. Therefore, after obtaining a global route, our proposed scheme tries to fill the gap by pre-processing the global route offline. More details are provided in the next section.

As previously mentioned in 2.4.2, behavioral planning layer can take many forms depending on the complexity considered. Since the thesis focus is not on behavioral planning, we use an FSM behavioral planner to cover most of the possible cases a local planner may encounter while driving autonomously. Note that different scenarios covered by a behavioral planner may lead to the same qualitative action sent to the local planning layer. Fig. 3.2 shows the implemented behavioral planning FSM, six states are modeled and they are:

- **Track desired speed:** while following a desired path, maintain the reference speed.
- **Follow a leading vehicle:** while following the reference path, if there is a leading vehicle, maintain the leading vehicle speed with a safe distance until a safe DLC maneuver is possible.
- **Make a DLC maneuver:** to safely avoid a static or dynamic obstacle on the current lane, deviate temporarily from the reference path and make a DLC then converge back to the original reference path.
- **Decelerate to stop:** decelerate the vehicle speed until reaching a complete stop.
- **Stop:** Keep the vehicle stopped until it is safe to proceed.
- **Reroute:** If the error between the desired path and the actual location of the vehicle is large, or no path has been created, find a new route to the target destination.

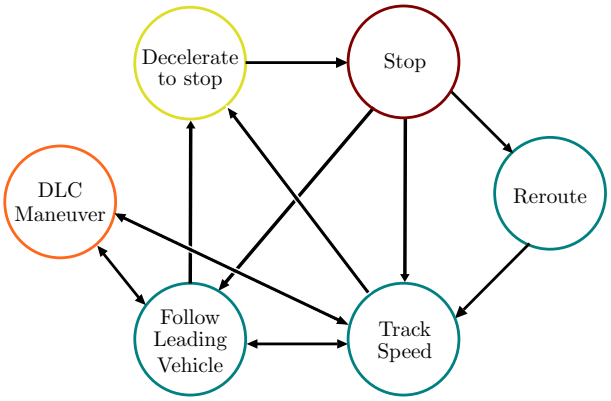


Figure 3.2: The adopted FSM behavioral planner.

3.3 Offline Route Feasibility Checker

3.3.1 Waypoints Refinement

Generally, two coordinate systems can be used for waypoint representation, which are S-L coordinate system and X-Y inertial coordinate system. Here, we adopt X-Y inertial coordinate system, as it is continuous in the whole space.

Describing global route as waypoints is very useful, but sometimes the obtained waypoints may be spaced unevenly. For a given global route $\mathcal{R}_{global}$ that is described by waypoints WPs and a waypoint linear spacing distance d_{wp}, first we check whether the waypoints of the global route satisfy the distance d_{wp}. If not, our approach interpolates new waypoints to ensure even spacing between waypoints. In case of an LC maneuver, the algorithm calls an LC planner to reroute the maneuver. LC planner is discussed in more details in the next subsection. Finally, the equally spaced route $\mathcal{R}_{global,ref}$ is then returned. Ensuring equal spacing between waypoints is a crucial step for the curve fitting introduced later in this chapter as it guarantees smoothness of path-following. The proposed methodology is shown in Alg. 1.

3.3.2 Feasibility-Guaranteed Lane Change Planner

Global planners may or may not incorporate vehicle constraints in planning an LC maneuver. Note that our scope here is for LC maneuvers required to reach a given destination and not for obstacle avoidance. The latter will be discussed in the next section. Hence, we adopt the work in [12] and [13] with some modifications to re-plan LC maneuvers. By using a sigmoid function, a smooth LC can be planned by defining the required longitudinal and lateral distances. For simplicity, we represent the LC maneuver by parameterizing y coordinate by x coordinate. To facilitate different road orientations, the equation can then be rotated by a road angle θ_{road}. The general sigmoid function is given by:

$$y(x) = \frac{B}{1 + e^{-a(x-c)}} \tag{3.1}$$

where B is the lateral displacement needed, c is the maneuver location along the x axis and a represents how fast an LC maneuver takes place. In [12] and [13], parameter c is used

Algorithm 1: Waypoints Spacing Checker

Input: $\mathcal{R}_{global}$, d_{wp}
Output: $\mathcal{R}_{global,ref}$

```
begin
    R_global,ref[0] ⟵ R_global[0]
    oldWP ⟵ 0
    for WP ∈ R_global do
        if LaneChange did not occur then
            Distance = getDistance( R_global,ref[end], WP)
            if Distance < d_wp OR LaneChange then
                oldWP ⟵ WP
                Continue
            else if Distance == d_wp then
                oldWP ⟵ WP
                AppendToR_global,ref(WP)
            else if Distance > d_wp then
                newWP = Interpolate(R_global,ref[end],oldWP, WP)
                AppendToR_global,ref(newWP)
        else
            Plan a lane change maneuver.
```

to determine the maneuver's longitudinal distance and a is used to represent the urgency of the maneuver. However, this representation assumes starting from the zero point along the x axis. Therefore, in our approach we rely on a to determine the longitudinal distance $d_{long,LC}$ of an LC maneuver. Fig. 3.3 shows the effect of changing a with respect to $d_{long,LC}$. After examining the sigmoid function behavior, $d_{long,LC}$ and a can be correlated regardless of the value of B and c by:

$$d_{long,LC} = \frac{8}{a} \tag{3.2}$$

The pivotal question is how to determine $d_{long,LC}$ value given a lateral displacement B. The key factor is the maximum curvature $\kappa_{max,v}$ that a vehicle can handle given its dynamic and kinematic constraints. Mathematically, curvature κ shows the speed of rotation of the tangent line to a curve at a certain point [65], and it is the reciprocal of the radius of curvature:

$$\kappa = \frac{1}{R} \tag{3.3}$$

It can also be formulated directly by:

$$\kappa = \frac{\left|x'y'' - y'x''\right|}{\left[(x')^2 + (y')^2\right]^{\frac{3}{2}}} \tag{3.4}$$

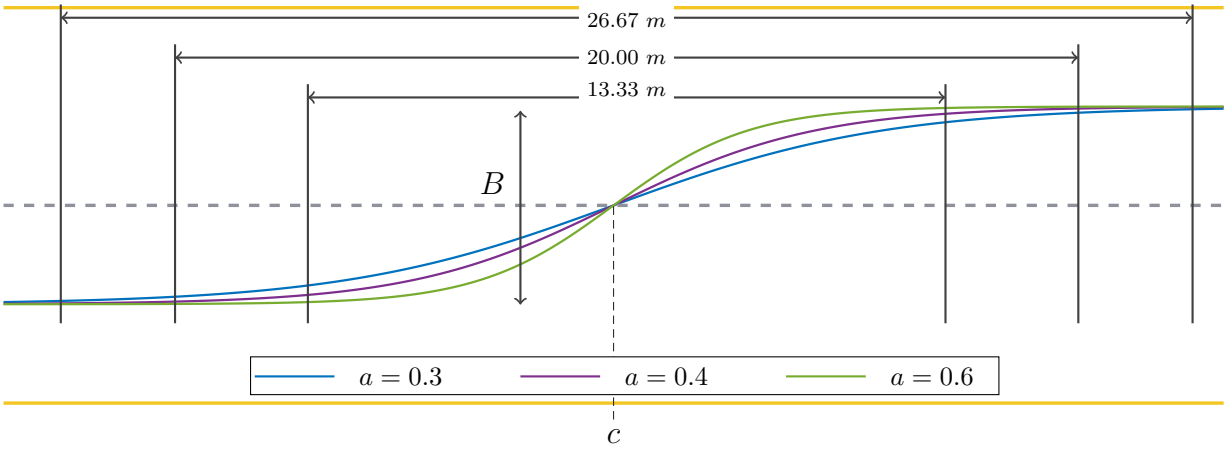


Figure 3.3: Three different sigmoid functions.

and for plane curves where $y = f(x)$, κ is reduced to:

$$\kappa = \frac{|y''|}{\left[1 + (y')^2\right]^{\frac{3}{2}}} \tag{3.5}$$

Therefore, to find κ of an LC maneuver represented by a sigmoid function, first and second derivatives of (3.1) are calculated as follows:

$$y'(x) = -a \cdot e^{-a(x-c)} \cdot \left[\frac{-B}{\left(1 + e^{-a(x-c)}\right)^2}\right]$$

and can be simplified to:

$$y'(x) = a \cdot y(x) \cdot \left[1 - \frac{y(x)}{B}\right] \tag{3.6}$$

Similarly:

$$y''(x) = a \cdot \left[y'(x) \cdot \left[1 - \frac{y(x)}{B}\right] - y(x) \cdot \frac{y'(x)}{B}\right] \tag{3.7}$$

Maximum curvature $\kappa_{max,LC}$ of an LC maneuver depends on B and a. As shown in table 3.1 and in Fig. 3.4, $\kappa_{max,LC}$ increases as a or B increase. Note that an increase in a results in a decrease in $d_{long,LC}$.

a \ B	1.0	2.0	3.0	4.0
0.05	0.0025	0.005	0.0075	0.01
0.1	0.01	0.02	0.03	0.04
0.15	0.0225	0.045	0.0675	0.09
0.2	0.04	0.08	0.12	0.16

Table 3.1: A sample of maximum curvature values for different values of a and B.

Mathematical derivation of a generic mathematical formula that links B, a and $\kappa_{max,LC}$ is not feasible because of the complexity of substituting equations (3.6) and (3.7) in (3.5).

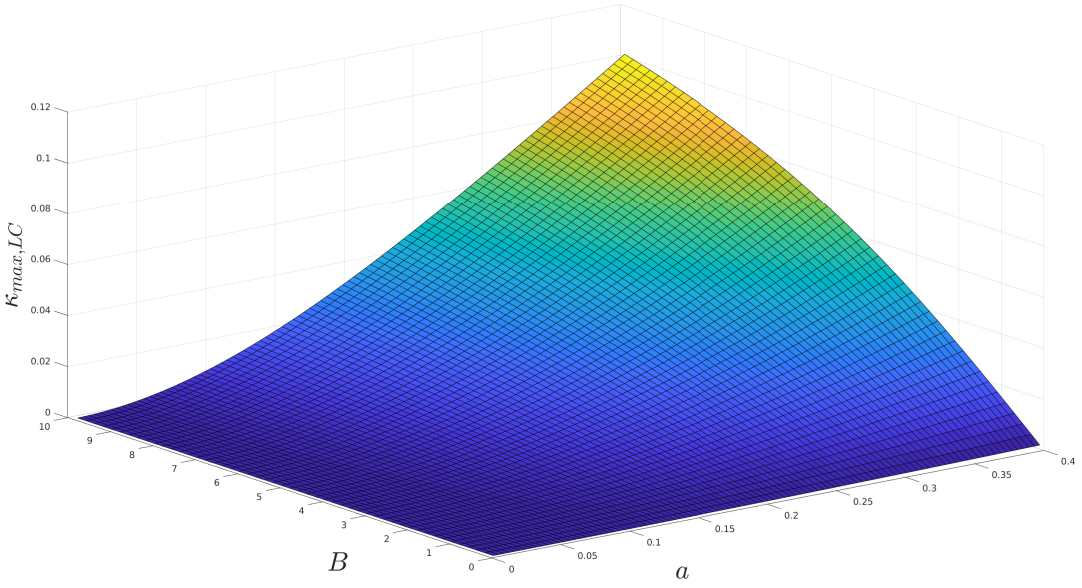


Figure 3.4: Effect of B and a on $\kappa_{max,LC}$.

To solve this issue, we propose two methods. The first one is simple and more straight forward, it sets the B value to a typical lateral displacement $B_{lat,LC}$ needed to perform a complete LC maneuver. Then, using the numerically generated data, we fit an equation that links a and $\kappa_{max,LC}$ such that $a = f(\kappa_{max,LC})$. Therefore, for a given $\kappa_{max,v}$, value of a that reflects $d_{long,LC}$, can be calculated easily and directly.

The second method uses the numerically generated data to fit a two dimensional function that maps the values of a and B to the values of $\kappa_{max,LC}$, such that $\kappa_{max,LC} = f(a, B)$. After that, for a given B and $\kappa_{max,LC}$, the value of a can then be numerically calculated by finding the roots of the resulting equation. Fig. 3.5 shows the obtained curve for both methods.

In the results shown in Fig. 3.5, $B_{lat,LC} = 3.6\ m$ for the first method and the equation obtained for a is given by:

$$a(\kappa) = 8.616 \times 10^6\kappa^5 - 1.23e \times 10^6\kappa^4 + 6.587 \times 10^4\kappa^3 - 1677\kappa^2 + 26.6\kappa + 0.02233 \quad (3.8)$$

where $\kappa = \kappa_{max,LC}$ for simplicity. For the second method, the obtained equation is:

$$\begin{aligned}\kappa(a, B) = &-0.0003849 + 0.01688a + 6.824 \times 10^{-5}B - 0.1094a^2 - 0.01148aB \\ &+ 8.455 \times 10^{-5}B^2 + 0.1768a^3 + 0.1703a^2B + 0.001221aB^2 - 1.808 \times 10^{-5}B^3 \\ &- 0.1052a^3B - 0.006476a^2B^2 - 8.759 \times 10^{-6}aB^3 + 8.83 \times 10^{-7}B^4\end{aligned}$$

where $a \in [0, 0.4]$ and $B \in [0, 10]$. By substituting given values of B and $\kappa_{max,LC}$, the previous equation can be reduced to:

$$\gamma_3a^3 + \gamma_2a^2 + \gamma_1a + \gamma_0 = 0 \quad (3.9)$$

where $\gamma_0, \ldots, \gamma_3$ are constants and a can be found by solving for the roots of (3.9) and taking the root λ such that $\lambda \in [0, 0.4]$.

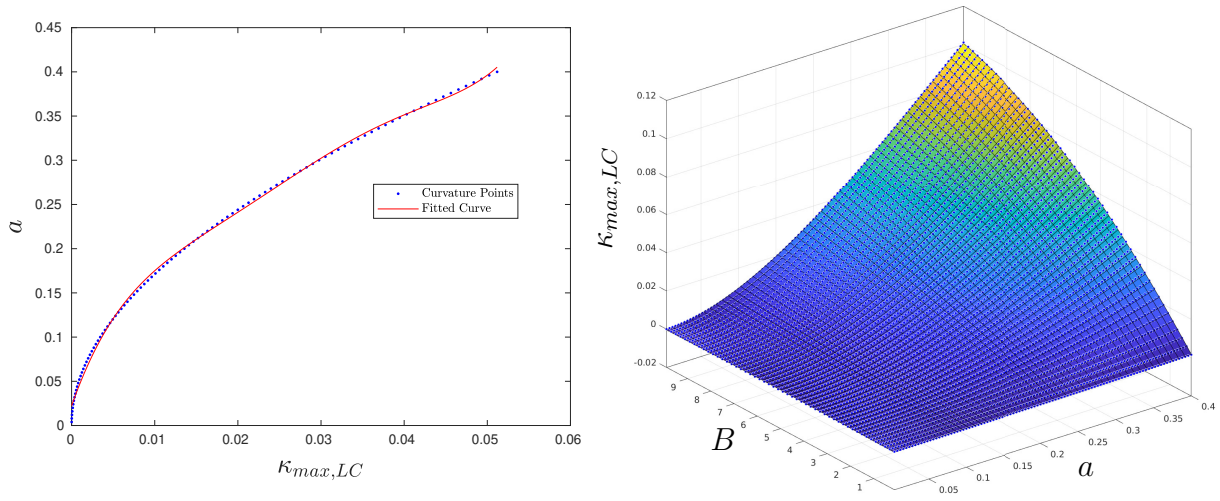


Figure 3.5: Left: first method, values of a and $\kappa_{max,LC}$ for $B = B_{lat,LC}$ and the fitted equation. Right: second method, values of $\kappa_{max,LC}$ for different values of a and B and the fitted two dimensional equation.

Value of $\kappa_{max,LC}$ should be less than or be equal to $\kappa_{max,v}$ at any given instance. Furthermore, $\kappa_{max,v}$ must indicate dynamic and kinematic limits of the ego vehicle. We assume that the maximum kinematically admissible curvature for a vehicle is $\kappa_{max,kin}$. The value of $\kappa_{max,kin}$ depends on the minimum turning radius R_{min} of the vehicle. The recommended values of R_{min} for passenger vehicles lies in $[10.4 - 10.7]$ [m] [66]. The maximum dynamically admissible curvature for a vehicle $\kappa_{max,dyn}$ depends on vehicle lateral acceleration a_{lat} and vehicle speed v at a certain instance such that:

$$\kappa_{max,dyn} = \frac{a_{lat}}{v^2} \tag{3.10}$$

Since this step is done offline, speed limit of the road along with maximum comfortable lateral acceleration are used to find $\kappa_{max,dyn}$. Finally, $\kappa_{max,v}$ is set to be the minimum of $\kappa_{max,kin}$ and $\kappa_{max,dyn}$ such that:

$$\kappa_{max,v} = \min\{\kappa_{max,kin}, \kappa_{max,dyn}\} \tag{3.11}$$

Although both methods introduced above use over-fitted equations, the expected values of a and B in real cases should not be out of the specified range. Thus, by using the first or second method, the obtained sigmoid equation that describes an LC maneuver is dynamically and kinematically feasible for the ego vehicle to follow. The two proposed previous methods are summarized in Alg. 2 and Alg. 3 where X_{LC} and Y_{LC} are the coordinates of LC maneuver in the global frame, θ_{road} is the road orientation and v_{lim} is the speed limit.

Algorithm 2: First method

Input: X_{LC}, Y_{LC}, θ_{road}, v_{lim}, $\kappa_{max,kin}$, $B_{lat,LC}$
Output: LC_{wp}

begin
- $\kappa_{max,dyn} = \frac{a_{lat}}{v_{lim}^2}$.
- $\kappa_{max,v} = \min\{\kappa_{max,kin}, \kappa_{max,dyn}\}$.
- $a_{LC} = a(\kappa_{max,v})$ using (3.8)
- $B = B_{lat,LC}$.
- Generate $y(x)$ using equation (3.1).
- Rotate $y(x)$ by θ_{road}.
- Translate $y(x)$ by X_{LC} and Y_{LC}.
- Generate LC_{wp}.

Algorithm 3: Second method

Input: X_{LC}, Y_{LC}, θ_{road}, v_{lim}, $\kappa_{max,kin}$, B
Output: LC_{wp}

begin
- $\kappa_{max,dyn} = \frac{a_{lat}}{v_{lim}^2}$.
- $\kappa_{max,v} = \min\{\kappa_{max,kin}, \kappa_{max,dyn}\}$.
- Find $\gamma_0, \ldots, \gamma_3$ using $\kappa_{max,v}$ and B.
- Solve equation (3.9) for a_{LC}.
- Generate $y(x)$ using equation (3.1).
- Rotate $y(x)$ by θ_{road}.
- Translate $y(x)$ by X_{LC} and Y_{LC}.
- Generate LC_{wp}.

3.3.3 Curvature Calculation and Velocity Profile Refinement

Curvature of the route directly affects the speed of a vehicle as explained in equation (3.10). However, since the reference global route $\mathcal{R}_{global,ref}$ is represented by equally spaced waypoints and not a mathematical equation, curvature formula given in equation (3.4) cannot be used.

Instead, we introduce Menger curvature. Menger curvature of three points in multi-dimensional euclidean space corresponds to the reciprocal of a radius of a circle that passes through these three points [67]. Therefore, curvature of the route can be calculated using Menger curvature iteratively. For three points in the Euclidean space P_1, P_2 and P_3, Menger curvature is given by:

$$\kappa = \frac{4A}{|dist(P_1, P_2)| \cdot |dist(P_2, P_3)| \cdot |dist(P_3, P_1)|} \tag{3.12}$$

where A is the area of triangle formed by P_1, P_2 and P_3, and $dist(.,.)$ is the Euclidean distance between two points. Since waypoints spacing d_{wp} is relatively small, deviation in Z direction (height) of waypoints can be neglected. Therefore, we assume that waypoints lie on the X-Y plane. Accordingly, the area A can be found iteratively for three points using the following equation:

$$A = \left| \frac{P_{1,x}(P_{2,y} - P_{3,y}) + P_{2,x}(P_{3,y} - P_{1,y}) + P_{3,x}(P_{1,y} - P_{2,y})}{2} \right| \tag{3.13}$$

where $P_{1,x}$ and $P_{1,y}$ are X-Y components of P_1 and so on. Thus, curvature information of $\mathcal{R}_{global,ref}$ can now be calculated using equations (3.12) and (3.13).

To generate a velocity profile for $\mathcal{R}_{global,ref}$ we rely on speed limit information for each waypoint on the route. However, speed limits are set based on road regulations and do not usually consider sharp turns. For example, in an urban driving scenario, speed limit usually lies within 50 km/h - 60 km/h. However, to make a sharp turn at an intersection, the vehicle is not expected to drive on the speed limit. Therefore, after generating the

velocity profile based on speed limit information, we refine it based on the route curvature information. This can be done by reformulating equation (3.10) as follows:

$$v = \sqrt{\frac{a_{lat}}{\kappa_{route}}} \tag{3.14}$$

where a_{lat} is the maximum comfortable lateral acceleration and κ_{route} is the calculated route curvature using equations (3.12) and (3.13). We will use route curvature information again for curve fitting, which will be discussed in the next section.

3.4 Online Parameterized Curves Generator

In this section we discuss our methodology to follow the refined route $\mathcal{R}_{global,ref}$ from the starting point to a given destination.

3.4.1 Vehicle Positioning Relative to Global Route and Dynamic Lookahead Distance

As discussed in the previous section, $\mathcal{R}_{global,ref}$ consists of a series of equally spaced waypoints. This representation raises a problem of determining the nearest waypoint on the route to the actual location of the vehicle. One way to solve this issue is by searching the whole route for the nearest waypoint to the actual location. However, as it is being performed online while driving autonomously, it may be too computationally expensive in case of long routes. Instead, we propose an efficient localization algorithm that can find the nearest waypoint efficiently.

The algorithm starts by finding the actual location of the ego vehicle. This can be done using different sensors such as GNSS and IMUs. Next we find the euclidean distance between the current location and the previous location recorded for the vehicle. After that, this euclidean distance is projected on the global route $\mathcal{R}_{global,ref}$. Note that the actual location of the ego vehicle is ahead of the projected distance. Therefore, this projected distance is used to generate a search window that covers a distance at within which the vehicle lies. Then, the euclidean distance between every waypoint in the search space and the actual location is calculated, and the waypoint with the smallest euclidean distance is used as the actual location of the ego vehicle. An example is given in Fig. 3.6 where the red brackets are the search window. Note that although the example shows that the current location of the vehicle is perfectly on the route $\mathcal{R}_{global,ref}$, the algorithm can handle vehicle deviation from the route as well.

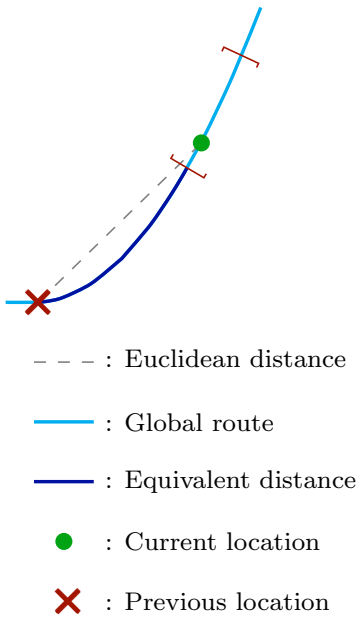


Figure 3.6: Vehicle positioning relative to the global route.

Here, dynamic lookahead distance approach is introduced. Generally, lookahead distance term refers to a

point ahead of the vehicle by a certain distance. Usually short lookahead distances are being used to locally plan the vehicle motion. However, in our approach we decouple the obstacle avoidance and the local motion planning into two separate problems. Given the global route $\mathcal{R}_{global,ref}$, and the nearest waypoint to the actual location of the vehicle, a subset of the waypoints is used to generate a parametric curve for the vehicle to follow. As road shapes vary significantly based on the road network design, a fixed lookahead distance may result in a poor curve generation, or it may cover relatively short distance. Instead, our approach uses road curvature information to dynamically change the lookahead distance, so that the generated parametric curve covers the longest possible distance while maintaining minimum error to the route $\mathcal{R}_{global,ref}$. For example, if the road is straight, retrieved curvature value will be zero, and thus, maximum lookahead distance is set. On the other hand, for relatively big curvature values, a smaller lookahead distance is used. Here, the dynamic lookahead distance $d_{LA,dyn}$ is set such that $d_{LA,dyn} \in [12.5, 100]$ m. The proposed algorithm for vehicle position, dynamic lookahead distance and parametric curve generation is summarized in Alg. 4.

Algorithm 4: Generate a new parametric curve $\mathcal{P}$

Input: Search Length, d_{wp}, $\mathcal{R}_{global,ref}$
Output: $\mathcal{P}$

```
begin
    Search Index = 0
    while did not Arrive do
        Location = Get ego vehicle location
        Euclidean Distance = getDistance(Location, R_global,ref[Search Index])
        Search Index = Search Index + Euclidean Distance/d_wp
        Last Search Index = Search Index + Search Length
        for n in range(Search Index, Last Search Index) do
            SearchArray.append(getDistance(Location, WP[n]))
        FitIndStart = SearchIndex + SearchArray.index(min{SearchArray})
        d_LA,dyn = getLookAhead(getCurvatureInfo)
        FitIndEnd = FitIndStart + d_LA,dyn / d_wp
        P = GenerateSpline(FitIndStart, FitIndEnd)
```

3.4.2 Curve Fitting Problem Formulation

In this subsection, we present our proposed approach for the curve fitting optimization problem. As we decouple the obstacle avoidance and the local motion planning, the purpose here is to generate a parametric curve using a subset of $\mathcal{R}_{global,ref}$ regardless of any obstacles. Obstacle avoidance is discussed in the next subsection of this chapter.

Generally, in a curve fitting problem, the purpose is to construct a mathematical equation that provides the best fit to a series of data. One way to perform curve fitting is by using Least Squares (LS), at which an optimization problem is formulated to obtain the optimal curve parameters. A general optimization problem has the following formulation:

$$\min_{\mathbf{u}} \quad \Psi(\mathbf{u}) \tag{3.15a}$$

$$\text{s.t.} \quad \mathbf{g}_1(\mathbf{u}) \leq 0, \tag{3.15b}$$

$$\mathbf{g}_2(\mathbf{u}) = 0. \tag{3.15c}$$

where Ψ is the optimization criterion, $\mathbf{u}$ is a vector of optimization variables, $\mathbf{g}_1(\mathbf{u})$ is a vector of inequality constraints and $\mathbf{g}_2(\mathbf{u})$ is a vector of equality constraints. For an LS curve fitting problem, let us denote the series of points that we need to fit by Λ such that $\Lambda = \lambda_1, \lambda_2, \ldots, \lambda_n$ and $\lambda_i = [x_i, y_i]^\top$, and the equation that we are trying to fit to these points by $f(\xi)$. Therefore, the optimization criterion Ψ is the summation of the squared error between Λ and $f(\xi)$ such that:

$$\Psi = \sum_{i=1}^{n} \left|\left|\lambda_i - f(\xi_i)\right|\right|^2, \tag{3.16}$$

where n is the number of points, and the optimization variables $\mathbf{u}$ are the parameters of the desirable equation $f(\xi)$. $\mathbf{g}_1(\mathbf{u})$ and $\mathbf{g}_2(\mathbf{u})$ constraints are not necessarily needed unless the output equation must meet certain constraints. For example, if a certain point must lie exactly on the output equation, it can be enforced as equality constraints.

To generate a parametric curve $\mathcal{P}$ parameterized by a path variable ξ for a series of waypoints, we use two parametric equations, one for X direction and one for Y direction as follows:

$$\mathcal{P}(\xi) = \begin{bmatrix} \bar{x}(\xi) \\ \bar{y}(\xi) \end{bmatrix}, \tag{3.17}$$

and for each parametric equation, we use a 5th order polynomial (quintic spline):

$$\begin{bmatrix} \bar{x}(\xi) \\ \bar{y}(\xi) \end{bmatrix} = \begin{bmatrix} \alpha_5\xi^5 + \alpha_4\xi^4 + \alpha_3\xi^3 + \alpha_2\xi^2 + \alpha_1\xi + \alpha_0 \\ \beta_5\xi^5 + \beta_4\xi^4 + \beta_3\xi^3 + \beta_2\xi^2 + \beta_1\xi + \beta_0 \end{bmatrix}. \tag{3.18}$$

In addition, the path variable ξ is set such that $\xi_i \in [0, 1]$ and $\Delta\xi = \frac{1}{n}$. Since ξ values are equally spaced, global route waypoints must be equally spaced as well to ensure unbiased fitting and smooth behavior as mentioned earlier. Therefore, the curve fitting optimization problem becomes:

$$\min_{\mathbf{u}} \quad \Psi(\mathbf{u}) = \sum_{i=1}^{n} \left|\left|\lambda_i - f(\xi_i)\right|\right|^2 \tag{3.19a}$$

and the optimization variables:

$$\mathbf{u} = \begin{bmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \alpha_3 & \alpha_4 & \alpha_5 & \beta_0 & \beta_1 & \beta_2 & \beta_3 & \beta_4 & \beta_5 \end{bmatrix}^\top \tag{3.20}$$

To solve the curve fitting optimization problem, we use an open-source numerical optimization tool called CasADi [68], which uses the interior point method (IPOPT) [69]. One point to note is that the number of waypoint n is fixed regardless of the lookahead distance used. Therefore, the computation time needed to generate the parametric curve $\mathcal{P}$ is almost constant and can be synchronized. Note that at time $t = k$, the parametric curve $\mathcal{P}_k$ must overlap with the previous parametric curve $\mathcal{P}_{k-1}$ to ensure smoothness and continuity of the reference path. Fig. 3.7 illustrates an example of two overlapping curves.

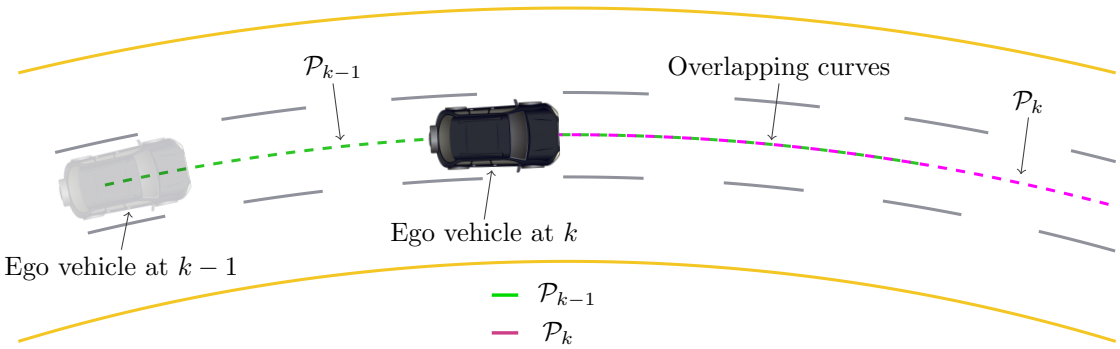


Figure 3.7: Overlapping in parametric curves generation.

3.4.3 Feasibility-Guaranteed Double Lane Change Planner

In literature, different methods have been proposed to achieve obstacle avoidance as outlined in 2. To the best of the author's knowledge, DLC maneuver has not been addressed using bell-shaped exponential functions. We propose using a bell-shaped function as a collision-free trajectory for the vehicle to perform a DLC maneuver safely and smoothly. Similar to our approach in 3.3.2, for simplicity, we study the behavior of a bell-shaped function in X-Y plane given by:

$$y(x) = \frac{2H}{1 + e^{(mx)^2}}, \tag{3.21}$$

where H is the maximum lateral shift needed to perform a DLC maneuver, or generally, obstacle avoidance, and m is the sharpness of the maneuver. Note that m also determines the longitudinal distance $d_{long,DLC}$ needed to perform a DLC maneuver. After examining the behavior of equation (3.21), $d_{long,DLC}$ and m can be correlated by the following equation:

$$d_{long,DLC} = \frac{4}{m}. \tag{3.22}$$

Fig. 3.8 shows three different plots for different values of m and their corresponding $d_{long,DLC}$. Selection of $d_{long,DLC}$ depends on the maximum curvature the vehicle can handle

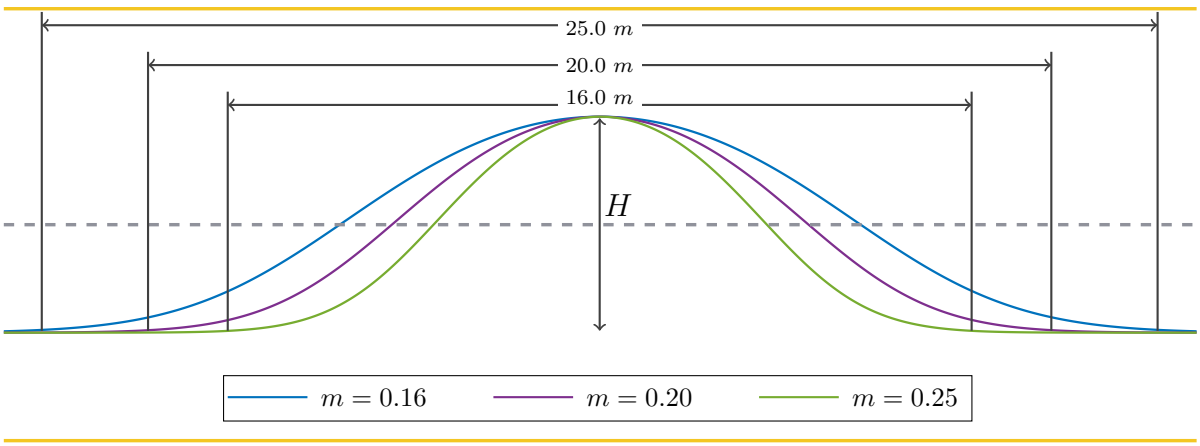


Figure 3.8: Three different bell-shaped functions.

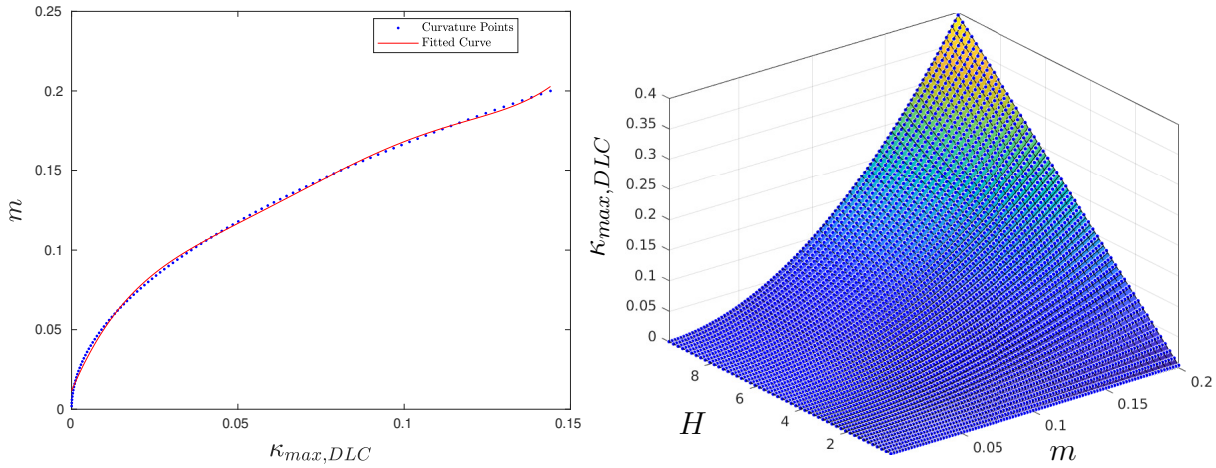


Figure 3.9: Left: first method, values of m and $\kappa_{max,DLC}$ for $H = H_{lat,DLC}$ and the fitted equation. Right: second method, values of $\kappa_{max,DLC}$ for different values of m and H and the fitted two dimensional equation.

given its speed and maximum comfortable lateral acceleration. To derive an expression for the maximum curvature of DLC maneuver $\kappa_{max,DLC}$, we base our derivation on equation (3.5). Therefore, the first and second derivative of equation (3.21) should be calculated as follows:

$$y'(x) = \frac{-2H}{\left(1 + e^{(mx)^2}\right)^2} \cdot 2mx \cdot m \cdot e^{(mx)^2}, \tag{3.23}$$

and can be simplified to:

$$y'(x) = -2m^2 x \cdot y(x) \cdot \left[1 - \frac{y(x)}{2H}\right]. \tag{3.24}$$

Similarly:

$$y''(x) = 2m^2 \cdot \left[x \cdot y(x) \cdot \frac{y'(x)}{2H} - x \cdot y'(x) \cdot \left[1 - \frac{y(x)}{2H}\right] - y(x) \cdot \left[1 - \frac{y(x)}{2H}\right]\right]. \tag{3.25}$$

Similar to our approach in 3.3.2, we numerically generate values for maximum curvature $\kappa_{max,DLC}$. In addition, as finding a mathematical relation between $\kappa_{max,DLC}$ and H and m is unfeasible, we use the same methods introduced before in 3.3.2. First method, which is suitable for a typical DLC maneuver, fixes the value of H to a typical lane width denoted by $H_{lat,DLC}$. Then, an equation of m in terms of $\kappa_{max,DLC}$ is generated such that $m = f(\kappa_{max,DLC})$.

The second method fits a two dimensional function using the numerically generated data such that the obtained function maps m and H values to $\kappa_{max,DLC}$ values so that $\kappa_{max,DLC} = f(m, H)$, where $H \in [0, 10]$ and $m \in [0, 0.2]$. Finally, for a given H and $\kappa_{max,DLC}$, m value can be numerically calculated by finding the roots of the resulting equation. Fig. 3.9 shows the obtained curve for both methods. For $H_{lat,DLC} = 3.6$, the resulting equation becomes:

$$m(\kappa) = 2.598 \times 10^4 \kappa^5 - 1.042 \times 10^4 \kappa^4 + 1565\kappa^3 - 111.9\kappa^2 + 4.928\kappa + 0.01156 \tag{3.26}$$

where $\kappa = \kappa_{max,DLC}$ for simplicity. For the second method, the obtained equation is:

$$\kappa(m, H) = m^2 H$$

which can be reduced for a value of H and $\kappa_{max,DLC}$ to:

$$\gamma_1 m^2 + \gamma_0 = 0 \tag{3.27}$$

where γ_0 and γ_1 are constants and m can be found by solving for the roots of (3.27) and taking the root λ such that $\lambda \in [0, 0.2]$.

Value of $\kappa_{max,DLC}$ should be less than or equal to $\kappa_{max,v}$ at any given instance. We rely on equation (3.11) to set the value of $\kappa_{max,DLC}$. Since in an obstacle avoidance scenario, the obstacle itself might be dynamic and moving. Therefore, the variable x denoted in equation (3.21) represents the relative distance between the obstacle and the ego vehicle in longitudinal direction.

As mentioned before, we decouple local motion planning and obstacle avoidance into two separate problems. Since in a generic obstacle avoidance scheme the lateral distance H is not necessarily equal to the lane width, the optimal H value may need to be found using a search algorithm. However, our approach reduces the search space into one dimensional search. That is to say, optimal value of H should be the safest lateral distance to the obstacle and road boundaries regardless of the obstacle relative longitudinal distance to the ego vehicle, and H sign determines the direction of the DLC maneuver. Fig. 3.10 shows an example of obstacle blocking a small part of the left lane. In such a case, the optimal H value will be greater than typical lane width. Thus, we reformulate the equation (3.21) to:

$$\delta(\rho) = \frac{2H}{1 + e^{(m\rho)^2}}, \tag{3.28}$$

where δ is the lateral shift in vehicle coordinate frame and ρ is the relative longitudinal distance between ego vehicle and the obstacle in vehicle coordinate frame. Note that there is no longitudinal shift in the vehicle coordinate frame. To integrate our collision avoidance

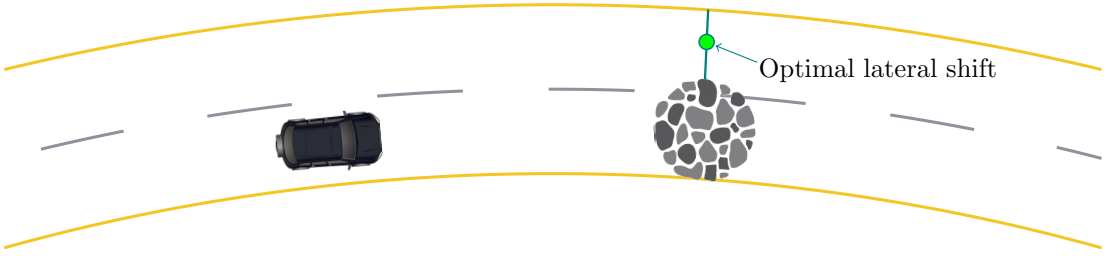


Figure 3.10: An illustration of finding the optimal H value. Note that the optimal H is not at the center of the left lane.

methodology with the obtained parametric curve $\mathcal{P}$, the resulting bell-shaped equation is simply added to $\mathcal{P}$ since it is already in the vehicle coordinate frame. Thus, the reference collision-free path $\mathcal{P}_{clear}$ can be obtained by incorporating road orientation angle θ_{road} as follows:

$$\mathcal{P}_{clear}(\xi, H, \rho) = \begin{bmatrix} \bar{x}(\xi) \\ \bar{y}(\xi) \end{bmatrix} + \delta(\rho, H) \begin{bmatrix} \cos(\theta_{road}) \\ -\sin(\theta_{road}) \end{bmatrix} \tag{3.29}$$

This proposed methodology can handle much faster response for collision-free paths generation as it temporarily shifts the reference path to avoid obstacles while allowing the ego vehicle to smoothly converge back to the original route. In addition, shifting the path $\mathcal{P}$ temporarily does not affect the vehicle positioning relative to the global route, as the the algorithm will find the corresponding waypoint on the path and use it as the actual location of the vehicle to generate the next.

3.5 Model Predictive Path-Following Control

3.5.1 MPFC Scheme

NMPC, or MPC in general, has been attracting attention in recent years to tackle the control aspect of autonomous driving because of its promising performance in dealing with nonlinear systems [16]. As stated in 2.2, the key point of NMPC is formalizing an OCP and solving it iteratively to obtain the optimal control actions. In general, this formulation depends on the nature of the application and the desired performance of the controller. Broadly speaking, two of the main problems in the control of robots are setpoint stabilization and trajectory tracking problems. Setpoint stabilization refers to controlling the motion of a robot to reach a final target state regardless of the traversed path. In trajectory tracking, a time varying reference is being used to control the speed and traversed path of a robot [70]. However, autonomous driving, and some other applications, require higher priority for error minimization between a given reference path and the ego vehicle, rather than maintaining a strict reference speed. Therefore, vehicle speed might be modified to achieve better path-following accuracy [70].

For this type of applications, the objective is to design a controller which decides on the optimal speed to follow a given path in order to minimize the deviation between the reference path and the actual location of the vehicle. The path-following problem in this case is referred to as *Output Path Following with Velocity Assignment*, where *output space* refers to the output of the system $\mathbf{y}$ in terms of system states $\mathbf{x}$.

For two dimensional output space, some assumptions are made to define the path-following problem:

- The reference path has no pre-defined timing information.
- The reference path, denoted by $\mathcal{S}$, is a parameterized curve.
- The reference path $\mathcal{S}$ is continuously differentiable.

$$\mathcal{S} = \{\mathcal{G} \in \mathbb{R}^3 \mid \xi \in [\xi_0, \xi_f] \rightarrowtail \mathcal{G} = s(\xi)\}. \tag{3.30}$$

where $\mathcal{G}$ contains the information of a point coordinates in the global X-Y frame and its orientation, ξ is the path parameter, ξ_0 and ξ_f are the upper and lower bounds of ξ, and $s : \mathbb{R} \rightarrow \mathbb{R}^3$. Note that ξ is time dependent, but its dynamics is not specified. Specifying ξ time evolution will result in a trajectory tracking problem.

To solve the *Output Path Following with Velocity Assignment* problem, the controller should achieve the following conditions:

1. Path convergence: The system output $\mathbf{y}$, which is function of the system states, converges to the reference path $\mathcal{S}$:

$$\lim_{t\to\infty} \Big|\Big|\mathbf{y}(\mathbf{x}) - s\big(\xi(t)\big)\Big|\Big| = 0.$$

2. Velocity convergence: The path velocity $\dot{\xi}(t)$ converges to a predefined $\dot{\xi}_{ref}(t)$ such that:

$$\lim_{t\to\infty} \Big|\Big|\dot{\xi}(t) - \dot{\xi}_{ref}(t)\Big|\Big| = 0.$$

3. Constraints satisfaction: Constraints on system states and control actions must be satisfied for $t \in [t_0, \infty)$.

One more point, defining path velocity $\dot{\xi}_{ref}(t)$ is not equivalent to trajectory tracking formulation. That is because defining $\dot{\xi}_{ref}(t)$ does not enforce a timed reference state on the controller.

The proposed method is named Model Predictive Path-Following Control (MPFC) [70, 71], and it treats the path parameter ξ as an extra virtual state of the system. This virtual state $\xi(t)$ behavior is governed by an Ordinary Differential Equation (ODE), referred to as *timing law*, that is based on an additional virtual control action ϑ such that $\vartheta \in \mathbb{R}$. The timing law is often chosen as an integrator chain. Here, we use a single integrator as a timing law such that:

$$\dot{\xi}(t) = \vartheta(t) \tag{3.31}$$

Note that $\vartheta(t) \in [0, \infty)$, which means that the path parameter $\xi(t)$ is monotonically increasing towards the reference path parameter ξ_{ref} since $\xi(t)$ is the integration of $\vartheta(t)$. The reference path parameter ξ_{ref} is set to be equal to ξ_f, which marks the end of the path.

The work in [17] shows the effectiveness of MPFC [70] to solve path-following problem for autonomous vehicles. However, it cannot deal with long routes, lane changes or obstacle avoidance since the path is required to defined a priori. Therefore, by proposing our online parametric curve generator and DLC planner, we extend MPFC [70] to solve path-following problem for any desired route length while safely avoiding dynamic obstacles.

3.5.2 Vehicle Model and OCP Formulation and Discretization

The vehicle model used in our MPFC is the vehicle kinematic model (2.1). We re-state the model here again for the reader's convenience:

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = f\Big(\mathbf{x}(t), \mathbf{u}(t)\Big) = \begin{bmatrix} v\cos(\theta + \beta_s) \\ v\sin(\theta + \beta_s) \\ \frac{v\cos(\beta_s)\tan(\theta)}{L} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}, \mathbf{u} = \begin{bmatrix} v \\ \delta_f \end{bmatrix} \tag{3.32}$$

where:

$$\beta_s = \arctan\big(\frac{l_r \tan\delta_f}{L}\big) \tag{3.33}$$

where x and y are the position of the vehicle in X-Y global frame, θ is the vehicle orientation in the global frame, v is the velocity of the ego vehicle at its C.G., β is the side-slip angle of the vehicle, $\mathbf{u}$ is the control actions of the system, l_r is the distance between the rear axle and the C.G., L is the wheel base length of the vehicle, and δ_f is the steering angle. This model is used to control the ego vehicle to follow the collision-free path $\mathcal{P}_{clear}$. Recall that $\mathcal{P}_{clear}$ is parameterized by ξ, H and ρ. Since the lateral shift $\delta(\rho, H)$ is temporary, we focus on following the path $\mathcal{P}(\xi)$ and integrate $\delta(\rho, H)$ later.

Now we augment the system (3.32) and (3.31) together to get:

$$\dot{\mathbf{z}} = \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} f\big(\mathbf{x}(t), \mathbf{u}(t)\big) \\ \vartheta \end{bmatrix}, \mathbf{w} = \begin{bmatrix} \mathbf{u} \\ \vartheta \end{bmatrix}. \tag{3.34}$$

The reference path $\mathcal{S}$ in our case is the parameterized curve $\mathcal{P}$ and its path parameter is ξ. The reference state vector $\mathbf{x}_{ref}$ for a certain ξ can be described as:

$$\mathbf{x}_{ref} = \mathbf{p}(\xi) = \begin{bmatrix} x_{ref} \\ y_{ref} \\ \theta_{ref} \end{bmatrix} \tag{3.35}$$

where x_{ref} and y_{ref} are equal to $\bar{x}(\xi)$ and $\bar{y}(\xi)$ defined in equation (3.18), and θ_{ref} is given by:

$$\theta_{ref} = \arctan\left(\frac{\frac{\partial y_{ref}}{\partial \xi}}{\frac{\partial x_{ref}}{\partial \xi}}\right). \tag{3.36}$$

One point to note is that the second requirement 2 of the controller is replaced with a vehicle reference speed. This reference speed implicitly assigns a reference path velocity.

To formulate the constrained NMPC problem, first we introduce the cost functional J to be minimized in continuous time:

$$J\big(\mathbf{x}, \xi, \mathbf{u}, \vartheta\big) = \int_{t_0}^{t_0+T_H} \Big[\big|\big|\mathbf{x}_{ref} - \mathbf{x}\big|\big|_Q^2 + \big|\big|\dot{\mathbf{u}}\big|\big|_R^2 + \big|\big|v_{ref} - v\big|\big|_\iota^2 + \big|\big|\xi_{ref} - \xi\big|\big|_\mu^2\Big] dt. \tag{3.37}$$

where $Q \in \mathbb{R}^{3\times3}$, $R \in \mathbb{R}^{2\times2}$, $\iota \in \mathbb{R}$ and $\mu \in \mathbb{R}$ are diagonal weighting matrices, t_0 is the initial time, and T_H is the prediction horizon. Therefore the constrained NMPC problem formulation is:

$$\min_{\mathbf{u}(t), \vartheta(t)} \quad J\big(\mathbf{x}(t), \xi(t), \mathbf{u}(t), \vartheta(t)\big) \tag{3.38a}$$

$$\text{s.t.} \quad \dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mathbf{u}(t)\big), \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38b}$$

$$\dot{\xi}(t) = \vartheta(t), \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38c}$$

$$\dot{\mathbf{u}}(t)_{min} \leq \dot{\mathbf{u}}(t) \leq \dot{\mathbf{u}}(t)_{max}, \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38d}$$

$$\mathbf{u}_{min}(t) \leq \mathbf{u}(t) \leq \mathbf{u}_{max}(t), \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38e}$$

$$\xi_0(t) \leq \xi(t) \leq \xi_{ref}(t), \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38f}$$

$$0 \leq \vartheta(t) < \infty, \qquad \forall t \in [t_0, t_0 + T_H] \tag{3.38g}$$

where $\dot{\mathbf{u}}(t)_{min}$ and $\dot{\mathbf{u}}(t)_{max}$ are the upper and lower limits on the vehicle acceleration and the steering angle rate, $\mathbf{u}_{min}$ and $\mathbf{u}_{max}$ are the upper and lower limits on the vehicle speed

and the steering angle, $\xi_0(t)$ represents the starting point of the path and $\xi_0(t) = 0$ in our case, and $\xi_{ref}(t)$ represents the end point of the path and is set such that $\xi(t)_{ref} = 1$.

In order to solve the OCP (3.38) numerically, we discretize it using Runge-Kutta 4th order method (RK4). The discretized OCP is then reformulated as Nonlinear Programming (NLP) problem by multiple shooting technique using an open-source numerical optimization tool called CasADi [68]. Finally, the NLP problem is solved using CasADi and the interior point optimizer (IPOPT) [69].

The aforementioned NMPC controller can accurately follow the parametric curve $\mathcal{P}$ with minimal deviation and while tracking the reference speed as close as possible. However, although this formulation shows promising results in path-following and has high potential to improve the current existing control of autonomous vehicles [17], it is not suitable for long routes, and does not handle obstacle avoidance problem.

Therefore, as stated before, a global route $\mathcal{R}_{global,ref}$ cannot be described as a one parametric curve. Thus, by continuously generating new parametric curves that start from the current location of the vehicle and covers the global route $\mathcal{R}_{global,ref}$ for a certain dynamic lookahead distance $d_{LA,dyn}$, the MPFC scheme can be extended for much longer trips. Note that at each time the parametric curve $\mathcal{P}$ is updated, path parameter ξ needs to be reset such that $\xi = \xi_{reset}$, and the ξ_{reset} value depends on the vehicle desired speed. Lastly, the method proposed in section 3.4.3 is used to solve obstacle avoidance problem by adding the lateral shift $\delta(\rho, H)$ directly to $\mathbf{x}_{ref}$ to form $\mathcal{P}_{clear}$. Note that ρ and H are not included in OCP (3.38) as they are externally set based on the interaction with obstacles.

Finally, several simulations have been conducted and discussed in 6.2 to validate all of the aforementioned proposed techniques with comparisons to other existing schemes in the literature.

Chapter 4

Path-following and Adjustable Driving Behavior of Autonomous Vehicles using Dual-Objective Nonlinear MPC [1]

4.1 Introduction

Driving behavior is a main concern in automotive industry and road traffic safety because of its effects on the environment in terms of energy consumption as well as the safety of other road users [10, 11]. Particularly, autonomous vehicles are expected to achieve better driving behavior compared to human driving to ensure both the safety of other road users, and the comfort of the vehicle passengers [16]. Moreover, this driving behavior should be adjusted according to the preference of the vehicle passengers.

Here, we propose a control methodology that changes the driving behavior of autonomous vehicles using a dual-objective NMPC. The controller changes the driving behavior based on the passengers preference, energy saving mode or sport mode, while following a given reference path. This is accomplished by changing the weights of each term in the dual-objective cost function used in NMPC. A kinematic bicycle model is used as the prediction model of the NMPC controller based on the comparison made in 2.3.3.

A real-time simulation model adapting the kinematic bicycle model is developed on Gazebo dynamic simulator [19]. Moreover, the proposed controller implementation is linked to the simulator using the Robot Operating System (ROS) [73]. The proposed controller performance is tested and validated with different road shapes.

[1] Some of the content of this chapter was published as Mohamed A. Daoud, Mostafa Osman, Mohamed W. Mehrez, and William W. Melek, "Path-following and Adjustable Driving Behavior of Autonomous Vehicles using Dual-Objective Nonlinear MPC", in *IEEE Int. Conf. of Vehicular Electronics and Safety (ICVES), 2019* [72].

4.2 Bicycle and Energy Consumption Models

In this section, first we present the kinematic bicycle model used for the motion prediction of the vehicle. Next, we describe the adopted vehicle energy consumption model.

4.2.1 Bicycle Model

The kinematic bicycle model used here is derived by expressing wheels' rolling and sliding constraints. By combining the motion of each wheel under these constraints, the bicycle model is obtained [30]. This model expresses vehicle's motion in terms of the front wheel angular speed and its steering angle, which can be used to compute rear wheel speed. Fig. 4.1 shows a schematic diagram of the bicycle model, where l is the vehicle's wheel base, θ is the heading angle relative to the inertial frame, $C.G.$ is the center of gravity of the vehicle, ICR is the instantaneous center of rotation, and δ_f is the front wheel steering angle.

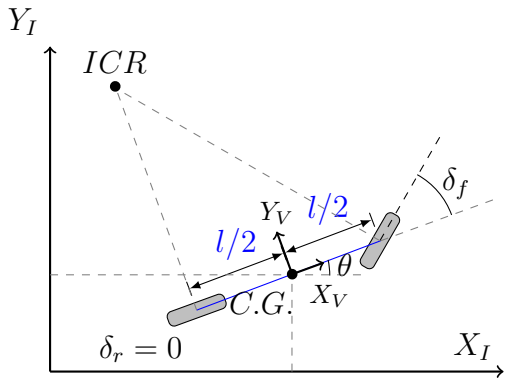


Figure 4.1: Schematic diagram of the bicycle model, where I is the inertial frame and V is the vehicle local frame.

The discrete-time bicycle kinematic model can be written as follows:

$$\underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix}}_{\mathbf{x}_k} = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_k) = \mathbf{x}_{k-1} + t_s \underbrace{r\dot{\phi}_f \begin{bmatrix} \cos\delta_f \cos\theta - \frac{\sin\delta_f}{2} \sin\theta \\ \cos\delta_f \sin\theta + \frac{\sin\delta_f}{2} \cos\theta \\ \frac{\sin\delta_f}{l} \end{bmatrix}}_{\mathbf{f}_c(\mathbf{x}(t),\mathbf{u}(t))},$$

where $\mathbf{f}_c$ is the continuous time model and $\mathbf{f}_d$ is its Euler discretization, k is the current time step, $\mathbf{x} = \begin{bmatrix} x & y & \theta \end{bmatrix}^\top \in \mathbb{R}^3$ is the state vector of the vehicle which is composed of the planar Cartesian coordinates of the center of the vehicle $[x, y]$ as well as the heading angle θ, and $t_s \in \mathbb{R}_{>0}$ is the sampling time. For the completeness of the mathematical formulation, the relation between rear wheel angular speed and front wheel angular speed is given by:

$$\dot{\phi}_r = \dot{\phi}_f \cos(\delta_f).$$

4.2.2 Energy Consumption Model

In order to model the energy consumption in an electric ground vehicle, motor torque expressed in equation (4.1) is used.

$$T_m = \frac{r}{G}\left(m_v a_{long} + m_v g C_R + \frac{1}{2}\rho_a C_d A_f v^2\right) \tag{4.1}$$

where T_m is the electric motor torque, r is the wheel radius, G is the total reduction ratio, m_v is the vehicle mass, a_{long} is the vehicle longitudinal acceleration, g is the gravitational acceleration, C_R is the rolling resistance, ρ_a is the air density, C_d is the drag coefficient, A_f is the frontal area of the vehicle, and v is the vehicle longitudinal speed. The motor torque equation (4.1) considers only motion in the longitudinal direction of the vehicle without taking into consideration the lateral forces. This is because the longitudinal motion consumes the major amount of energy in a ground vehicle.

Using equation (4.1), the total input energy $\mathcal{E}_{in}$ to the vehicle can be calculated through integrating the input power over time as shown below.

$$\mathcal{E}_{in} = \int_0^{t_f} \frac{T_m\ \omega}{\eta} dt, \tag{4.2}$$

where $\eta \in \mathbb{R}_{>0}$ represents the electric motor efficiency, ω is the angular speed of the electric motor in rad/sec, and t_f is the final time of the vehicles operation.

4.3 Proposed Dual-Objective NMPC

In this section, we start by explaining the path-following control problem and our proposed solution approach. Next, we formulate the OCP of the nonlinear model predictive control (NMPC) scheme. Finally, we introduce the proposed dual-objective cost function utilized to change the behavior of the controller as discussed in Section 4.1.

4.3.1 Path-following Control Problem

We are considering a path-following control using both vehicle's speed and steering as control inputs. Let us define the vehicle current state by $\mathbf{x}_k$, and the final target state by $\mathbf{x}_f$, where $\mathbf{x}_k, \mathbf{x}_f \in [\mathbf{x}_{min}, \mathbf{x}_{max}]$, and $\mathbf{x}_{min}$, $\mathbf{x}_{max}$ are the upper and lower limits of state $\mathbf{x}$, respectively. Moreover, we assume that $\mathbf{x}_k$ and $\mathbf{x}_f$ are points on a geometric path $\mathcal{S}$. Therefore, the path-following problem now requires that the vehicle state $\mathbf{x}_k$ moves along the path $\mathcal{S}$ towards the path end point $\mathbf{x}_f$, while satisfying the state and control constraints.

Since the driving behavior changes based on the desired mode, $\mathcal{S}$ is not parameterized in time as the path timing itself is not known a priori [74]. Consequently, we parameterize $\mathcal{S}$ in a scalar path variable ε, i.e. $\mathcal{S} = s(\varepsilon)$. Now, the proposed path-following problem is summarized in Alg. 5. In this algorithm, at each iteration, we calculate the actual path variable ε_{act} based on the current state $\mathbf{x}_k$ and the parametric representation of $\mathcal{S}$. Then, we add an increment $\psi \in \mathbb{R}$ to ε_{act} to obtain the reference path variable ε_{ref}. Finally, ε_{ref}

is used to find the next reference state $\mathbf{x}_{ref}$. Here, we use κ_{step} as a path evolution step-size parameter that determines the smoothness of the transition along the path $\mathcal{S}$. Fig. 4.2 shows a schematic of the proposed path-following methodology. Note that the proposed path-following approach in Fig. 4.2 is simpler than the method used in 3.5.

Algorithm 5: Path-following

foreach *time step* k **do**
  $Error = norm(\mathbf{x}_f - \mathbf{x}_k)$
  $Allowable\_Error = 0.3$
  if $Error > Allowable\_Error$ **then**
    $\psi = \kappa_{step} - \frac{\kappa_{step}}{1 + Error}$
    $\varepsilon_{act} = s^{-1}(\mathbf{x}_k)$
    $\varepsilon_{ref} = \varepsilon_{act} + \psi$
    $\mathbf{x}_{ref} = s(\varepsilon_{ref})$
  else
    End of path reached
  end
end

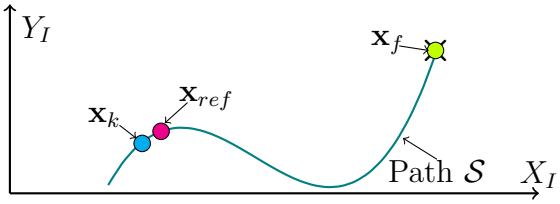


Figure 4.2: A schematic of the path-following problem.

Notice that the formulation of the increment ψ allows it to automatically decrease as the error between the current state and target state becomes smaller as shown below, i.e. $\lim_{\mathbf{x}_k \to \mathbf{x}_f} \psi = 0$.

4.3.2 NMPC Formulation

The constrained nonlinear model predictive control (NMPC) problem can be formulated as shown in equation (4.3). As can be seen in the equation, the problem is to minimize a cost function (introduced below) subject to the system kinematic model as well as the bounded constraints on the states, the controls, and the change in controls.

$$\min_{\mathbf{U}} \quad J(\mathbf{U}) \tag{4.3a}$$

$$\text{s.t.} \quad \mathbf{x}_{k+1} = \mathbf{f}_d(\mathbf{x}_{k-1}, \mathbf{u}_k), \quad \forall k \in [k, k+N] \tag{4.3b}$$

$$\Delta\mathbf{U}_{min} \leq \Delta\mathbf{U} \leq \Delta\mathbf{U}_{max}, \forall k \in [k, k+N-1] \tag{4.3c}$$

$$\mathbf{U}_{min} \leq \mathbf{U} \leq \mathbf{U}_{max}, \quad \forall k \in [k, k+N-1] \tag{4.3d}$$

$$\mathbf{x}_{min} \leq \mathbf{x} \leq \mathbf{x}_{max}, \quad \forall k \in [k, k+N] \tag{4.3e}$$

In (4.3), $J : \mathbb{R}^{2\times N} \to \mathbb{R}_{\geq 0}$ is the cost function, $\mathbf{u}_k := [\dot{\phi}_f, \delta_f]^\top \in \mathbb{R}^2$ is the control action at time k, and $\mathbf{U} := \{\mathbf{u}_k, ..., \mathbf{u}_{k+N-1}\} \in \mathbb{R}^{2\times N}$ is the set of control actions over the prediction horizon N, and $\mathbf{U}_{min}$, $\mathbf{U}_{max}$ are the upper and lower limits on $\mathbf{U}$, respectively. Also, $\Delta\mathbf{u}_k = \mathbf{u}_k - \mathbf{u}_{k-1}$ is the change in the control action at time k, where for $k = 0$, $\Delta\mathbf{u}_0 = \mathbf{u}_0 - \mathbf{p}$ where $\mathbf{p}$ is the current measured wheel angular speed $\dot{\phi}_f$ and steering angle δ_f. $\Delta\mathbf{U} := \{\Delta\mathbf{u}_k, ..., \Delta\mathbf{u}_{k+N-1}\} \in \mathbb{R}^{2\times N}$ is the set of $\Delta\mathbf{u}_k$ over N, and $\Delta\mathbf{U}_{min}$, $\Delta\mathbf{U}_{max}$ are the upper and lower limits on $\Delta\mathbf{U}$, respectively.

4.3.3 Dual-Objective Cost Function

Here, we discuss our main contribution which is the use of a dual-objective cost function to consider different driving behaviors. By changing the weight of each element in the cost function, the NMPC solution differs to adjust the controller behavior based on the passengers preference.

Furthermore, by adding an economic term that represents the energy consumption to the cost function, NMPC considers minimizing the power at each iteration. Therefore, the produced control actions reduce the needed driving power as well as the error to the reference signal.

To compute the total needed energy for a certain given path, equation (4.2) is used. To reduce the total consumed energy, either the motor efficiency or the output power, represented in terms of torque and angular speed, can be added to the cost function. In our method, we use the output power in the cost function. This showed an efficient and more comfortable driving behavior in our analysis. The proposed quadratic dual-objective cost function is given by:

$$J = \underbrace{\sum_{k=0}^{N} ||\mathbf{x}_{ref} - \mathbf{x}_k||_Q^2 + \sum_{k=0}^{N-1} ||\Delta \mathbf{u}_k||_R^2}_{J_r} + \underbrace{\sum_{k=0}^{N} ||T_m\ \omega||_E^2}_{J_e}, \tag{4.4}$$

where J_r is the regulating cost, J_e is the economic cost, $Q \in \mathbb{R}^{3\times3}$, $R \in \mathbb{R}^{2\times2}$, and $E \in \mathbb{R}$ are diagonal weighting matrices.

By reducing the weight of the output power in J_e, the NMPC controller focuses more on minimizing the error to the reference path signal and vice versa. Thus, we can obtain different driving behaviors by modifying the weights Q, R, and E as shown in Fig. 4.3.

Figure 4.3: Effect of changing E in the dual-objective function.

We present our results for the proposed approach in 6.3.

Chapter 5

Obstacle Representation using Fourier Series

5.1 Introduction

Modeling the static map and the obstacles in the ego vehicle environment is crucial to ensure the safety of the obtained motion plans from the approach proposed in Chapter 3. Since collision detection is performed online, it should be reliable, accurate and computationally efficient. Collision detection requires proper obstacle spatial representation in the environment of the vehicle. The majority of the current approaches for obstacle representation use conservative approximations such as unions of overlapping spheres [60] to ensure the safety of the obtained plans. However, since obstacles usually have irregular shapes, there is a trade-off between accuracy of the representation and its computational complexity.

Inspired by the work in [75], in this chapter we propose the usage of Complex Fourier Series (CFS) to create mathematical models for geometric shapes and obstacles an ego vehicle may encounter while driving autonomously. Briefly, the proposed method works as follows:

1. Sample the boundary of the obstacle equally.
2. Determine number of frequencies needed based on the obstacle shape.
3. Compute Fourier Series.

In the next section, we give a brief on CFS mathematical background. Afterwards, we present our methodology which uses CFS for obstacle representation.

5.2 Complex Fourier Series Expansion

Joseph Fourier has shown from the study of two differential equations, the wave and the heat equations, that real periodic mathematical equations can be expanded using infinite

trigonometric series named Fourier Series (FS) expansion [76, 77]. Firstly, Fourier trigonometric series is given by:

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \Big[a_n \cos(n \cdot 2\pi t) + b_n \sin(n \cdot 2\pi t) \Big] \tag{5.1}$$

where a_0, a_n, b_n are called Fourier constants, and $n \in \mathbb{N}$ such that $n = 1, 2, \ldots, \infty$. Since FS is periodic, the periods of $\cos(n \cdot 2\pi t)$ and $\sin(n \cdot 2\pi t)$ are $\frac{1}{n}$. Accordingly, Fourier series (5.1) can represent periodic functions that are of period 1 [76]. The Fourier coefficients can be calculated using the following expressions:

$$\begin{aligned} a_n &= 2 \textstyle\int_0^1 f(t) \cos(n \cdot 2\pi t)\ dt, \quad n = 0, 1, 2, \ldots \\ b_n &= 2 \textstyle\int_0^1 f(t) \sin(n \cdot 2\pi t)\ dt, \quad n = 1, 2, 3, \ldots \end{aligned} \tag{5.2}$$

However, equation (5.1) can be further extended and written in the complex exponential form using Euler's formula (5.3).

$$\begin{aligned} e^{\pm j\theta} &= \cos\theta \pm j \sin\theta \\ \cos\theta &= \tfrac{1}{2}(e^{j\theta} + e^{-j\theta}) \\ \sin\theta &= \tfrac{1}{2j}(e^{j\theta} - e^{-j\theta}) \end{aligned} \tag{5.3}$$

Therefore, a periodic complex function $f(t)$ can be written as a sum of complex exponentials as follows:

$$f(t) = \sum_{n=n_{low}}^{n_{up}} c_n e^{n \cdot j2\pi t} \tag{5.4}$$

where $t \in [0, 1]$, n_{low} and n_{up} are lower and upper frequency limits (ideally $n_{low} = -\infty$ and $n_{up} = \infty$), and c_n are complex constants. The constants c_n can be calculated by:

$$c_n = \int_0^1 f(t) \cdot e^{-n \cdot j2\pi t} dt. \tag{5.5}$$

The output space of a periodic complex function $f(t)$ is two dimensional, and is referred to as the complex space. Usually, Argand diagram is used to represent complex numbers geometrically using Cartesian axes, where the real number line is assigned to the *x-axis*, while the imaginary number line is assigned to the *y-axis* as shown in Fig. 5.1.

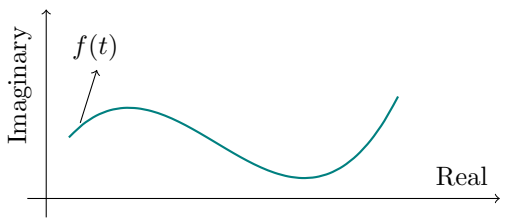


Figure 5.1: The Argand diagram for a complex function $f(t)$.

In the next section, we will show how CFS can be used for obstacle representation.

5.3 Obstacle Representation

A proper obstacle representation technique should be computationally efficient, stable, scalable, and can handle different of shapes. Since autonomous driving problem is mainly considered in two dimensional environments, obstacles can be treated as complex functions. These complex functions model the boundaries of the obstacles, making obstacle representation more accurate and robust. The main reason behind using CFS instead of FS is to find a proper representation by calculating only one integration numerically, equation (5.5), rather than four different integrations as in equation (5.2). Since only one integration operation is performed using CFS as opposed to four, CFS will be much faster, and hence, computationally efficient for obstacle representation in real time implementation.

In order to make such a representation for an obstacle $\mathcal{O}_i$, first we evenly sample the outer boundary of $\mathcal{O}_i$ into p_i points as shown in Fig. 5.2. Next, to find the complex

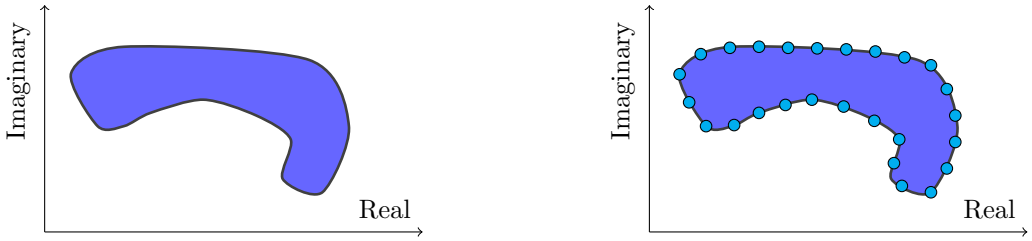


Figure 5.2: An obstacle and its sampled boundary.

coefficients c_n, equation (5.5) is calculated numerically by converting it into a summation such that:

$$c_n = \sum_{n=0}^{p_i} f(\tau) \cdot e^{-\tau \cdot j2\pi}, \ \tau = \frac{n}{p_i}. \tag{5.6}$$

Note that $\tau \in [0, 1]$. Increasing the number of boundary samples p_i of obstacle $\mathcal{O}_i$ increases the accuracy of the obtained representation. Note that equations (5.1) and (5.2) can be used to obtain the same representation, but it requires doing the analysis for the values of the *x-axis* and the values of the *y-axis* separately. Instead, after finding the complex constants c_n, FS real representation for *x-axis* values and *y-axis* values can be obtained knowing that:

$$c_n = A_n + jB_n \tag{5.7}$$

and by using Euler's formula (5.3):

$$c_n \cdot e^{n \cdot j2\pi\tau} = \underbrace{[A_n \cos(n \cdot 2\pi\tau) - B_n \sin(n \cdot 2\pi\tau)]}_{x(\tau)} + j \underbrace{[B_n \cos(n \cdot 2\pi\tau) + A_n \sin(n \cdot 2\pi\tau)]}_{y(\tau)} \tag{5.8}$$

and

$$a_{x,n} = A_n, \ b_{x,n} = -B_n, \ a_{y,n} = B_n, \ b_{y,n} = A_n \tag{5.9}$$

where $a_{x,n}$, $b_{x,n}$, $a_{y,n}$ and $b_{y,n}$ are Fourier constants for Fourier representation of x values and y values respectively. This is useful to omit the usage of complex numbers after obtaining such a representation.

The key question is how to use CFS to create obstacles representation. We propose two different methods to create these representation depending on the nature of the considered obstacle and the mode of operation:

- Offline method: For the most common obstacles, create a library offline of accurate representations using a top view image of these obstacles.
- Online method: While driving, create a simple representation for the environment using the available sensory information.

5.3.1 Offline Method

Since this method is entirely performed offline, the computational cost of obtaining an accurate representation is ignored. To sample obstacle boundary with relatively large number of samples, a top view image is used. We use computer vision techniques to suppress the obstacle to its outer boundary. Afterwards, by taking the outer boundary pixels as samples, an accurate representation can be obtained. Note that the scale between the used image and the actual obstacle is required to enlarge the obtained representation to the obstacle actual size.

After getting the outer boundary, its pixels are used as boundary samples p_i and accordingly, equation (5.6) is used to find c_n complex constants. To ensure the safety of the obtained obstacle representation, we scale up the representation by 1.5%. This percentage is sufficient to cover any small protruding parts of the obstacle that was not modeled while generating the representation, and acts as a safety factor in case of inaccuracy in the scale between the actual obstacle size and its top view image. Fig. 5.3 shows an example of obstacle representation generation using the offline method.

Increasing the number of frequencies, e.g. increasing n_{up} or decreasing n_{low}, required to generate such a representation will result in a more accurate representation. However, as the number of frequencies increases, the computational cost of collision detection becomes higher. Alg. 6 summarizes the proposed offline method, where $\mathcal{I}$ is the top view image, and $\mathcal{SC}$ is the scale between the image and actual size.

Algorithm 6: Obstacle Representation: Offline Method

Input: $\mathcal{I}$, $\mathcal{SC}$,n_{low},n_{up}
Output: $a_{x,n}$,$b_{x,n}$,$a_{y,n}$,$b_{y,n}$
1 Outer_Boundary = getBoundary($\mathcal{I}$)
2 xVal, yVal = getOrderedCoordinates(Outer_Boundary)
3 $a_{x,n}$, $b_{x,n}$, $a_{y,n}$, $b_{y,n}$ = getFourierRep($\mathcal{SC}$, xVal, yVal, n_{low}, n_{up})

5.3.2 Online Method

Since the scenarios autonomous vehicles may encounter while driving are countless, modeling all kinds of obstacles is unfeasible. Herein, we propose our online obstacle representation method for unknown obstacles based on the available sensory information. Usually,

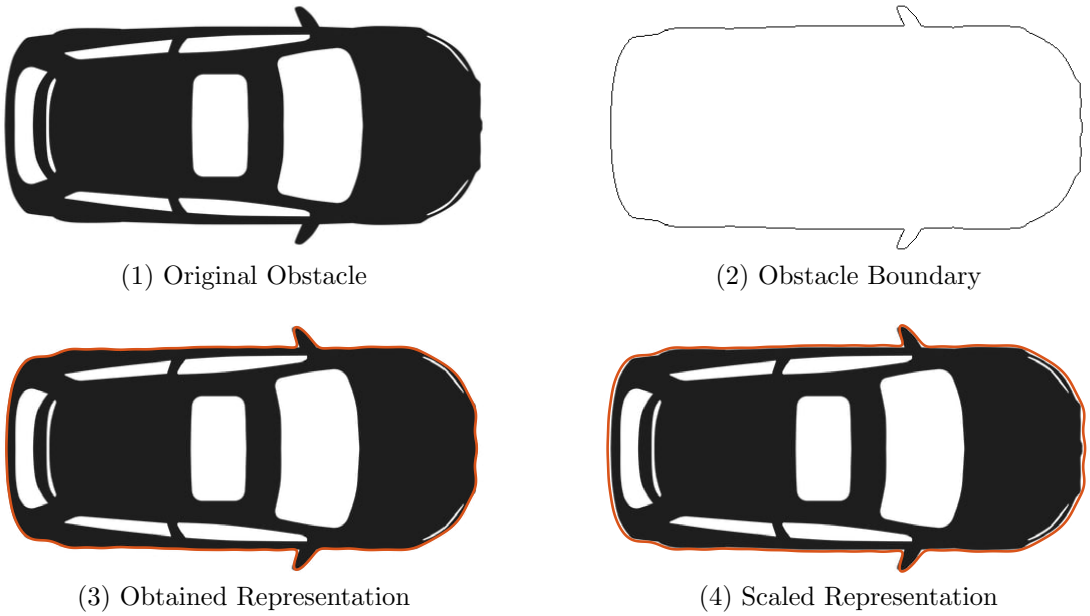


Figure 5.3: An example of the offline method used to generate obstacle representation.

an autonomous vehicle sensor suite is equipped by a stereo vision system, a lidar, a radar, and ultrasonic sensors [78]. These sensors provide depth information of the surrounding environment.

Since the depth information of obstacles in the scene are measured from the prospective of the ego vehicle, this information shows the significant parts of the obstacles that the ego vehicle is facing. Therefore, the measured free space around the ego vehicle can be modeled as one mathematical equation using the aforementioned approach.

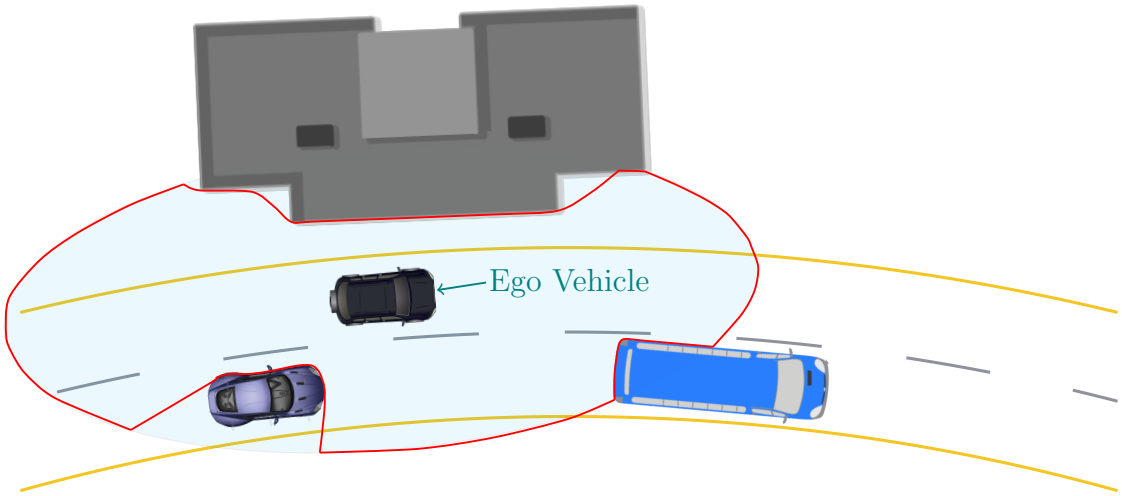


Figure 5.4: An example of the online method used to generate free space representation.

Fig. 5.4 shows an example of a map with three obstacles in the scene, where the ellipse shows the range of the ego vehicle sensors, and the red loop shows the measured depth information. Note that this approach does not require any computer vision preprocessing and relies entirely on the available depth information.

Because of the dynamic nature of the obstacles in the scene, e.g. relative speed between the ego vehicle and static and dynamic obstacles, the relative distance to the ego vehicle is variable in time. Consequently, the red loop shown in Fig. 5.4 is dynamically changing. Therefore, the dynamic generation of the free space representation using the available sensory data must be done ahead of time to allow for corrective action when risk of collision exists. The proposed algorithm provides fast generation of the free space since it is not an optimization-based method. Instead, it relies on numerical integration to find such a representation. This representation would be useful to ensure that the ego vehicle does not collide with any obstacles in the scene. Alg. 7 summarizes the proposed online method where $\mathcal{D}$ is depth information.

Algorithm 7: Free Space Representation: Online Method

Input: $\mathcal{D}$

Output: $a_{x,n}$,$b_{x,n}$,$a_{y,n}$,$b_{y,n}$

1 Outer_Boundary = getBoundary($\mathcal{D}$)

2 xVal, yVal = getOrderedCoordinates(Outer_Boundary)

3 $a_{x,n}$, $b_{x,n}$, $a_{y,n}$, $b_{y,n}$ = getFourierRep(xVal, yVal, n_{low}, n_{up})

A significant note on FS approach is that it is vulnerable to discontinuities. This means that, it cannot be used directly to model open loops. The reason behind this limitation is that FS method is suitable only for periodic functions. To overcome this limitation, the open loop can be sampled such that $\tau \in [0, 0.5]$ in equation (5.6), and resampled again in the reverse direction such that $\tau \in (0.5, 1]$. This reverse sampling will form closed loops over the open loops and will represent them accurately. In such case, to use the obtained representation, there will be no need to use the full range of τ, e.g. $\tau \in [0, 1]$. Instead, half of the period would be sufficient.

To conclude, in this chapter we proposed the usage of Fourier Series for obstacle representation to create accurate mathematical models in relatively short time. Two methods were proposed; offline and online methods. The offline method creates accurate models for well known obstacles, while the online method creates models for the free space in case of interacting with unknown obstacles based on the available depth information. We run the proposed methodology on different obstacles and maps to validate its efficacy and the results obtained are presented in 6.4.

Chapter 6

Simulation Results and Benchmarking

6.1 Introduction

In this chapter we present the results obtained for the proposed methodologies in Chapter 3, Chapter 4, and Chapter 5. First, in section 6.2 we show the validation of the Simultaneous Local Motion Planning and Control scheme. Afterwards, in section 6.3 we present the results of the proposed Driving Behavior Control scheme. Finally, in section 6.4 we illustrate the efficacy of Fourier Series usage for obstacle representation.

6.2 Simultaneous Local Motion Planning and Control Results

Here, we present our results obtained for the methodology proposed in Chapter 3. To assess the proposed scheme, we test it using three different cases:

1. **Case 1:** Two LC maneuvers with different longitudinal distance.
2. **Case 2:** Two DLC maneuvers to avoid obstacle vehicles and one LC maneuver.
3. **Case 3:** One DLC for multiple obstacles avoidance.

These cases were performed using CARLA simulator [18], which is an open-source simulator for autonomous driving research. CARLA simulator has the potential to simulate cars in realistic scenarios, where the environment and other road users interact dynamically with the ego vehicle. A screen recording videos for each case has been uploaded on YouTube at the following link: www.youtube.com/playlist?list=PL5vYvbYXildsANRkOL73jP1fmbezyEqES

To generate $\mathcal{R}_{global}$ for each case, we rely on CARLA builtin route planner. The ultimate goal for each case is to pre-process the $\mathcal{R}_{global}$ using the Offline Route Feasibility Checker 3.3 to obtain $\mathcal{R}_{global,ref}$. Upon obtaining the reference route $\mathcal{R}_{global,ref}$, the ego vehicle follows it using the Online Parameterized Curves Generator 3.4 and Model Predictive Path-Following Control 3.5. Table 6.1 summarizes the details for each case.

Table 6.1: Cases performed to test the proposed scheme.

Case	Route Length	LC maneuvers	Dynamic Obstacles
1	386.67 [m]	2	0
2	976.06 [m]	1	2 separated obstacles
3	657.74 [m]	0	3 obstacles in a row

6.2.1 Offline Route Feasibility Checker Results

As mentioned before, global planners may or may not consider vehicles constraints in planning LC maneuvers. CARLA global planner used in our testing ignores vehicles constraints. Hence, three different LC maneuvers have been tested and rerouted by the introduced method in 3.3 using sigmoid functions. To obtain feasibility-guaranteed maneuvers, the longitudinal distance $d_{long,LC}$ has been calculated based on speed limit and maximum comfortable lateral acceleration as described by equations (3.2),(3.8) and (3.10). Fig. 6.1 shows the three LC maneuvers and table 6.2 shows their parameters.

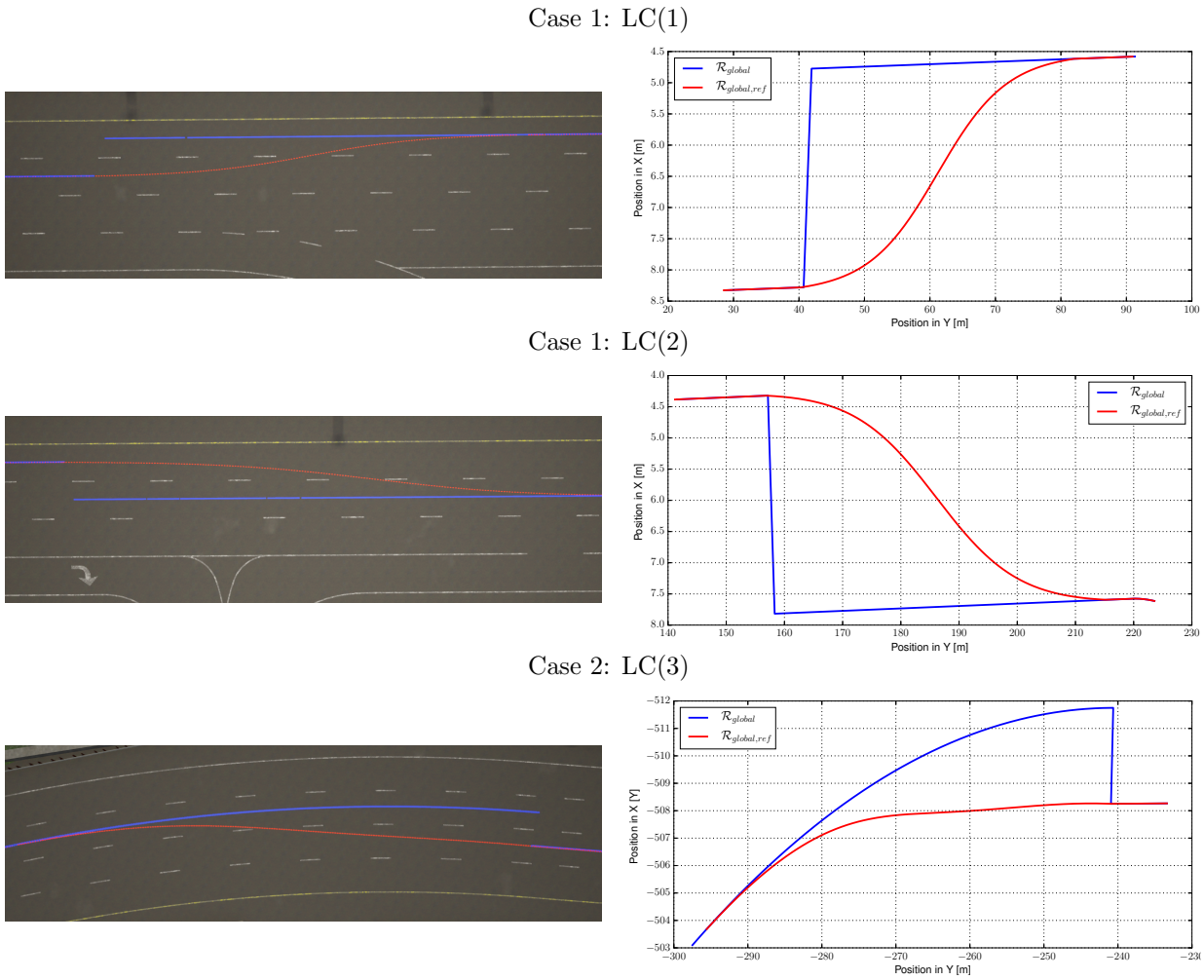


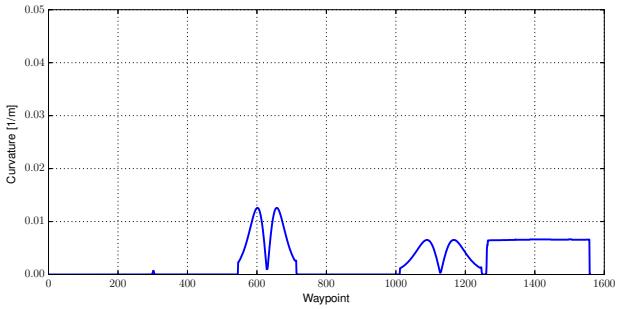
Figure 6.1: Left: Snapshots for the three LC maneuvers from CARLA, where $\mathcal{R}_{global}$ is in blue and $\mathcal{R}_{global,ref}$ is in red. Right: Plots showing longitudinal and lateral distances for each maneuver.

Table 6.2: LC maneuvers performed to test the proposed scheme.

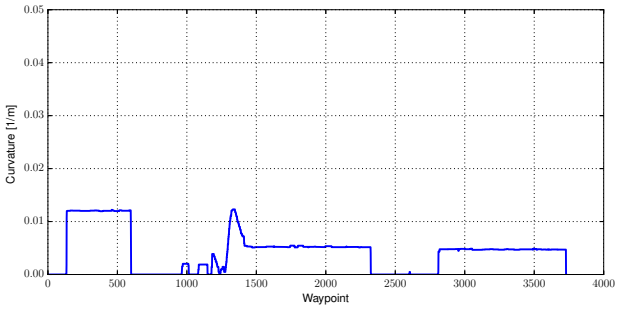
Lane Change	a_{lat}	v_{lim}	κ	a	$d_{long,LC}$
Case 1: LC(1)	1.0 [m/s^2]	30 [km/h]	0.014400 [1/m]	0.191799	41.71 [m]
Case 1: LC(2)	1.0 [m/s^2]	50 [km/h]	0.005184 [1/m]	0.137464	58.20 [m]
Case 2: LC(3)	1.0 [m/s^2]	50 [km/h]	0.005184 [1/m]	0.137464	58.20 [m]

Although [12] and [13] have used sigmoid functions to plan LC maneuvers to ensure their smoothness, equation generation has not been reported. On the other hand, our introduced approach finds the best feasible maneuver based on the vehicle constraints. Moreover, as it can be seen in Fig. 6.1, the introduced framework succeeds in planning LC maneuvers and integrates it with the original route regardless of road orientation.

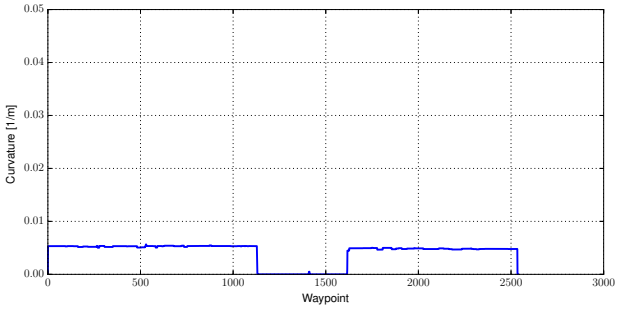
After obtaining $\mathcal{R}_{global,ref}$, road curvature information is then calculated using equations (3.12) and (3.13). Road curvature information is used in the online parameterized curves generator as proposed in 3.4 to change the lookahead distance dynamically. Calculated road curvatures for each case are shown in Fig. 6.2.



Case 1: curvature



Case 2: curvature



Case 3: curvature

Figure 6.2: Road curvature information for each scenario.

6.2.2 Online Curve Generator Results

Given the reference global route $\mathcal{R}_{global,ref}$ represented as waypoints and road curvature information, this module finds the nearest waypoint on the global route to the actual location of the vehicle. Afterwards, a subset of the waypoints is taken to generate a reference parametric curve $\mathcal{P}$ using two fifth order splines, one for X values and one for Y values.

Fig. 6.3 illustrates samples of the results. Note that the length of the fitted curve differs from one sample to another. Table 6.3 outlines the the maximum and absolute average of fitting error and computational time needed where the maximum values are highlighted in yellow. The recorded time is the total time for finding the nearest waypoint to the ego vehicle location, determining the dynamic lookahead distance, waypoints preparation, and solving the LS problem to generate the parametric curve. The maximum obtained error 0.0162 [m] value is acceptable compared to typical lane width (3.6 - 3.75) [m]. To synchronize curve generation time, we use a frequency of 2 Hz to generate new parametric curve. In other words, the reference parametric curve $\mathcal{P}$ is updated twice every second. The maximum computation time (0.1188 [s]) in the worst case is less than a quarter of the corresponding time of the used update frequency ($0.5/4 = 0.125$ [s]) which makes the worst case acceptable.

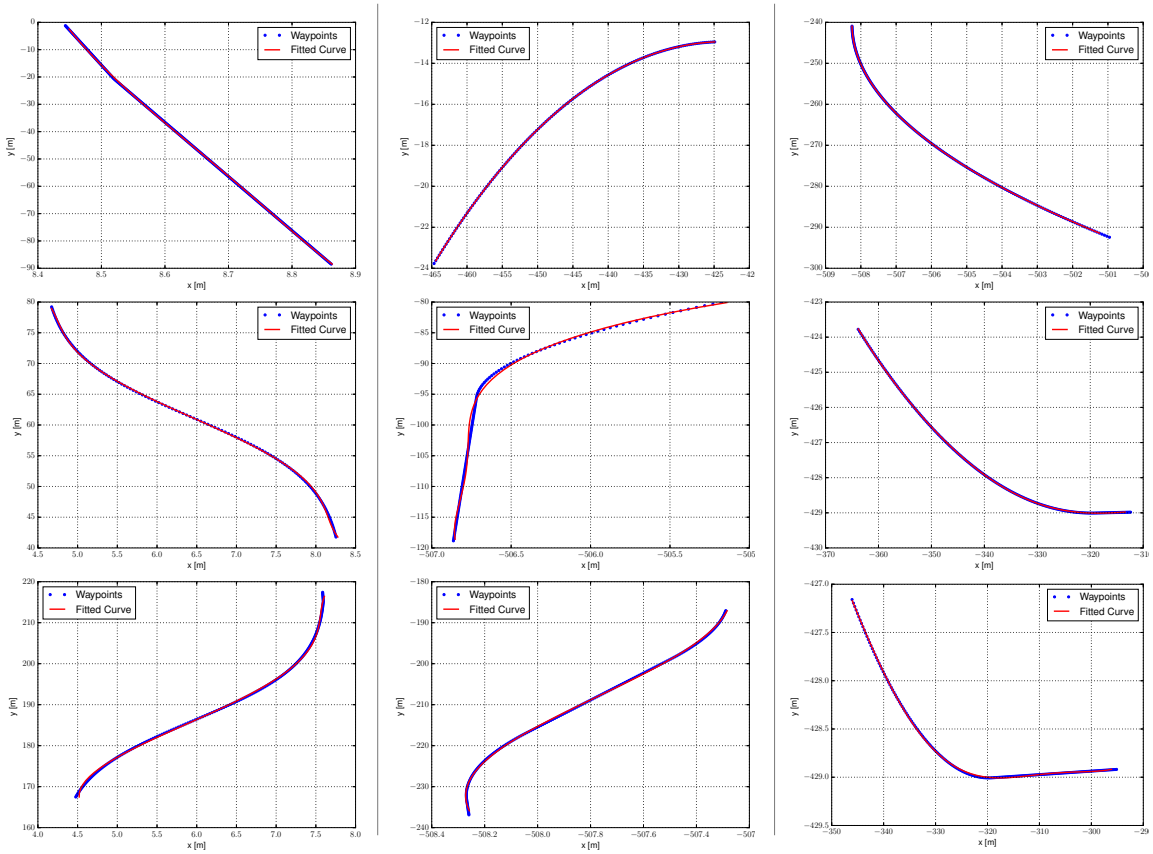


Figure 6.3: Sample of curve fitting results and waypoints for the three scenarios. Right: Case 1. Middle: Case 2. Left: Case 3.

Table 6.3: Curve fitting computation time and error.

Case	Avg. Fitting Error	Max. Fitting Error	Avg. Fitting Time	Max. Fitting Time
1	0.0088 [m]	0.0714 [m]	0.0539 [s]	0.0792 [s]
2	0.0162 [m]	0.1384 [m]	0.0791 [s]	0.1107 [s]
3	0.0054 [m]	0.0469 [m]	0.0821 [s]	0.1188 [s]
Mean	**0.0365 [m]**	**0.0856 [m]**	**0.0717 [s]**	**0.1029 [s]**

As stated before, the lookahead distance $d_{LA,dyn}$ is set dynamically based on road curvature. The lookahead distance varies such that $d_{LA,dyn} \in [12.5, 100]$ m, and is determined according to the following piece-wise equation:

$$d_{LA,dyn}(\kappa_{max}) = \begin{cases} 100 \text{ [m]} & , \kappa_{max} < 0.0002 \\ 87.5 \text{ [m]} & , 0.0002 \leq \kappa_{max} < 0.0010 \\ 75.0 \text{ [m]} & , 0.0010 \leq \kappa_{max} < 0.0016 \\ 62.5 \text{ [m]} & , 0.0016 \leq \kappa_{max} < 0.0100 \\ 50.0 \text{ [m]} & , 0.0100 \leq \kappa_{max} < 0.0200 \\ 37.5 \text{ [m]} & , 0.0200 \leq \kappa_{max} < 0.0350 \\ 25.0 \text{ [m]} & , 0.0350 \leq \kappa_{max} < 0.0360 \\ 12.5 \text{ [m]} & , 0.0360 \leq \kappa_{max} \end{cases} \tag{6.1}$$

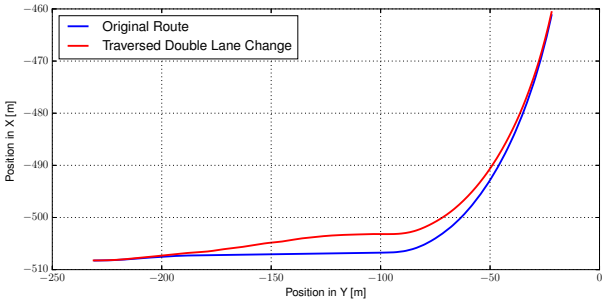
Since a new $\mathcal{P}$ is obtained every half a second, the shortest lookahead distance is still relatively feasible. That is because the ego vehicle cannot reach the end of the path before receiving the next one since its speed cannot cover the current parametric curve in less than 0.5 second.

Dynamic obstacle avoidance is performed using the proposed method in 3.4.3. Given the location and the velocity of the obstacles on the road, a bell-shaped function is created to perform a feasibility-guaranteed DLC maneuver for each obstacle. In case of multiple obstacles moving in row, or long obstacles like buses, the longitudinal distance is kept zero until going back to the original lane is safe. Three cases have been tested using the introduced solution to show its efficacy. All obstacles were moving at a speed of 20 km/h. The first two cases involved the ego vehicle performing a DLC to evade an upfront dynamic obstacle. Meanwhile, in the third case, the ego vehicle performed a DLC to evade three dynamic obstacles in a row ahead of it. Fig. 6.4 shows the three cases. Note that in the third case, the longitudinal distance needed to perform the DLC maneuver is relatively longer than the first two DLC maneuvers.

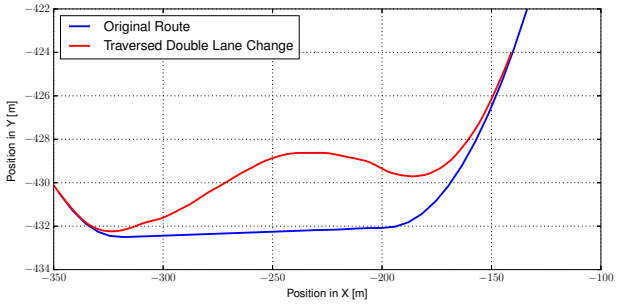
Comparing our method to the one proposed in [14], which uses hard constraints to omit obstacle region from the solution space, the results obtained are smoother and guaranteed to be feasible for the ego vehicle. The computational time needed to obtain the bell-shaped function is negligible compared to searching for the optimal path like in [15, 16]. In addition, none of the previous methods incorporated road shape in their solution.

6.2.3 Model Predictive Path-following Control Results

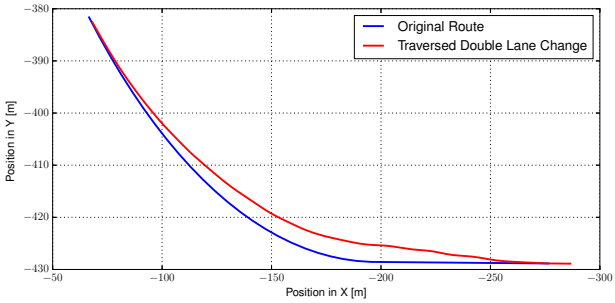
After receiving the parametric curve $\mathcal{P}_{clear}$, the MPFC module follows it with minimum possible error while tracking the reference speed with less priority as stated before. First, we present the set constraints on the optimization variables in the OCP (3.38) in table 6.4.



Case 2: DLC (1)



Case 2: DLC (2)



Case 3: DLC (3)

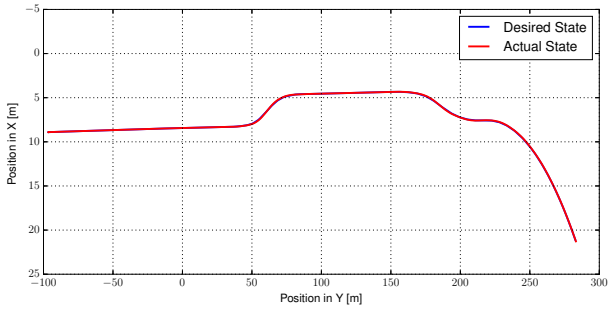
Figure 6.4: Original parametric curve $\mathcal{P}$ and the traversed parametric curve $\mathcal{P}_{clear}$.

Table 6.4: Constraints set on the optimization variables.

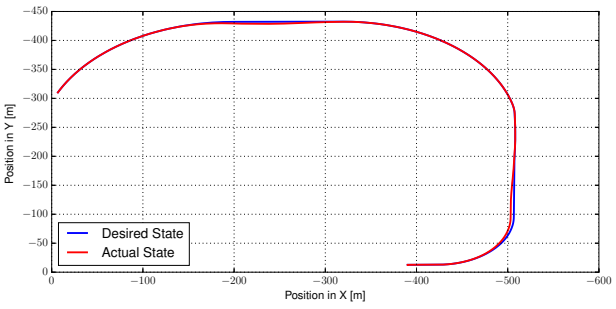
Parameter	Lower Bound	Upper Bound
$\dot{\mathbf{u}}(t)$	$\dot{\mathbf{u}}(t)_{min} = \begin{bmatrix} -3 \text{ m/s}^2 \\ -1 \text{ rad/s} \end{bmatrix}$	$\dot{\mathbf{u}}(t)_{max} = \begin{bmatrix} 3 \text{ m/s}^2 \\ 1 \text{ rad/s} \end{bmatrix}$
$\mathbf{u}(t)$	$\mathbf{u}(t)_{min} = \begin{bmatrix} -16 \text{ m/s} \\ -\frac{\pi}{4} \text{ rad} \end{bmatrix}$	$\mathbf{u}(t)_{max} = \begin{bmatrix} 16 \text{ m/s} \\ \frac{\pi}{4} \text{ rad} \end{bmatrix}$

Second, the reference path and the traversed path for every scenario are illustrated in 6.5. The two LC maneuvers can clearly be seen in case (1), while the three DLC maneuvers in case (2) and case (3) are shown as deviation between $\mathcal{R}_{global,ref}$ and the traversed route.

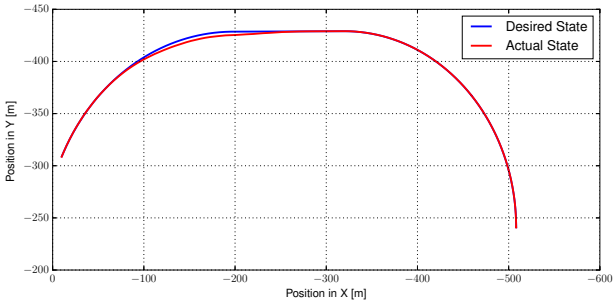
As it can be seen in Fig. 6.5, the ego vehicle was able to follow the path with minimum possible error. Path-following error is presented in Fig. 6.7 and summarized in table 6.5 where the maximum values are highlighted in yellow. The error for all cases is very small and is negligible relative to a standard lane width as it does not exceed 12.5 cm. Computation time is also very small especially that the OCP is nonlinear. Since tracking the reference speed is not a strict requirement in our problem, there is some deviation between the reference speed (blue) and the NMPC output speed command (green) and the actual vehicle speed (red) as shown in Fig. 6.7. The matching between the NMPC speed and the actual speed is more important as it reflects the ability of the NMPC to consider the dynamics of the vehicle while solving for the control actions. In addition, the NMPC output speed is robust regardless of the shape of the reference speed signal. Fig. 6.6 shows the relation between desired speed, MPC speed, and actual speed.



Case 1



Case 2



Case 3

Figure 6.5: Reference route $\mathcal{R}_{global,ref}$ and the traversed route for each case.

Table 6.5: NMPC computation time and path-following error.

Case	Avg. Following Error	Max. Following Error	Avg. Computation Time	Max. Computation Time
1	0.0670 [m]	0.1241 [m]	0.0248 [s]	0.1153 [s]
2	0.0587 [m]	0.1214 [m]	0.0370 [s]	0.1164 [s]
3	0.0539 [m]	0.1117 [m]	0.0383 [s]	0.0873 [s]
Mean	**0.0599 [m]**	**0.1190 [m]**	**0.0334 [s]**	**0.1063 [s]**

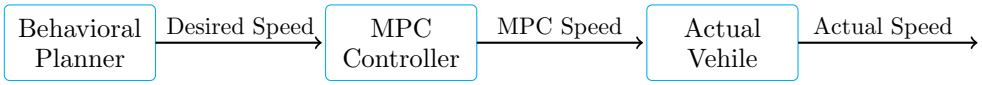


Figure 6.6: Relation between desired speed, MPC speed, and actual speed.

In comparison to the solution proposed in [15], our approach has smaller error and less computational time, while driving on almost the same speeds. Although the method proposed in [17] uses the same path-following scheme, it uses only a constant reference speed, and hence, the effect of different reference speeds is not studied. In addition, it does not handle long trips as the reference parametric curve is created beforehand. Finally, obstacle avoidance has not been integrated in any other scheme for autonomous vehicles using the path-following method introduced in [9].

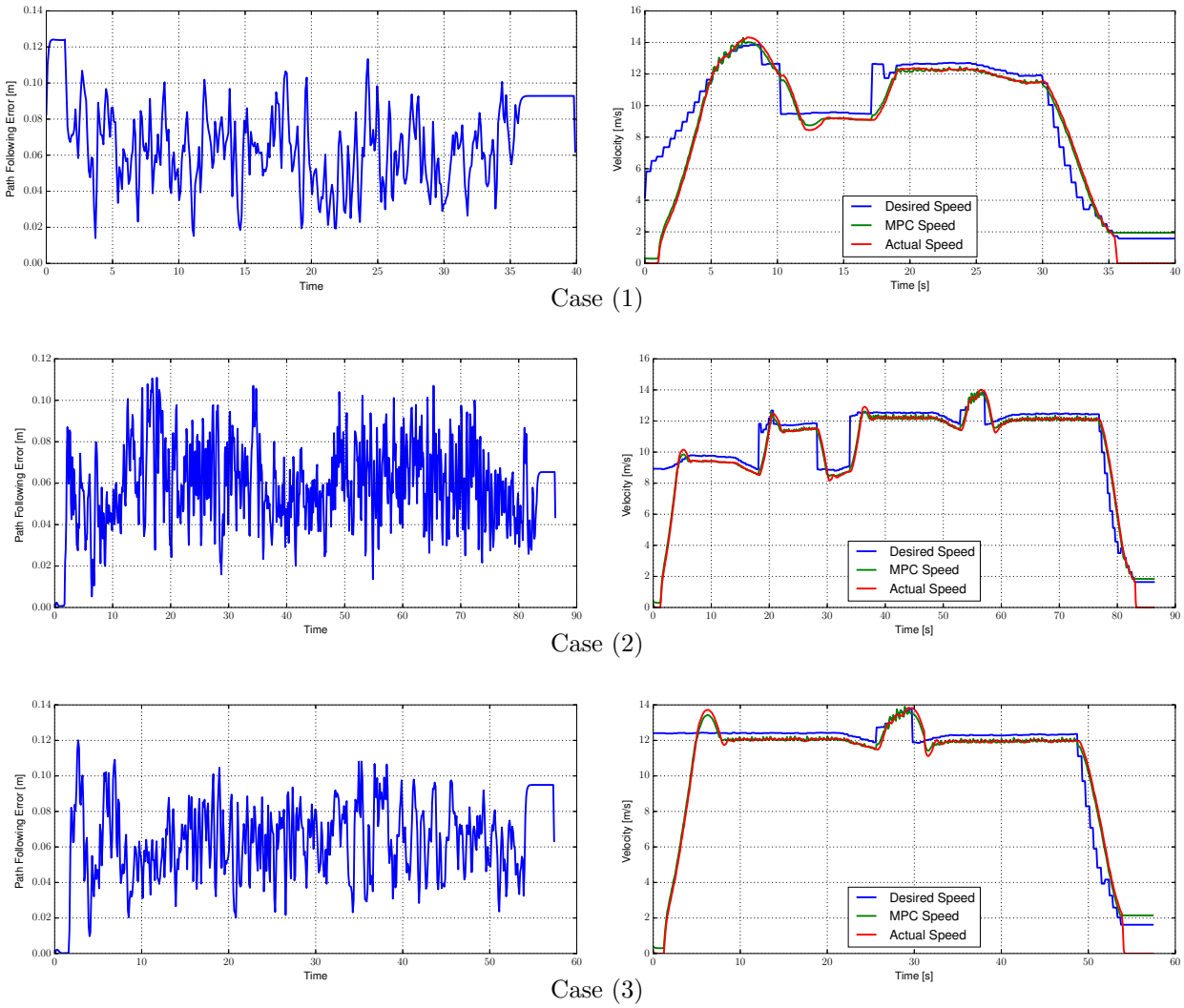


Figure 6.7: Left: Path-following error for each case. Right: velocity tracking output for each case.

6.3 Adjustable Driving Behavior Control Results

Here, we present our results obtained for the methodology proposed in Chapter 4.

6.3.1 Dynamic Simulation Scenarios

The proposed controller is tested on a car model developed in Gazebo [19] simulator. The parameters of the model are for a Chevrolet Bolt car, see Table 6.6. The NMPC controller is formulated using CasADi optimization library [68] and solved using the interior point method (IPOPT) [69]; the controller is implemented in Python programming language. The simulator and the controller are integrated under the Robot Operating System (ROS) [73]. The developed vehicle dynamic model satisfies Ackermann condition, i.e. when a vehicle is moving slowly on a curve, it turns slip-free [31]. Ackermann condition is given by:

$$\cot(\delta_o) - \cot(\delta_i) = \frac{w}{l}, \tag{6.2}$$

where δ_i and δ_o are the inner and the outer steering angles, respectively. The dimensions w and l are illustrated in Fig. 6.9.

Table 6.6: Simulated vehicle parameters

Parameter	Value	Parameter	Value
m_v	1615 [kg]	G	7
l	2.6 [m]	A	2.0 [m^2]
w	1.5 [m]	ρ_a	1.2 [kg/m^3]
r	0.2 [m]	C_d	0.45
C_r	0.015	η	70% - 95%

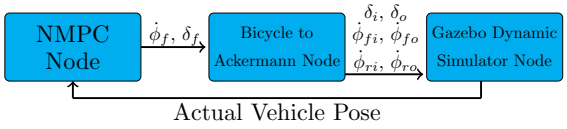


Figure 6.8: The developed ROS package block diagram.

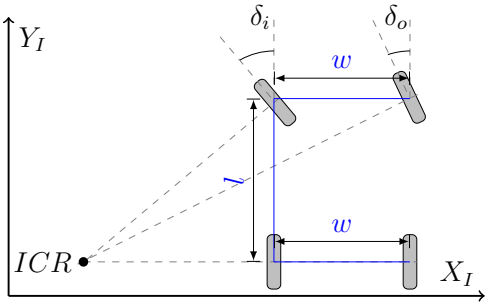


Figure 6.9: Ackermann vehicle model

Fig. 6.8 shows the developed ROS package. In this figure, $\dot{\phi}_{fi}$ and $\dot{\phi}_{fo}$ are the spinning speed of the front inner and outer wheels, respectively; and $\dot{\phi}_{ri}$ and $\dot{\phi}_{ro}$ are the spinning speed of the rear inner and outer wheels, respectively. It is worth mentioning that this implementation is suitable for electric vehicles in which each wheel is driven independently (using in-wheel motors) without a main motor and a drive-line transmission.

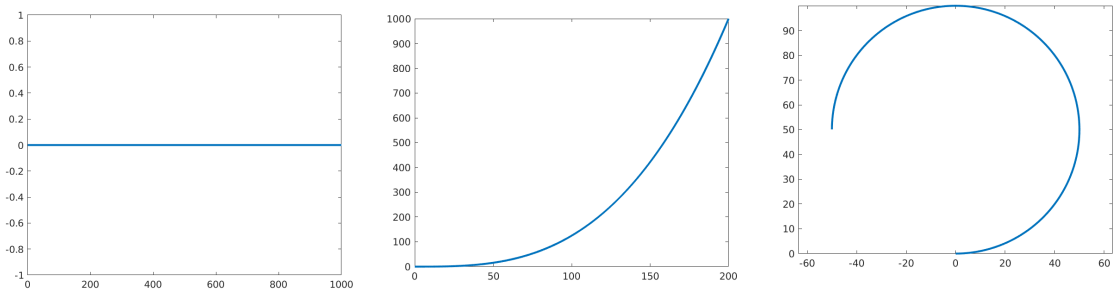
The reference paths used in our implementation are geometric splines simulating road and traffic lanes as suggested in [79]. We consider three main scenarios to test the proposed dual-objective NMPC in solving the path-following control problem given in Alg. 5. The reference paths in these scenarios are modeled using geometric equations as shown in Fig. 6.10.

6.3.2 Real-time Simulation Results and Discussion

In this section, we present our real-time simulation results. We consider mainly two driving modes: energy-efficient driving mode, and sport driving mode. The two driving modes are tested with each reference path shown in Fig. 6.10. Here, we first present the tuning parameters of the proposed NMPC controller. Then, we present the simulation results for the considered driving scenarios.

Tuning parameters for NMPC

Tuning the cost function weights of each term is important to ensure satisfactory NMPC performance. Table 6.7 shows the used parameters and weights, which led to the best



(a) Straight Path
$\varepsilon \in [0, 1000]$
$x = \varepsilon,\ y = 0$

(b) Third Order Polynomial Path
$\varepsilon \in [0, 200]$
$x = \varepsilon\ ,\ y = \frac{\varepsilon^3}{8000}$

(c) Circular Path
$\varepsilon \in [\frac{-\pi}{2}, \pi]$
$x = 50\cos(\varepsilon),\ y = 50\ (1 + \sin(\varepsilon))$

Figure 6.10: Different test cases used for path-following.

path-following performance, for each test scenario. All simulations were conducted using a prediction horizon of $N = 20$ and a sample time of $t_s = 0.2$ [sec], $\mathbf{U}_{min} = [-15, \frac{-\pi}{6}]^\top$, $\mathbf{U}_{max} = [100, \frac{\pi}{6}]^\top$, $\Delta\mathbf{U}_{min} = [-8, -0.1]^\top$, and $\Delta\mathbf{U}_{max} = [4, 0.1]^\top$. The path evolution step-size parameter κ_{step} depends on the road shape. This is because it represents increments in meters in the first two cases, while it represents increments in radians in the third case, see Fig. 6.10.

Table 6.7: Parameters and weights used for each test case

Driving Scenario	Params	Efficient Mode	Sport Mode
Straight Line	κ_{step}	30	30
	Q	$diag(4.75, 4.75, 100)$	$diag(4.75, 4.75, 100)$
	R	$diag(3, 20)$	$diag(1, 1)$
	E	2.5×10^{-10}	0
Spline Path	κ_{step}	20	20
	Q	$diag(10, 10, 2)$	$diag(10, 10, 2)$
	R	$diag(1, 1)$	$diag(1, 1)$
	E	2.5×10^{-10}	0
Circular Path	κ_{step}	0.157	0.157
	Q	$diag(100, 100, 0)$	$diag(100, 100, 0)$
	R	$diag(1, 1)$	$diag(1, 1)$
	E	2.5×10^{-10}	0

Straight Line Path

Most of city roads can be modeled as straight segments. In this test scenario, we consider the reference path in Fig. 6.10a; the path length is 1000 meter. Results obtained are shown in Fig. 6.11, 6.12 and 6.13 and summarized in table 6.8, where RMSE is Root Mean Squared Error (RMSE). As can be seen in Fig. 6.12, the velocity profile shows slower acceleration and deceleration for the energy-efficient mode. Meanwhile, sport mode showed

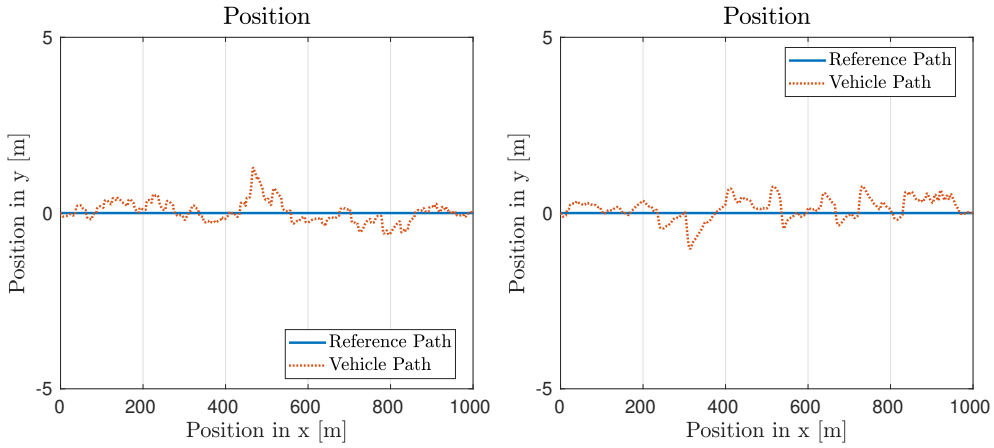


Figure 6.11: Path-following simulation results for the straight line reference path. Left: Energy-efficient mode. Right: Sport mode.

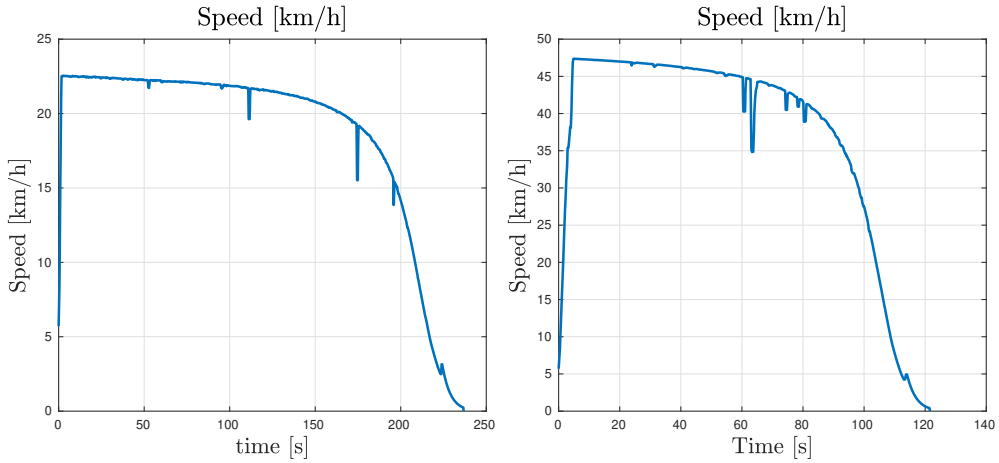


Figure 6.12: The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode.

a higher speed velocity profile. Fig. 6.13 shows the efficacy of the proposed controller in saving energy, while following the desired path, in case of the energy-efficient mode when compared to the sport mode. The summary of the results in Table 6.8 shows that the energy-efficient mode was able to reduce the total consumed energy by 48% compared to the sport mode. Also, sport mode was able to arrive faster than the energy-efficient mode by 51.4%. The RMSE for both scenarios is very small. Small RMSE shows the effectiveness of the proposed controller in following the reference path.

Third Order Spline Path

Curved roads are very common cases especially in highways, and they can be modeled using polynomial splines. Now, we consider the reference path in Fig. 6.10b with a total length of 175.5 meter. Results obtained are shown in Fig. 6.14 and 6.15, and summarized in table 6.9. The results show a similar behavior compared to the straight line path. However, as this path is shorter, total consumed energy for both scenarios is less than that for the straight line path.

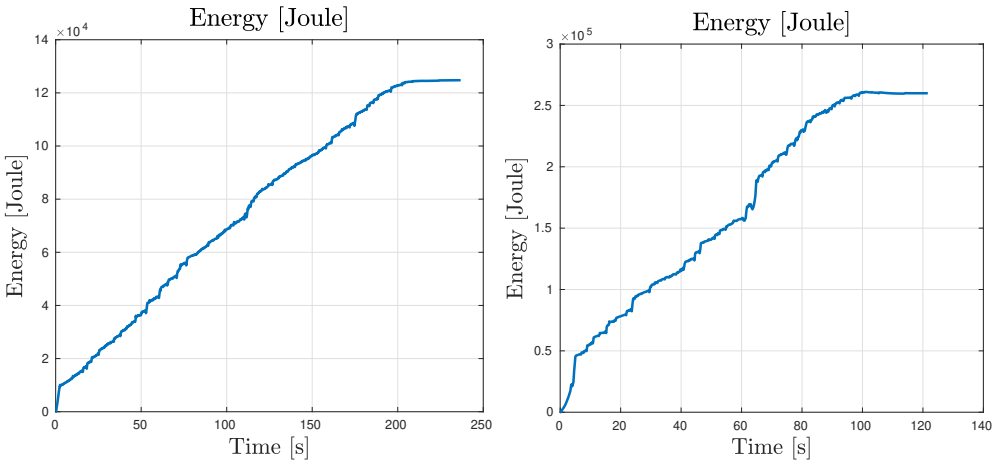


Figure 6.13: Consumed energy (Joules) by the vehicle simulation results. Left: Energy-efficient mode. Right: Sport mode.

Table 6.8: Comparison between driving modes for straight line path

Parameter	Efficient Mode	Sport Mode
Total Energy Consumption	124.764 [kJ]	259.93 [kJ]
Total Travel Time	236.79 [sec]	121.62 [sec]
Average Speed	15.20 [km/h]	29.60 [km/h]
RMSE	0.2871 [m]	0.3161 [m]
End Point Error	0.0168 [m]	0.0017 [m]

Circular Path

Finally, we use a circular reference path, Fig. 6.10c, to demonstrate the ability of the proposed controller in maintaining a fixed turning radius while driving. The total path length is 235.62 meter. Results obtained are shown in Fig. 6.16 and 6.17, and summarized in table 6.10.

Table 6.9: Comparison between driving modes for the spline path

Parameter	Efficient Mode	Sport Mode
Total Energy Consumption	26.86 [kJ]	44.2 [kJ]
Total Travel Time	45.67 [sec]	34.89 [sec]
Average Speed	13.83 [km/h]	18.11 [km/h]
RMSE	1.1335 [m]	0.9628 [m]
End Point Error	0.0512 [m]	0.0824 [m]

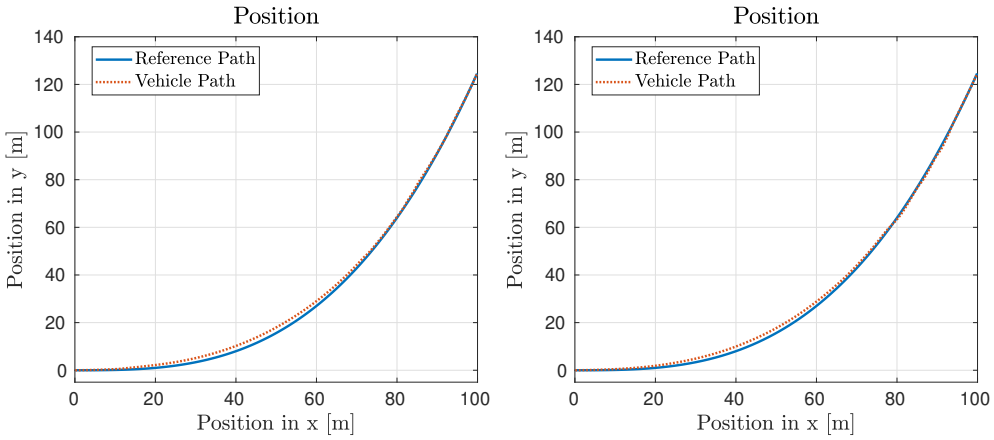


Figure 6.14: Path-following simulation results for the spline reference path. Left: Energy-efficient mode. Right: Sport mode.

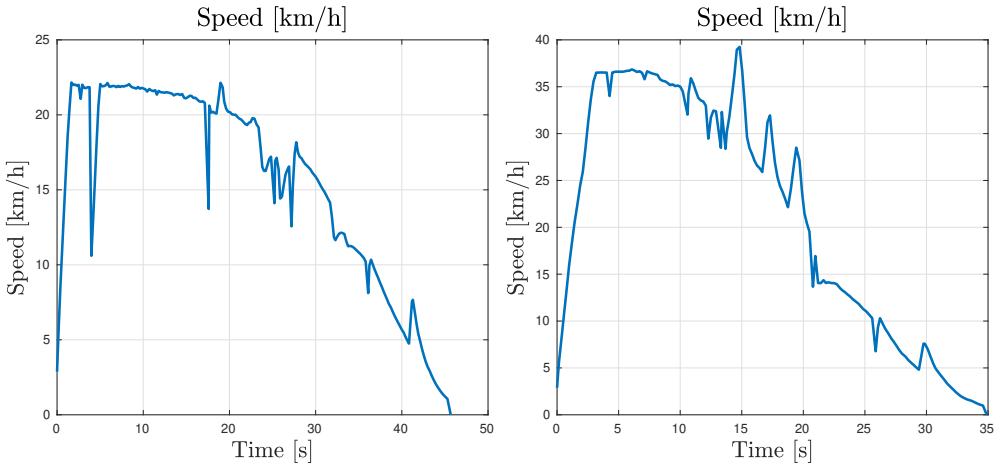


Figure 6.15: The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode.

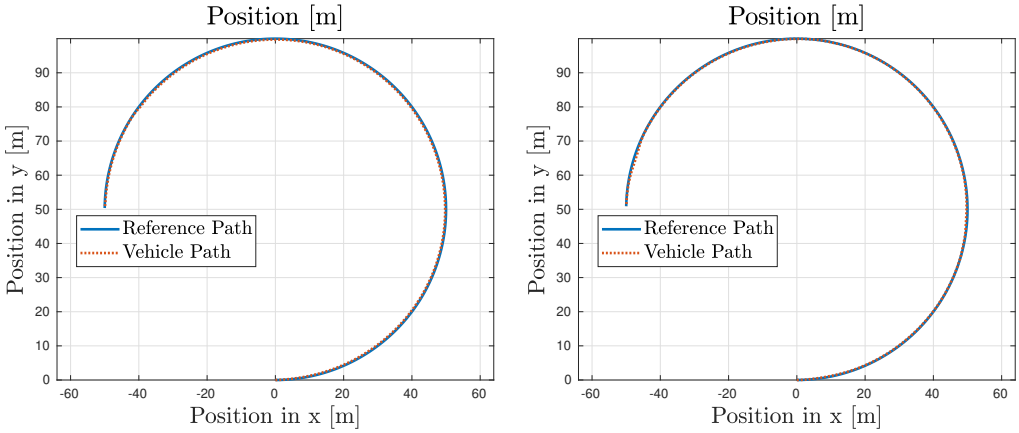


Figure 6.16: Path-following simulation results for the circular reference path. Left: Energy-efficient mode. Right: Sport mode.

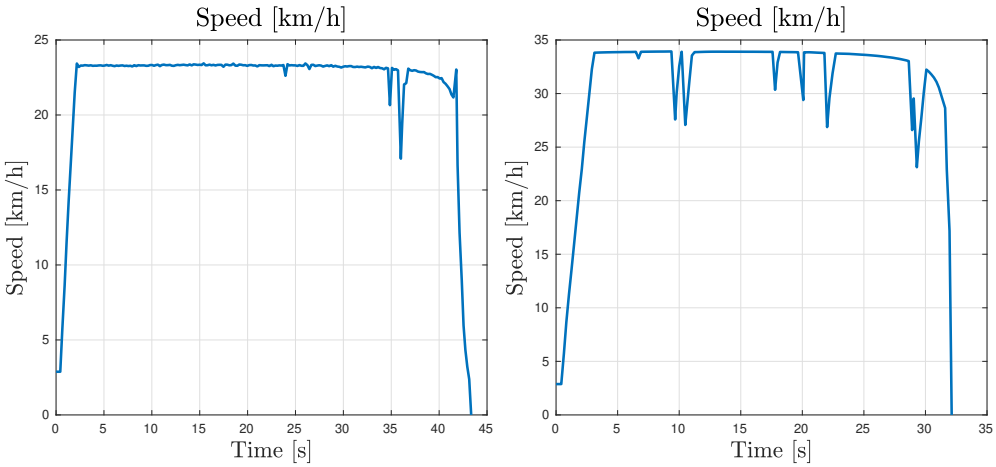


Figure 6.17: The overall vehicle speed in km/h for both driving modes. Left: Energy-efficient mode. Right: Sport mode.

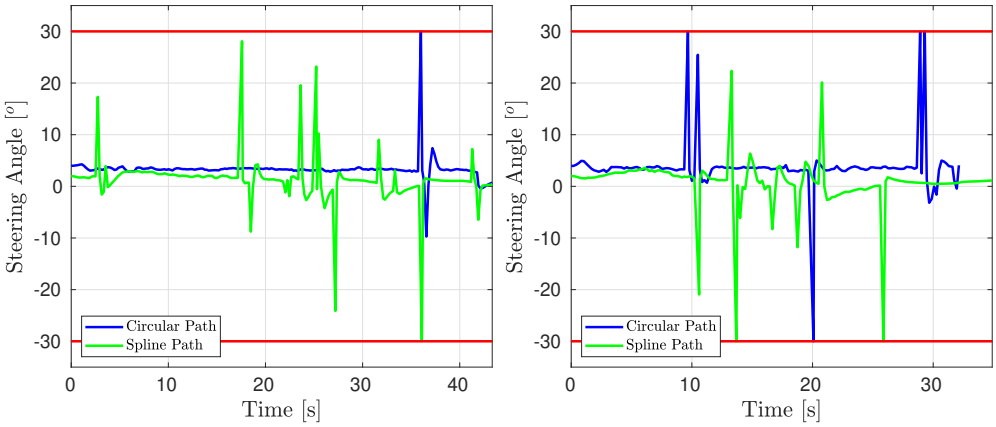


Figure 6.18: Steering angle control action. Red lines represent the upper and lower limits. Left: Energy-efficient mode. Right: Sport mode.

Similar to the previous results, energy-efficient mode was able to reduce energy consumption, while sport mode was able to minimize total travel time.

The proposed controller maintained the input and state constraints throughout all simulations. Fig. 6.18 shows the computed steering angles for the spline path and the circular path for both driving modes. It can be noted from the figure that the control actions did not go beyond their limits. Path-following results for the three test cases are available at the following link: https://youtu.be/2dSwTR1gk20.

Table 6.10: Comparison between driving modes for circular path

Parameter	Efficient Mode	Sport Mode
Total Energy Consumption	31.06 [kJ]	55.48 [kJ]
Total Travel Time	43.34 [sec]	32.13 [sec]
Average Speed	19.57 [km/h]	26.40 [km/h]
RMSE	0.4832 [m]	0.3395 [m]
End Point Error	0.0284 [m]	0.0015 [m]

6.4 Obstacle Representation using FS Results

6.4.1 Obstacle Representation

The most common obstacle an ego vehicle may encounter while driving are other vehicles on the road. The methodology proposed in Chapter. 5 is used to represent different vehicles with different shapes and sizes. The accuracy of obtained representation depends heavily on two main factors; number of samples of the outer boundary, and most importantly, number of frequencies f_n used in modeling. Fig. 6.19 shows the results for five different f_n for the same vehicles. Notice the effect of the used f_n on the accuracy of the representation.

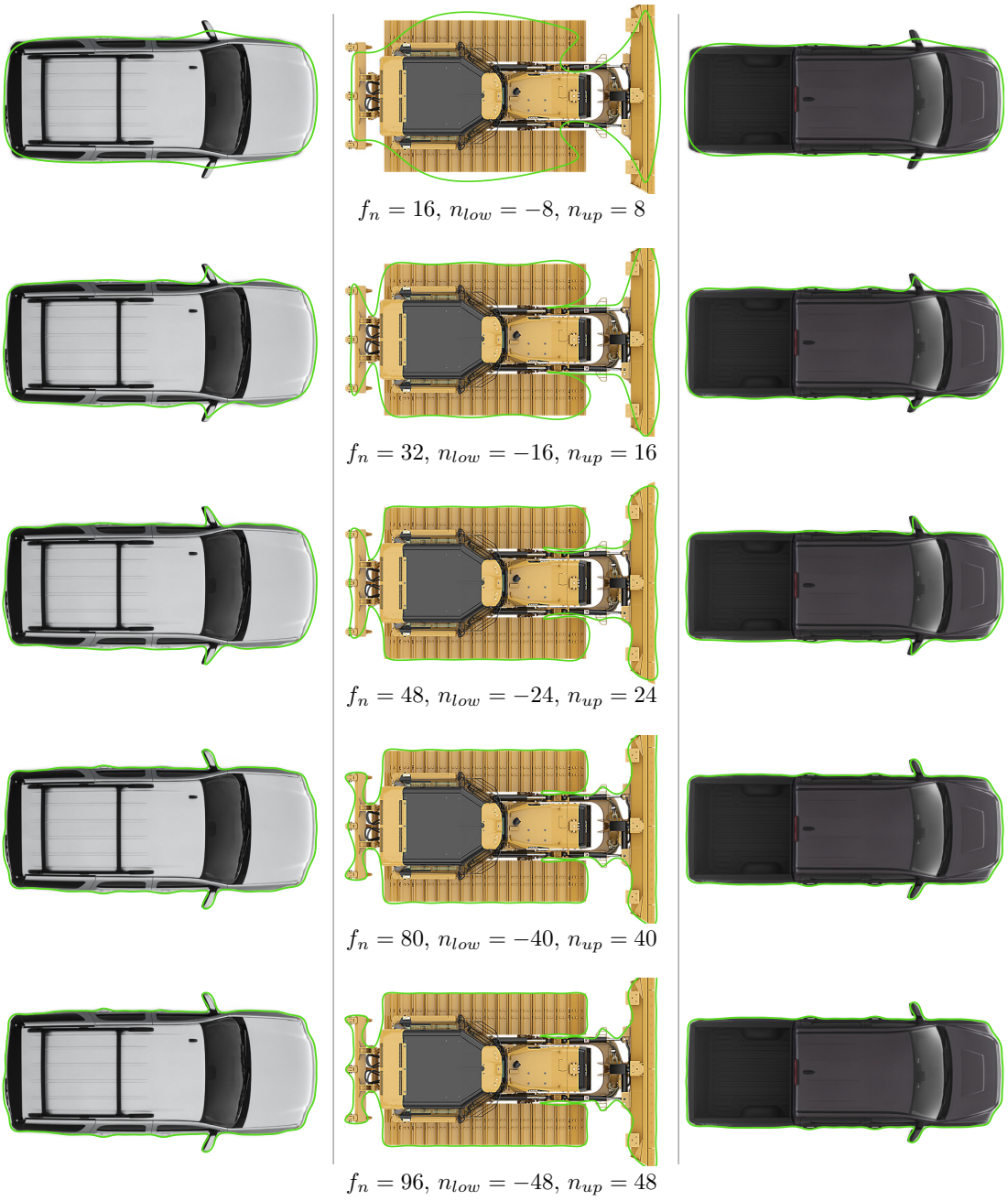


Figure 6.19: Effect of the number of frequencies on the accuracy of the obtained representation.

The number of frequencies f_n is set depending on the nature of the application and the complexity of the shape. For fast collision detection, a lower f_n would be more useful. Note that we rely on edge detection of the outer boundary of the object to create its representation. To simplify the complexity of an object shape, the outer boundary can be simplified first.

The effect of the frequencies is further studied for different shapes. Higher frequencies usually show smaller magnitudes. For a number of frequencies $f_n = 80$, the magnitude of every frequency is plotted in Fig. 6.20. Note that small frequencies are plotted using a smaller scale. The zero frequency is responsible for the C.G location of the object, and hence, its magnitude does not affect the representation but only translates it. To reduce

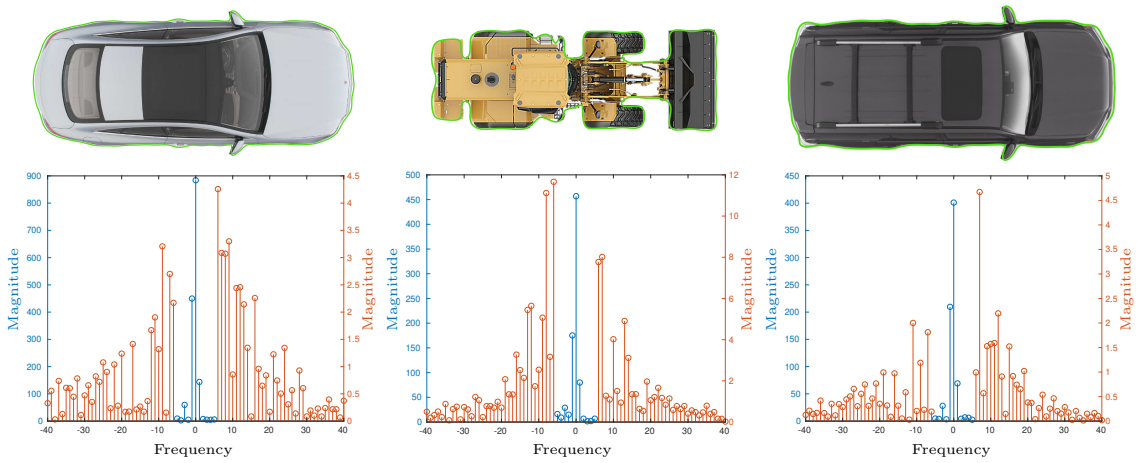
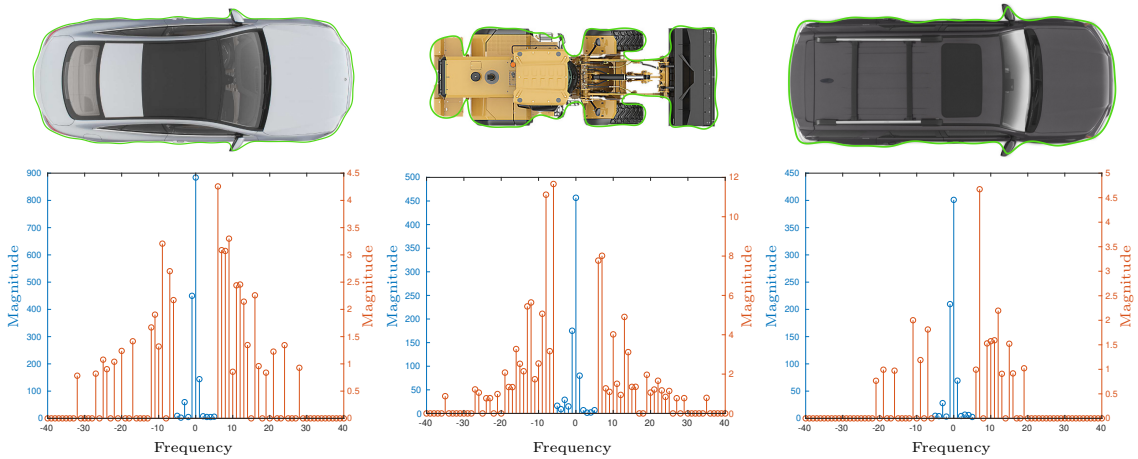


Figure 6.20: Magnitude of every frequency for three different objects.

the mathematical complexity, frequencies with small magnitudes can be omitted. Keeping the high magnitude components only should be able to model the object with relatively high accuracy. Fig. 6.21 shows the same objects in Fig. 6.20 after removing components with magnitudes smaller than 0.75.



Frequencies removed: 42
Remaining frequencies: 38

Frequencies removed: 29
Remaining frequencies: 51

Frequencies removed: 54
Remaining frequencies: 26

Figure 6.21: Obtained representation after filtering out small magnitude frequencies.

6.4.2 Static Map Representation

Similarly, the road network map can be represented using the same approach. Such representation may aid in identifying the drivable regions on the road in adverse weather conditions as well as help in vehicle localization. Similar to modeling the outer boundary of obstacles, creating map representation can be done using the road boundaries, which can be obtained from the HD maps. Note that HD maps are represented using different standards such as OpenStreetMap [80].

The proposed methodology has been tested on CARLA Town01 as shown in Fig. 6.22, where the green lines are the obtained representations. Each closed loop is modeled using $f_n = 80$ as a mathematical equation, and hence, this map is represented by eight mathematical equations.

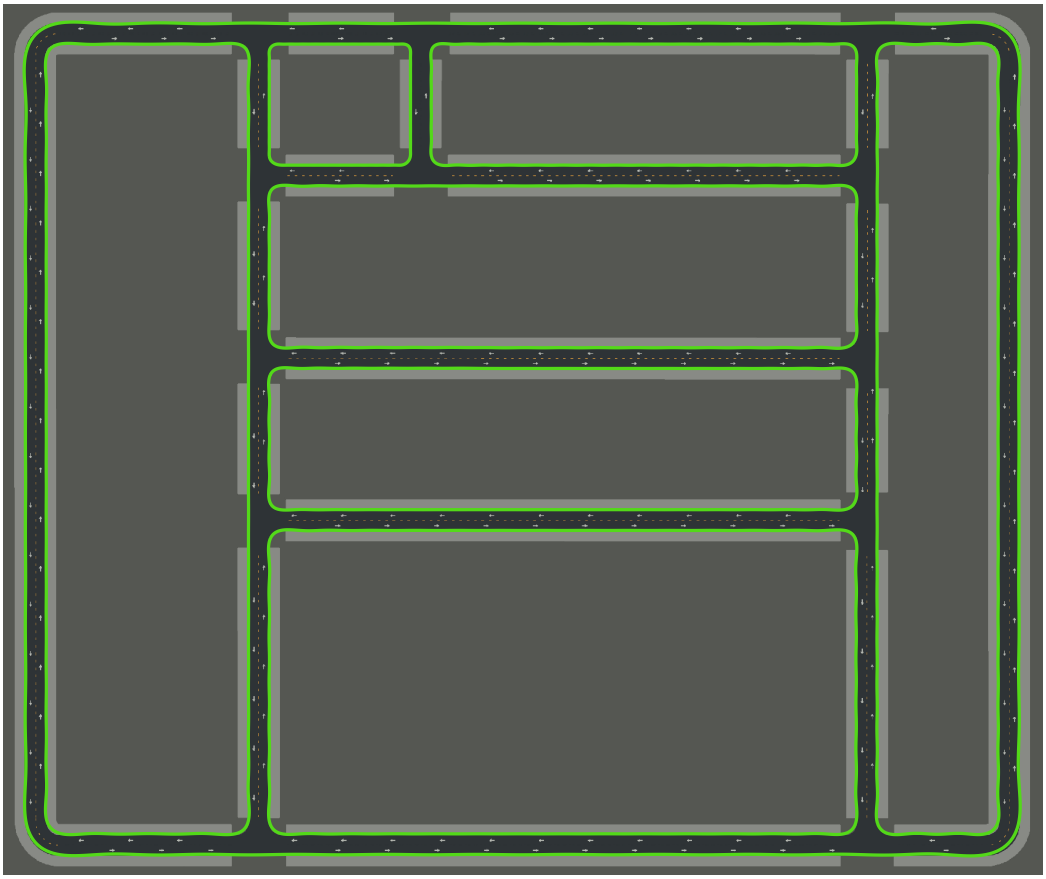


Figure 6.22: Obtained representation for CARLA Town01.

When a portion of the map is required to be modeled and not the entire map, this portion can be modeled by the same way. However, it will introduce some open loops. This is actually normal since any map is connected with another bigger road network, and hence, open loops must exist. To tackle the existence of open loops, we use the reverse sampling method proposed in Chapter 5.

To show its validity, we apply it on five-way intersection taken from CARLA Town03. Fig. 6.23 shows the results for road boundary modeling with and without reverse sampling. Note that in both cases, every green open loop is represented using one equation. We use

$f_n = 80$ for the representation on the left and $f_n = 50$ for the representation on the right. As it can be seen, modeling with reverse sampling gives a much better result, even in case of smaller number of frequencies.

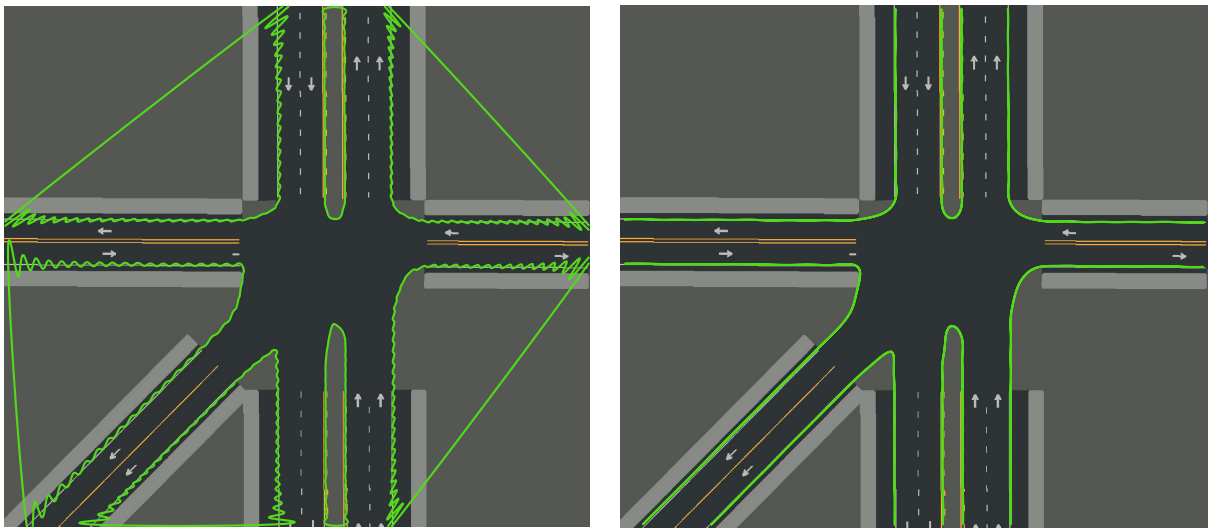


Figure 6.23: Obtained representation for five-way intersection from CARLA Town03. Left: representation without reverse sampling. Right representation with reverse sampling

The proposed methodology can also be used in autonomous mobile robots and can be further extended to model objects in 3D. Such a technique has the potential to improve mapping and 3D construction techniques. In case of modeling the surrounding environment, the magnitude and phase of each frequency would represent the features of the environment.

Chapter 7

Conclusion and Future Work

In this chapter, we summarize the major contributions proposed in the thesis and describe some of the possible future work. The main thesis focus was to introduce faster decision-making schemes. The primary problems considered and discussed in the thesis were local motion planning and control, driving behavior change, and obstacle representation. The main contributions proposed in the thesis are:

1. A simultaneous local motion planning and path-following control scheme.
2. An adjustable driving behavior using dual-objective cost function methodology.
3. An obstacle representation using Fourier series framework.

7.1 Simultaneous Local Motion Planning and Control

In chapter 3, we proposed a simultaneous local motion planning and path-following control scheme. The proposed approach enables autonomous driving for long trips while reducing the burden of computational complexity and ensuring obstacle avoidance functionality. The proposed scheme works simultaneously through utilizing a parallel architecture for local motion planning and control layers. In addition, novel feasibility-guaranteed lane change and double lane change planners were introduced for path planning and obstacle avoidance. Moreover, a dynamic lookahead distance based on road curvature information was proposed. Finally, an online parameterized curve generator was proposed and integrated with Model Predictive Path-Following Control [9]. To show the validity of the proposed methodology, different real-time simulation scenarios were conducted using CARLA open source simulator. The results, demonstrated in 6.2, showed the efficacy of the proposed decision-making architecture.

7.2 Adjustable Driving Behavior

In chapter 4, we introduced a new path-following controller based on a dual-objective NMPC. The proposed controller adjusts the driving behavior of the autonomous vehicle

based on the passenger's preference. Two different controller behaviors were designed, namely, energy-efficient mode and sport mode. Moreover, a dual-objective cost function is used to shift between these two modes. In order to validate the proposed controller, different simulation scenarios were designed and implemented using Gazebo dynamic simulator. The results, presented in 6.3, showed the efficacy of the proposed controller.

7.3 Obstacle Representation using Fourier Series

In chapter 5, a novel static map and obstacle representation using Fourier Series framework was proposed. The framework relies on Complex Fourier Series analysis to reduce the computation time. The output of the framework is mathematical equations that describe the shape of either the obstacle or the map. Furthermore, two methods were proposed, namely; offline method and online method. The offline method was used to create accurate representations for static maps and most common obstacles an ego vehicle may encounter. The online method was used to model the free space around the vehicle when dealing with uncertain environments. To test the proposed framework, different obstacles with different shapes and sizes were considered as well as two static maps. The results, illustrated in 6.4, showed the efficacy of the proposed framework.

7.4 Possible Future Work

While simulation results presented in thesis validate all the proposed contributions, experimental work will show the efficacy of the proposed contributions in real life scenarios. Although all of the experimental facilities are available on campus, the experimental validation had to be suspended in order to abide to the emergency orders by Ontario public health because of Covid-19 pandemic which required campus closure.

The decision-making presented in chapter 3 uses FSM for behavioral planning. However, RL has the potential to deal with more unexpected scenarios. Therefore, an RL behavioral planner may make the proposed scheme more generic. The parameterized curve generator can also be optimized to obtain new splines with shorter time. Finally, prediction models for the dynamic obstacles may improve the proposed collision avoidance techniques.

In chapter 4, we presented two different driving modes. However, different driving behaviors beside the sport and energy-efficient modes can also be designed. The stability of the proposed controller is to be rigorously proven with and without stabilizing constraints [25]. Finally, the effect of switching between different modes throughout vehicle operation will be studied in terms of the controller stability.

The approach proposed in chapter 5 can also be further extended to model objects in 3D. Such a technique has the potential to improve mapping and 3D reconstruction techniques. In addition, this approach may help in solving loop closure problem in Simultaneous Localization and Mapping (SLAM) since it represents the features of each coordinate of the map separately. Finally, such a technique can replace spline generations used in lane markings detection and path planning. Lastly, all the aforementioned techniques can be integrated together to obtain a decision-making system that incorporates all of the proposed features.

References

[1] Early Estimate of Motor Vehicle Traffic Fatalities in 2018. National Highway Traffic Safety Administration, Washington, D.C., USA, 2019.

[2] Critical reasons for crashes investigated in the national motor vehicle crash causation survey. National Highway Traffic Safety Administration, Washington, D.C., USA, 2018.

[3] Study: Long commutes to work by car. Statistics Canada, 2019.

[4] No Hands Across America Journal. https://www.cs.cmu.edu/afs/cs/usr/tjochem/www/nhaa/Journal.html, 1995. [Online]; accessed 12-November-2019.

[5] The DARPA Grand Challenge: Ten Years Later. https://www.darpa.mil/news-events/2014-03-13, 2014. [Online]; accessed 12-November-2019.

[6] Hao Zhu, Ka-Veng Yuen, Lyudmila Mihaylova, and Henry Leung. Overview of environment perception for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(10):2584–2601, 2017.

[7] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, 1(1):33–55, 2016.

[8] Yadollah Rasekhipour, Amir Khajepour, Shih-Ken Chen, and Bakhtiar Litkouhi. A potential field-based model predictive path-planning controller for autonomous road vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1255–1267, 2016.

[9] Timm Faulwasser, Benjamin Kern, and Rolf Findeisen. Model predictive path-following for constrained nonlinear systems. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 8642–8647. IEEE, 2009.

[10] Jack N Barkenbus. Eco-driving: An overlooked climate change initiative. *Energy Policy*, 38(2):762–769, 2010.

[11] Tomer Toledo and Tsippy Lotan. In-vehicle data recorder for evaluation of driving behavior and safety. *Transportation Research Record*, 1953(1):112–119, 2006.

[12] Carlo Ackermann, Jakob Bechtloff, and R Isermann. Collision avoidance with combined braking and steering. In *6th International Munich Chassis Symposium 2015*, pages 199–213. Springer, 2015.

[13] Path tracking for autonomous vehicles based on nonlinear model: Predictive control method, author=Li, Shaosong and Li, Zheng and Zhang, Bangcheng and Zheng, Shunhang and Lu, Xiaohui and Yu, Zhixin, year=2019, institution=SAE Technical Paper. Technical report.

[14] Alberto Franco and Vitor Santos. Short-term path planning with multiple moving obstacle avoidance based on adaptive MPC. In *2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 1–7. IEEE, 2019.

[15] Chaoyong Zhang, Duanfeng Chu, Shidong Liu, Zejian Deng, Chaozhong Wu, and Xiaocong Su. Trajectory planning and tracking for autonomous vehicle based on state lattice and model predictive control. *IEEE Intelligent Transportation systems magazine*, 11(2):29–40, 2019.

[16] Jie Ji, Amir Khajepour, Wael William Melek, and Yanjun Huang. Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints. *IEEE Transactions on Vehicular Technology*, 66(2):952–964, 2016.

[17] Robert Ritschel, Frank Schrödel, Juliane Hädrich, and Jens Jäkel. Nonlinear Model Predictive Path-Following Control for Highly Automated Driving. *IFAC-PapersOnLine*, 52(8):350–355, 2019.

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[19] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[20] Tom M. Gasser *et al.* Legal consequences of an increase in vehicle automation. BASt (German Federal Highway Institute), 2013. BASt-Report F83.

[21] Preliminary statement of policy concerning automated vehicles. US National Highway Trafc Safety Administration. National Highway Traffic Safety Administration, Washington, D.C., USA, 2013.

[22] Taxonomy and defnitions for terms related to driving automation systems for on-road motor vehicles. Society for Automotive Engineers, 2018.

[23] D Subbaram Naidu. *Optimal control systems*. CRC press, 2002.

[24] Donald E Kirk. *Optimal control theory: an introduction*. Courier Corporation, 2004.

[25] Lars Grüne and Jürgen Pannek. Nonlinear model predictive control. In *Nonlinear Model Predictive Control*, pages 45–69. Springer, 2017.

[26] James B Rawlings. Tutorial overview of model predictive control. *IEEE control systems magazine*, 20(3):38–52, 2000.

[27] Shuo Cheng, Liang Li, Hong-Qiang Guo, Zhen-Guo Chen, and Peng Song. Longitudinal collision avoidance and lateral stability adaptive control system based on MPC of autonomous vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[28] Mohamed W Mehrez, Karl Worthmann, Joseph PV Cenerini, Mostafa Osman, William W Melek, and Soo Jeon. Model Predictive Control without terminal constraints or costs for holonomic mobile robots. *Robotics and Autonomous Systems*, 127:103468, 2020.

[29] Karl Worthmann, Mohamed W Mehrez, Mario Zanon, George KI Mann, Raymond G Gosine, and Moritz Diehl. Model predictive control of nonholonomic mobile robots without stabilizing constraints and costs. *IEEE Transactions on Control Systems Technology*, 24(4):1394–1406, 2015.

[30] Roland Siegwart, Illah Reza Nourbakhsh, Davide Scaramuzza, and Ronald C Arkin. *Introduction to autonomous mobile robots*. MIT press, 2011.

[31] Reza N Jazar. *Vehicle dynamics: theory and application*. Springer, 2017.

[32] Jason Kong, Mark Pfeiffer, Georg Schildbach, and Francesco Borrelli. Kinematic and dynamic vehicle models for autonomous driving control design. In *2015 IEEE Intelligent Vehicles Symposium (IV)*, pages 1094–1099. IEEE, 2015.

[33] Edward M Kasprzak, Kemper E Lewis, and Douglas L Milliken. Inflation pressure effects in the nondimensional tire model. *SAE Transactions*, pages 1781–1792, 2006.

[34] Hannah Bast, Daniel Delling, Andrew Goldberg, Matthias Müller-Hannemann, Thomas Pajor, Peter Sanders, Dorothea Wagner, and Renato F Werneck. Route planning in transportation networks. In *Algorithm engineering*, pages 19–80. Springer, 2016.

[35] Rahul Kala. *On-road intelligent vehicles: Motion planning for intelligent transportation systems*. Butterworth-Heinemann, 2016.

[36] Edsger W Dijkstra et al. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[37] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[38] Andrew V Goldberg and Chris Harrelson. Computing the shortest path: A search meets graph theory. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 156–165. Society for Industrial and Applied Mathematics, 2005.

[39] Robert Geisberger, Peter Sanders, Dominik Schultes, and Christian Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[40] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius Brito Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago Meireles Paixão, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, page 113816, 2020.

[41] Martin Buehler, Karl Iagnemma, and Sanjiv Singh. *The DARPA urban challenge: autonomous vehicles in city traffic*, volume 56. springer, 2009.

[42] Magnus Olsson. Behavior trees for decision-making in autonomous driving, 2016.

[43] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic MDP-behavior planning for cars. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1537–1542. IEEE, 2011.

[44] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.

[45] John T Betts. Survey of numerical methods for trajectory optimization. *Journal of guidance, control, and dynamics*, 21(2):193–207, 1998.

[46] Lydia E Kavraki, Mihail N Kolountzakis, and J-C Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[47] Mihail Pivtoraiko, Ross A Knepper, and Alonzo Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

[48] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[49] Yanbo Li, Zakary Littlefield, and Kostas E Bekris. Sparse methods for efficient asymptotically optimal kinodynamic planning. In *Algorithmic foundations of robotics XI*, pages 263–282. Springer, 2015.

[50] Aurelio Piazzi and C Guarino Lo Bianco. Quintic g/sup 2/-splines for trajectory planning of autonomous vehicles. In *Proceedings of the IEEE Intelligent Vehicles Symposium 2000 (Cat. No. 00TH8511)*, pages 198–203. IEEE, 2000.

[51] Alonzo Kelly and Bryan Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[52] Yu Zhang, Huiyan Chen, Steven L Waslander, Tian Yang, Sheng Zhang, Guangming Xiong, and Kai Liu. Toward a more complete, flexible, and safer speed planning for autonomous driving via convex optimization. *Sensors*, 18(7):2185, 2018.

[53] Jingjing Jiang and Alessandro Astolfi. Lateral control of an autonomous vehicle. *IEEE Transactions on Intelligent Vehicles*, 3(2):228–237, 2018.

[54] Khaled Sailan and Klaus Dieter Kuhnert. Modeling and design of cruise control system with feedforward for all terrian vehicles. *Computer Science & Information Technology (CS & IT)*, 2013.

[55] Gabriel M Hoffmann, Claire J Tomlin, Michael Montemerlo, and Sebastian Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *2007 American Control Conference*, pages 2296–2301. IEEE, 2007.

[56] Jarrod M Snider et al. Automatic steering methods for autonomous automobile path tracking. *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RITR-09-08*, 2009.

[57] Gilles Tagne, Reine Talj, and Ali Charara. Higher-order sliding mode control for lateral dynamics of autonomous vehicles, with experimental validation. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 678–683. IEEE, 2013.

[58] Yinan Wang, Ting Qu, Jianxin Chu, Shuyou Yu, and Hongyan Guo. Trajectory Planning and Tracking Control of Vehicle Obstacle Avoidance based on Optimization Control. In *2019 Chinese Control Conference (CCC)*, pages 3157–3162. IEEE, 2019.

[59] Hans Pacejka. *Tire and vehicle dynamics*. Elsevier, 2005.

[60] Kevin M Lynch and Frank C Park. *Modern Robotics*. Cambridge University Press, 2017.

[61] Xiaohan Chen, Yingmin Jia, Junping Du, and Fashan Yu. Formation control of mobile robots with input constraints: an elliptic approximation approach. In *2011 11th International Conference on Control, Automation and Systems*, pages 186–191. IEEE, 2011.

[62] Mohamed W Mehrez, George KI Mann, and Raymond G Gosine. Formation stabilization of nonholonomic robots using nonlinear model predictive control. In *2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–6. IEEE, 2014.

[63] Exploding The Map. a16z.com/2017/09/16/exploding-map-evolution-cartography-deep-map, 2014. [Online]; accessed 12-November-2019.

[64] Niels Joubert, Tyler GR Reid, and Fergus Noble. Developments in modern gnss and its impact on autonomous vehicle architectures. *arXiv preprint arXiv:2002.00339*, 2020.

[65] Curvature and Radius of Curvature. https://www.math24.net/curvature-radius/, 2014. [Online]; accessed 6-June-2020.

[66] Vehicle turning paths. https://www.dimensions.com/collection/vehicle-turning-paths-radius. [Online]; accessed 12-June-2020.

[67] Jean-Christophe Léger. Menger curvature and rectifiability. *Annals of mathematics*, 149:831–869, 1999.

[68] Joel A E Andersson, Joris Gillis, Greg Horn, James B Rawlings, and Moritz Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, In Press, 2018.

[69] 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, Mar 2006.

[70] Timm Faulwasser and Rolf Findeisen. Nonlinear model predictive control for constrained output path following. *IEEE Transactions on Automatic Control*, 61(4):1026–1039, 2015.

[71] Timm Faulwasser, Janine Matschek, Pablo Zometa, and Rolf Findeisen. Predictive path-following control: Concept and implementation for an industrial robot. In *2013 IEEE International Conference on Control Applications (CCA)*, pages 128–133. IEEE, 2013.

[72] Mohamed A Daoud, Mostafa Osman, Mohamed W Mehrez, and William W Melek. Path-following and Adjustable Driving Behavior of Autonomous Vehicles using Dual-Objective Nonlinear MPC. In *2019 IEEE International Conference of Vehicular Electronics and Safety (ICVES)*, pages 1–6. IEEE, 2019.

[73] Morgan Quigley, Brian Gerkey, Ken Conley, Josh Faust, Tully Foote, Jeremy Leibs, Eric Berger, Rob Wheeler, and Andrew Ng. Ros: an open-source robot operating system. In *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA) Workshop on Open Source Robotics*, Kobe, Japan, May 2009.

[74] Mohamed W Mehrez, Karl Worthmann, George KI Mann, Raymond G Gosine, and Timm Faulwasser. Predictive path following of mobile robots without terminal stabilizing constraints. *IFAC-PapersOnLine*, 50(1):9852–9857, 2017.

[75] Grant Sanderson. But what is a Fourier series? from heat flow to circle drawings — de4. https://www.youtube.com/watch?v=r6sGWTCMz2k, June 2019.

[76] RUSSELL L Herman. An Introduction to Fourier and complex analysis with applications to the spectral analysis of signals. *University of North Carolina Wilmington, Wilmington, NC, online publication, http://people. uncw. edu/hermanr/-mat367/FCABook/Book2013/FunctionSpaces. pdf*, 2013.

[77] Elias M Stein and Rami Shakarchi. *Fourier analysis: an introduction*, volume 1. Princeton University Press, 2011.

[78] Jaycil Z Varghese, Randy G Boone, et al. Overview of autonomous vehicle sensors and systems. In *International Conference on Operations Excellence and Service Engineering*, pages 178–191, 2015.

[79] Alexey Abramov, Christopher Bayer, Claudio Heller, and Claudia Loy. A flexible modeling approach for robust multi-lane road estimation. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1386–1392. IEEE, 2017.

[80] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017.