Department of Precision and Microsystems Engineering

A Tentacle-Based Motion Planning Algorithm for Automated Vehicles

Zisen Li

Report no	: 2025.034
Coach	: Dr. B. Shyrokau
Professor	: Dr. S.H. HosseinNia Kani
Specialisation	: Mechatronic System Design (MSD)
Type of report	: Master's Thesis
Date	: 14 July 2025

A Tentacle-Based Motion Planning Algorithm for Automated Vehicles

Ensuring Real-Time Performance and Passenger Comfort through Limiting Jerk

by

Zisen Li

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday July 21, 2025 at 10:00 AM.

Student number:	5932483	
Project duration:	September 1, 2024 – July 21, 2025	
Thesis committee:	Dr. ir. B. (Barys) Shyrokau,	TU Delft, supervisor
	Dr. S. H. (Hassan) Hossein Nia Kani,	TU Delft
	Dr. C. (Cosimo) Della Santina	TU Delft
	Ir. A. (Alberto) Bertipaglia,	TU Delft

Style: TU Delft Report Style, with modifications by Daan Zwaneveld

Abstract

Automated driving systems are expected to be revolutionary technologies that will reconstruct mobility by improving safety and efficiency. Among the components of automated driving systems, motion planning plays a critical role as it determines how the vehicle reaches its target location from a given origin. The current challenges of motion planning lie in real-time performance and passenger comfort, which is also the main objective of motion planning algorithms. These challenges often occur simultaneously but are treated separately. Neither of these challenges could be solved at a global level of motion planning, which is mainly concerned with route selection and travel time minimization, and could be computed in advance. Consequently, this thesis primarily focuses on local motion planning related to the maneuvering of a vehicle, ensuring real-time performance and passenger comfort.

To address these two challenges, an extended tentacle-based motion planning algorithm is developed. This is an interpolation curve-based algorithm that could work in real-time because there are no optimizers or learning processes that cost a lot of computational resources in the algorithm. The most important factor that affects passenger comfort in a ride is jerk, which is also known as the time derivative of acceleration. The proposed algorithm manages to control the jerk in both lateral and longitudinal directions, thus ensuring ride comfort. Begin with the current state of the vehicle, including velocity, attitude, and steering angle, a series of geometry curves called tentacles is generated and evaluated. The maximum lateral jerk is limited during tentacle generation to avoid excessive impact on passengers during maneuvering.

In most motion planning algorithms, geometry path planning and speed planning are treated separately. However, in the proposed planning algorithm, the speed profile is generated based on the selected best tentacle, thus making the speed profile more rational and adaptable to current maneuvers. In addition, the target speed of the speed profile is decided based on road curvature and traffic conditions, ensuring safety and avoiding wasting time on some low-speed traffic participants. More importantly, the speed profile limits the maximum longitudinal jerk, thus making jerk limited in all directions and ensuring passenger comfort.

To evaluate the performance of the proposed motion planning algorithm, simulations using a high-fidelity vehicle model through IPG CarMaker and MATLAB/Simulink are implemented under various conditions. Because this report does not focus on the design of the path-following controller, the vehicle directly uses the output of the planner as control input during simulations. Even so, the algorithm still manages to complete static and dynamic obstacle avoidance as well as adaptive following and overtaking maneuvers at various speed ranges and road conditions. A virtual map of part of the campus of TU Delft is also constructed using Carmaker and used for validation of the proposed algorithm under urban traffic conditions. The proposed algorithm works effectively in complex environments and can operate in real time at a frequency of $20\ Hz$ *while constraining the total jerk.*

This research illustrated the capability of the developed tentacle-based motion planning algorithm to ensure passenger comfort and safety under various traffic conditions. Although primarily serving as a concept emphasizing feasibility through simulations rather than immediate on-road verification, the proposed algorithm establishes a foundation for future real-vehicle implementation, thus contributing towards resolving normal driving conditions essential for achieving fully automated driving.

Nomenclature

Abbreviations

Abbreviation	Definition
AHP	Analytic Hierarchy Process
AV	Automated Vehicle
ADS	Automated Driving Systems
CoG	Center of Gravity
GPS	Global Positioning System
IMU	Inertial Measurement Unit
LiDAR	Light Detection And Ranging
LQR	Linear Quadratic Regulator
MPC	Model Predictive Control
OCP	Optimal Control Problem
PRM	Probabilistic Road Map
RMS	Root Mean Square
RRT	Rapidly-exploring Random Trees
SQP	Sequential Quadratic Programming
TTC	Time-To-Collision

Symbols

Symbol	Definition	Unit
a_{lat}	Lateral acceleration	m$/s^2$
a_{lat}^{RMS}	RMS of lateral acceleration	m$/s^2$
a_{long}	Longitudinal acceleration	m$/s^2$
a_{long}^{RMS}	RMS of longitudinal acceleration	m$/s^2$
$C_{\alpha f}$	Lateral stiffness of the front axle	N/rad
$C_{\alpha r}$	Lateral stiffness of the rear axle	N/rad
d	Length of the tentacle between the detection point and the obstacle	m
d_s	Minimum braking distance	m
f	Refreshing rate	Hz
I_z	Moment of inertia in vertical direction	kg·m^2
i_s	Steering ratio	$-$
j_{lat}	Lateral jerk	m/s^3
j_{lat}^{RMS}	RMS of lateral jerk	m/s^3
j_{long}	Longitudinal jerk	m/s^3
j_{long}^{RMS}	RMS of longitudinal jerk	m/s^3
K_{us}	Understeer gradient	$-$
L	Wheelbase	m
L_c	Collision distance	m
L_t	Safe length of the best tentacle	m
l	Length of the vehicle	m
l_c	Crash distance	m
l_f	Distance from center of gravity to the front axle	m
l_r	Distance from center of gravity to the rear axle	m

Symbol	Definition	Unit
l_s	Minimum lateral safe distance	m
m	Mass	kg
n	Number of tentacles	$-$
R	Curvature radius	m
r	Yaw rate	rad/s
t	Time	s
v	Velocity of the ego vehicle	m/s
v_{lat}	Lateral velocity	m/s
v_{long}	Longitudinal velocity	m/s
v_{obs}	Velocity of the obstacle	m/s
v_r	Relative velocity	m/s
v_{ref}	Reference velocity	m/s
v_{tar}	Target velocity	m/s
w	Width of the ego vehicle	m
w_0	Width of the unsafe region	m
w_o	Width of the obstacle	m
ρ	Curvature	m^{-1}
ϕ	Heading angle	rad
δ	Front wheel steering angle	rad
δ_s	Steering wheel angle	rad

Contents

List of Figures

List of Tables

1 Introduction

1.1. Background

Recently, the number of private vehicles has grown by leaps and bounds. Automobiles not only bring convenience to people's lives, but also begin to gradually become the third space of people's lives. However, the drawbacks of increasing car ownership, led by serious traffic safety problems, cannot be ignored. According to the data, there were 1.19 million road fatalities in 2021 [1]. It has been shown that most of the traffic accidents are caused by driver errors [2]. In order to reduce the possibility of driver error and traffic accidents, automated driving technology is gradually being developed.

Research in automated driving technology began with early concepts in the early 1900s and has continued to advance with improvements in computational, artificial intelligence, and sensor technologies. In the 1980s, the pioneering work of E. Dickmanns and the PROMETHEUS program drove the initial development of automated driving technology, which enabled partially automated driving of vehicles in closed environments [3]. In the mid-2000s, the DARPA Challenge Events became an important milestone in the field of automated driving, bringing together experts from different fields and promoting the interdisciplinary convergence of technologies [4]. These challenge events significantly accelerated research and investment in automated driving technology, furthering automated driving research and development in universities and industry.

Automated Driving Systems (ADS) consist of multiple collaborative subsystems to enable perception, decision-making, and control. The main systems include acquisition, perception, communication, evaluation, path planning, and control modules. The architecture of an automated driving system is shown in Figure 1.2. The system acquires data from the vehicle and the environment through proprioceptive sensors and exteroceptive sensors. Proprioceptive sensors are responsible for acquiring information about the vehicle's own state, such as speed and acceleration, while exteroceptive sensors, such as light detection and ranging (LiDAR), cameras, and radar, detect changes in the external environment. The perception system is responsible for processing the sensor data and realizing functions such as object detection, tracking, and lane recognition to understand the surrounding environment. The positioning system combines the Global Positioning System (GPS) and the Inertial Measurement Unit (IMU) to determine the precise position of the vehicle on the road. The assessment system analyzes environmental risks, including predicting the behavior of surrounding vehicles and pedestrians to improve driving safety.

Based on this, the decision-making system makes behavioral decisions and generates a drivable path from the current position to the target position, which is divided into global path planning and local path planning to ensure the safety and efficiency of the path. The control system translates path planning results into specific maneuvering commands for the vehicle, such as accelerating, braking, and steering, to ensure that the vehicle accurately executes the planned path. The actuation module is responsible for implementing control commands to specific actuators such as the steering wheel, throttle, and brake pedals. This module directly drives the vehicle movements to accomplish driving tasks. The seamless collaboration between the systems enables automated driving in complex and dynamic

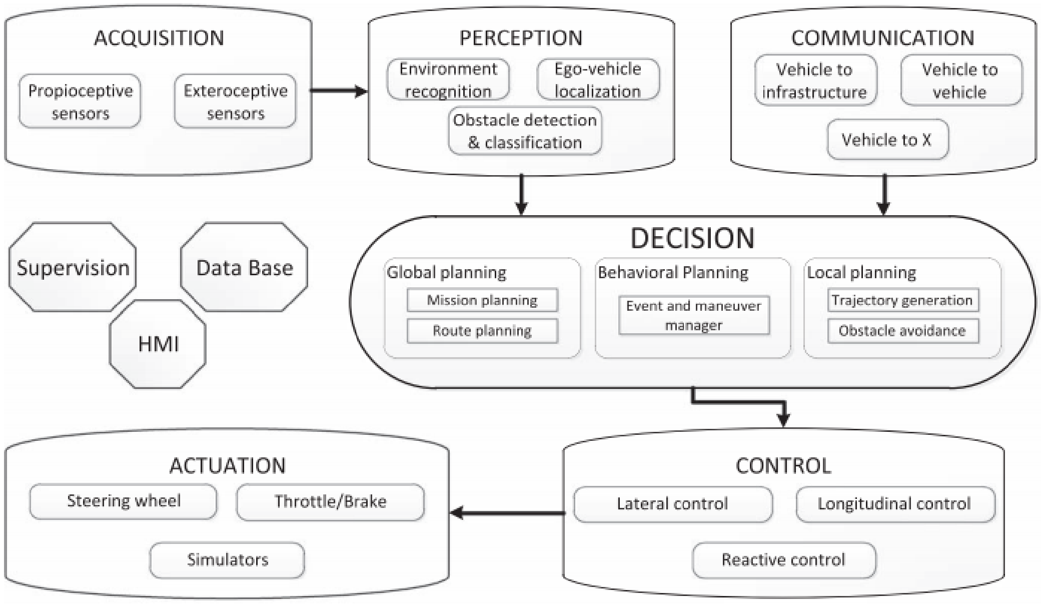


Figure 1.1: Architecture of ADS [5]

road environments. Despite the significant progress of path planning algorithms in recent years, they still face challenges regarding real-time performance, obstacle avoidance capability, and passenger comfort. Therefore, continuous improvement of path planning methods, including techniques such as graph search-based, sampling-based, optimization-based, and learning-based methods, has become an important research direction in the field of automated driving.

1.2. Problem Statement

Traditional path planning algorithms usually focus on the feasibility and safety of the path, but less research on the comfort of vehicle driving. Especially the lack of effective restrictions on the vehicle lateral and longitudinal acceleration derivatives (jerk) in the path generation process, which results in planned paths that are difficult to achieve comfortable driving in practical applications. In addition, the more common path planning methods based on learning or numerical optimization, although they can achieve higher planning accuracy, have high computational complexity and a large computational burden, resulting in algorithms that cannot run in real time, which cannot meet the stringent requirements for response speed in the actual automated driving environment. Therefore, how to take into account both driving comfort and real-time performance in the motion planning process, especially under the condition of strictly restricting the longitudinal and lateral jerk, to quickly plan a highly comfortable path has become an important problem that needs to be solved. Based on this, this study will focus on the following research question:

How to design an effective path planning method to ensure the real-time performance and computational efficiency of the planning algorithm while significantly reducing the lateral and longitudinal jerk?

It should be noted that this research mainly focuses on the design of the motion planning algorithm itself, and does not involve research at the perception and control levels. Specifically, at the perception level, we assume that the environmental information captured by the sensors carried by the vehicle is accurate and complete, with no omissions or errors, and is not affected by weather, light or environmental conditions; information such as obstacle locations, auto-vehicle states, and road boundaries are assumed to be accurately known, and thus the effect of perceptual uncertainty is not considered. At the control level, it is further assumed that the controller used by the vehicle is an ideal controller that can perfectly track the path generated by the planning algorithm without any lateral or longitudinal tracking errors. These assumptions allow this research to focus on the performance of the motion planning algorithm, clearly defining the research boundaries and focus of this report.

Figure 1.2: A complex urban scenario

1.3. Main Contributions

1.3.1. Clothoid-Based Tentacle Generation for Lateral Jerk Limitation

This report proposes a clothoid-based tentacle generation method to solve the problem of sudden curvature changes in the baseline method. Different from the circular tentacles, the curvature of clothoid tentacles varies with arc length. The lateral jerk is limited under $2m/s^3$ through limiting the slope of curvature growth. Besides, the weighting parameters used in the tentacle selection process are varied depending on whether an obstacle is encountered or not, thus balancing the performance between tracking the reference path and obstacle avoidance. Simulation results show that the proposed method can reduce the maximum curvature jump at the joints by more than 80% compared with the baseline method, which can significantly alleviate the vibration and lateral bumps caused by the discontinuous path curvature of the vehicle, and improve the ride comfort and robustness of the vehicle maneuvering.

1.3.2. Obstacle-Aware Adaptive Speed Planning with Longitudinal Jerk Constraints

In order to further enhance the driving comfort, this report proposes a novel speed planning approach that incorporates an explicit longitudinal jerk constraint. The speed planning algorithm takes the result of tentacle selection and plans the most appropriate target speed based on the best tentacle, making the speed profile more rational and adaptable to the current maneuver. The maximum longitudinal jerk of the vehicle is limited within $3\ m/s^3$ by a saturated PD controller that takes the speed error and outputs the longitudinal jerk. In addition, the proposed speed planning algorithm dynamically adapts traffic and road conditions by providing additional acceleration while overtaking moving obstacles and reducing target speed when encountering curved roads.

1.3.3. Multi-scenario simulation with high-fidelity vehicle simulator

In order to verify the feasibility and robustness of the proposed motion planning algorithm in real scenarios, a joint simulation platform of IPG CarMaker and MATLAB/Simulink is constructed. In highway conditions, the simulated vehicle cruises at the velocity of $90\ km/h$ and performs path replanning and avoidance in front of dynamic obstacles. In the urban condition, a stretch of road network around the TU Delft campus was chosen as a simulation scenario, a speed limit of $30\ km/h$ is set, and static as well as dynamic obstacles (cyclists and vehicles) are introduced. The simulation process is performed by calling the path and speed profiles generated by the proposed algorithm in real time, and detailed vehicle dynamics simulation is performed by CarMaker. The test metrics include replanning time and lateral and longitudinal acceleration and jerk. The results show that the maximum lateral and longitudinal acceleration is limited within $2m/s^2$ in all conditions, which satisfies the comfort requirements. This validation not only proves the practicality of the algorithm under multiple complex road conditions but also provides a reusable joint simulation scheme for subsequent engineering deployment.

1.4. Thesis Structure

The rest of the report is organized as follows. In Chapter 2, the literature review on sampling-based and interpolation curve-based planning algorithms is presented, discussing their strengths and weaknesses with respect to real-time performance and path comfort. Followed by a section showing the baseline method of this research, the tentacle-based planning algorithm, and the challenges faced within the scope of this thesis. In Chapter 3, the proposed methodology is introduced. Firstly, the chapter shows how to generate a series of clothoid-based tentacle paths that restrict lateral jerk. Then, the criteria for choosing the best tentacle path to execute are presented with a discussion of the selection of parameters in the algorithm. A speed planning algorithm is also proposed with jerk control and adaptation to curved roads and overtaking conditions. In Chapter 4, a series of simulations using CarMaker and Matlab/Simulink are conducted. The simulations involve highway and urban road conditions as well as dynamic and static obstacle avoidance. Simulation results, including lateral and longitudinal acceleration and jerk as well as real-time performance, are presented. Then, the overall strengths and limitations of the proposed method are discussed. Chapter 5 draws final conclusions and makes meaningful recommendations for future works.

2 Literature Review

This chapter elaborates on related works regarding motion planning techniques for automated vehicles and explains the baseline method upon which the proposed algorithm is constructed. Firstly, some representative approaches of sampling-based methods and interpolating curve-based methods are represented and discussed regarding real-time performance and ride comfort. Then, the baseline tentacle-based motion planning method is introduced, highlighting its advantages regarding both of these metrics as well as drawbacks.

2.1. Motion Planning for Automated Vehicles

In the existing literature, there is a wide range of types of motion planning algorithms, which can be broadly categorized into sampling-based methods, graph search-based methods, optimization-based methods, learning-based methods, and interpolation curve-based methods etc. However, each method has its own shortcomings: graph search methods are limited by the mesh resolution, computationally intensive and difficult to smooth; optimization methods, despite the direct constraints on the comfort of the non-linear solution, are seriously time-consuming and prone to fall into the local optimum; learning methods require a large amount of labeled data, are time-consuming to train, and have poor model interpretability. Given that the goal of this study is to balance real-time performance with smooth curvature and low jerk, we choose a relatively balanced sampling-based and interpolation curve-based planning framework, which can quickly generate feasible paths and ensure smooth and continuous curvature through spline or polynomial interpolation.

2.1.1. Sampling-based Method

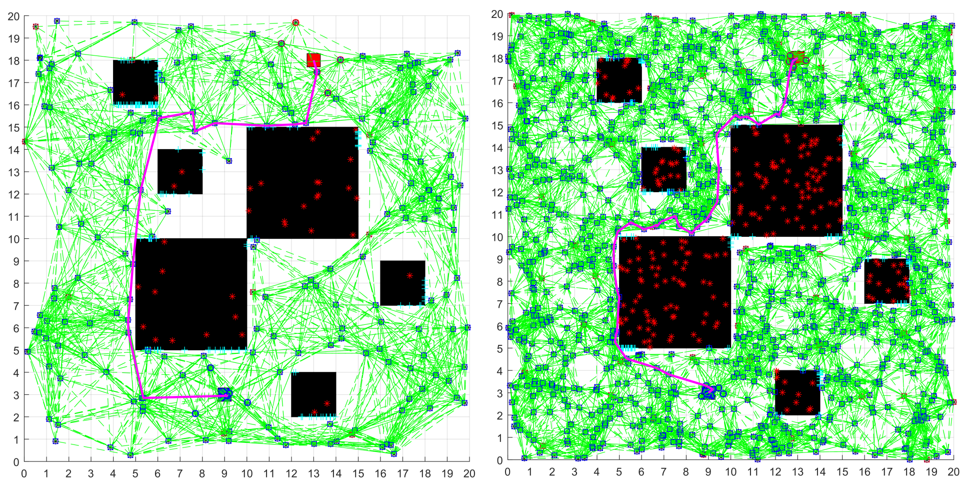
After research, basically all sampling-based methods could be classified into Probabilistic Maps (PRM) and Rapidly-Exploring Random Trees (RRT). The PRM is designed to find collision-free paths in high-dimensional configuration spaces. In the learning phase, PRM randomly generates collision-free nodes within the free space, then attempts to connect each node to nearby nodes via a simple local planner, producing edges only if the straight-line connection is collision-free. To improve coverage in challenging regions such as narrow passages, additional sampling and connectivity checks are performed selectively, ensuring the roadmap graph is sufficiently dense where it matters most. Once the roadmap is built, the query phase integrates the user's start and goal configurations by linking them to their nearest roadmap nodes. A standard graph search method— such as Dijkstra [6] or A* [7] —then finds the shortest node-to-node path. If a path exists, it is stitched together with the start and goal segments and optionally smoothed to improve quality. As more samples are added, the roadmap becomes better connected and the resulting path length typically decreases.

PRM's strengths lie in its ability to bypass exhaustive exploration of the configuration space—random sampling allows it to scale to high degrees of freedom—and its robustness across varied obstacle layouts. However, its weaknesses include variable path quality, which depends heavily on the number of samples and often requires post-processing to improve smoothness or optimality. Figure 2.1 shows the visualization of PRM with a different number of sampling points. The red and blue points represent the start and goal configuration, and the black blocks represent obstacles. The hollow blue points are the random sampling points in the space, and the green lines are the edges between two neighboring points. The small red dots show the points sampled inside the obstacle that are removed from the roadmap. In Figure 2.1a, a feasible path is generated and shown in a pink line, which is suboptimal. In figure 2.1b, the path generated has a shorter distance from start to goal when the number of sampling points increases to 1000. PRM is also inherently designed for static environments; in dynamic settings with moving obstacles, the precomputed roadmap may quickly become invalid, leading to planning failure. Moreover, narrow regions may remain under-sampled unless sampling density is locally increased, which raises computational costs.

To address PRM's suboptimal convergence, an improved algorithm called PRM* is proposed in [8], it adapts the connection radius as a function of the current number of samples, ensuring each new sample attempts connections within a gradually shrinking radius r, expressed as the following equation.

$$r(n) = \gamma \left(\frac{\log(n)}{n} \right)^{\frac{1}{d}} \tag{2.1}$$

where γ is a constant, d is the dimension of the space and n is the number of sampling points. This dynamic radius maintains a target average number of connections per node and enables continuous, incremental path cost optimization during roadmap construction. PRM* is probabilistically complete—if a collision-free path exists, the probability of finding it approaches one as sampling continues—and asymptotically optimal, meaning the cost of the best path in the roadmap converges to the true optimal cost as the sample count grows without bound.



(a) 200 sampling points **(b)** 1000 sampling points

Figure 2.1: PRM visualization

In addition to PRM algorithm, the Rapidly-Exploring Random Tree (RRT) [9] quickly finds collision-free paths in high-dimensional, non-holonomic spaces by growing a tree through random sampling. Starting from the initial state, each iteration samples a random free-space configuration, finds the nearest tree node, and "steers" a fixed step toward it. If the new segment is collision-free, it's added to the tree. Sampling continues until a node reaches the goal region, yielding a feasible path. RRT's principal advantages are its rapid exploration of large spaces and its probabilistic completeness. Its expected runtime and failure probability can be bounded in terms of path length, clearance from obstacles, and sampling density, ensuring predictable performance in practice.

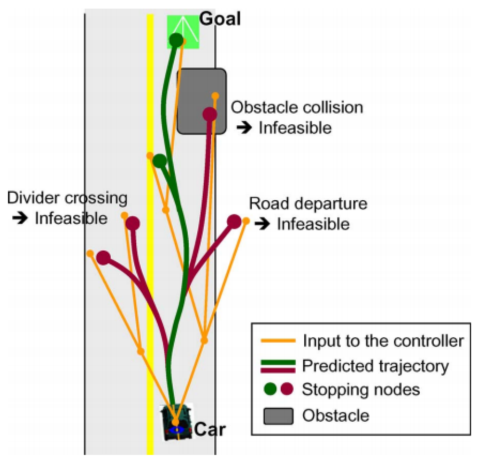
However, RRT does not aim for optimality. As the number of samples increases, the cost of the best path in the RRT tree converges to a suboptimal value. To address this, RRT* is proposed in [8], it augments the basic RRT framework with a "rewiring" mechanism and a dynamically shrinking connection radius. When a new node is generated, RRT* identifies all existing nodes within a radius. Among these neighboring nodes, RRT* chooses the parent that yields the lowest total cost to the new node, and then attempts to reconnect each of those neighbors through the new node if this would lower their cost. By continually performing this local optimization at each insertion, RRT* guarantees asymptotic optimality, the cost of the best path in the tree converges to the globally optimal cost. Although these local rewiring steps increase computational overhead, they produce significantly smoother, shorter paths than basic RRT, making RRT* the preferred choice when path quality is critical.

Extensions of the RRT framework have been developed to handle additional practical constraints. The Closed-Loop RRT (CL-RRT) proposed in [10] samples in the space of control inputs (e.g., steering and acceleration commands) rather than configurations, using a feedback controller to predict closed-loop system responses, shown in Figure 2.2a. CL-RRT also employs environmentally dependent biasing schemes, focusing sampling near intersections or along likely lanes to improve efficiency in structured environments, and it incorporates a collision risk penalty in its cost function. CL-RRT has been demonstrated in urban driving scenarios to safely navigate dynamic obstacles.

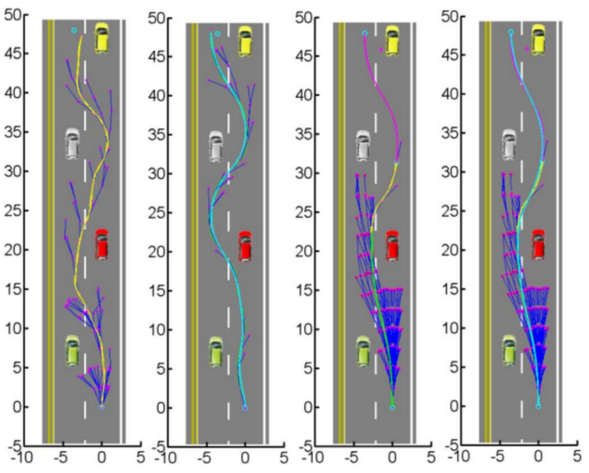
For vehicles with non-holonomic dynamics such as cars that must follow curvature constraints, RRT* can be further adapted by replacing the “steer” step with a model-specific steering function [11]. In Dubins RRT*, each potential connection uses the minimum length Dubins path (combining straight lines and maximum steering arcs). As with RRT*, rewiring then optimizes cost locally. This yields feasible and asymptotically optimal paths for simple car-like models.

For more complex, nonlinear dynamics—such as a double integrator or a bicycle model that captures yaw dynamics and load transfer—approximate steer functions solve two-point boundary-value problems via shooting methods in [12]. A repropagation step then adjusts descendant nodes whenever a parent changes, preserving tree consistency. Heuristic “cost-to-go” estimation and conservative reachable set checking prune unlikely branches, improving runtime without sacrificing optimality. In high-speed automated driving, mapping tire-friction limits into elliptical acceleration constraints at a pseudo-flat output (the vehicle’s front center of oscillation [13]) enables dynamically feasible RRT* paths that satisfy both steering and acceleration limits.

Practical variants such as Fast RRT [14] combine offline rule templates, which are predefined maneuver segments for typical traffic scenarios with aggressive extensions that attempt direct connections to the goal when possible, shown in Figure 2.2b. An environmental assessment module dynamically selects between standard and aggressive extensions to accelerate convergence. The static obstacle avoidance method adds a pruning step to eliminate redundant nodes and smooth the resulting path via Bézier curve fitting. Sampling can be focused on a Gaussian sector ahead of the vehicle, and adaptive sampling regions are defined by recent success rates, further improving efficiency.



(a) CL-RRT [10]



(b) Fast RRT [14]

Figure 2.2: RRT approaches

Sampling-based path planning methods have the advantages of adaptability and simplicity of implementation in high-dimensional complex spaces. However, they may produce unsmooth, suboptimal paths. In high-dimensional or dynamic environments, sampling inefficiency increases search time and hampers real-time performance. Random sampling yields variable path quality, risking local optima or failure in narrow passages. Repeated collision checks also incur high computational costs. Moreover, these methods weakly address vehicle kinematics and dynamics, and handling multi-objective or constrained scenarios demands complex extensions. Overall, their limitations in path quality, efficiency, and adaptability need further optimization or hybrid approaches.

2.1.2. Interpolating Curve-based Method

To address sampling-based methods' shortcomings in smoothness and efficiency, we introduce interpolation curve–based motion planning methods. The core idea is to define key control points along a global reference path and use analytical interpolation to generate continuous paths with smoothly varying curvature. This approach inherently satisfies vehicle kinematic and dynamic constraints while avoiding the heavy computations of random sampling. The resulting paths exhibit no sharp turns or abrupt curvature changes, improving compatibility with high-speed tracking. Moreover, curvature, acceleration, or other constraints can be readily incorporated, enhancing flexibility in complex environments. In the following, representative studies using polynomial, Bezier, and spline curves are presented.

In polynomial curve methods, cubic or quintic polynomials are constructed, which define vehicle paths as functions of one or more independent variables (typically time or arc length). Coefficients of polynomials are chosen to satisfy boundary constraints: typically position, velocity, and acceleration at the start and end points. This smoothness is crucial for high-speed maneuvers, complex lane changes, and respect for vehicle dynamics.

Building on these foundations, several real-time planners use polynomial-based endpoint interpolation followed by optimization. One such framework [15] samples lateral and longitudinal offsets along the road centerline, then connects pairs of sampled endpoints via cubic polynomials. The coefficients are computed by gradient descent under curvature and heading constraints to avoid curvature discontinuities during successive replanning, thus realizing a quadratic polynomial expansion at the current attitude of the vehicle. An “inverse method” then discretizes speed profiles and fits cubic polynomials to match vertex constraints. Finally, each candidate path is scored by weighted costs including length, curvature, jerk and offset, and optimized via non-derivative simplex methods, yielding marked improvements in path quality and runtime.

A specialized overtaking planner [16] models the overtaking maneuver in three phases (lane departure, side-by-side passing, and lane return), using cubic polynomials in the first and third segments, and matching positions and velocities at the phase boundaries. A nonlinear adaptive controller then tracks the generated path, estimating the overtaken vehicle's speed online to reduce the dependence on external measurements.

A FRENETIX framework [17] offers a modular, high-performance planner capable of city and highway driving. It updates the vehicle's state, and samples a lattice of end states in time, lateral offset, and longitudinal velocity. Quintic polynomials generate lateral paths while cubic polynomials generate longitudinal ones. After transforming candidates back to Cartesian space, kinematic feasibility (acceleration, curvature, and derivative limits) is checked, and paths are scored on comfort, offset, obstacle distance, and collision probability. The feasible path with the best score is selected; if none exists, an emergency braking path is generated.

Invented by Pierre Bezier for automotive body design, Bezier curves are parametric curves defined by a series of control points and Bernstein polynomial, and can be calculated using the de Casteljau algorithm [19] by recursively subdividing a Bezier curve into two segments. Early motion planning methods proposed in [19] fit cubic Bezier curves at turns using midpoint constraints on corner bisectors, then optimize control points to minimize curvature variation and curvature jumps between segments, yielding smoother steering profiles.

A subsequent work [20] integrates obstacle avoidance by combining Bezier fitting with a time-varying potential field representing road boundaries and obstacles defined in Frenet coordinates. A quartic Bézier curve with five control points is optimized under curvature, curvature derivative, and dynamics constraints using Sequential Quadratic Programming (SQP), with initial guesses from a simple heuristic. The speed profile is then computed by a quadratic polynomial matching path length and velocity

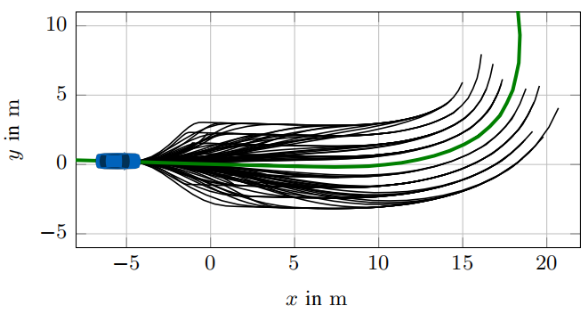


Figure 2.3: Polynomial curve method [17]

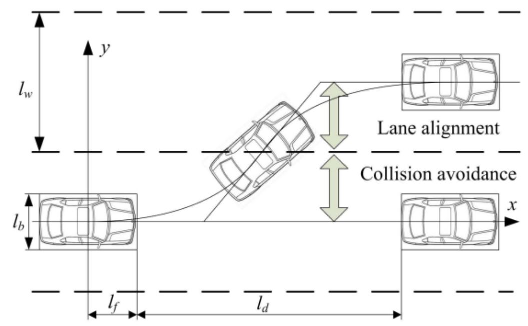


Figure 2.4: Bezier curve method [18]

boundaries.

In [18], a trajectory planning method is proposed using three-dimensional quartic Bezier curves for collision avoidance and lane alignment, treating vehicle speed as a vertical coordinate alongside planar position. By expressing lateral and velocity control points in terms of a single optimization variable (the longitudinal coordinate of the final point), the planner minimizes obstacle proximity, curvature variation, maneuver time, and acceleration constraints, yielding integrated path and speed solutions.

Splines are piecewise polynomial curves composed of segments defined over sub-intervals. They offer local control, which means adjusting one control point affects only its neighboring segments, and avoids the oscillations typical of high-degree polynomials. The first type of spline is polynomial spline, where each segment is a low-degree polynomial joined at knots with continuity constraints up to a specified derivative order. An efficient lane change algorithm is proposed in [21], it uses G^2-quintic splines: by constraining lateral jerk to have a single extremum, it bounds maneuver time to prevent overshoot, and samples candidate paths within a safe region derived from map and dynamic information. Sampling density is adapted to relative speeds, and experiments show robust performance in urban driving.

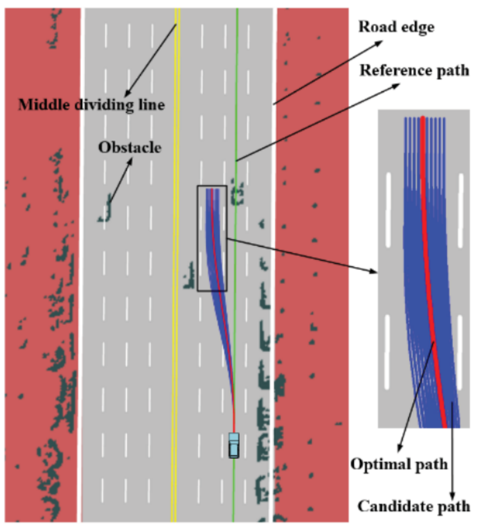


Figure 2.5: G^2 polynomial spline method [21]

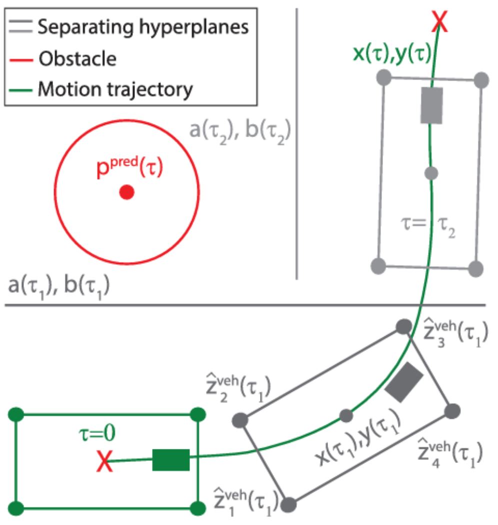


Figure 2.6: B-spline method [22]

In addition to polynomial splines, a more commonly used type of spline is the B-spline curve. B-spline curves are developed to address the shortcomings of Bézier curves, which could not locally control the direction of the curve. A move of one control point on the Bézier curve causes the entire curve to change. A B-spline curve is defined by a number of control points P_i and a basis function $N_{i,k}(u)$. An early approach [23] computes a safe corridor along mine walls via the Minkowski sum, then fits a quadratic B-spline whose curvature variation is minimized to reduce jerk and maximize smoothness.

Obstacle avoidance and vehicle dynamics constraints form a nonlinear program solved by SQP. Simultaneously, a velocity profile is optimized through acceleration, constant speed, and deceleration phases. Compared to a prior piecewise 7th-degree method, this scheme yields smoother paths, shorter travel times, and higher efficiency, particularly well suited to industrial settings with static obstacles.

Building on string stability theory, [24] embeds inter-vehicle perturbation attenuation directly into the trajectory planner. Control points of a B-spline representation are optimized under objectives for smoothness, spacing error, and lateral offset while enforcing dynamics, collision avoidance, and minimum inter-vehicle distances. To mitigate communication delays, a real-time prediction model adjusts control points, and only a small subset of trajectory parameters need to be shared among vehicles, reducing bandwidth requirements. Simulations confirm smooth, stable platooning even with significant latency.

In [22], the infinite-dimensional Optimal Control Problem (OCP) is reconstructed by representing all state and control paths as B-splines. The nonlinear variation of variables transforms the original constraints, as well as the constraints to avoid moving obstacles by separating hyperplanes through time parameterization, into spline coefficient constraints. Each obstacle is predicted linearly, and the collision check is simplified to verifying that the vehicle's polygon vertices remain on one side of a moving hyperplane. A receding-horizon SQP solver updates and optimizes every cycle, minimizing travel time. Extensive simulations demonstrate robust, collision-free performance in dynamic settings.

Reference [25] divides a lane-change maneuver into four phases, each modeled by a quasi-uniform cubic B-spline whose control points anchor to phase start and end positions. The return to the original lane mirrors the departure curve for continuity. Constraints enforce collision clearance, rollover and side-slip limits, and road-boundary adherence. A unified cost function weights path safety, smoothness, and comfort. Indicators are normalized to $[0, 1]$ and combined; heading angle discretization generates a limited candidate set, from which the lowest cost path is chosen.

Interpolation curve-based motion planning algorithms excel at producing smooth, continuous paths with minimal jerk, making them highly efficient and comfortable in static or structured environments. By fitting low-order polynomials or splines through predefined control points, they guarantee path and derivative continuity, ensure passenger comfort, and simplify implementation. However, their reliance on fixed waypoints and the need to recompute entire curves whenever obstacles move undermines real-time performance. In complex, unstructured, or highly dynamic scenarios, where obstacles and goals change unpredictably, interpolation methods struggle to adapt quickly and maintain computational efficiency, limiting their practical applicability.

2.2. Baseline Method

2.2.1. Tentacle-based Motion Planning Algorithm

The tentacle algorithm is a motion planning method that mimics the behaviour of insects that use their tentacle to detect and avoid obstacles, which is commonly used in automated vehicles. The core idea is to generate a series of predefined paths (called tentacles) centered on the vehicle, representing the possible motion paths at the current speed and direction. The shape of the tentacle can be circular [26], straight, clothoid curves [27], or self-defined curves [28]. After tentacle generation, the best tentacle is selected based on multiple criteria. Since only local planning is considered in each computation of tentacles, the response of the tentacle-based algorithm is fast and suitable for dynamic environments. Therefore, it is able to optimize path planning in combination with other methods (e.g., interpolating curves, occupancy grids, and sampling-based methods) to enhance adaptability.

A real-time path planning algorithm for automated vehicles combining the tentacle algorithm and B-spline curve is proposed in [29]. Firstly, a two-dimensional occupancy grid map with obstacle and road boundary information around the self-vehicle is constructed. On the grid map, a set of circular tentacles is generated offline when neglecting the vehicle's initial steering angle. Vehicle speed is divided into 20 ranges, and for each range, 81 tentacles with different radii are generated. The radius r^k of the k^{th} tentacles is defined based on the current speed set, shown by:

$$r_k = \begin{cases} p^k R_j, & k = 0, \dots, 39, \\ \infty, & k = 40, \\ p^{80-k} R_j, & k = 41, \dots, 80. \end{cases} \tag{2.2}$$

where p is the correction factor, and R_j is the initial radius of speed set j related to the current speed. The vehicle's speed is assumed not to change on each tentacle path. The generated tentacle path set is shown in Figure 2.7.

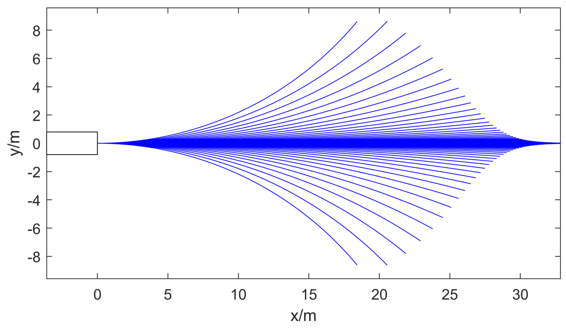


Figure 2.7: Tentacle paths for a speed set [29]

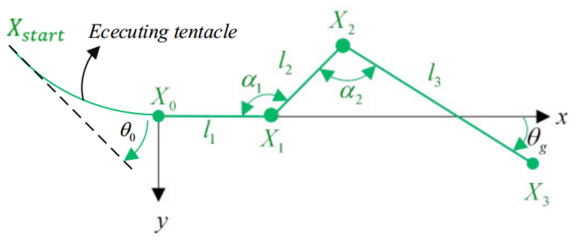


Figure 2.8: Control path segment [29]

The proposed method considers only the safety distance as the criterion for best tentacle selection. The safety distance is derived by collision check of the detection area, which is generated by expanding with the safe width along the path point on each tentacle. The execution time on the selected best tentacle is set according to the speed set. In a situation where the safety distance is less than the minimum, the vehicle will brake in the current direction and replan at the next interval.

After generating and selecting tentacle paths, cubic B-spline curves are used to generate follow-up paths. The B-spline curve is generated using three control segments with four control points X_0, X_1, X_2, X_3 as shown in Figure 3.3, where X_{start} and X_0 is the start and end point of the executing tentacle path, θ_g is the heading angle of the target point, and θ_0 is the heading angle of X_0. l_1, l_2, l_3 are lengths of each segments and α_1, α_2 are angles between two segments. In order to limit the maximum curvature of the B-spline curve, the distance L of adjacent vertices and the angle α between them need to be satisfied as:

$$L \geq L_{\min} = \frac{1}{6}\frac{\sin\alpha}{\kappa_{\max}}\left(\frac{1-\cos\alpha}{8}\right)^{-1.5} \tag{2.3}$$

When the coordinate (x_g, y_g) of the target point X_3 is determined by demand of the maneuver, and α_1 and l_1 are given, the expression of l_2 and l_3 can be derived as:

$$\begin{cases} l_3 = \dfrac{(l_1 - x_g)\sin\alpha_1 + y_g\cos\alpha_1}{\sin\theta_g\cos\alpha_1 - \cos\theta_g\sin\alpha_1}, \\ l_2 = \dfrac{l_3\cos\theta_g + l_1 - x_g}{\cos\alpha_1}. \end{cases} \tag{2.4}$$

Then, the points in the transverse direction of the reference target point are sampled to generate a set of B-spline curves based on Equation 2.4. The target points are selected as shown in Figure 2.9, where the blue curve is the reference path, and the red curve is the best tentacle. G is the reference target point, i.e., the point on the reference path where the obstacle is detected. The distance between sampling points can be determined according to the actual scene. Based on the optimal tentacle path, we only need to sample target points that are close to the sampling area. After collision detection of the generated B-spline path set, the optimal path is selected in terms of path length, curvature, curvature derivative, and offset between the target point and the reference path. The B-spline chosen and the best tentacle paths are combined to form the total obstacle avoidance path.

In order to evaluate the performance of the aforementioned motion planning algorithm, the authors design the simulations and experiments. The simulation is conducted in a lane-changing scenario to avoid obstacles. As shown in Figure 2.10, the frame at the bottom of the figure stands for the self-vehicle, and the black rectangle in front of the self-vehicle is the obstacle. The first subfigure from the left shows the tentacles generated for obstacle detection; the second subfigure shows the best tentacle; the third subfigure demonstrates the B-spline set (blue curve) generated based on sampling points and the executed tentacle path (red curve), and the fourth subfigure shows the selected optimal path. The real-time performance of the proposed algorithm is also demonstrated with an average time of 4.9 ms for the best tentacle path and 9.9 ms for B-spline path generation.

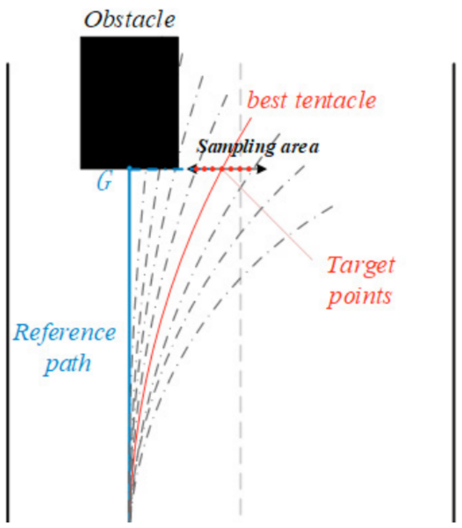


Figure 2.9: Tentacle paths for a speed set [29]

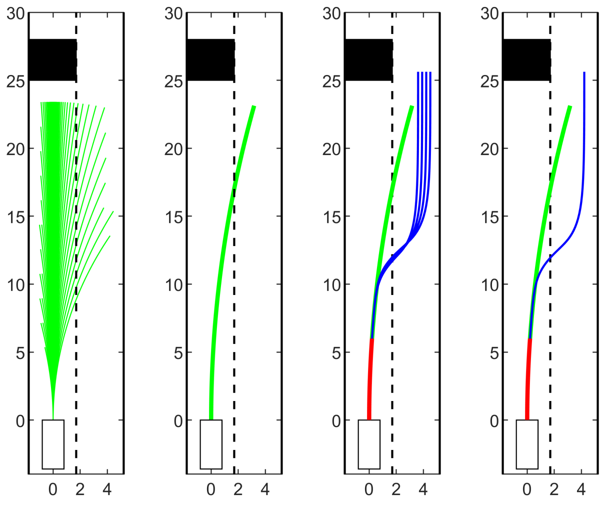
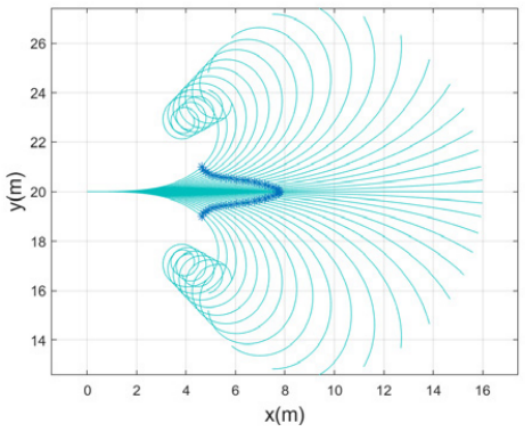


Figure 2.10: Control path segment [29]

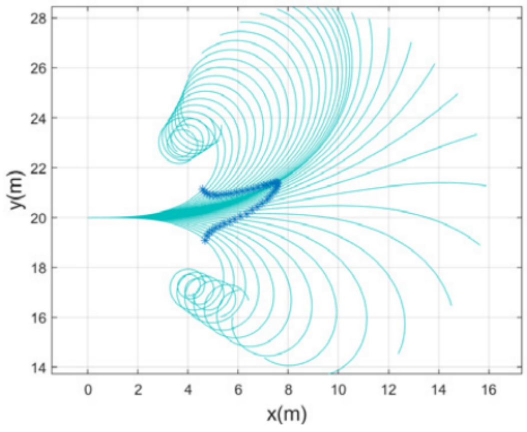
The proposed method in [29] is improved in a later work [30]. Instead of circular tentacles that ignore the current steering angle, the improved tentacles are generated by first defining the curvature derivatives:

$$\frac{d\rho}{ds} = \beta s^{0.5} + C_0 \tag{2.5}$$

where s is the curve length and C_0 is the initial curvature derivative related to the current steering angle. Taking the indefinite integral of Equation 2.5 yields the curvature function that can be converted to Cartesian coordinates. The improved tentacles with different initial steering angles are represented in Figure 2.11.



(a) C_0=0



(b) C_0=0.02

Figure 2.11: Tentacles at different steering angles [30]

After collision detection, the best tentacle is determined by defining the total value as follows:

$$V = \omega_1 V_{ld} + \omega_2 V_\alpha + \omega_3 V_C \tag{2.6}$$

where V_{ld} represents the distance criteria, which is defined based on the deviation distance of the tentacle path from the reference path, V_{α} represents the angle criteria, which is defined based on the angle difference between the tentacle and reference path, V_C represents curvature criteria which is defined based on curvature derivative of the tentacle. Weights ω_i are chosen via Analytic Hierarchy Process (AHP): pairwise comparisons based on accuracy and smooth criteria, eigenvalue consistency check, yielding $\omega_1 = 0.4954$, $\omega_2 = 0.2716$, and $\omega_3 = 0.2330$. Simulations show significantly reduced lateral error and curvature variation versus clothoid tentacles, at the cost of higher average planning time $(20.57\ ms)$ compared to [29].

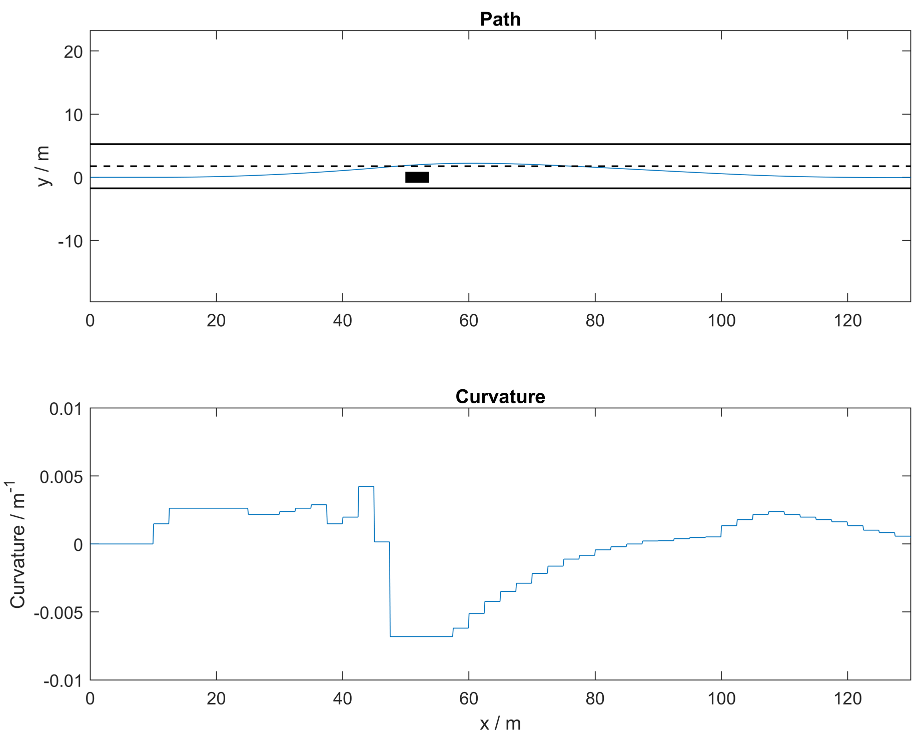


Figure 2.12: Path and curvature of an obstacle avoidance scenario

2.2.2. Limitations

For the problem at hand, the tentacle-based motion planning method is a promising approach, which has the advantage that a series of pre-defined paths can effectively handle dynamic environments, and the lower computational cost of the tentacles makes the algorithm real-time guaranteed. However, the baseline method mentioned above is subject to some limitations that are presented below.

- The baseline algorithm considers only overtaking maneuvers in a straight line, so the tentacles are designed to be circular and each has a constant curvature. Consequently, when the planner decides to choose a different tentacle from the last replanning, a sudden change in curvature appears, shown in Figure 2.12. The sudden change in path curvature leads to jerky lateral acceleration based on the relationship $a_{lat} = v^2\kappa$ with a_{lat} the lateral acceleration, v the velocity and κ the path curvature, which can greatly affect ride comfort. **The improved algorithm should also consider the maneuver in a curved road and thus generate tentacles regarding the self-vehicle's initial steering and heading angles .**
- The baseline algorithm chooses the best tentacle only based on the safe distance, which is considered too simple. **The improved algorithm should implement more comprehensive criteria for the best tentacle selection based on both safety (including safe distance and time to collision) and the executability and comfort of each tentacle.**

- The baseline algorithm assumes the vehicle maintains a constant velocity when executing the collision-avoidance path, which is unrealistic. **Therefore, the improved algorithm will need a velocity planning method to determine the velocity profile and the timing of acceleration and braking.**
- The proposed speed range for the baseline algorithm is under 40 km/h, which is suitable for urban roads but too low for highways. **The improved algorithm should consider the scenario of high-speed overtaking maneuver when the vehicle may exhibit different dynamic properties.**
- The baseline algorithm manages to plan a lane-changing path when facing an obstacle in front, but the path returns to the original lane after overtaking is not generated. **The improved algorithm should also generate the lane-changing path after finishing overtaking while making sure the vehicle would not exceed the lane line boundary.**
- The baseline algorithm validates only the path planning of static obstacle avoidance. The effectiveness of the algorithm for dynamic obstacle avoidance has yet to be verified. **Therefore, the improved algorithm should contain a frequent replanning process to handle sudden obstacles and complex environments.**

3 Methodology

To achieve a real-time motion planning algorithm that can cope with complex environments while satisfying ride comfort, the proposed framework is presented in this chapter. The approach builds upon the tentacle-based real-time path planning methodology [29] discussed in the previous chapter and manages to fix the poor ride comfort by lateral jerk control. Besides, a novel speed planning method with longitudinal jerk control is proposed to improve the feasibility in various traffic scenarios. This chapter is organized as follows: firstly, the tentacle generation method is introduced to generate a series of executable tentacles, then the tentacle selection is implemented to find a safe tentacle to get to the destination among all candidates. Finally, the speed planning method is proposed based on the selected best tentacle.

3.1. Tentacle Generation

3.1.1. Coordinate System

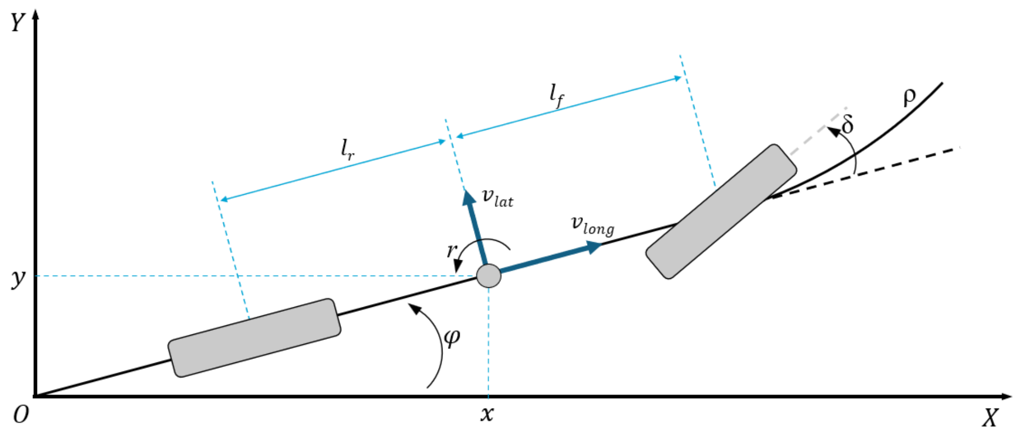
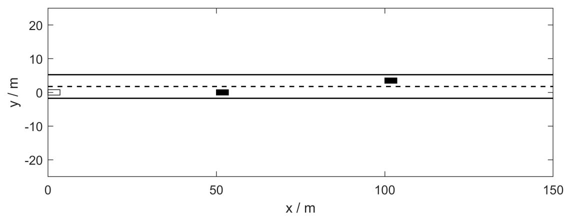


Figure 3.1: Coordinate system

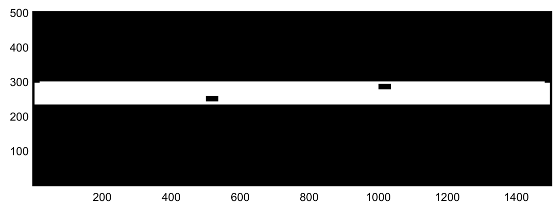
As shown in Figure 3.1, we adopt a right-handed, global inertial coordinate frame (X, Y) in the plane, with its origin at a fixed reference point (e.g., map origin). The vehicle is simplified in the diagram to a bicycle model. The vehicle's position in this frame is given by (x, y), representing the coordinates of its center of gravity (CoG). Its orientation (heading angle) is denoted by ϕ. Longitudinal and lateral velocity of the vehicle CoG are given by v_{long} and v_{lat}, and the steering angle δ is defined positive for a left (counterclockwise) turn relative to the vehicle's heading. The curvature of the vehicle's travel path is denoted by ρ, with positive direction same as the steering angle. These variables and sign conventions will be used throughout the proposed motion planning formulation.

3.1.2. Occupancy Grid Map

Occupancy grid map is a fundamental tool in the perception and navigation module of ADS, providing a discrete, metric map of the environment around a vehicle by tessellating space into square cells, each of which holds an estimate of whether that region is free or occupied. To map the occupancy grid, external perception sensors - most commonly LiDAR, but also radar or camera systems - scan the surroundings at fixed intervals. As mentioned in Chapter 1, this thesis assumes that the sensor measurements are accurate, and the value of square cells could only be 0 or 1, which stands for free or occupied. In each interval (at the same frequency as the planner's operating frequency), the ideal sensor generates an ego-centered, two-dimensional grid map with 1500 × 500 cells, with the size of each cell being 0.1 m × 0.1 m. The extent of the physical space environment around the vehicle is therefore known to be 150 m × 50 m. Figure 3.2 shows an example of an occupancy grid map. The left panel shows a road scene with two obstacles (solid rectangle) in front of the ego-vehicle (hollow rectangle). The right panel shows the occupancy grid map of this scene, with the black area representing the occupied cell because of obstacles and road boundaries, and the white area representing the navigable space.



(a) Road scene



(b) Occupancy grid map

Figure 3.2: Occupancy grid map example

3.1.3. Tentacle Generation Considering Lateral Jerk

The curvature discontinuity problem of the baseline method mentioned in Chapter 2 could cause serious problems on passenger comfort, considering lateral jerk j_{lat}, which can be calculated as follows:

$$j_{lat}(t) = \frac{da_{lat}(t)}{dt} = \frac{d}{dt}\big[v(t)^2\,\rho(t)\big] = 2\,v(t)\,\frac{dv(t)}{dt}\,\rho(t) \;+\; v(t)^2\,\frac{d\rho(t)}{dt}. \tag{3.1}$$

where a_{lat} is lateral acceleration, v is vehicle velocity, and ρ is curvature of the path. Although practically the vehicle speed varies with time, the vehicle speed is assumed to be constant in order to compare more intuitively the lateral jerk performance differences between the different algorithms. With a constant speed v, Equation 3.1 becomes:

$$j_{lat}(t) = v^2\,\frac{d\rho(t)}{dt}. \tag{3.2}$$

In this report, the maximum lateral jerk is limited to 2 m/s^3 as the comfort threshold. Not only does the maximum jerk need to be limited, but frequent changes in the jerk during the ride may also lead to poor comfort, so a criterion is also needed to evaluate the average comfort on the road. The Root Mean Square (RMS) of lateral jerk in a period of time is defined as follows:

$$\mathrm{RMS}\big(j_{lat}\big) = \sqrt{\frac{\sum\limits_{i=1}^{N-1}\big(j_{lat,i}\big)^2\,\Delta t_i}{\sum\limits_{i=1}^{N-1}\Delta t_i}}\,, \quad \Delta t_i = t_{i+1} - t_i. \tag{3.3}$$

where $t = [t_1, t_2, , t_N]$ is the time period.

Spline Interpolation Approach

Cubic natural splines are used to connect the tentacles selected for execution in two successive planning processes because cubic spline curves are not only guaranteed to have continuous function values at the nodes, but also continuous first-order derivatives (tangent direction) and second-order derivatives (curvature derivative). Besides, natural splines implicitly minimize the integral of the square of the second-order derivative of the overall curvature while guaranteeing passage through a given point.

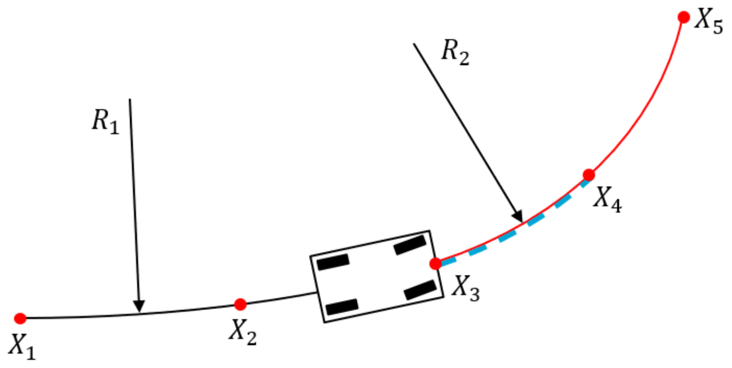


Figure 3.3: Spline interpolation

Figure 3.3 shows the situation that the vehicle has traveled to the end of the current tentacle $\overline{X_1X_3}$, and has planned the tentacle to be executed in the next interval $\overline{X_3X_5}$. X_2 and X_4 are midpoints of $\overline{X_1X_3}$ and $\overline{X_3X_5}$, R_1 and R_2 are radius of $\overline{X_1X_3}$ and $\overline{X_3X_5}$. If $R_1 \neq R_2$, then direct change from tentacle $\overline{X_1X_3}$ to tentacle $\overline{X_3X_5}$ will cause curvature discontinuity, and this is where the cubic spline comes into play. The algorithm takes coordinates of discrete points of arc $\overline{X_2X_3}$ and $\overline{X_4X_5}$ as nodes $\{(x_i, y_i)\}_{i=1}^{n}$ to define the spline. In each interval $[x_i, x_{i+1}]$, the spline is represented by a cubic polynomial:

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad x \in [x_i, x_{i+1}] \tag{3.4}$$

There will be a total of $N - 1$ segments, each with 4 unknown coefficients (a_i, b_i, c_i, d_i) to be identified. To solve these coefficients, several boundary conditions are introduced, the first is an interpolation condition, which ensures the spline passes through the given (x_i, y_i) at all nodes:

$$S_i(x_i) = y_i, \quad S_{i+1}(x_{i+1}) = y_{i+1}, \quad i = 1, \ldots, N - 1 \tag{3.5}$$

Then come the derivative continuity conditions: at the interior node $x_2, ..., x_{N-1}$, it is required that both the first and second order derivatives of the preceding and following segments are the same:

$$S_i'(x_{i+1}) = S_{i+1}'(x_{i+1}), \quad S_i''(x_{i+1}) = S_{i+1}''(x_{i+1}), \quad i = 1, \ldots, N - 2 \tag{3.6}$$

Finally is the natural boundary condition: at the leftmost and rightmost endpoints, it is required that:

$$S_i''(x_1) = 0, \quad S_i(x_N) = 0 \tag{3.7}$$

With all the boundary conditions, coefficients (a_i, b_i, c_i, d_i) could be solved, thus defining the spline curve $\overline{X_3X_3}$ shown in Figure 3.3 by a blue dotted line to connect the two tentacles. This process could be done in MATLAB using the function `spline`. Since curvature is defined by first and second order derivatives:

$$\rho(x) = \frac{S''(x)}{[1 + (S'(x))^2]^{3/2}} \tag{3.8}$$

The derivative continuity conditions shown in Equation 3.6 ensure smooth transition of first and second derivatives, thus ensuring smooth transition of curvature from $\rho_1 = 1/R_1$ to $\rho_2 = 1/R_2$.

Although there are splines to smooth the connections of curvature neighboring planning tentacles, if the difference between the curvatures of neighboring planning is too large, it is still possible that the lateral jerk exceeds the maximum limit. A tentacle selection post-processing method is introduced in Algorithm 1 to limit the curvature difference between two consecutive plannings. In each replanning process, the maximum allowed curvature difference $\Delta\rho_{max}$ is calculated based on current speed v and maximum allowed jerk j_{max} according to Equation 3.2. After the tentacle set T is generated, the tentacle selection method chooses the best tentacle (with index k) to execute, as described in Chapter 3.2. A saturation function then limits the change in the best tentacle index and selects the index k_{sat} that is closest to k under the condition that the difference from the tentacle index k_{pre} used in the previous planning is within the permissible range. Finally, the `Spline` function takes the previous tentacle and current best tentacle as input and outputs the path smoothing by cubic spline.

Algorithm 1: Spline-based approach

```
1 Require $T$, $v$, $j_{max}$ ;
2 foreach planning do
3     $\Delta\rho_{max} \leftarrow$ CalculateMax$\Delta\rho(v, j_{max})$;
4     $k_{pre} \leftarrow k_{sat}$;
5     $k \leftarrow$ TentacleSelection($T$);
6     $k_{sat} \leftarrow$ Saturation($k, k_{pre}, \Delta\rho_{max}$);
7     $p \leftarrow$ Spline($T(k_{sat}), T(k_{pre})$);
8     return $p$
```

A comparison between the spline-based approach and the baseline approach is shown in Figure 3.4. The ego vehicle travels at a constant speed of $10m/s$ and there are two obstacles $50m$ and $90m$ ahead of it, the planner works at a frequency of $20Hz$. Results show that the curvature changes of paths generated by the spline-based approach are smoother compared to the baseline method, resulting in a maximum lateral jerk of $2m/s^{-3}$, whereas the baseline method has no jerk limit and has a maximum value of $98.72m/s^{-3}$. Besides, the spline-based approach also outperforms the baseline method in RMS of lateral jerk, with a value of $0.53m/s^{-3}$ compared to $5.56m/s^{-3}$. It is worth noting that the path planned by the spline-based approach has more lateral offset than the baseline method due to the inability to quickly convert the tentacles after overtaking an obstacle, caused by the limitation of maximum lateral jerk.

Clothoid Approach

A novel clothoid curve-based tentacle generation approach is proposed in this section. Unlike a circular curve with constant curvature, a clothoid curve has a curvature that varies with arc length [31], which improves the smoothness of the generated path. Additionally, the initial curvature of every clothoid tentacle starts with the current curvature of the executing path ρ_0, effectively avoiding current changes in curvature. The clothoid tentacle generation approach is described in Algorithm 2. Firstly, the maximum curvature of the planned path at the speed of v is defined as follows [31]:

$$\rho_{max} = \frac{a_{lat}^{max}}{v^2} \tag{3.9}$$

where $a_{lat}^{max} = 4m/s^2$ is the maximum allowed lateral acceleration that guarantees the stability of the vehicle. The terminal curvature $\rho_{ter}(i)$ for each tentacle could then be defined as follows:

$$\rho_{ter,i} = -\rho_{max} + \frac{2(i-1)\rho_{max}}{n-1}, \quad i = 1, 2, \ldots, n. \tag{3.10}$$

where n is the number of tentacles, taken as 121 in this report. Hence, $\rho_{ter,\ i} = -\rho_{max}$, $\rho_{ter,\ n}(n) = \rho_{max}$ and $\rho_{ter,\ i}$ increases linearly in equal increments from $-\rho_{max}$ to ρ_{max}. Next, the maximum slope of curvature over arc length s could be calculated as:

$$\frac{d\rho}{ds} = \frac{d\rho}{dt}\frac{dt}{ds} = \frac{d\rho}{dt}\frac{1}{v} \tag{3.11}$$

where $\frac{d\rho}{dt}$ is the time derivative of curvature obtained from Equation 3.2. For the tentacle that has the largest change in curvature, the longest arc length is required to grow from the current curvature ρ_0 to

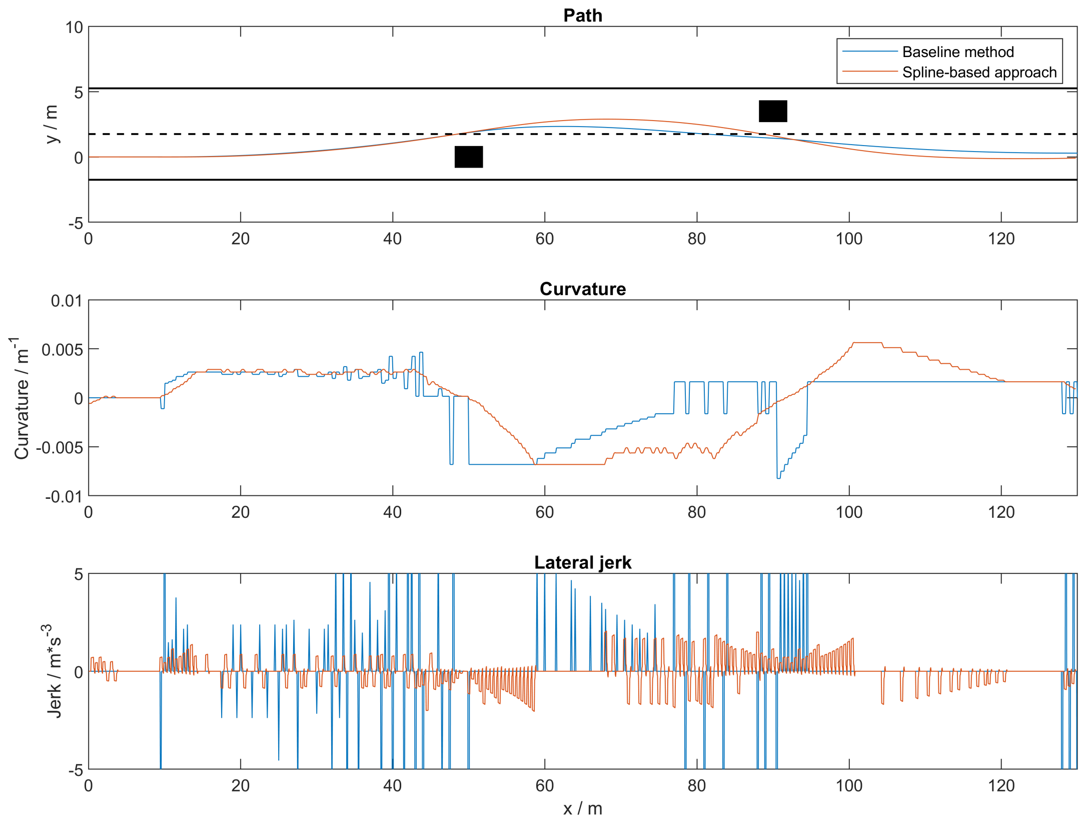


Figure 3.4: Collision avoidance of spline-based approach

the maximum curvature ρ_{max}. Hence, the minimum length l_{min} required for this tentacle to increase curvature should be:

$$l_{min} = \frac{max(-\rho_{max} - \rho_0,\ \rho_{max} - \rho_0)}{\frac{d\rho}{ds}_{max}} \tag{3.12}$$

We assume all the tentacles have the same length L, which is related to velocity:

$$L = t_0 v \tag{3.13}$$

where $t_0 = 6s$ shows that the tentacle can show the possible position of the vehicle for at most t_0 seconds. The faster the vehicle travels, the longer the tentacles, the better the advance judgment of road conditions. For every tentacle, the curvature increases from ρ_0 to the corresponding terminal curvature $\rho_{ter,\ i}$ over the length of l_{min}, which makes sure that the slopes of all tentacles are less than $\frac{d\rho}{ds}_{max}$, and thus limiting the maximum lateral jerk. The tentacles maintain their respective terminal curvature $\rho_{ter,\ i}$ in the portion of their length beyond l_{min} and less than L. However, if $l_{min} \geq L$, which means the tentacle with the greatest curvature difference could not increase to the maximum curvature within the length L. In this case, all tentacles should increase the curvature with the slope of $d\rho_{max}$.

After obtaining an expression for the curvature of each tentacle ρ_i as a function of arc length s, the heading angle could be computed by integrating the curvature over the arc length:

$$\phi_i(s) = \phi_0 + \int_0^s \rho_i(u)\,\mathrm{d}u, \quad i = 1, \dots, n \tag{3.14}$$

where ϕ_0 is the current heading angle of the vehicle, and the coordinates of each tentacle $x_i(s)$ and $y_i(s)$ could be computed by integrating the heading angle over the arc length with initial conditions x_0

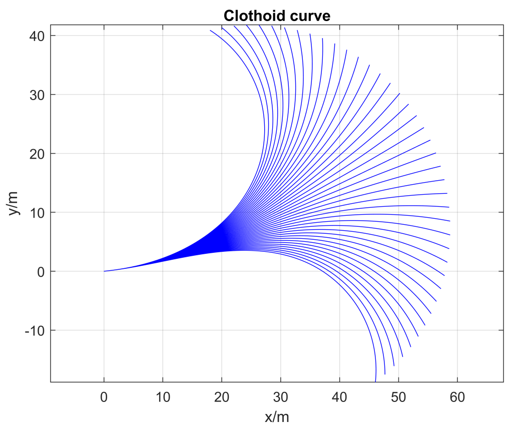
and y_0:

$$\begin{aligned} x_i(s) &= x_0 + \int_0^s \cos\big(\varphi_i(u)\big)\,\mathrm{d}u \\ y_i(s) &= y_0 + \int_0^s \sin\big(\varphi_i(u)\big)\,\mathrm{d}u \end{aligned} \tag{3.15}$$

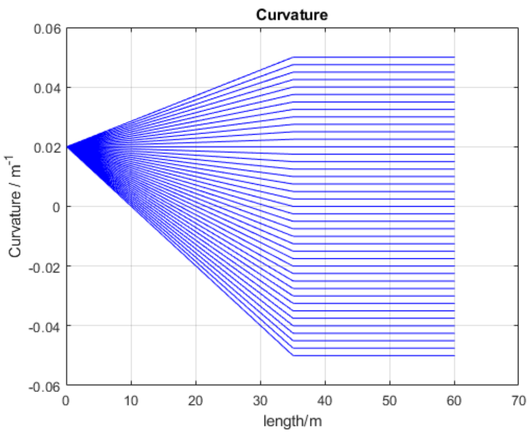
Algorithm 2: Clothoid tentacle generation

```
1  Require $\rho_0$, $v$, $j_{max}$, $\phi_0$, $x_0$, $y_0$ $L$;
2  foreach planning do
3      Calculate $\rho_{max}$ using Equation 3.9 ;
4      $d\rho_{max} \leftarrow$ CalculateMax$d\rho(v, j_{max})$;
5      $l_{min} \leftarrow$ CaculateMinLength$(d\rho_{max}, v)$ ;
6      $\rho_{ter} \leftarrow$ linspace$(-\rho_{max} - \rho_0,\ \rho_{max} - \rho_0,\ n)$ ;
7      for $i = 1 \ldots n$ do
8          if $l_{min} \leq L$ then
9              Curvature $\rho(i)$ grows from $\rho_0$ to $\rho_{ter}$ within $l_{min}$ and stay constant ;
10         else
11             Curvature $\rho(i)$ grows from $\rho_0$ at a slope of $d\rho_{max}$ within $L$ ;
12         $\phi(i) \leftarrow$ Intergrate$(\phi_0, \rho(i))$ ;
13         $T(i) \leftarrow$ Integrate$(x_0, y_0, \phi(i))$ ;
14     Tentacle selection method ;
```

Figure 3.5 shows a set of clothoid curves as well as their curvatures with initial coordinate $x_0 = 0m$, $y_0 = 0m$, initial heading angle $\phi_0 = 0.1rad$, and initial curvature $\rho_0 = 0.02/m$ at the speed of $10m/s$. The tentacles are drawn at intervals for a clearer presentation, as shown in Figure 3.5b, the curvatures of all the tentacles reach their respective terminal curvatures $\rho_{ter,\ i}$ starting from the current curvature ρ_0, and the terminal curvatures of the first and last tentacles are ρ_{max} and $-\rho_{max}$, respectively.



(a) Clothoid curve



(b) Curvature of clothoid curve

Figure 3.5: Occupancy grid map example

Figure 3.6 shows the collision avoidance results of the proposed novel approach using the clothoid curve compared with the baseline method [29], the spline-based approach proposed previously, and the clothoid curve-based approach proposed in [31]. It could be seen that compared to the spline-based approach, the path generated by the proposed clothoid approach is smoother in curvature, thus having smaller jerk averages. As shown in Table 3.1, the proposed clothoid approach has the minimum

RMS of lateral jerk among all four approaches with $0.58m/s^3$, while limiting the maximum value to $2m/s^3$. Although the clothoid approach proposed in [31] has the smallest extreme value on lateral jerk, this method of making curvature grow all the way through the length of the tentacle makes the lateral acceleration change so slowly that it is difficult to adjust back to the original lane quickly after avoiding the first obstacle, causing a lot of oscillation. Taken together, the proposed novel clothoid tentacle generation method has the best restriction on the lateral jerk, so all subsequent simulations are performed by this method.

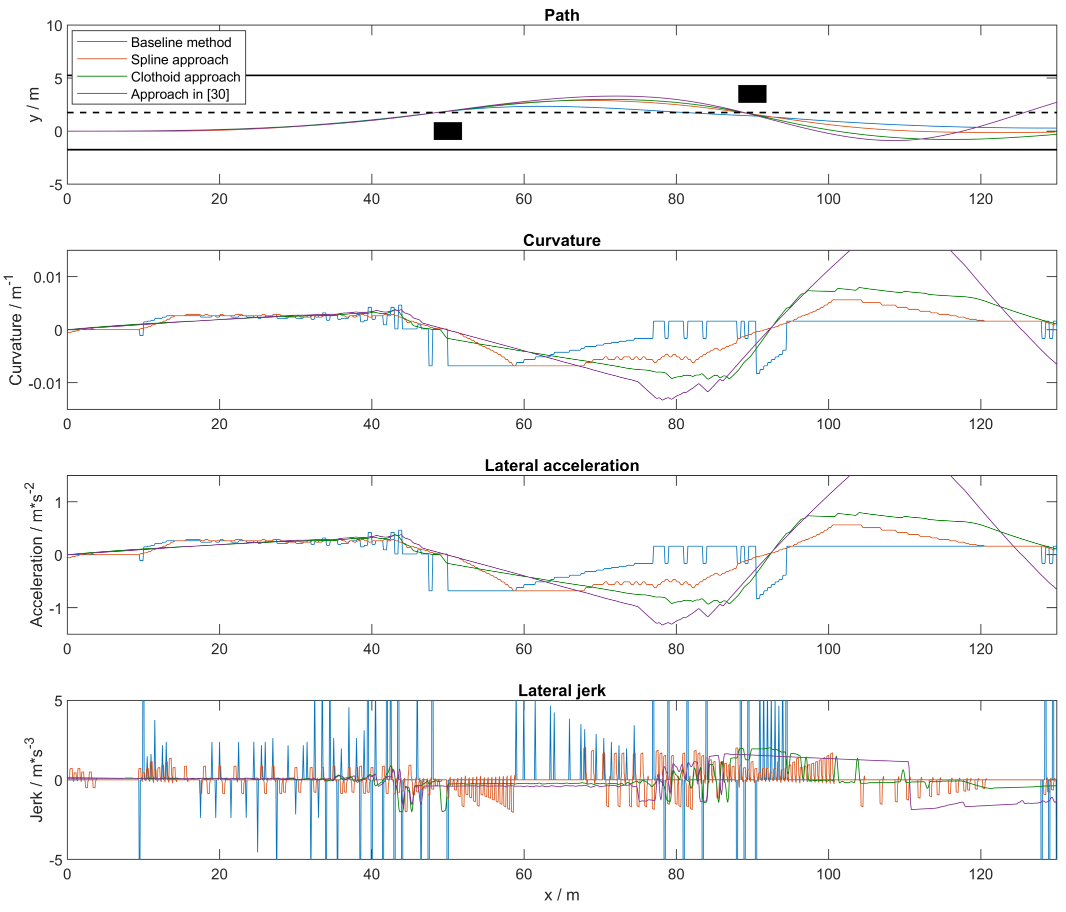


Figure 3.6: Collision avoidance of clothoid approach

Table 3.1: Comparison of algorithms on lateral jerk performance

Algorithm	Maximum lateral jerk (m/s^3)	RMS of lateral jerk (m/s^3)
Baseline algorithm [29]	98.72	5.56
Spline-based approach	2	1.08
Clothoid approach	2	0.58
Approach in [31]	1.64	0.96

3.1.4. Collision Detection

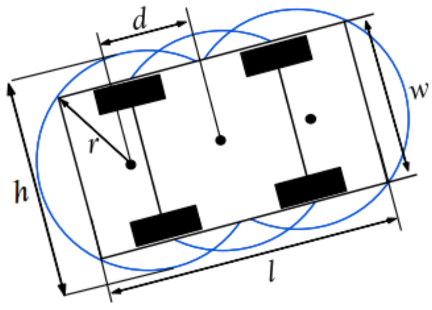


Figure 3.7: Collision detection model [29]

After obtaining the set of tentacles, not all parts of all tentacles are safe and navigable due to the presence of obstacles, so it is necessary to cut off the unsafe tentacles in order to choose the best path among the safe ones. First, all tentacles are discretized into sets of points spaced $0.1m$ apart, then a collision detection model [29] shown in Figure 3.7 is used to detect whether there is a collision between the vehicle and obstacles. The collision detection model is constructed by three equal-sized circles with centers on the vehicle's central axis. The size of parameters is designed to be [29]:

$$r = \sqrt{\left(\frac{l-2d}{2}\right)^2 + \left(\frac{w}{2}\right)^2} \tag{3.16}$$

where w and l are width and length of the vehicle, $h = 2r$ is the safe width and $d = l/3$ is the distance between centers of circle. The detailed collision detection method is introduced in Algorithm 3. In order to improve computational efficiency, the algorithm detects collision every five discrete points on a tentacle, that is, the interval between two detections is $0.5m$. For each detection, the CoG of the vehicle is assumed to be on the detection point, the locations of the centers of the three circles C_1, C_2, and C_3 are calculated according to the detection model shown in Figure 3.7. Then based on the occupancy grid map constructed in Chapter 3.1.2 to determine whether an obstacle (or road boundary) falls into the detection model: if not, it continues to check the next point; if so, only the portion of the tentacle before this point is retained, and the rest is pruned. Finally, the length of the remaining part of each tentacle is defined as the safe length L_0.

Algorithm 3: Collision detection

```
1 Require $T$, $Map$;
2 for $i = 1 \ldots n$ do
3     foreach point $j$ at interval of 5 points do
4         $[C_1, C_2, C_3] \leftarrow$ CalculateCenter$(T(i), j)$ ;
5         if IsColliding($Map$, $C_1$) + IsColliding($Map$, $C_2$) + IsColliding($Map$, $C_3$) == True then
6             Break ;
7     $T'(i) =$ Prune$(T(i))$ Calculate the safe distance for each tentacle $L_0(i)$
8 return $T', L_s$
```

Actually, cutting off the tentacles directly in front of the obstacle does not actually guarantee safety, because the obstacle may be moving and its speed may change at any time, so it is necessary to maintain a certain safe distance between the ego vehicle and the obstacle in the longitudinal direction at all times. The safe distance is defined by Time-To-Collision (TTC):

$$TTC = \frac{d}{v_r} \tag{3.17}$$

where d is the length of the tentacle between the detection point and the obstacle shown in Figure 3.8, representing the distance to the obstacle, and $v_r = v - v_{obs}$ is the relative velocity. The minimum allowed TTC is decided to be $1.5s$ in this report. Figure 3.8 shows the pruning process, where tentacles exceeding the road boundary will be pruned straight away (the part that was cut off is shown in red), while the tentacles for tentacles intersecting obstacles, only the portion with a TTC greater than $1.5s$ is retained (blue lines).

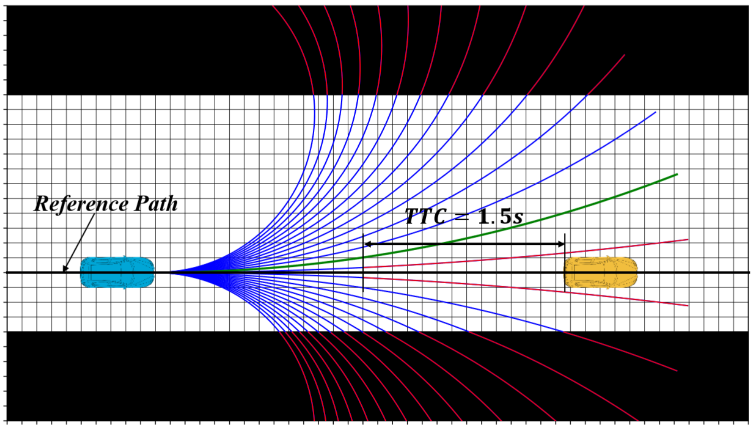


Figure 3.8: Tentacle pruning considering TTC

3.2. Tentacle Selection

This section introduces the method that selects the best tentacle to execute after obtaining the safe parts of all tentacles. Firstly, the criteria of tentacle selection are introduced, then an improvement is proposed by varying the parameters depending on the distance to the obstacle. The proposed algorithm also considers the lateral safe distance to the obstacle.

3.2.1. Selection Criteria

Only one of all the tentacles is selected as the best tentacle and the path to be tracked by the vehicle through the weighted sum of two decision-affecting values, namely $V_{clearance}$ and $V_{trajectory}$, which are defined below. The sum of the two values is represented as follows:

$$V = a_0 V_{clearance} + a_1 V_{trajectory} \tag{3.18}$$

where a_0 and a_1 are weighting parameters. In this report, we use $a_0 = 2$ and $a_1 = 0.5$.

Clearance Criterion

The clearance criterion represents the safety of the tentacle based on the maximum distance for the vehicle to drive along a tentacle before encountering an obstacle while maintaining longitudinal safe distance, which is defined as below [26]:

$$V_{\text{clearance}}(L_0) = \begin{cases} 0 & \text{free tentacle} \\ 2 - \dfrac{2}{1 + e^{-c \cdot L_0}} & \text{otherwise} \end{cases} \tag{3.19}$$

where L_0 is the safe length of each tentacle calculated in Chapter 3.1.4, and c is a constant to yield $V_{clearance}(L_{0.5}) = 0.5$ at a distance $L_{0.5} = 20m$ taken in our implementation as:

$$c = \frac{ln(1/3)}{L_{0.5}} \tag{3.20}$$

The function defined in Equation 3.19 is a sigmoid-like function, as shown in Figure 3.9, its value is naturally normalized to the range $[0, 1]$: when $L_0 = 0m$, $V_{clearance} = 1$; when $L_0 \to \infty$, $V_{clearance} \to 0$.

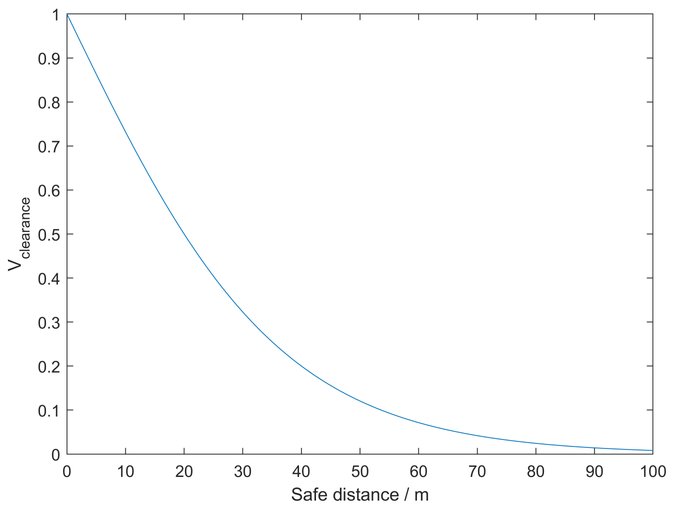


Figure 3.9: $V_{clearance}$

Trajectory Criterion

In addition to ensuring safety, it should also be ensured that the chosen tentacle allows the vehicle to follow a given reference path, e.g., generated by GPS waypoints or a global path planning algorithm, which is the role played by $V_{trajectory}$. The simplest approach to estimate $V_{trajectory}$ is to measure the lateral and angular error of a single point on the tentacle taken at a collision distance L_c and its corresponding point on the reference path shown in Figure 3.10. Collision distance L_c is defined as follows:

$$L_c = t_1 v \tag{3.21}$$

where $t_1 = 1.5s$, the collision distance increases as the vehicle speed, which means the error is always calculated at the position of the vehicle after $1.5s$, so that if the error is too large, the vehicle can be given the same reaction time to adjust the path. The error between each tentacle and the reference path is calculated as follows: as shown in Figure 3.10, the measurement V_{dist} is calculated for each tentacle by computing the distance b between the point on the tentacle and the corresponding point on the trajectory and its relative tangent direction α [31]:

$$v_{dist} = b + c_a \alpha \tag{3.22}$$

where $c_a = 0.3m/rad$ represents a scale between the linear distance and the tangent orientations. The $V_{trajectory}$ is the normalized value of V_{dist}, calculated in Equation 3.23 by V_{max} and V_{min} which are the maximum and minimum values of V_{dist} calculated for all tentacles.

$$V_{trajecotry} = \frac{V_{dist} - V_{min}}{V_{max} - V_{min}} \tag{3.23}$$

3.2.2. Varying Parameters

Results in Figure 3.4 and 3.6 are obtained using the tentacle selection criteria introduced previously. It could be seen that the vehicle does not immediately return to the initial lane after passing the first obstacle ($50m$ to $60m$), resulting in excess oscillation ($100m$ to $120m$) caused by not starting to avoid the second obstacle in time. Therefore, different weights should be given to the parameters a_0, a_1 for the obstacle avoidance phase and the completion of the obstacle avoidance phase. In the overtaking phase, $V_{clearance}$ should be weighted higher for efficiently selecting a safer tentacle to avoid obstacles, while after passing the obstacle, $V_{trajectory}$ should be weighted higher for fast tracing of the reference path.

No extra sensors are needed to check if the vehicle has passed the obstacle: according to Chapter 3.1.4, when performing collision detection, if a tentacle hits an obstacle, it is pruned so that the TTC of

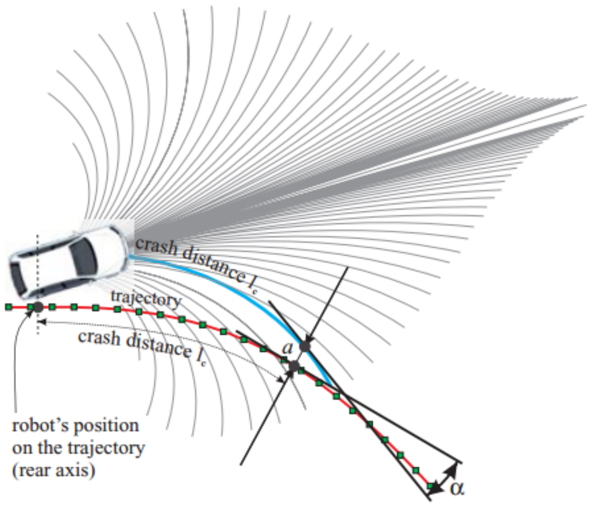
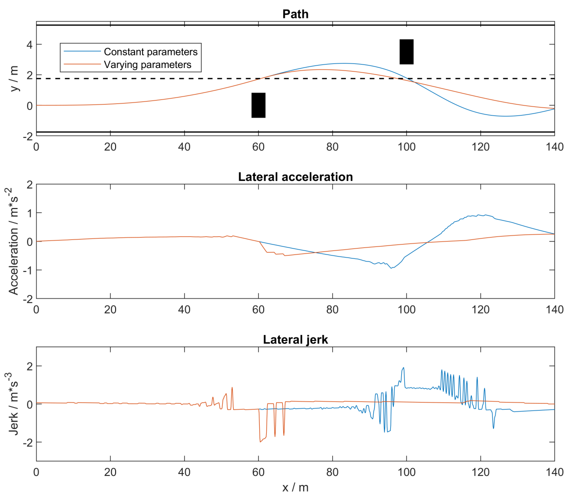
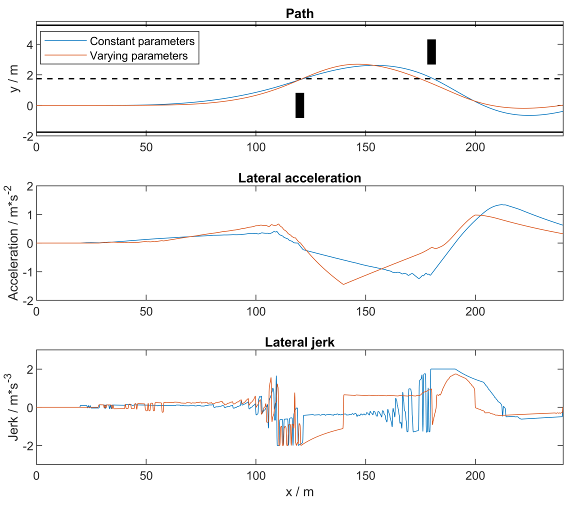


Figure 3.10: $V_{trajectory}$ [26]

the ego vehicle is always greater than $1.5s$. If all tentacles do not meet an obstacle, it can be considered that the overtaking is complete or that they have not yet encountered an obstacle; in this case, the weighting parameters of the clearance criterion and trajectory criterion are selected as $a_0 = 2.5$ and $a_1 = 0.5$. If any of the tentacles hit an obstacle, it could be assumed that the ego vehicle needs to avoid this obstacle; in this case, the weighting parameters are selected as $a_0 = 0.5$ and $a_1 = 0.5$. Figure 3.13 shows the comparison of constant and varying weighting parameters at different speeds. Two obstacles are located in two different lanes. At a ego vehicle speed of $10m/s$ shown in Figure 3.11a, the vehicle using parameter-varying method starts to steer to the reference path than the parameter-constant method after passing the first obstacle at $60m$, resulting smoother path when avoiding the second obstacle, with a RMS value of lateral jerk of $0.27m/s^3$, which is 41% less than the parameter-constant method, which has a RMS value of $0.46m/s^3$. At a ego vehicle speed of $20m/s$ shown in Figure 3.11b although the difference between the two methods is not as great as at $10m/s$, the parameter-varying method still manages to return to the reference path faster, achieving a RMS value of lateral jerk of $0.7m/s^3$, which is 9% less than the parameter-constant method, which has a RMS value of $0.78m/s^3$.



(a) $10m/s$



(b) $20m/s$

Figure 3.11: Comparison of constant and varying weighting parameters at different speeds

3.2.3. Consider Lateral Safe Distance

Although the collision detection and pruning process introduced in Chapter 3.1.4 ensures that the executed path is at a sufficient longitudinal safety distance from obstacles, we still need to make sure that the vehicle will have a sufficient lateral safety distance when overtaking the obstacle to cope with unexpected situations. As shown in Figure 3.12, compared to Figure 3.8, the best tentacle selected (green line) has changed from 1 to 2 because tentacle 1 does not have enough lateral safety distance. An unsafe region around the obstacle, considering lateral distance, is defined in Figure 3.12, shown in red. The width of the unsafe region w_0 is defined as follows:

$$w_0 = w + w_o + 2l_s \tag{3.24}$$

where w is the width of the ego vehicle, w_o is the width of the obstacle given by the bounding box of the obstacle derived by the obstacle recognition algorithm of the sensing module of ADS. l_s is the minimum safe lateral distance between the ego vehicle and the obstacle when passing through the obstacle; in this report, $l_s = 1m$ is selected. Tentacles that fall into the unsafe region will be given more cost and hard to select as the best tentacle. As shown in Figure 3.12, the best tentacle is shown by the green line because it is the closest to the reference path while maintaining obstacle-free and outside the unsafe region. Figure 3.12 illustrates the difference between taking into account or not the lateral safety distance when choosing the optimal path under different velocities. Under both velocities, the algorithm enables the vehicle to maintain a safe lateral distance of at least $1m$ from the obstacle (computing the edge of the vehicle and obstacle), while the average jerk is increased because the vehicle needs more steering for greater lateral distance.

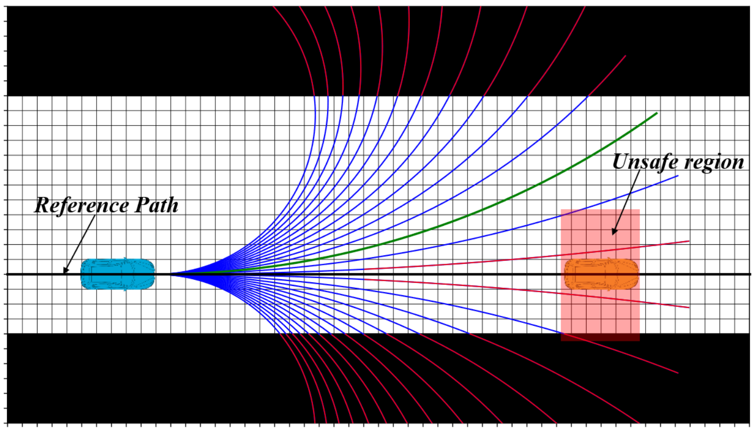


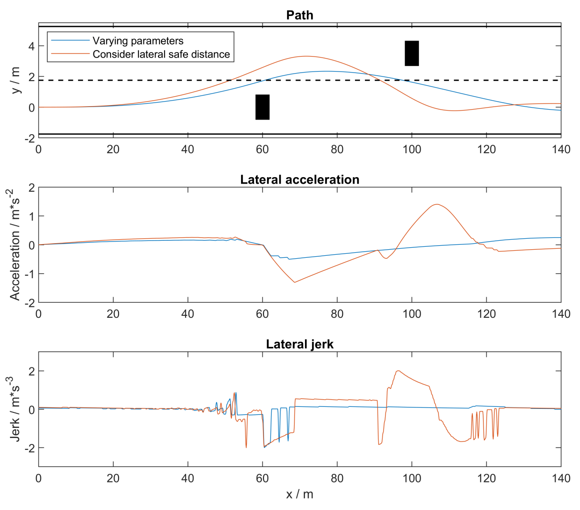
Figure 3.12: Tentacle pruning considering TTC

3.3. Speed Planning

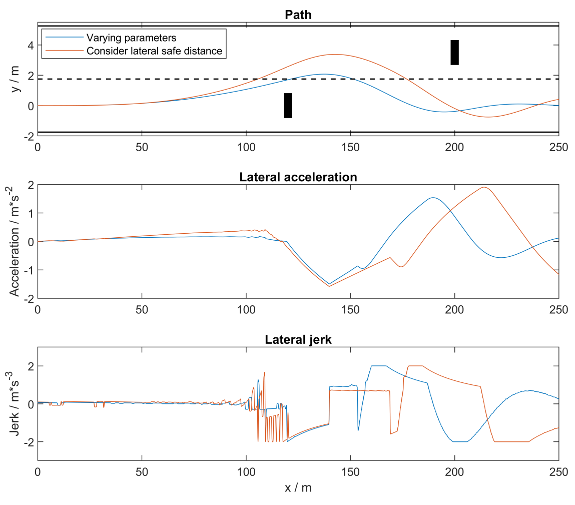
For an automated vehicle traveling at high speeds, it is necessary to slow down and avoid obstacles in front of it to ensure safety. At the same time, for a low-speed vehicle, it is necessary to increase the speed to save time within the regulatory limits. In this section, a speed profile is generated, including time information. The speed profile must comply with traffic rules, take into account static and dynamic obstacles, and be adapted to current road conditions. Firstly, a jerk-limited speed profile generation method is introduced, then an adjustment term that specifically accounts for overtaking scenarios and road curvature is proposed.

3.3.1. Limitations

Limitations on velocity, longitudinal acceleration, and jerk are important in speed planning in order to ensure the ego vehicle complies with road regulations and improves driving comfort. There are two limitations on vehicle velocity [32], expressed in Equation 3.25. The first limitation $V_{lim,1}$ concerns traffic rules (traffic lights and road signs), and is presumed to be known by the sensing module of the vehicle. The second limitation $V_{lim,2}$ is enforced to ensure vehicle stability and comfort by limiting the lateral acceleration below the maximum allowed value of $a_{lat}^{max} = 4m/s^2$. This limit is calculated at every point



(a) $10m/s$



(b) $20m/s$

Figure 3.13: Comparison of lateral safe distance at different speeds

on the tentacle with the curvature ρ of this point known. v_{lim} is the lesser of the two limitations, which is also the maximum speed that the vehicle must not exceed during the planning process.

The maximum longitudinal acceleration a_{long} should also be limited to ensure ride comfort. A maximum acceleration threshold $a_{acc}^{max} = 2m/s^2$ and a maximum deceleration $a_{dec}^{max} = -4m/s^2$ threshold is defined. The longitudinal jerk j_{long} is limited to a maximum value of $j_{long}^{max} = 3m/s^3$.

$$\begin{aligned} V_{lim,1} &= V_{road} \\ V_{lim,2} &= \sqrt{a_{lat}^{max}/\rho} \\ v_{lim} &= \text{min}(V_{lim,1}, V_{lim,2}) \end{aligned} \tag{3.25}$$

3.3.2. Jerk-limited Speed Planning

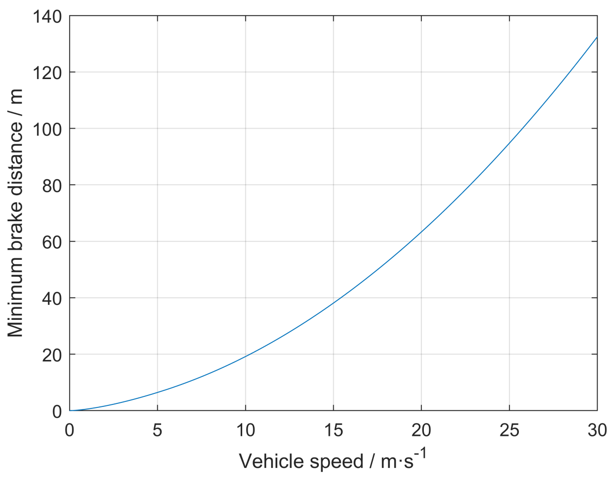


Figure 3.14: Minimum braking distance-velocity

Based on the selected best tentacle to execute, a reference speed control model shown in the following equation is established:

$$v_{tar} = \begin{cases} v_{interpolate}, & L_t < d_s + d_0 \\ v_{ref}, & L_t \geq d_s + d_0 \end{cases} \tag{3.26}$$

where v_{ref} is the reference speed, which is the speed that the vehicle will eventually need to reach. v_{lim} is the maximum velocity the vehicle could reach, which is derived in Chapter 3.3.1, L_t is the safe length of the best tentacle derived in Chapter 3.2. d_s is the minimum braking distance to satisfy maximum deceleration a_{dec}^{max} and jerk j_{long}^{max} limitations at current speed and acceleration; specific calculations are presented in Appendix A. $v_{interpolate}$ is the velocity when the shortest braking distance is L_t. $v_{interpolate}$ is obtained by interpolating the braking distance-velocity diagram shown in Figure 3.14. Each point on the curve represents the shortest braking distance at different speeds, computed offline using the method in Appendix A.

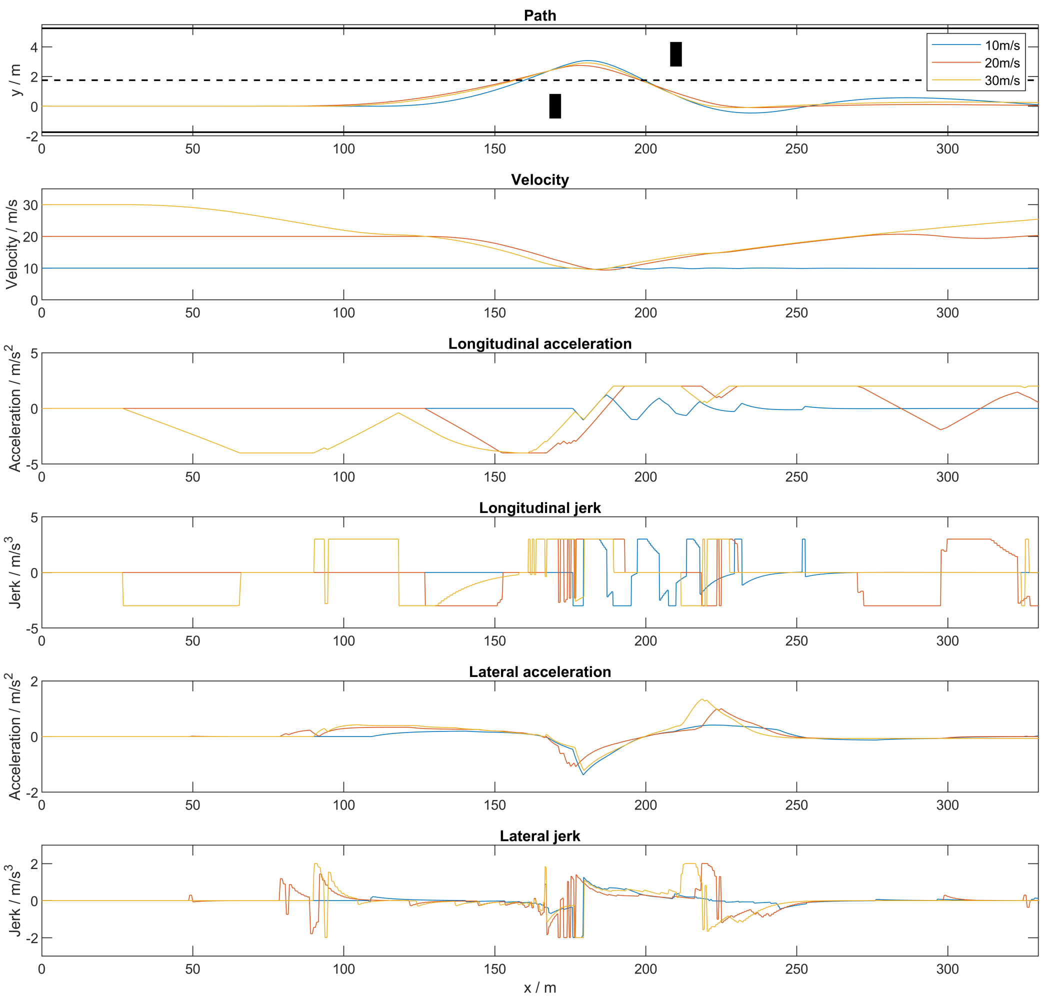


Figure 3.15: Speed planning

After obtaining the reference velocity v_{ref}, the reference acceleration a_{ref} is computed as follows. a_{ref} is set to the maximum acceleration if the current speed is significantly lower than the reference speed, while it is set to the maximum deceleration if the current speed is significantly higher than the reference speed. Δ is a small value to make sure the reference acceleration is set to zero when the current velocity is close to the reference velocity to avoid oscillations at the steady state.

$$a_{ref} = \begin{cases} a_{acc}^{max}, & v < v_{ref} - \Delta \\ a_{dec}^{max}, & v > v_{ref} + \Delta \\ 0, & \text{else} \end{cases} \tag{3.27}$$

The longitudinal jerk is derived as the output of a PD controller that takes the error between reference acceleration and current acceleration as input. The PD controller is defined as follows:

$$
\begin{aligned}
e(t) &= a_{ref}(t) - a(t) \\
j(t) &= K_p\, e(t) + K_d \frac{\mathrm{d}e(t)}{\mathrm{d}t},
\end{aligned}
\tag{3.28}
$$

where K_p is the proportional gain, providing an immediate correction proportional to the current error. K_d is the derivative gain, reducing steady-state error by accumulating past errors. The output of the PD controller j is saturated between $-j_{long}^{max}$ and j_{long}^{max} to meet the limitation on longitudinal jerk. The longitudinal jerk is assumed to be constant between two plannings, and the acceleration and velocity profiles can be derived by integrating over the longitudinal jerk.

Figure 3.15 illustrates the results of the proposed speed planning algorithm. The vehicle has to avoid two obstacles $40m$ apart under different initial velocities. When the reference speed is $10m/s$, the vehicle hardly needs to slow down to avoid the obstacle. When the reference speed is $20m/s$, the vehicle starts to brake when the length of the best tentacle is shorter than the minimum braking distance under the current speed. As soon as the speed drops to around $10m/s$, the vehicle starts to steer and starts to accelerate after passing the first obstacle, finally reaching the reference speed. When the reference speed is $30m/s$, the vehicle first brakes for a while and then releases the brake. This is because the best tentacle changes from the one that follows the reference path to the one that avoids the first obstacle. The length of the best tentacle is greater than the minimum braking distance, so the vehicle stops to brake till the length of the best tentacle is so short that it has to slow down again because of the second obstacle.

3.3.3. Modification of v_{ref}

In this section, v_{ref} is modified to adapt to dynamic traffic conditions and curvy roads. This parameter now consists of several sub-functions, each of which deals with a different aspect of road conditions and the dynamic environments.

Relative Velocity

When overtaking dynamic obstacles, it is critical to accelerate to ensure smooth road traffic flow and increase time efficiency. If overtaking occurs in urban environments, acceleration also reduces the exposure of the vehicle to oncoming traffic, thus ensuring safety. If the act of overtaking has been decided, the acceleration should be maximum when the relative velocity v_r is zero, decreasing as the relative velocity increases [32]. Therefore, the first sub-function f_1 is defined as the function of the relative velocity v_r, represented in a cubic polynomial:

$$
f_1(v_r) = a_1 v_r^3 + b_1 v_r^2 + c_1 v_r + d_1, \quad v_r \in [0, v] \tag{3.29}
$$

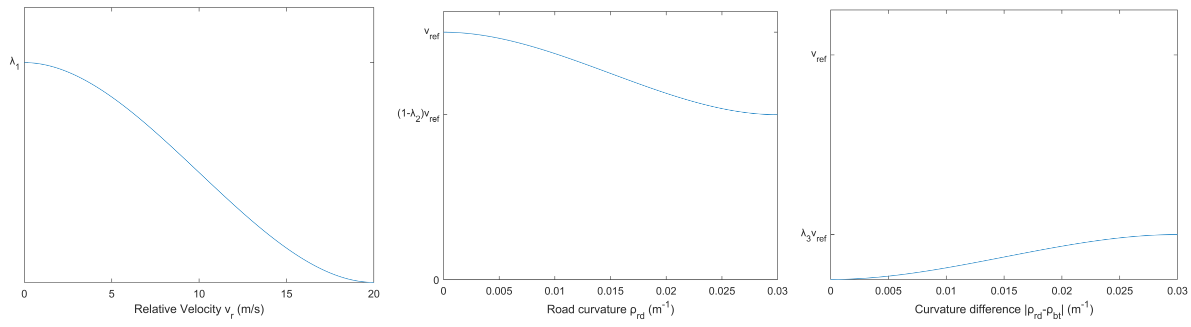
where a_1, b_1, c_1, d_1 are polynomial coefficients, v is the current speed of ego vehicle. To calculate these coefficients, several boundary conditions are constructed:

$$
\begin{aligned}
f_1(\frac{1}{2}v - k_1) = \lambda_1, \quad f_1(\frac{1}{2}v + k_1) = 0 \quad \text{where} \quad k_1 \in [\frac{1}{2}v, \infty] \\
f_1'(\frac{1}{2}v - k_1) = f_1'(\frac{1}{2}v + k_1) = 0
\end{aligned}
\tag{3.30}
$$

where $\lambda_1 = 0.2v$ is a gain to specify the upper bound of the function. Bringing the boundary conditions back to Equation 3.29 yields that k_1 is a parameter to be solved that uniquely determines the shape of f_1. Figure 3.16a represents the shape of f_1 with $k_1 = \frac{1}{2}v$ under $v = 20m/s$, the value of f_1 reaches the maximum when relative velocity is zero and minimum (0) when relative velocity is v, which means the obstacle is static. The purpose of developing f_1 is to provide additional acceleration to the vehicle when overtaking dynamic obstacles [32].

Road Curvature

The reference speed should decrease with increasing road curvature and the condition that the best tentacle is significantly different from the curvature of the road, even in straight roads (for instance,



(a) f_1 **(b)** f_2 **(c)** f_3

Figure 3.16: Comparison of lateral safe distance at different speeds

during lane changes). Hence, two sub-functions f_2 and f_3 are defined as the function of the road curvature ρ_{rd} and curvature difference between the best tentacle ρ_{bt} and the road ρ_{rd}:

$$f_2(\rho_{rd}) = a_2\rho_{rd}^3 + b_2\rho_{rd}^2 + c_2\rho_{rd} + d_2, \quad \rho_{rd} \in [0, \rho_{rd}^{max}] \tag{3.31}$$

$$f_3(|\rho_{rd}-\rho_{bt}|) = a_3|\rho_{rd}-\rho_{bt}|^3 + b_3|\rho_{rd}-\rho_{bt}|^2 + c_3|\rho_{rd}-\rho_{bt}| + d_3, \quad |\rho_{rd}-\rho_{bt}| \in [0, |\rho_{rd}-\rho_{bt}|^{max}] \tag{3.32}$$

where $a_2, \ldots, d_2$ and $a_3, \ldots, d_3$ are polynomial coefficients, ρ_{rd}^{max} is the maximum possible curvature of the road, which is taken as $0.1/m$ for urban conditions and $0.03/m$ for highway conditions. Several boundary conditions are constructed to calculate the coefficients of f_2 and f_3:

$$\begin{aligned} f_2(\frac{1}{2}\rho_{rd} - k_2) = v_{ref}, \quad f_2(\frac{1}{2}\rho_{rd} + k_2) = (1-\lambda_2)v_{ref} \quad \text{where} \quad k_2 \in [\frac{1}{2}\rho_{rd}^{max}, \infty] \\ f_2'(\frac{1}{2}\rho_{rd} - k_2) = f_2'(\frac{1}{2}\rho_{rd} + k_2) = 0 \end{aligned} \tag{3.33}$$

$$\begin{aligned} f_3(\frac{1}{2}|\rho_{rd}-\rho_{bt}| - k_2) = 0, \quad f_3(\frac{1}{2}|\rho_{rd}-\rho_{bt}| + k_2) = \lambda_3 v_{ref} \quad \text{where} \quad k_3 \in [\frac{1}{2}|\rho_{rd}-\rho_{bt}|^{max}, \infty] \\ f_3'(\frac{1}{2}|\rho_{rd}-\rho_{bt}| - k_2) = f_3'(\frac{1}{2}|\rho_{rd}-\rho_{bt}| + k_3) = 0 \end{aligned} \tag{3.34}$$

where $\lambda_2 = 0.4$ and $\lambda_3 = 0.2$ provide a lower bound for f_2 and an upper bound for f_3 respectively, v_{ref} is the reference velocity obtained in Equation 3.26. Similar to f_1, k_2 and k_3 are parameters to be solved that uniquely determine the shape of f_2 and f_3. The diagrams of f_2 and f_3 are depicted in Figure 3.16. Thus, starting from the zero-curvature base speed kb specified in the base frame and the optimal path, f_2 and f_3 act to gradually reduce the speed of the vehicle as the curvature increases [32].

The target velocity v_{tar} is defined in terms of f_1, f_2, and f_3 as the modification of v_{ref}:

$$v_{tar} = k_f f_1 + f_2 - f_3 \tag{3.35}$$

k_f is a boolean value that determines whether or not to activate f_1, while f_2 and f_3 are always activated. If the behavioral decision to overtake the front vehicle is made and the relative speed v_r is less than a certain value, then $k_f = 1$; otherwise, $k_f = 0$. The limitation on v_r is imposed because if there is a large relative velocity, the obstacle can be considered as a static one, and too small a value of f_1 can cause unwanted jerks.

As the f_1 varies with the relative velocity, and f_2 and f_3 vary with continuous curvature. Its first derivative (for acceleration) and second derivative (for jerk) must be limited to the maximum allowable acceleration and jerk. Take f_1 as an example, the constraints on acceleration and jerk are given as follows:

$$|\frac{\partial f_1}{\partial t}| = |(3a_1v_r^2 + 2b_1v_r + c_1)\frac{\partial v_r}{\partial t}| \le |a_{dec}^{max}| \tag{3.36}$$

$$|\frac{\partial^2 f_1}{\partial t^2}| = |(6a_1 v_r + 2b_1)\left(\frac{\partial v_r}{\partial t}\right)^2 + (3a_1 v_r^2 + 2b_1 v_r + c_1)\frac{\partial^2 v_r}{\partial t^2}| \leq |j_{long}^{max}| \tag{3.37}$$

According to the boundary conditions of f_1 shown in Equation 3.30, a_1, b_1, c_1, d_1 could all be expressed in terms of k_1, so the range of values of k_1 could be derived after solving Equation 3.36 and 3.37. See Appendix B for details of the solution process. Suppose k_1^{min} is the smallest value that k_1 can take. If $k_1^{min} < \frac{1}{2}v$, then take $k_1 = \frac{1}{2}v$ to define f_1, and conversely take $k_1 = k_1^{min}$ to define f_1. Similar approaches could be used to solve the value ranges of k_2 and k_3 that can fully define f_2 and f_3. The target velocity v_{tar} could therefore be derived using Equation 3.35.

4 Simulation and Results

In this chapter, the proposed tentacle-based motion planning algorithm is validated through comprehensive simulations using IPG CarMaker combined with MATLAB/Simulink. Previous analyses and preliminary algorithm designs treated the vehicle as a simplified point mass, neglecting the intricate dynamics associated with the actual behavior of vehicles and tires. This simplification, while conducive to theoretical analysis, may ignore critical behaviors that only occur under real driving conditions. Thus, performing simulations on a high-fidelity platform is essential to evaluate the algorithm's performance accurately. By evaluating scenarios including straight and curved lane highway driving, as well as complex urban environments exemplified by the TU Delft campus scenario, this simulation phase is crucial in demonstrating the algorithm's effectiveness in maintaining passenger comfort, ensuring safety, and confirming its real-time performance. Results obtained from these simulations validate the robustness and practicality of the proposed approach, highlighting its capacity to manage dynamic and static obstacles while adhering strictly to lateral and longitudinal jerk constraints.

4.1. Framework

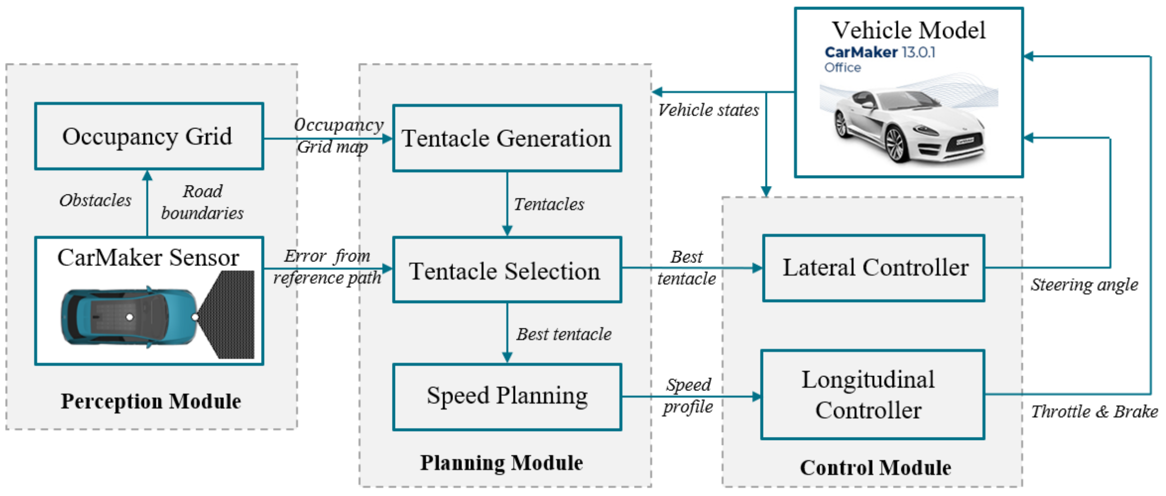


Figure 4.1: Framework of simulation of the proposed algorithm

The framework of the simulation process of the proposed tentacle-based motion planning algorithm is shown in Figure 4.1. The framework consists of three modules: perception, planning, and control, and forms a closed-loop structure with a high-fidelity vehicle dynamics simulation environment provided by CarMaker. First, the perception module acquires road environment information, including obstacle locations and road boundaries, through the object sensor and road sensor defined in CarMaker, and then generates an occupancy grid map. The planning module first generates a set of tentacles and gets them pruned according to the occupancy grid, then selects the optimal tentacle after considering indicators of safety and path deviation, and further performs speed planning to meet comfort requirements based on the best tentacle. The control module employs lateral and longitudinal controllers to track the best tentacle and speed profile, and outputs corresponding steering angle and throttle/brake commands to the CarMaker vehicle model. The simulation model provides real-time feedback of vehicle state data to the planning and control module, enabling closed-loop control. This effectively validates the proposed algorithm's real-time performance, robustness, and ability to balance safety and comfort in complex dynamic environments.

4.2. Vehicle Control

In this section, simple lateral and longitudinal controllers for the vehicle are developed to enable the vehicle to accurately follow the path and speed profile planned by the planning module. The parameters of the vehicle model we use in CarMaker are given in Table 4.1. Both controllers work at a frequency of 100Hz.

Table 4.1: Vehicle Parameters

Symbol	Description	Unit	Value
m	Mass	kg	2065
I_z	Moment of inertia in vertical direction	kg·m^2	2900
i_s	Steering ratio	-	16.46
L	Wheelbase	m	1.97
l_f	Distance from front axle to CoG	m	1.17
l_r	Distance from rear axle to CoG	m	1.80
l	Vehicle length	m	4.64
w	Vehicle width	m	1.89
$C_{\alpha f}$	Front axle cornering stiffness	N/rad	115000
$C_{\alpha r}$	Rear axle cornering stiffness	N/rad	97000

4.2.1. Lateral Control

For the lateral controller, the goal is to find a suitable sequence of front wheel angle deltas that will cause the vehicle to travel along the planned path. Since the focus of this chapter is on the verification of the planning algorithm, a simple open-loop controller based on the bicycle model is implemented. The dynamic bicycle model of a vehicle is shown as follows [33]:

$$\begin{aligned} \dot{v_{lat}} &= -\left(\frac{C_{\alpha f}+C_{\alpha r}}{m v_{long}}\right) v_{lat} + \left(-v_{long} + \frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{m v_{long}}\right) r + \frac{C_{\alpha f}}{m}\delta \\ \dot{r} &= \left(\frac{l_r C_{\alpha r} - l_f C_{\alpha f}}{I_z v_{long}}\right) v_{lat} + \left(-\frac{l_f^2 C_{\alpha f} + l_r^2 C_{\alpha r}}{I_z v_{long}}\right) r + \left(\frac{l_f C_{\alpha f}}{I_z}\right)\delta \end{aligned} \tag{4.1}$$

where v_{lat} and v_{long} are lateral and longitudinal velocity defined in Figure 3.1, r is yaw rate. $C_{\alpha f}$ and $C_{\alpha r}$ are front and rear axle cornering stiffness, which a characteristics of the tire. I_z is the moment of inertia of the vehicle in the vertical direction. δ is the steering angle of the front wheel. Based on the bicycle model, the steady-state curvature response gain G_{cur}^{ss} could be derived as follows [33]:

$$G_{\text{cur}}^{ss} = \frac{\rho}{\delta}\bigg|_{ss} = \frac{1}{L + \dfrac{K_{us} v_{long}^2}{g}} \tag{4.2}$$

where $L = l_f + l_r$ is the wheelbase of the vehicle. g is the gravitational constant. K_{us} is the understeer gradient that indicates the understeer or oversteer characteristics of a vehicle [33], which is defined in

Equation 4.3.

$$K_{us} = \frac{mg}{L}\left(\frac{l_r}{C_{\alpha f}} - \frac{l_f}{C_{\alpha r}}\right) \tag{4.3}$$

Equation 4.2 gives a relationship between path curvature ρ and steering angle δ, which could be used as the control law of an open-loop controller. Given the curvature sequence of the best tentacle obtained in Chapter 3.2, one can derive the front wheel steering angle sequence. Then, the steering wheel angle sequence δ_s is computed by the following relationship and input into the CarMaker vehicle model for further response.

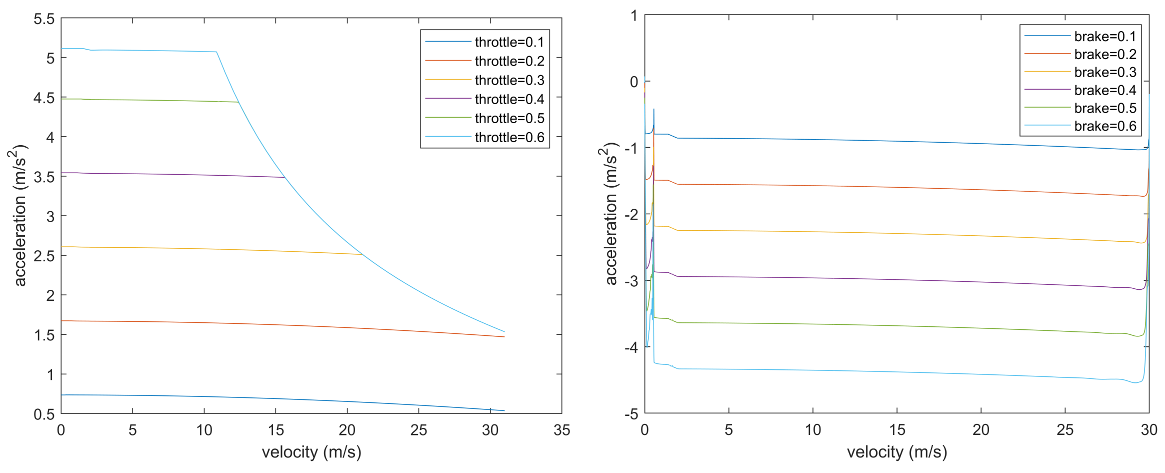
$$\delta_s = i_s \delta \tag{4.4}$$

where i_s is the steering ratio, which refers to the ratio between the steering wheel angle and the front wheel angle.

4.2.2. Longitudinal Control

Different from the simulations in Chapter 3, the longitudinal controller in this chapter could not directly control acceleration, but instead indirectly controls acceleration by controlling the throttle and brake openings. Figure 4.2 illustrates the mapping relationship between throttle and brake openings and vehicle acceleration. As shown in Figure 4.2a, before reaching a certain speed, acceleration barely changes with speed. After that speed, the speed and acceleration change inversely proportional to each other since the selected vehicle model is an electric vehicle. However, Figure 4.2b represents that the deceleration remains virtually unchanged across all speed ranges.

Similar to Chapter 3.3.2, a PI controller is implemented for longitudinal control. The controller takes the error between the current velocity and the target velocity v_{tar} obtained in Chapter 3.3.3, and outputs the target acceleration a_{tar}. Then the desired throttle or brake pedal position is obtained by interpolating the target acceleration in the look-up table constructed through the data in Figure 4.2. In addition, since different throttle or brake openings at different speeds will result in different accelerations, another look-up table is constructed to obtain the increment of throttle or brake to meet the limitation of maximum longitudinal jerk under different velocities.



(a) Throttle **(b)** Brake

Figure 4.2: mapping relationship between throttle and brake openings and acceleration

4.3. Common Road Scenario

In this section, the simulation focuses on the vehicle's driving and obstacle avoidance performance on ordinary roads outside built-up areas (Buiten de bebouwde kom) in the Netherlands. Ordinary roads outside built-up areas typically connect towns, rural areas, and suburban regions, characterized by two-way traffic without a physical central median. Such roads typically have one or two lanes in each direction, with a speed limit generally set at $80\ km/h$, and a standard road width of 3.5 meters per lane.

4.3.1. Scenario

Figure 4.3 shows the scenario of the simulation. The initial speed of the ego vehicle (white rectangle) is set to $50\ km/h$, and the target speed is $80\ km/h$. In the simulation scenario, two types of slower vehicles (black rectangles) are placed in front of the vehicle in different lanes: one traveling at $36\ km/h$ and the other at $54\ km/h$. This scenario is designed to simulate real-life overtaking situations where there are significant differences in speed.

During the simulation process, the ego vehicle should first decelerate, gradually adjusting its speed until it reaches the same speed as the slow-moving vehicle ahead ($36\ km/h$), while maintaining a safe following distance. Thereafter, the vehicle will execute an overtaking maneuver when conditions permit, i.e., by changing lanes to the adjacent lane to overtake the slow-moving vehicle ahead, and ultimately returning to its target cruising speed ($80\ km/h$).

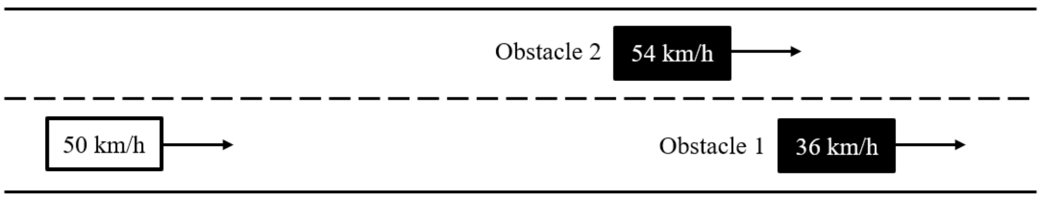


Figure 4.3: Scenario 1

4.3.2. Results

Figure 4.4 shows the simulation results of this scenario. The ego vehicle first accelerates to a cruising speed of $80\ km/h$, then brakes to the speed of the obstacle 1 ($36\ km/h$) to follow it. At this point, the vehicle cannot overtake obstacle 1 because obstacle 2 is still in the left lane. After obstacle 2 passes obstacle 1, the ego vehicle can speed up and overtake obstacle 1 while keeping a safe distance from obstacle 2.

The level of longitudinal jerk remains basically below the maximum value of $3\ m/s^3$. However, some oscillations appeared when the vehicle approaches the cruising speed (from $10s$ to $12s$ and after $38s$), causing a jerk up to $4\ m/s^3$. This happens because of the resistance encountered by the vehicle: when approaching the target speed, the target acceleration should be zero, but when the throttle is set to zero, the speed will decrease due to resistance, at which point the target acceleration will again be greater than zero, causing a cycle of oscillation. The level of lateral jerk also remains below the maximum value of $2\ m/s^3$ for most of the time, except for some moments before $27s$ when the vehicle has just started to overtake obstacle 1, the best tentacle is switching back and forth between following and overtaking before finally passing obstacle 1. This happens because of multiple reasons, such as the error of the lateral controller, and the error of estimation of the current curvature.

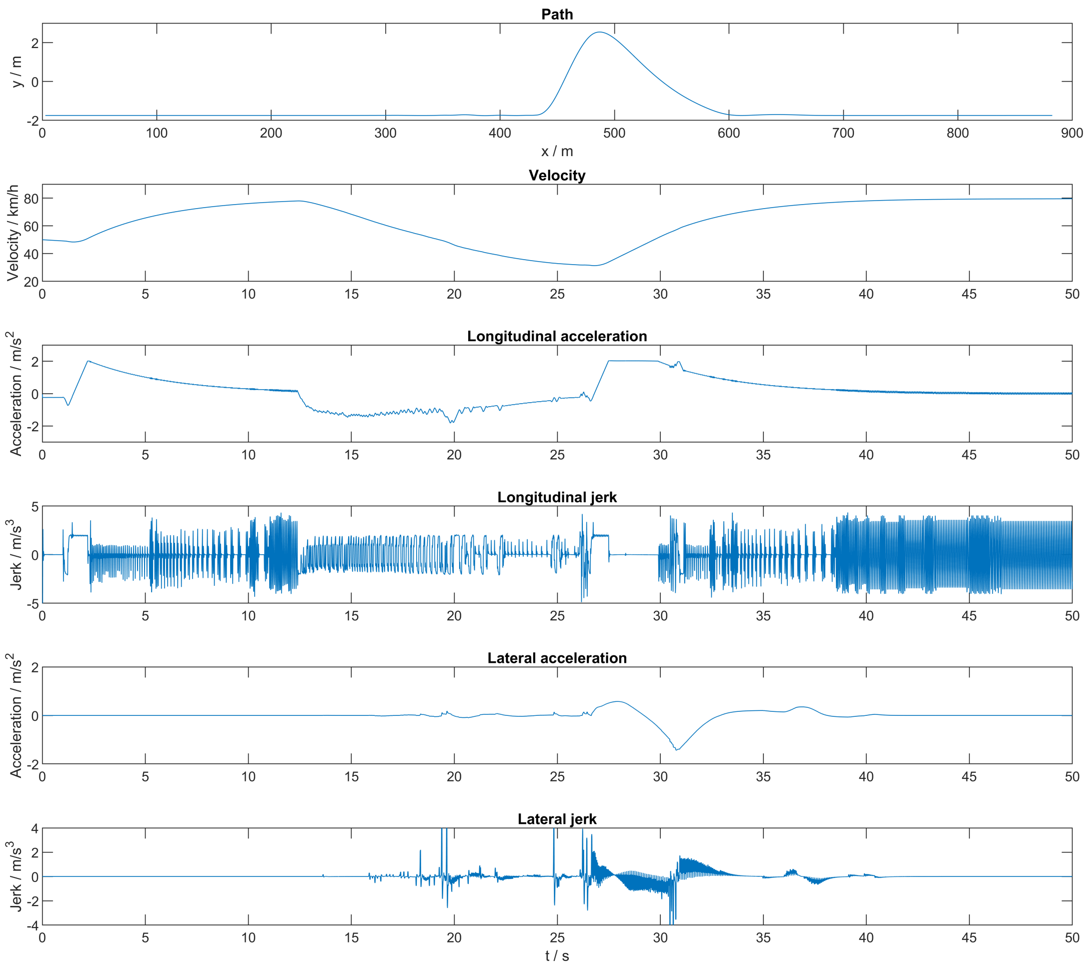


Figure 4.4: Simulation results of scenario 1

4.4. Highway Scenario

The simulation verification in this section primarily focuses on the vehicle's driving and obstacle avoidance performance on Dutch highways (autosnelweg). The highways are designed for high-speed vehicle traffic, typically featuring physical central dividers and multiple lanes. The maximum speed limit on highways is $130\ km/h$. Furthermore, due to the theoretical minimum speed limit of $60\ km/h$ on highways, autonomous vehicles will only encounter dynamic obstacles traveling at speeds similar to their own, and will not encounter obstacles traveling at much slower speeds or static obstacles.

4.4.1. Scenario

Figure 4.5 shows the scenario of this simulation. The vehicle is traveling on a two-lane, one-way highway, with each lane measuring 3.5 meters in width. The highway used in this simulation combines straight and curved sections, with smooth curvature transitions at the junctions between straight and curved sections. The curvature profile of the highway is represented in the first sub-figure in Figure 4.6. The initial speed of the vehicle is set at $90\ km/h$. In the simulation scenario, a moving obstacle traveling slightly slower than the vehicle, at a speed of $80\ km/h$, is positioned ahead of the vehicle in the same lane. The vehicle needs to accelerate to overtake the vehicle in front as quickly as possible. This scenario is designed to simulate typical overtaking requirements at high speeds.

4.4.2. Results

Figure 4.7 illustrates the results of the simulation of the highway scenario. The vehicle is cruising at $90\ km/h$ for the first 25 seconds, then the tentacle encounters the moving obstacle. After calculating the

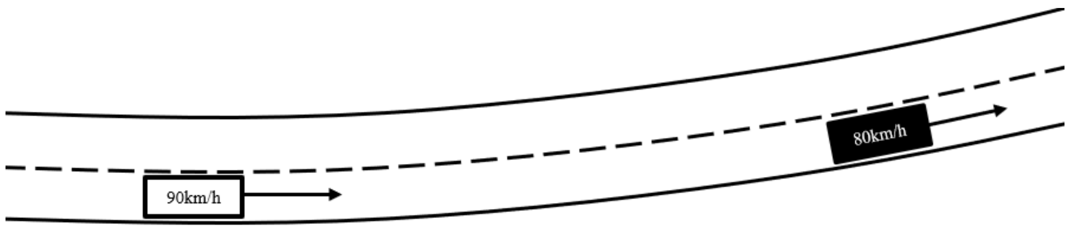


Figure 4.5: Scenario 2

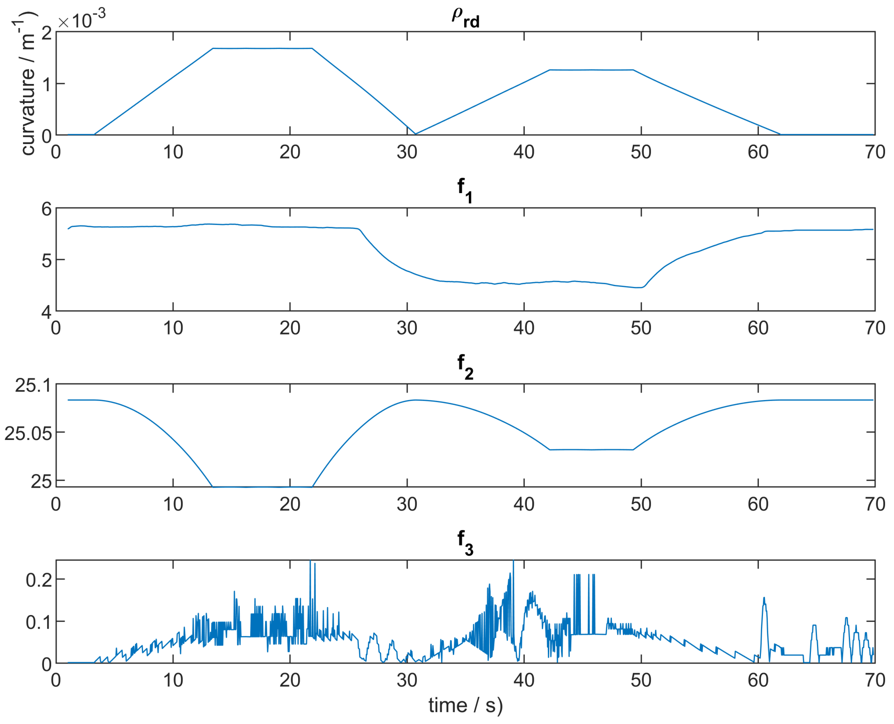


Figure 4.6: Profiling the sub-functions f_1, f_2, and f_3 in scenario 2

relative velocity between the ego vehicle and the obstacle, the vehicle decides to overtake. Meanwhile, the value of sub-function f_1 introduced in Chapter 3.3.3 is activated and computed to obtain an extra acceleration for overtaking. As shown in Figure 4.6, the value of f_1 decreases as the relative velocity decreases when accelerating. The ego vehicle finally accelerates to around $105\ km/h$ to overtake the obstacle. After completing the overtaking maneuver, f_1 is deactivated to return to the original cruising speed of $90\ km/h$.

Due to the road curvature, sub-functions f_2 and f_3 are also calculated, shown in Figure 4.6. The value of f_2 remains at the cruising speed on the initial straight road and decreases when the road curvature increases, but the decline is very limited because the maximum allowed curvature on a highway usually does not exceed $0.002m^{-1}$. When the curvature increases, there is a difference between the road curvature and the optimal tentacle curvature, and the value of f_3 is also affected.

Both longitudinal and lateral jerks are within the maximum allowed range, except for a few points. The longitudinal jerk reaches a high level when a change in the curvature of the road ahead is detected at the beginning, and then remains at that level until the vehicle needs to accelerate to pass the obstacle. The lateral jerk reaches a high level when the vehicle decides to steer and change lanes to overtake.

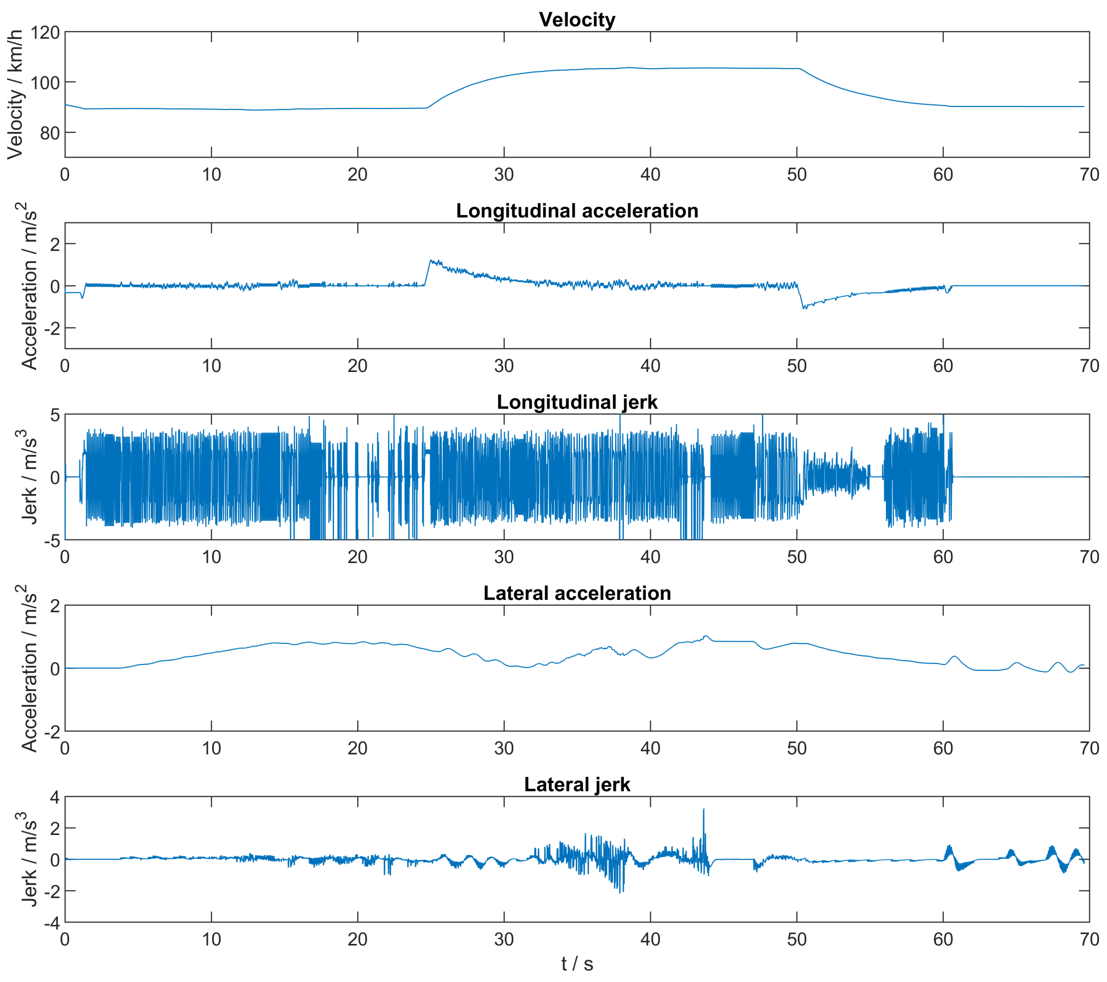


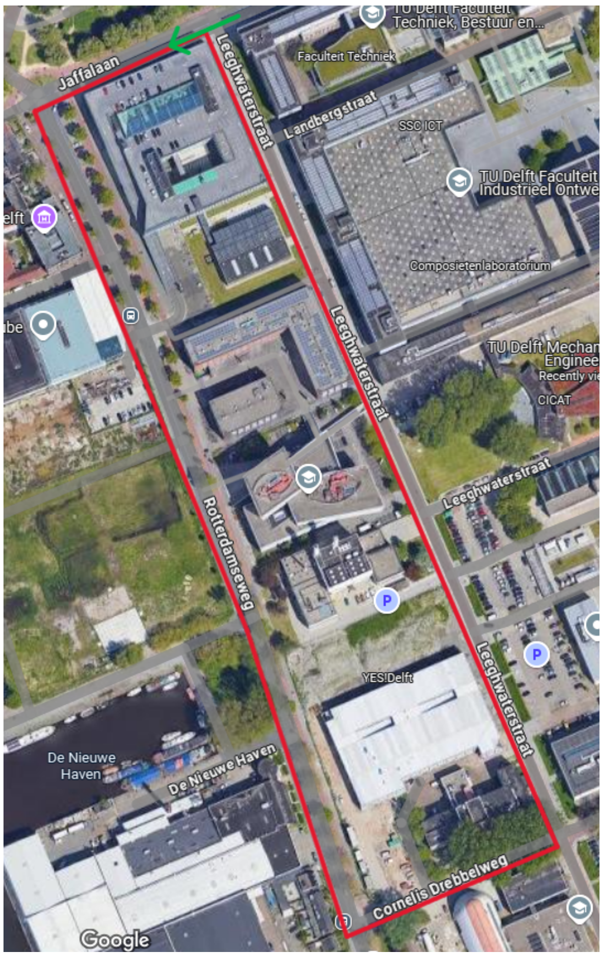
Figure 4.7: Simulation results of scenario 2

4.5. Urban Scenario

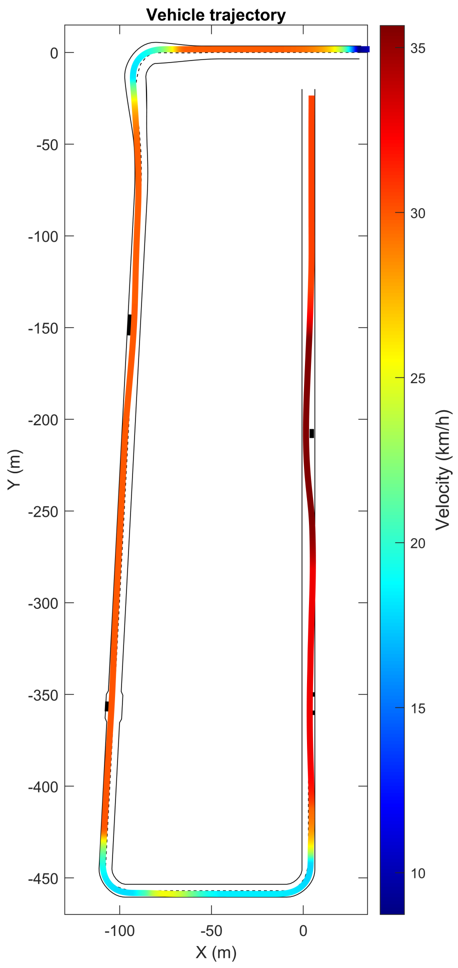
In urban road environments, automated vehicles face not only static obstacles but also dynamic participants such as vehicles, bicycles, and pedestrians. Such environments are particularly typical in university campus settings, where traffic participant density is high and behavioral patterns are complex and variable, imposing strict safety and adaptability requirements on motion planning algorithms. This section will validate the effectiveness of the proposed motion planning algorithm in such complex scenarios through simulation. The simulation environment specifically selects the TU Delft campus with a road network layout. The scenario includes both static roadside parked vehicles and bicycles moving at different speeds, fully reflecting the complexity of urban road traffic. Through this simulation, the algorithm's performance in real-world applications can be directly evaluated.

4.5.1. Scenario

The scenario of TU Delft is represented in Figure 4.8a, the vehicle starts from the green arrow, drives along the red route around the campus, and finally returns to the initial position. Jaffalan and Rotterdamseweg are both two-lane, two-way roads, with a speed limit of $30\ km/h$. After the vehicle turned onto Cornelis Drebbelweg, it became a two-way single lane without a center line. In this scenario, the vehicle will encounter several typical situations, including 1) overtaking stationary vehicles parked on the side of the road; 2) following the vehicle ahead at different speeds; 3) overtaking moving cyclists and vehicles, then returning to the reference path. During these operations, the road curvature dynamically changes from low-curvature straight paths to high-curvature corners. The initial velocity of the vehicle is set to $10\ km/h$, and the cruising speed is set to $30\ km/h$.



(a) TU Delft campus in Google Maps



(b) Vehicle trajectory

Figure 4.8: TU Delft campus and vehicle trajectory

4.5.2. Results

Figure 4.8b shows the vehicle trajectory during the simulation as well as the velocity at each point along the trajectory. The vehicle first accelerates along Jaffalaan to cruising speed, then decelerates to get through the first corner. After entering Rotterdamseweg, the vehicle maintains a cruising speed of $30\ km/h$ while successively overtaking a bus and a van parked at the side of the road. On Cornelis Drebbelweg, the ego vehicle follows the vehicle in front at a speed of $18\ km/h$ because there is not enough space to overtake. On Leeghwaterstraat, the ego vehicle accelerates to overtake two cyclists traveling at $12\ km/h$ and a vehicle traveling at $20\ km/h$. The maneuver is also shown in the speed profile in Figure 4.9. The lateral acceleration reaches a maximum of around $1.6m/s^2$ during right-angle turns at $18s$, $75s$, and $95s$, and the longitudinal acceleration is limited between $-2\ m/s^2$ and $2\ m/s^2$. The longitudinal jerk reaches the maximum of $\pm3\ m/s^3$ when the vehicle brakes and stays below ±3 m/s^3 in other cases, while the lateral jerk reaches the maximum of $2\ m/s^3$ at corners and the cases where the vehicle starts to avoid moving obstacles, indicating a comfortable and smooth drive.

Figure 4.11 represents the value of the sub-functions f_1, f_2, f_3 during simulation as well as the road curvature. The maximum road curvature reaches $0.15m^{-1}$ at right-angle turns while remaining zero at straight lanes. The sub-function f_1 provides additional acceleration when the ego vehicle overtakes the cyclists and the ahead vehicle on Leeghwaterstraat at around $100s$ and $115s$. Although f_1 is calculated when encountering previous static obstacles, f_1 is not activated at that time. The value of f_2 decreases when encountering the three right-angle turns, ensuring that the vehicle is slowed down sufficiently to pass through. Figure 4.10 represents the trajectory without f_2, where the vehicle could only be slowed

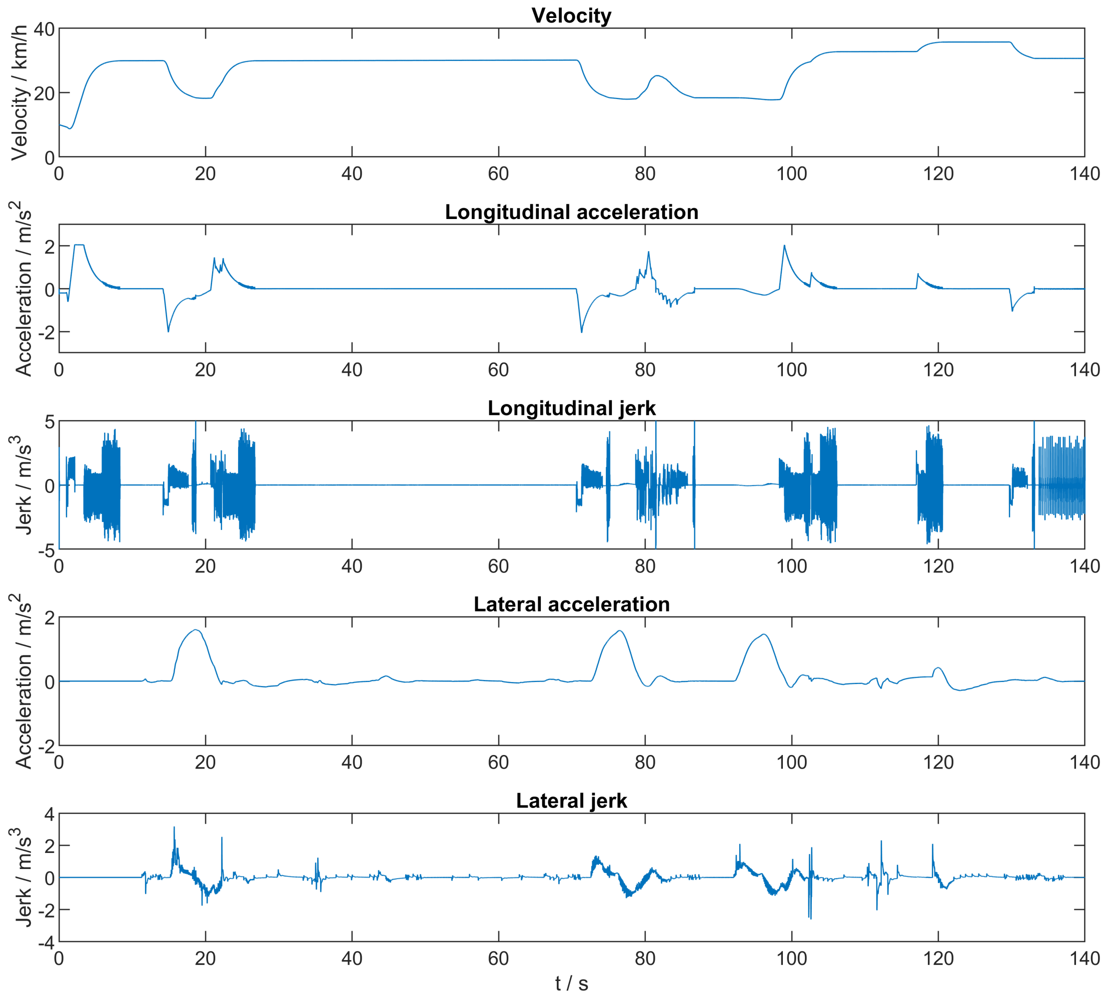


Figure 4.9: Simulation results of scenario 3

down due to the decrease in the length of the best tentacle. Compared to Figure 4.8b, the vehicle is unable to slow down sufficiently, causing it to veer off the road. f_3 also only works when going through right-angle turns, further reducing the vehicle speed when the curvature of the best tentacle is different from the road curvature.

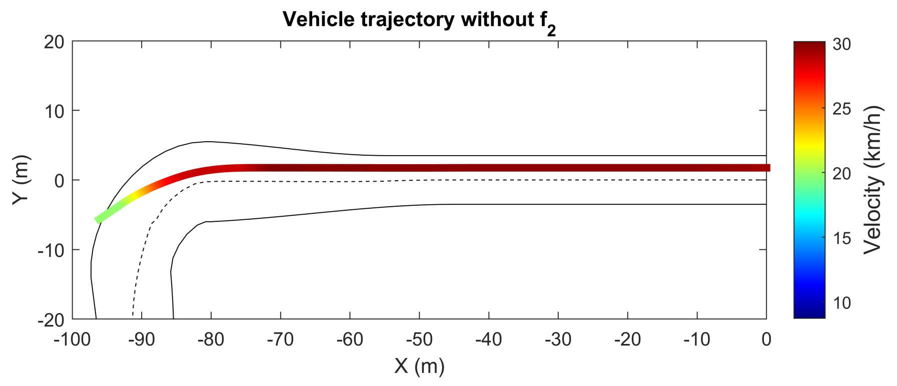


Figure 4.10: Vehicle trajectory without f_2

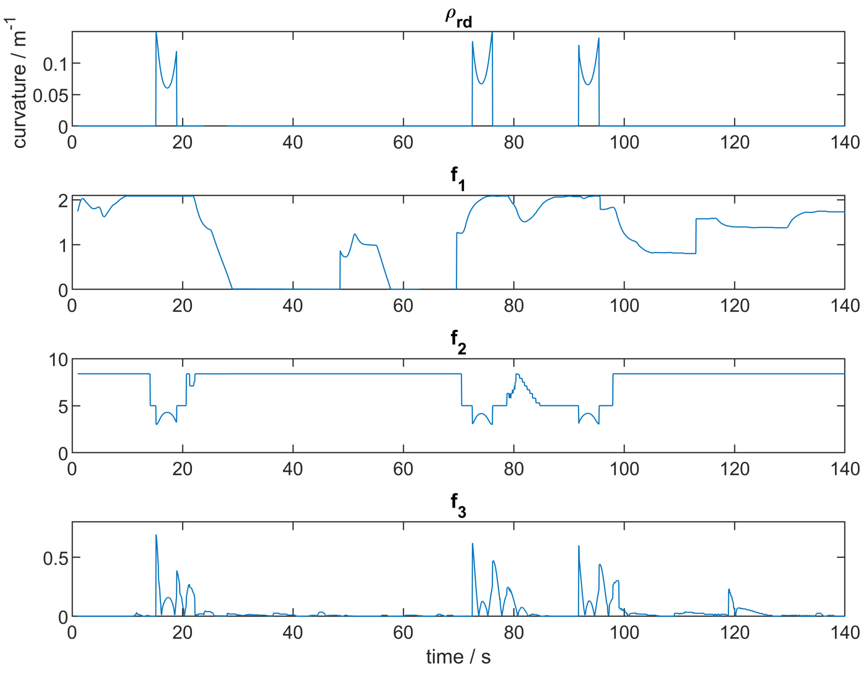


Figure 4.11: Profiling the sub-functions f_1, f_2, and f_3 in scenario 2

4.6. Comparative Analysis

4.6.1. Passenger Comfort

The maximum values and RMS values of the longitudinal and lateral acceleration, as well as the longitudinal and lateral jerk, in different simulation scenarios can be found in Table 4.2. Scenario 1 has the highest maximum longitudinal acceleration and the RMS of longitudinal acceleration because the vehicle needs to brake hard to follow the front vehicle that has a significant different speed. All three scenarios have relatively large maximum longitudinal jerks that exceed the maximum allowed value of $3\ m/s^3$, which could cause longitudinal discomfort at certain moments. Scenario 2 has the highest RMS of longitudinal jerk because cruising on winding roads requires frequent adjustments to the throttle opening. The maximum and RMS lateral acceleration as well as the RMS lateral jerk remained at low levels in all three scenarios, indicating that the passengers will not feel uncomfortable due to greater lateral forces. The maximum lateral jerk exceeded the permissible range of $2\ m/s^3$ in both scenarios 1 and 2 when the vehicle decides whether to overtake the obstacle, indicating possible discomfort. Overall, the proposed algorithm can guarantee passenger comfort in both lateral and longitudinal directions in the vast majority of cases.

4.6.2. Real-time Performance

The planning time of each simulation scenario is represented from Figure 4.12 to 4.14. The maximum, minimum, and average planning times for each scenario are given in Table 4.3. We could tell that the planning time is related to the vehicle speed because the length of the tentacles is defined to be directly proportional to speed in Equation 3.13. A shorter tentacle needs less time in collision detection in the occupancy grid map. As a result, the urban scenario shown in Chapter 4.5, which has the lowest average vehicle speed (under $30km/h$), has the lowest computational time, while the highway scenario shown in Chapter 4.4 has the longest planning time due to the highest average speed. Although the most planning time is required in highway scenarios, the planner could still work at a frequency of $20Hz$ with a maximum planning time of $49.4ms$, verifying the real-time performance of the proposed algorithm in various scenarios.

Table 4.2: Dynamic Metrics of Each Simulation Scenario

Metric	Scenario 1	Scenario 2	Scenario 3
a_{long}^{max} (m/s^2)	2.27	1.28	2.05
a_{long}^{RMS} (m/s^2)	1.09	0.28	0.45
j_{long}^{max} (m/s^3)	4.57	6.53	5.56
j_{long}^{RMS} (m/s^3)	2.29	3.24	2.11
a_{lat}^{max} (m/s^2)	1.18	0.98	1.58
a_{lat}^{RMS} (m/s^2)	0.29	0.66	0.53
j_{lat}^{max} (m/s^3)	5.22	3.24	3.17
j_{lat}^{RMS} (m/s^3)	0.58	0.38	0.35

Table 4.3: Planning Time Statistics for Each Scenario

Metric	Scenario 1	Scenario 2	Scenario 3
Max Planning Time (ms)	38.8	49.4	24.2
Min Planning Time (ms)	21.4	28.3	18.8
Mean Planning Time (ms)	30.4	38.2	20.0

4.7. Discussion

4.7.1. Evaluation

The proposed tentacle-based motion planning algorithm demonstrated robust and effective performance in multiple simulated scenarios. The extensive validation, carried out using the IPG CarMaker and MATLAB/Simulink joint simulation platform, highlights the algorithm's real-time capability and its effectiveness in maintaining passenger comfort by limiting lateral and longitudinal acceleration and jerk. The algorithm consistently performs safe maneuvers in a variety of driving scenarios, such as highway cruising, urban navigation in complex road networks, and static and dynamic obstacle avoidance. Specifically, the planner maintained real-time operation at a frequency of $20\ Hz$, ensuring prompt responses to dynamic changes in the environment. Furthermore, simulation results indicate significant reductions in jerk, with maximum lateral and longitudinal jerk limited to comfortable thresholds, thereby effectively improving ride comfort.

4.7.2. Limitations

Despite the promising results, the proposed algorithm has certain limitations that restrict immediate real-world deployment. First, the algorithm currently assumes perfect perception, with sensor inputs considered accurate and comprehensive, ignoring uncertainties typically encountered in practical scenarios such as sensor noise or adverse weather conditions. Second, although ride comfort is guaranteed most of the time, there are still occasional noticeable jerks in both the longitudinal and lateral directions that could cause discomfort. It might be partly due to that the performance of the planning module is limited by the simple open-loop lateral controller and PD longitudinal controller, which could cause significant tracking errors from the reference path and speed profile. Additionally, the algorithm has primarily been validated through simulation scenarios. Although these scenarios were relatively comprehensive, they could not fully cover all aspects of complex real-world driving dynamics. Therefore, the performance of the proposed planning algorithm in unpredictable, highly dynamic real-world environments has not yet been tested and requires further research.

4.7.3. Future Works

First, future research should focus on enhancing the robustness of the algorithm by integrating perception uncertainties and incorporating sensor noise models. Implementing methods such as probabilistic occupancy grid mapping and sensor fusion techniques could significantly improve the planner's robustness in realistic scenarios. Second, an advanced path-tracking controller should be implemented to reduce the influence of the controller on the performance evaluation of the planner. Optimal controllers that are based on vehicle dynamics models like Linear Quadratic Regulator (LQR) and Model Predictive Control (MPC) controllers should be considered. Third, the current planning method considers primarily ride comfort while neglecting emergency situations that require urgent maneuvering, which is also an important part of ensuring the safety of automated vehicles. Further improvements should be focused on the timing of widening the acceleration and jerk constraints to allow the vehicle to do emergency maneuvers. Additionally, subsequent work should include extensive testing on real vehicles to validate the algorithm under practical conditions, taking into account actuator delays and sensor data uncertainties. Addressing these enhancements will be crucial for bridging the gap between simulation results and practical applicability in fully autonomous driving systems.

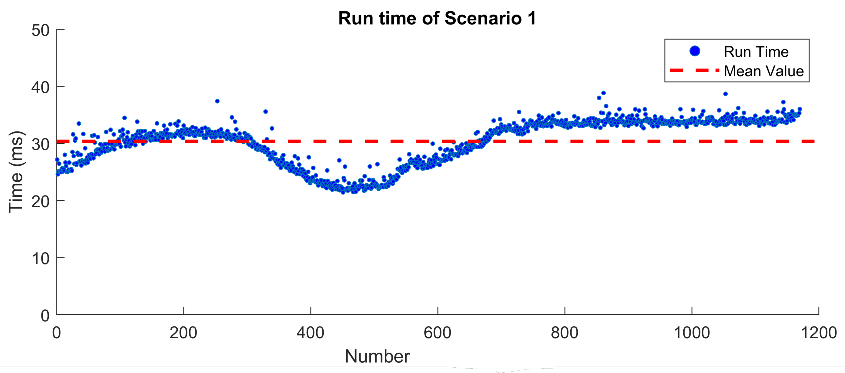


Figure 4.12: Planning time of scenario 1

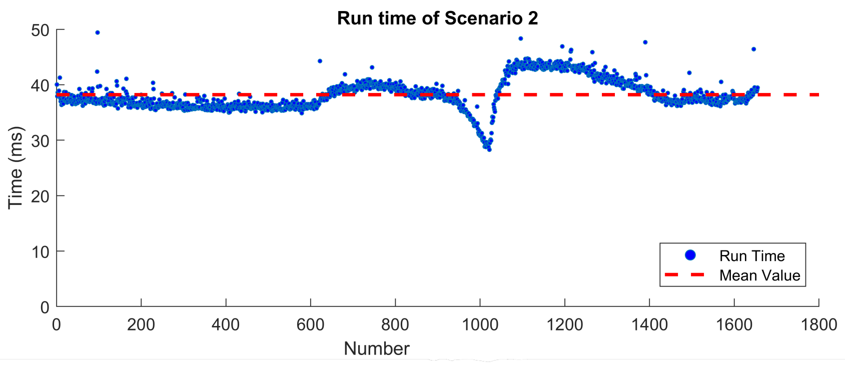


Figure 4.13: Planning time of scenario 2

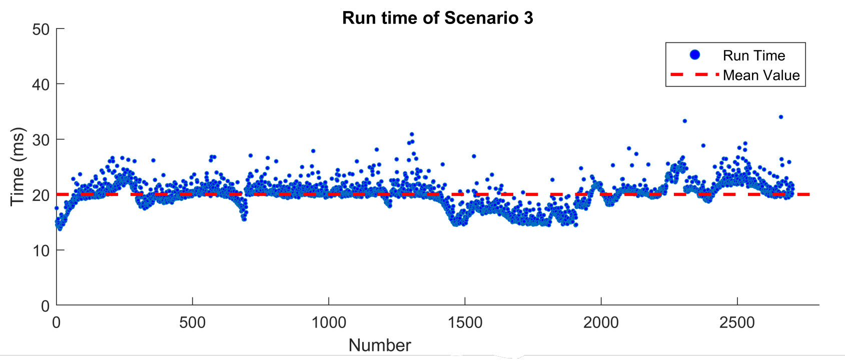


Figure 4.14: Planning time of scenario 3

5 Conclusion

This thesis report proposes a tentacle-based motion planning method for automated vehicles, aimed at addressing two key challenges in automated driving: real-time performance and ride comfort. Unlike traditional motion planning methods, which often separate geometric path planning from speed planning, this algorithm achieves an organic integration of the two by simultaneously generating and evaluating geometric paths (tentacles) and corresponding speed profiles. This integration significantly improves the adaptability and rationality of the velocity profile to driving maneuvers, thereby enhancing overall driving comfort.

The key contribution of this study lies in the constraint of jerks in both lateral and longitudinal directions. By utilizing clothoid tentacles, the algorithm ensures a continuous curvature profile, effectively eliminating abrupt curvature changes and significantly reducing lateral jerks compared to the baseline method. Additionally, by effectively constraining longitudinal jerk using a PD controller, the algorithm considers the vehicle's initial and final states, ensuring smooth acceleration and deceleration.

Comprehensive simulation validation across various traffic scenarios, including highways, common roads, and urban environments, has demonstrated the robustness and practicality of the proposed method. These simulations utilize a high-fidelity vehicle dynamics model integrated with IPG CarMaker and MATLAB/Simulink, demonstrating real-time performance at $20\ Hz$. The algorithm completed static and dynamic obstacle avoidance, and overtaking maneuvers while ensuring comfort. Specifically, both lateral and longitudinal acceleration were kept below $2\ m/s^2$, and lateral and longitudinal jerks are mostly kept below $2\ m/s^3$ and $3\ m/s^3$, respectively.

However, the algorithm still has certain limitations, primarily due to assumptions made in the report, such as perfect environmental perception and ideal vehicle control. The absence of sensor data uncertainties means that further validation is required before practical application. While the algorithm generally maintains ride comfort, occasional jerks persist due to limitations of the simple lateral and longitudinal controllers, causing tracking errors.

Future research directions include integrating sensor data, sensor data uncertainties, and actuator delays to enhance the algorithm's robustness under real-world complex conditions. In addition, advanced path-tracking controllers and emergency route generation methods need to be developed. Finally, testing on real vehicles is needed to provide valuable practical constraints and algorithm optimization insights, bridging the gap between simulation and actual deployment.

In summary, this paper successfully demonstrates that the proposed tentacle-based motion planning algorithm significantly improves passenger comfort and real-time computational efficiency, laying a foundation for future research toward automated driving.

References

[1] Global Road Safety Facility. *Global Road Safety Facility Annual Report 2024*. Annual Report. Washington, DC: Global Road Safety Facility, World Bank, 2024.

[2] Kateřina Bucsuházy et al. “Human factors contributing to the road traffic accident occurrence”. In: *Transportation Research Procedia* 45 (2020). Transport Infrastructure and systems in a changing world. Towards a more sustainable, reliable and smarter mobility.TIS Roma 2019 Conference Proceedings, pp. 555–561. ISSN: 2352-1465. DOI: https://doi.org/10.1016/j.trpro.2020.03.057. URL: https://www.sciencedirect.com/science/article/pii/S2352146520302192.

[3] Ernst Dieter Dickmanns and Volker Graefe. “Dynamic monocular machine vision”. In: *Machine Vision and Applications* 1.4 (1988), pp. 223–240. ISSN: 1432-1769. DOI: 10.1007/BF01212361. URL: https://doi.org/10.1007/BF01212361.

[4] M. Buehler, K. Iagnemma, and S. Singh. *The 2005 DARPA Grand Challenge: The Great Robot Race*. Springer Tracts in Advanced Robotics. Springer Berlin Heidelberg, 2007. ISBN: 9783540734291. URL: https://books.google.nl/books?id=rjxuCQAAQBAJ.

[5] David González et al. “A Review of Motion Planning Techniques for Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841.

[6] E. W. Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. ISSN: 0945-3245. DOI: 10.1007/BF01386390. URL: https://doi.org/10.1007/BF01386390.

[7] Nils J. Nilsson. “A Mobile Automaton: An Application of Artificial Intelligence Techniques”. In: *International Joint Conference on Artificial Intelligence*. 1969. URL: https://api.semanticscholar.org/CorpusID:12735356.

[8] Sertac Karaman and Emilio Frazzoli. “Sampling-based algorithms for optimal motion planning”. In: *The International Journal of Robotics Research* 30 (2011), pp. 846–894. URL: https://api.semanticscholar.org/CorpusID:14876957.

[9] Steven M. LaValle. “Rapidly-exploring random trees : a new tool for path planning”. In: *The annual research report* (1998). URL: https://api.semanticscholar.org/CorpusID:14744621.

[10] Yoshiaki Kuwata et al. “Real-Time Motion Planning With Applications to Autonomous Urban Driving”. In: *IEEE Transactions on Control Systems Technology* 17.5 (2009), pp. 1105–1118. DOI: 10.1109/TCST.2008.2012116.

[11] Sertac Karaman and Emilio Frazzoli. “Optimal kinodynamic motion planning using incremental sampling-based methods”. In: *49th IEEE Conference on Decision and Control (CDC)*. 2010, pp. 7681–7687. DOI: 10.1109/CDC.2010.5717430.

[12] Jeong hwan Jeon et al. “Optimal motion planning with the half-car dynamical model for autonomous high-speed driving”. In: *2013 American Control Conference*. 2013, pp. 188–193. DOI: 10.1109/ACC.2013.6579835.

[13] Jürgen Ackermann. “Robust decoupling, ideal steering dynamics and yaw stabilization of 4WS cars”. In: *Automatica* 30.11 (1994), pp. 1761–1768. ISSN: 0005-1098. DOI: https://doi.org/10.1016/0005-1098(94)90079-5. URL: https://www.sciencedirect.com/science/article/pii/0005109894900795.

[14] Liang Ma et al. “Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving”. In: *IEEE Transactions on Intelligent Transportation Systems* 16.4 (2015), pp. 1961–1976. DOI: 10.1109/TITS.2015.2389215.

[15] Wenda Xu et al. “A real-time motion planner with trajectory optimization for autonomous vehicles”. In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2061–2067. DOI: 10.1109/ICRA.2012.6225063.

[16] Plamen Petrov and Fawzi Nashashibi. “Modeling and Nonlinear Adaptive Control for Autonomous Vehicle Overtaking”. In: *IEEE Transactions on Intelligent Transportation Systems* 15.4 (2014), pp. 1643–1656. DOI: 10.1109/TITS.2014.2303995.

[17] Rainer Trauth et al. “FRENETIX: A High-Performance and Modular Motion Planning Framework for Autonomous Driving”. In: *IEEE Access* 12 (2024), pp. 127426–127439. DOI: 10.1109/ACCESS.2024.3436835.

[18] Te Chen et al. “Trajectory and Velocity Planning Method of Emergency Rescue Vehicle Based on Segmented Three-Dimensional Quartic Bezier Curve”. In: *IEEE Transactions on Intelligent Transportation Systems* 24.3 (2023), pp. 3461–3475. DOI: 10.1109/TITS.2022.3224785.

[19] Ji-wung Choi, Renwick Curry, and Gabriel Elkaim. “Path Planning Based on Bézier Curve for Autonomous Ground Vehicles”. In: *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*. 2008, pp. 158–166. DOI: 10.1109/WCECS.2008.27.

[20] Zheng Ling et al. “Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance”. In: *IET Intelligent Transport Systems* 14 (Jan. 2021), pp. 1882–1891. DOI: 10.1049/iet-its.2020.0355.

[21] Zhiyuan Li et al. “Efficent Lane Change Path Planning based on Quintic spline for Autonomous Vehicles”. In: *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2020, pp. 338–344. DOI: 10.1109/ICMA49215.2020.9233841.

[22] Tim Mercy, Ruben Van Parys, and Goele Pipeleers. “Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment”. In: *IEEE Transactions on Control Systems Technology* 26.6 (2018), pp. 2182–2189. DOI: 10.1109/TCST.2017.2739706.

[23] Tomas Berglund et al. “Planning Smooth and Obstacle-Avoiding B-Spline Paths for Autonomous Mining Vehicles”. In: *IEEE Transactions on Automation Science and Engineering* 7.1 (2010), pp. 167–172. DOI: 10.1109/TASE.2009.2015886.

[24] Robbin van Hoek, Jeroen Ploeg, and Henk Nijmeijer. “Cooperative Driving of Automated Vehicles Using B-Splines for Trajectory Planning”. In: *IEEE Transactions on Intelligent Vehicles* 6.3 (2021), pp. 594–604. DOI: 10.1109/TIV.2021.3072679.

[25] Yulong Zhang et al. “An Obstacle Avoidance Path Planning and Evaluation Method for Intelligent Vehicles Based on the B-Spline Algorithm”. In: *Sensors* 23.19 (2023). ISSN: 1424-8220. DOI: 10.3390/s23198151. URL: https://www.mdpi.com/1424-8220/23/19/8151.

[26] Felix von Hundelshausen et al. “Driving with tentacles: Integral structures for sensing and motion”. In: *Journal of Field Robotics* 25.9 (2008), pp. 640–673. DOI: https://doi.org/10.1002/rob.20256. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.20256. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.20256.

[27] Hafida Mouhagir et al. “Evidential-Based Approach for Trajectory Planning With Tentacles, for Autonomous Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.8 (2020), pp. 3485–3496. DOI: 10.1109/TITS.2019.2930035.

[28] Neşet Ünver Akmandor and Taşkin Padir. “A 3D Reactive Navigation Algorithm for Mobile Robots by Using Tentacle-Based Sampling”. In: *2020 Fourth IEEE International Conference on Robotic Computing (IRC)*. 2020, pp. 9–16. DOI: 10.1109/IRC.2020.00009.

[29] Zhuoren Li et al. “Real-time Local Path Planning for Intelligent Vehicle combining Tentacle Algorithm and B-spline Curve”. In: *IFAC-PapersOnLine* 54.10 (2021). 6th IFAC Conference on Engine Powertrain Control, Simulation and Modeling E-COSM 2021, pp. 51–58. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.10.140. URL: https://www.sciencedirect.com/science/article/pii/S2405896321015421.

[30] Zhuoren Li et al. “A Real-time Path Planner based on Improved Tentacle Algorithm for Autonomous Vehicles”. In: *2022 3rd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. 2022, pp. 629–634. DOI: 10.1109/ICBAIE56435.2022.9985832.

[31] Chebly Alia et al. “Local trajectory planning and tracking of autonomous vehicles, using clothoid tentacles method”. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. 2015, pp. 674–679. DOI: 10.1109/IVS.2015.7225762.

[32] Fadel Tarhini, Reine Talj, and Moustapha Doumiati. “Safe and Energy-Efficient Jerk-Controlled Speed Profiling for On-Road Autonomous Vehicles”. In: *IEEE Transactions on Intelligent Vehicles* (2024), pp. 1–16. DOI: 10.1109/TIV.2024.3416551.

[33] M. Abe. *Vehicle Handling Dynamics: Theory and Application*. Butterworth-Heinemann, 2015. ISBN: 9780081003732. URL: https://books.google.nl/books?id=yOzHBQAAQBAJ.

A

Calculation of Minimum Braking Distance

Consider a vehicle at initial longitudinal speed v, acceleration a, and jerk j. We wish to compute the minimum stopping distance d_s subject to the constraints

$$a(\tau) \;\geq\; a_{dec}^{max}, \qquad \bigl|\dot{a}(\tau)\bigr| \;=\; \bigl|j(\tau)\bigr| \;\leq\; j_{long}^{max},$$

where $a_{dec}^{max} = -4m/s^3$ is the maximum allowable deceleration and $j_{long}^{max} = 3m/s^3$ is the maximum allowable magnitude of jerk.

We decompose the braking maneuver into two phases:

1. **Phase 1: Jerk-limited deceleration to a_{dec}^{max}.**
   Apply the maximum negative jerk j_{long}^{max} to reduce the current acceleration a down to a_{dec}^{max}. The duration of this phase is
   $$t_1 = \frac{a - \left(-a_{dec}^{max}\right)}{j_{long}^{max}} = \frac{a + a_{dec}^{max}}{j_{long}^{max}}.$$
   Under this jerk profile, acceleration varies as
   $$a(\tau) = a - j_{long}^{max}\,\tau, \quad 0 \leq \tau \leq t_1,$$
   and speed evolves according to
   $$v(\tau) = v + a\,\tau - \tfrac{1}{2}\,j_{long}^{max}\,\tau^2.$$
   Hence, the speed at the end of Phase 1 is
   $$v_1 = v + a\,t_1 - \tfrac{1}{2}\,j_{long}^{max}\,t_1^2,$$
   and the distance traveled is
   $$d_1 = \int_0^{t_1} v(\tau)\,\mathrm{d}\tau = v\,t_1 + \tfrac{1}{2}\,a\,t_1^2 - \tfrac{1}{6}\,j_{long}^{max}\,t_1^3.$$

2. **Phase 2: Constant-deceleration braking to zero speed.**
   At the end of Phase 1, apply the maximum deceleration a_{dec}^{max} until the vehicle comes to rest. The duration of this phase is
   $$t_2 = \frac{v_1 - 0}{a_{dec}^{max}} = \frac{v_1}{a_{dec}^{max}}.$$
   The speed in this phase is
   $$v(\tau) = v_1 - a_{dec}^{max}\,\tau, \quad 0 \leq \tau \leq t_2,$$

and the distance traveled is

$$d_2 = \int_0^{t_2} v(\tau)\,\mathrm{d}\tau = v_1\,t_2 - \tfrac{1}{2}\,a_{dec}^{max}\,t_2^2.$$

Summing the two phases gives the minimum braking distance:

$$D_{\min} \;=\; d_1 + d_2 \;=\; v\,t_1 + \tfrac{1}{2}\,a\,t_1^2 - \tfrac{1}{6}\,j_{long}^{max}\,t_1^3 \;+\; v_1\,t_2 - \tfrac{1}{2}\,a_{dec}^{max}\,t_2^2$$

with

$$t_1 = \frac{a + a_{dec}^{max}}{j_{long}^{max}}, \quad v_1 = v + a\,t_1 - \tfrac{1}{2}\,j_{long}^{max}\,t_1^2,$$

$$t_2 = \frac{v_1}{a_{dec}^{max}}.$$

This two-phase strategy yields the shortest stopping distance while respecting both acceleration and jerk limits.

B

Solve the inequalities

Take the inequalities in terms of sub-function f_1 as an example. For Equation 3.36, the parameters a_1, b_1, c_1 could be represented by k_1 as follows according to Equation 3.30. Let $\frac{\partial v_r}{\partial t} = m$.

$$
\begin{aligned}
a_1 &= \frac{\lambda_1}{4k_1^3} \\
b_1 &= -\frac{15\lambda_1}{2k_1^3} \\
c_1 &= \frac{3(10\lambda_1 - k_1\lambda_1)(k_1 + 10)}{4k_1^3}
\end{aligned}
$$

Target Inequality:

$$|-(3a_1v_r^2 + 2b_1v_r + c_1)\, m| < |a_{dec}^{max}|$$

Step 1: Simplify $3a_1v_r^2 + 2b_1v_r + c_1$

We have:

$$3a_1v_r^2 + 2b_1v_r + c_1 = \frac{3\lambda_1(v_r^2 - 20v + 100 - k_1^2)}{4k_1^3} = \frac{3\lambda_1\left[(v_r - 10)^2 - k_1^2\right]}{4k_1^3}$$

Step 2: Substitute into the original inequality and rearrange

$$|-\frac{3\lambda_1\left[(v_r - 10)^2 - k_1^2\right]}{4k_1^3} m| < |a_{dec}^{max}|$$

which gives

$$\frac{3\lambda_1\left[k_1^2 - (v_r - 10)^2\right]}{4k_1^3} m < |a_{dec}^{max}|$$

Multiply both sides by $4k_1^3$ ($k_1 > 0$):

$$4|a_{dec}^{max}|k_1^3 - 3\lambda_1 m k_1^2 + 3\lambda_1 m (v_r - 10)^2 > 0$$

Step 3: Write as a cubic equation and apply Cardano's method

$$4|a_{dec}^{max}|k_1^3 - 3\lambda_1 m k_1^2 + 3\lambda_1 m (v_r - 10)^2 = 0$$

Divide both sides by $4|a_{dec}^{max}|$:

$$k_1^3 - \frac{3\lambda_1 m}{4|a_{dec}^{max}|}k_1^2 + \frac{3\lambda_1 m (v_r - 10)^2}{4|a_{dec}^{max}|} = 0$$

Let

$$\alpha = \frac{3\lambda_1 m}{4|a_{dec}^{max}|}, \quad \gamma = \frac{3\lambda_1 m(v_r - 10)^2}{4|a_{dec}^{max}|}$$

so the cubic becomes

$$k_1^3 - \alpha k_1^2 + \gamma = 0$$

Step 4: Depress the cubic (eliminate the quadratic term) and use Cardano's formula
Let $k_1 = y + \frac{\alpha}{3}$, then

$$y^3 + py + q = 0, \quad p = -\frac{\alpha^2}{3}, \quad q = \gamma - \frac{2\alpha^3}{27}$$

Cardano's Formula:

$$y = \sqrt[3]{-\frac{q}{2} + \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\left(\frac{q}{2}\right)^2 + \left(\frac{p}{3}\right)^3}}$$

$$k_{10} = y + \frac{\alpha}{3}$$

Conclusion:
The inequality holds if $k_1 > k_{10}$.