Technische Universität München
TUM School of Engineering and Design

Local Trajectory Planning Approaches for Autonomous Race Cars

Levent Ögretmen

Vollständiger Abdruck der von der TUM School of Engineering and Design der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz:

Prof. Dr.-Ing. Hartmut Spliethoff

Prüfende der Dissertation:

1. Prof. Dr.-Ing. habil. Boris Lohmann
2. Prof. Dr.-Ing. habil. Ansgar Trächtler
3. Prof. Dr.-Ing. Johannes Betz

Die Dissertation wurde am 13.03.2025 bei der Technischen Universität München eingereicht und durch die TUM School of Engineering and Design am 11.09.2025 angenommen.

Abstract

Developing fully autonomous driving systems requires addressing demanding driving situations such as high speeds, operation at the vehicle's dynamic limits, and rapidly changing environments. These challenges, along with the rise of autonomous racing series, have led to the emergence of a dedicated research field in trajectory planning for autonomous racing. This thesis highlights the limitations of state-of-the-art methods and presents three planning approaches designed to overcome them. As the approaches are applied in the autonomous racing series of the Indy Autonomous Challenge (IAC) and the Abu Dhabi Autonomous Racing League (A2RL), particular emphasis is placed on their practical application.

First, a hybrid planning approach is introduced, which combines elements from sampling-based and graph-based methods. This thesis focuses on the sampling-based part and demonstrates improved vehicle performance and stability compared to an existing pure graph-based approach. The hybrid approach was applied to a full-scale race car, enabling high-speed overtaking in two-vehicle races on oval tracks. Second, a pure sampling-based planning approach is introduced. Unlike existing methods that are limited to oval tracks, this approach can be applied to arbitrarily complex race tracks. Three-dimensional track effects, such as banking and slope, are incorporated, and the impact of online racing line generation on local planning is investigated. The approach was applied to a full-scale race car in races with up to four vehicles on a complex road course. Finally, an interaction-aware approach based on Reinforcement Learning (RL) is presented and compared with a conventional and a game-theoretic approach in an interactive blocking scenario. Additionally, a safety layer is introduced that intervenes when an invalid trajectory is detected, generating a similar but valid alternative. Simulative results show improved success rates for the RL-based approach, and necessary extensions for future real-world applications are discussed.

Danksagung

Diese Arbeit wäre ohne die Unterstützung vieler Menschen nicht möglich gewesen, denen ich zutiefst dankbar bin. Zunächst gilt mein besonderer Dank Prof. Dr.-Ing. Boris Lohmann für die Möglichkeit, dass ich meine Forschung unter seiner Betreuung durchführen durfte. Sein Vertrauen, seine wertvollen Anregungen, seine stetige Unterstützung sowie seine Leidenschaft für Lehre und Präzision haben mich nachhaltig geprägt.

Das TUM Autonomous Motorsport Team war für diese Arbeit von zentraler Bedeutung. Vom ersten Rennen der Indy Autonomous Challenge im Jahr 2021 bis zur Abu Dhabi Autonomous Racing League im Jahr 2024 hatte ich das Privileg, mit hochmotivierten und talentierten Forschenden zusammenzuarbeiten, die gemeinsam zum Erfolg des Teams beigetragen haben. Die intensiven Diskussionen und gemeinsamen Erfahrungen habe ich sehr geschätzt.

Ebenso danke ich meinen ehemaligen Kolleginnen und Kollegen am Lehrstuhl für Regelungstechnik – insbesondere Maximilian Herrmann, Matthias Rowold, Johannes Schwarz und Tobias Thoma. Das offene, freundschaftliche und zugleich professionelle Arbeitsumfeld hat jeden meiner Tage bereichert und mir ermöglicht, mich sowohl in der Forschung als auch in der Lehre weiterzuentwickeln.

Des Weiteren bedanke ich mich bei Felix Anhalt, Matthias Rowold und Simon Hoffmann für das Korrekturlesen, sowie bei allen Studierenden, die zum Gelingen dieser Arbeit beigetragen haben.

Abschließend möchte ich meiner Mutter, meinem Vater, meinen beiden Brüdern und meinen Großeltern von Herzen danken – für ihr Verständnis, ihre beständige Unterstützung und ihre Liebe.

Hannover, 19. Oktober 2025
Levent Ögretmen

Contents

List of Figures

List of Tables

List of Acronyms

2D	two-dimensional	12
3D	three-dimensional	12
A2RL	Abu Dhabi Autonomous Racing League	3
AD	autonomous driving	1
CH	constant heading	120
CLP	constant lateral position	120
DARPA	Defense Advanced Research Projects Agency	3
GAE	Generalized Advantage Estimation	113
IAC	Indy Autonomous Challenge	3
LiDAR	Light Detection and Ranging	6
LVMS	Las Vegas Motor Speedway	71
MDP	Markov Decision Process	106
MPC	Model Predictive Control	7
MPCB	Mount Panorama Circuit in Bathurst	94
NN	neural network	21
ODD	operational design domain	2
PPO	Proximal Policy Optimization	114
PVD	path-velocity decomposition	15
QSS	quasi-steady-state	11
RADAR	Radio Detection and Ranging	6
RH	receding horizon	7
RL	Reinforcement Learning	11
RRT	Rapidly Exploring Random Tree	17
SAE	Society of Automotive Engineers	2
SL	safety layer	105
YMCA	Yas Marina Circuit in Abu Dhabi	94

List of Symbols

Not all symbols used in this thesis are included in this list. Symbols that are used only locally and have no broader relevance are omitted to maintain clarity. Additionally, the meanings of some index variables may vary depending on the planning approach, as clarified within this list.

General

$\mathbf{0}$	Zero matrix or vector
$\boldsymbol{I}$	Unit matrix
$\boldsymbol{e}_{\mathrm{x},\mathcal{A}}$	Unit vector in the direction of the x-axis of frame $\mathcal{A}$
$\boldsymbol{S}(\boldsymbol{a})$	Cross product matrix of vector $\boldsymbol{a} \in \mathbb{R}^3$
$\boldsymbol{R}_{\mathrm{x}}$	Elementary rotation around the x-axis
$\boldsymbol{R}_{\mathrm{y}}$	Elementary rotation around the y-axis
$\boldsymbol{R}_{\mathrm{z}}$	Elementary rotation around the z-axis
$\boldsymbol{R}_{\mathcal{AB}}$	Rotation matrix from frame $\mathcal{B}$ to $\mathcal{A}$
$\boldsymbol{\omega}_{\mathcal{AB}}$	Angular velocity vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
$\boldsymbol{\Omega}_{\mathcal{AB}}$	Spatial curvature vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
$\boldsymbol{x}_{\mathcal{AB}}$	Position vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
$\boldsymbol{v}_{\mathcal{AB}}$	Velocity vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
$\hat{\boldsymbol{a}}_{\mathcal{AB}}$	Acceleration vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
$\tilde{\boldsymbol{a}}_{\mathcal{AB}}$	Apparent acceleration vector of frame $\mathcal{B}$ relative to $\mathcal{A}$
g	Gravitational acceleration
$\boldsymbol{g}$	Gravitational acceleration vector
t	Time

Race track

$\mathcal{I}$	Inertial frame
$\mathcal{R}$	Road frame
$\mathfrak{R}$	Set of all points describing the reference line
$\mathfrak{S}$	Set of all points describing the race track surface
$\boldsymbol{c}(s)$	Curve describing the reference line dependent on its arc length s

s_f	Length of reference line
d_l	Distance from reference line to left track boundary
d_r	Distance from reference line to right track boundary
$\boldsymbol{b}_\mathrm{l}$	Position vector of the left track boundary
$\boldsymbol{b}_\mathrm{r}$	Position vector of the right track boundary
θ	Orientation angle of the race track
μ	Slope angle of the race track
φ	Banking angle of the race track
Ω_x	Geodesic (or relative) torsion of $\mathcal{R}$
Ω_y	Normal curvature of $\mathcal{R}$
Ω_z	Geodesic curvature of $\mathcal{R}$

Vehicle model

$\mathcal{V}$	Velocity frame
s	Progress of the vehicle along the reference line
d	Lateral offset of the vehicle from the reference line
$\hat{\chi}$	Relative orientation of the vehicle to the reference line
x	Position of the vehicle in the x-axis of $\mathcal{I}$
y	Position of the vehicle in the y-axis of $\mathcal{I}$
z	Position of the vehicle in the z-axis of $\mathcal{I}$
V	Velocity of the vehicle in the x-axis of $\mathcal{V}$
w	Velocity of the vehicle in the z-axis of $\mathcal{V}$
$\hat{a}_\mathrm{x}$	Acceleration of the vehicle in the x-axis of $\mathcal{V}$
$\hat{a}_\mathrm{y}$	Acceleration of the vehicle in the y-axis of $\mathcal{V}$
$\hat{a}_\mathrm{z}$	Acceleration of the vehicle in the z-axis of $\mathcal{V}$
$\tilde{a}_\mathrm{x}$	Apparent acceleration of the vehicle in the x-axis of $\mathcal{V}$
$\tilde{a}_\mathrm{y}$	Apparent acceleration of the vehicle in the y-axis of $\mathcal{V}$
$\tilde{g}$	Apparent acceleration of the vehicle in the z-axis of $\mathcal{V}$
ω_x	x-component of angular velocity of $\mathcal{R}$
ω_y	y-component of angular velocity of $\mathcal{R}$
ω_z	z-component of angular velocity of $\mathcal{R}$
$\hat{\kappa}$	Curvature of the path traversed projected on the road plane
$\hat{\kappa}_\mathrm{max}$	Maximum curvature describing the kinematic constraints
$\tilde{a}_\mathrm{x,max}$	Maximum longitudinal acceleration in the gg-diagrams
$\tilde{a}_\mathrm{y,max}$	Maximum lateral acceleration in the gg-diagrams
$\tilde{a}_\mathrm{x,min}$	Minimum longitudinal acceleration in the gg-diagrams
ρ	Shape factor for the gg-diagrams

Local planning interfaces

k	Index variable for discrete data points
N_{map}	Number of data points for the race track map
$\boldsymbol{z}_0$	Estimated state of the ego vehicle
$\square_0$	Variables associated with the estimated ego vehicle state
N_{rl}	Number of data points for the racing line
T_{rl}	Temporal horizon of the racing line
$\square_{\mathrm{rl}}$	Variables associated with the racing line
N_{opp}	Number of detected opponent vehicles at current planning cycle
l	Index variable for detected opponent vehicles
$\square_{\mathrm{opp},l}$	Variables associated with the state of opponent vehicle l
N_{pr}	Number of data points for the prediction
T_{pr}	Temporal horizon of the prediction
$\square_{\mathrm{pr},l}$	Variables associated with the prediction of the opponent vehicle l
V_{limit}	Speed limit
N_{traj}	Number of data points for the planned trajectory
T_{traj}	Temporal horizon of the planned trajectory

Common planning symbols

D_{w}	Vehicle width
D_{l}	Vehicle length
D_{saf}	Safety distance used in collision checks
τ_{prev}	Trajectory planned in the previous planning cycle
τ_{new}	Trajectory planned in the current planning cycle
$\bar{\boldsymbol{z}}_0$	Projection of the ego vehicle state onto the trajectory planned in the previous planning cycle τ_{prev}
$\tilde{\boldsymbol{z}}_0$	Start state of a planning cycle
$\tilde{\square}_0$	Variables associated with the start state
J	Overall cost of an edge or trajectory candidate
$J_{\mathrm{pr},l}$	Prediction cost for opponent vehicle l
Δd	Lateral position difference between racing line and edge or trajectory candidate
V_{target}	Target velocity
V_{follow}	Velocity to follow the front opponent vehicle in no-passing zones
ΔV	Difference between target velocity V_{target} and velocity of edge or trajectory candidate
c_{d}	Cost parameter weighting the lateral distance term
c_{V}	Cost parameter weighting the velocity term

$c_{\hat{\kappa}}$	Cost parameter weighting the curvature term
c_{pr}	Cost parameter weighting the prediction term
Δt_{sim}	Time step for simulations

Hybrid planning approach

N_{s}	Number of layers of the spatial graph
D_{s}	Layer spacing along the reference line
N_{d}	Number of spatial nodes in a layer
D_{d}	Lateral spacing of nodes in a layer
N_{V}	Number of intervals dividing the velocity dimension
N_{t}	Number of intervals dividing the time dimension
i	Index variable for layers
ι	Index of initial layer
j	Index variable for spatial nodes in a layer
j_{rl}	Index of spatial node aligned laterally with the racing line
p	Index variable for intervals dividing the velocity dimension
q	Index variable for intervals dividing the time dimension
$\mathfrak{L}_i$	Layer indexed with i (set of spatial nodes with same progress)
s_i	Progress of layer i
$\mathfrak{n}_{ij}$	Spatial node indexed with j at layer i
$\square_{ij}$	Variables associated with spatial node $\mathfrak{n}_{ij}$
Δs_{min}	Minimum distance for initial layer selection
$\mathfrak{E}_0$	Set of initial edges
$\mathfrak{E}_1$	Set of remaining spatio-temporal edges
t_{e}	End time of an initial edge
$N_{\underline{\mathrm{V}}}$	Number of velocity samples in low-speed range
$N_{\bar{\mathrm{V}}}$	Number of velocity samples in high-speed range
m	Index variable for sampled velocities
V_m	Velocity sample m
$\square_{jm}$	Variables associated with spatial node $\mathfrak{n}_{\iota j}$ and velocity sample m
T_{col}	Temporal horizon for the collision checks
$d_{\mathrm{x},l}$	Longitudinal distance between the ego and the opponent vehicle l from the opponent vehicle's perspective
$d_{\mathrm{y},l}$	Lateral distance between the ego and the opponent vehicle l from the opponent vehicle's perspective
p_{x}	Longitudinal length of the elliptical prediction cost distribution
p_{y}	Lateral length of the elliptical prediction cost distribution
h	Scaling function for the prediction cost term

Sampling-based planning approach

$\mathfrak{T}_{\text{long}}$	Set of longitudinal curves
$\mathfrak{T}_{\text{lat}}$	Set of lateral curves
$N_{\dot{\text{s}}}$	Number of longitudinal velocity samples
$K_{\dot{\text{s}}}$	Scaling factor for the upper limit of longitudinal velocity sampling
N_{d}	Number of lateral position samples per velocity sample
i	Index variable for sampled longitudinal velocities
j	Index variable for sampled lateral position
$\dot{s}_{\text{e},i}$	Longitudinal velocity sample i
$\tilde{d}_{\text{rl},i}$	Lateral profile required to traverse the racing line path with longitudinal curve s_i
$d_{\text{e},ij}$	Lateral position sample j for longitudinal velocity sample i
$\square_{ij}$	Variables associated with longitudinal velocity sample i and lateral position sample j
p_{s}	Parameter specifying the length of the elliptical prediction cost distribution
p_{d}	Parameter specifying the width of the elliptical prediction cost distribution
H_{rl}	Spatial horizon of the racing line
$D_{\text{saf,rl}}$	Safety distance for collision checks of the racing line
$\tilde{a}_{\text{scl}}$	Scaling factor of the gg-diagrams used for the racing line
$\tilde{a}_{\text{mgn}}$	Absolute margin of the gg-diagrams used for the racing line
D_{snr}	Sensor range used for simulations

Reinforcement Learning theory

$\mathfrak{s}$	Set of all possible states (state space)
$\mathfrak{a}$	Set of all possible actions (action space)
$\mathfrak{r}$	Set of all possible rewards
t_{d}	Discrete time step
$S_{t_{\text{d}}}$	State at time step t_{d}
$A_{t_{\text{d}}}$	Action at time step t_{d}
$R_{t_{\text{d}}}$	Reward received at time step t_{d}
π	Policy of the agent
P	Dynamics of the environment
σ and σ'	Any state from $\mathfrak{s}$
α	Any action from $\mathfrak{a}$
r	Any reward from $\mathfrak{r}$

N_{ep}	Number of time steps in an episode
π^*	Optimal policy
G_{t_d}	Return (discounted sum of all future rewards)
γ	Discount factor
v_π	State-value function for policy π
q_π	Action-value function for policy π
$\boldsymbol{\theta}$	Parameter vector for the policy
N_θ	Dimension of the policy parameter vector $\boldsymbol{\theta}$
J_π	Performance criterion for policy-based and actor-critic methods
η_θ	Step size for updating the policy parameter vector $\boldsymbol{\theta}$
b	Baseline used in policy-based and actor-critic methods
$\hat{v}_\pi$	Function approximation of the state-value function
$\boldsymbol{\phi}$	Parameter vector for the state-value function approximation
N_ϕ	Dimension of the state-value parameter vector $\boldsymbol{\phi}$
η_ϕ	Step size for updating the state-value parameter vector $\boldsymbol{\phi}$
$\square_\text{old}$	Parameter vector $\square$ before update step
$\square_\text{new}$	Parameter vector $\square$ after update step

Interaction-aware planning using Reinforcement Learning

$s_{\text{ego,init}}$	Initial progress of the ego vehicle
$d_{\text{ego,init}}$	Initial lateral position of the ego vehicle
$s_{\text{opp,init}}$	Initial progress of the opponent vehicle
$d_{\text{opp,init}}$	Initial lateral position of the opponent vehicle
$\hat{\chi}_\text{init}$	Initial relative orientation of the ego and opponent vehicle
V_init	Initial velocity of the ego and opponent vehicle
$\square_\text{ego}$	Variables associated with the ego vehicle
$\square_\text{opp}$	Variables associated with the opponent vehicle
δ	Steering angle
ω	Steering rate
β	Body slip angle
δ_max	Maximum steering angle
ω_max	Maximum steering rate
l_r	Distance from the center of gravity to the rear axle
l_f	Distance from the center of gravity to the front axle
K_P	Proportional gain in the blocking control law
K_D	Derivative gain in the blocking control law
e	Control error in the blocking control law
$\hat{\chi}_{\text{opp,d}}$	Desired relative orientation in the blocking control law

$\Delta\bar{d}$	Lateral gap in the blocking control law
K_{L}	Look-ahead distance in the blocking control law
$V_{\max}$	Maximum velocity of the vehicle
R_{success}	Reward received for a successfully finished episode
R_{failure}	Reward received for an unsuccessfully finished episode
$\bar{R}_{t_{\mathrm{d}}}$	Dense reward received at time step t_{d}
$\Delta\dot{s}$	Relative longitudinal velocity between the ego and opponent vehicle
$\Delta\dot{s}_{\max}$	Maximum relative velocity achieved up to the current time step
k_{scl}	Vehicle dimension scaling factor for the curriculum learning stages
$\square_{\mathrm{invalid}}$	Variables associated with the invalid trajectory used for the safety layer

1 Introduction

Throughout this thesis, key challenges in planning for autonomous racing are investigated, and various solutions to overcome them are explored. This chapter provides an overview of the research contributions presented in this work. Section 1.1 motivates research in autonomous racing in general and establishes the background in which this work was developed. Subsequently, Section 1.2 explains the specific task of planning and its associated requirements. Section 1.3 then offers a comprehensive review of existing planning approaches, highlighting their strengths and drawbacks. Building on this review, Section 1.4 identifies limitations in the current state of the art and outlines the contributions of this thesis that aim to address these limitations.

1.1 Motivation and Background

The potential benefits of autonomous driving (AD) for the transport sector's future are manifold. AD promises a better user experience, increased mobility for mobility-impaired people, increased road safety, and reduced energy consumption as well as emissions [1, 2]. Regarding road safety, the survey in [3] states that human error is the cause of 94 % of all crashes, allowing AD to reduce the number of accidents significantly. Meanwhile, the impact of autonomous vehicles on the environment is unclear. However, a reduction in energy consumption and greenhouse gas emissions - possibly by up to 80 % - is likely through several factors, including efficient traffic flow, less congestion, and vehicle weight reductions enabled by increased safety standards [4]. In the United States alone, these factors could lead to $800 billion in annual consumer and societal benefits by 2050 [5]. In addition to the benefits mentioned, it can generate significant value in the automotive industry, with a potential revenue of $300 to $400 billion in the passenger car market alone by 2035 [6].

Table 1.1: SAE levels of AD systems according to [7]

Level	Driving task		Fallback	ODD
	Longitudinal and lateral control	Object and event detection/response		
0	Driver	Driver	Driver	-
1	Driver & System	Driver	Driver	Limited
2	System	Driver	Driver	Limited
3	System	System	Driver	Limited
4	System	System	System	Limited
5	System	System	System	Unlimited

In [7], the Society of Automotive Engineers (SAE) classifies different levels of AD systems based on the degree of automation. For this, the driving task is divided into the subtasks of lateral and longitudinal vehicle motion control, as well as the detection and the appropriate reaction to objects and events. Also, a fallback is defined for the driving task to bring the vehicle into a minimal risk condition if required. Depending on the SAE level, this fallback may be performed by the driver or an alternative functionality of the AD system. Further, the operational design domain (ODD) defines the operating conditions for which an AD system is designed. With these definitions, the SAE levels are listed in Table 1.1. While in levels 0 to 2, the driver must perform at least one subtask, in levels 3 to 5, the AD system performs the entire driving task. While in level 3, the driver still serves as a fallback that must be prepared if the AD system requires an intervention, in level 4, the AD system also handles the fallback. Finally, in level 5, the AD system performs the driving task and the fallback under all conditions, i.e., with an unlimited ODD.

Currently, automotive manufacturers have achieved level 3 autonomy for the consumer market. In 2021, Honda became the first company to receive approval for a level 3 AD system in Japan, although only a limited number of cars were made available for lease [8]. Mercedes-Benz was the first manufacturer to sell level 3 cars, with approval in Germany and selected states in the United States [9]. Beyond the consumer market, level 4 vehicles are already used in driverless taxis and delivery services [10]. Nevertheless, transitioning to level 5 is particularly difficult, as it requires the handling of all potential edge cases arising

from the unlimited ODD. Such edge cases occur especially in highly dynamic environments and at the vehicle's handling limits, which are characterized by high speeds and accelerations. Autonomous racing competitions provide a controlled and safe environment where these edge cases frequently occur. Here, the AD software can be systematically tested and developed under extreme conditions, thus further narrowing the gap to level 5 AD systems [11, 12].

The challenges of the Defense Advanced Research Projects Agency (DARPA) are often regarded as the predecessors of the current autonomous racing series. The first competition was the DARPA Grand Challenge 2004, in which an off-road course had to be completed within a limited time [13]. None of the teams could complete the route, so the competition was repeated similarly the following year at the DARPA Grand Challenge 2005, with 5 out of 23 finalists reaching the finish line [14–17]. Whereas in the 2004 and 2005 challenges, the vehicles drove in isolation, in the DARPA Urban Challenge 2007, the focus shifted to structured road environments and traffic rules. An urban area course had to be covered in compliance with all traffic regulations, making detection and evasion of other vehicles necessary. In this challenge, 6 out of 11 finalists completed the course [18–23]. After the DARPA challenges, several other AD competitions emerged. In addition to many applications with 1:10 scale cars [24, 25], racing competitions with full-scale vehicles and high speeds evolved. The Roborace racing series [11, 26, 27] was held in various seasons from 2015 to 2021. The electric, autonomous race cars reached speeds of up to 220 km/h in single-vehicle laps on complex race tracks. In addition to high-speed single-vehicle driving, simplified overtaking maneuvers were also performed. In 2017, the Formula Student Driverless competition [28, 29] was introduced, in which the teams had to create both the mechanics and the software of the car for single-vehicle driving on a race track outlined by cones with speeds below 100 km/h. The Indy Autonomous Challenge (IAC) racing series [30–33] was launched in 2021 and has since included several single-vehicle and passing competitions on high-speed oval tracks. The passing competitions involved overtaking maneuvers between two vehicles at speeds of up to 270 km/h. The latest racing series is the Abu Dhabi Autonomous Racing League (A2RL). Its first race took place in April 2024, featuring a four-vehicle race on a Formula 1 track. Additional races have been announced.

(a) Race of the IAC (7 January 2022) at the Las Vegas Motor Speedway.

(b) Race of the A2RL (27 April 2024) at the Yas Marina Circuit in Abu Dhabi.

Figure 1.1: Vehicles (blue) of the TUM Autonomous Motorsport team competing in the IAC and A2RL. *Image Credits: Indy Autonomous Challenge / Technical University of Munich*

The challenges and competitions have different focuses, and the participating teams use individual conceptual approaches. Nevertheless, the various AD software approaches can fundamentally be divided into the sequential structure of perception, planning, and control.[1] In the perception module, the vehicle's sensor data is processed to perceive the surrounding world. This includes the state estimation of the vehicle to be controlled, the extraction of relevant road information, and the detection of other cars. This information is forwarded to the planning module, where the decision-making process is executed, i.e., the intended movement of the vehicle is planned. This plan is then passed to the control module, which realizes the planned movement by appropriately selecting the car's control inputs (steering angle, throttle, and brake).

The work presented in this thesis was developed from 2020 to 2024 within the TUM Autonomous Motorsport research team, which participates in the autonomous racing series IAC (Figure 1.1a) and A2RL (Figure 1.1b). This thesis discusses the limitations of existing planning approaches and presents three algorithms that address the challenges of planning for autonomous racing. These challenges include planning at the vehicle's handling limits on complex race tracks, solving non-convexity, considering interaction in multi-vehicle scenarios, and enabling fast reactions through low computation times.

[1] End-to-end approaches, in which the vehicle's control inputs are calculated directly from the sensor data, are excluded here, as they have not been utilized in previous competitions.

1.2 Planning for Autonomous Racing

The planning module is responsible for decision-making and, for instance, determines the braking point at the entrance to a turn or when and how to initiate an overtaking maneuver. This section introduces the basic planning terminology in Subsection 1.2.1, classifies the planning task within the overall AD software architecture in Subsection 1.2.2, and describes the division of the planning module into submodules to realize the planning task in Subsection 1.2.3.

1.2.1 Terminology

The following terms and distinctions are used throughout this thesis.

Single-Vehicle and Multi-Vehicle Racing Single-vehicle racing involves only one vehicle on the race track, corresponding to a qualifying lap. By contrast, two or more vehicles are simultaneously on the track in multi-vehicle racing, requiring overtaking and evasive maneuvers.

Ego Vehicle and Opponent Vehicles The ego vehicle is the vehicle controlled by the AD system under consideration. In this work, this is always the vehicle controlled by the proposed planning algorithms. In multi-vehicle racing, the remaining cars (controlled by the software of opposing teams in a competition) are referred to as opponent vehicles.

Oval Track and Road Course An oval track is an oval-shaped closed race track with turns in only one direction. Such tracks are characterized by their geometric simplicity, their usually banked turns, and the consistently high speeds that can be achieved. Meanwhile, road courses are closed race tracks of any complex geometry. They are characterized by the diversity of their turns and straights, resulting in strong variations in the driven speed profile.

Path and Trajectory A path is a sequence of spatial positions without time information, i.e., a spatial curve parameterized by its arc length. It may also contain additional geometric details, such as the orientation or curvature. A trajectory is a path with additional time stamps for each position, i.e., a spatial curve parameterized by time. Due to the temporal dimension, higher-order time derivatives such as velocity, acceleration, and jerk can also be provided.

Feasible and Collision-Free Trajectory A trajectory is considered feasible if it can be traversed by the ego vehicle, assuming the absence of opponent vehicles and track restrictions. Internal constraints indicate whether the vehicle can execute the trajectory under its kinematic and dynamic limitations. Additionally, a trajectory is collision-free if no collision occurs with the track boundaries or opponent vehicles when traversing the trajectory, taking into account the vehicle geometry. These limitations are also referred to as external constraints.

Planning Horizon The planning horizon specifies the spatial or temporal horizon of the trajectory to be planned. A sufficiently long planning horizon is necessary to ensure the availability of a feasible solution in every planning cycle. If the planning horizon is too short, for instance, a turn may be accounted for too late, causing the necessary braking point to be missed.

Racing Line The racing line is the time-optimal trajectory around the entire race track in the absence of opponent vehicles. The spatial planning horizon of the racing line corresponds to the track length, and the start and end states of the trajectory coincide.

1.2.2 Software Architecture and Planning Task

The vehicle is equipped with sensors to perceive the environment and actuators to control the system as desired. Commonly installed sensors include Global Navigation Satellite System (GNSS) units, Inertial Measurement Units (IMUs), cameras, Light Detection and Ranging (LiDAR), and Radio Detection and Ranging (RADAR) sensors. The control inputs consist of the steering angle, throttle position, and brake pressure.

Figure 1.2 shows the architecture of the AD software, which computes the control inputs from the sensor signals. As outlined in Section 1.1, the AD software can be divided into three sequentially executed modules: perception, planning, and control. The sensor data is used in the perception module to estimate the ego vehicle's state (state estimation) and to generate an object list consisting of the states of the opponent vehicles (detection). This information is forwarded to the planning module, along with the map of the race track and the racing flags. In contrast to the other planning inputs, the map can be generated offline before driving, as the race track is known in advance in the

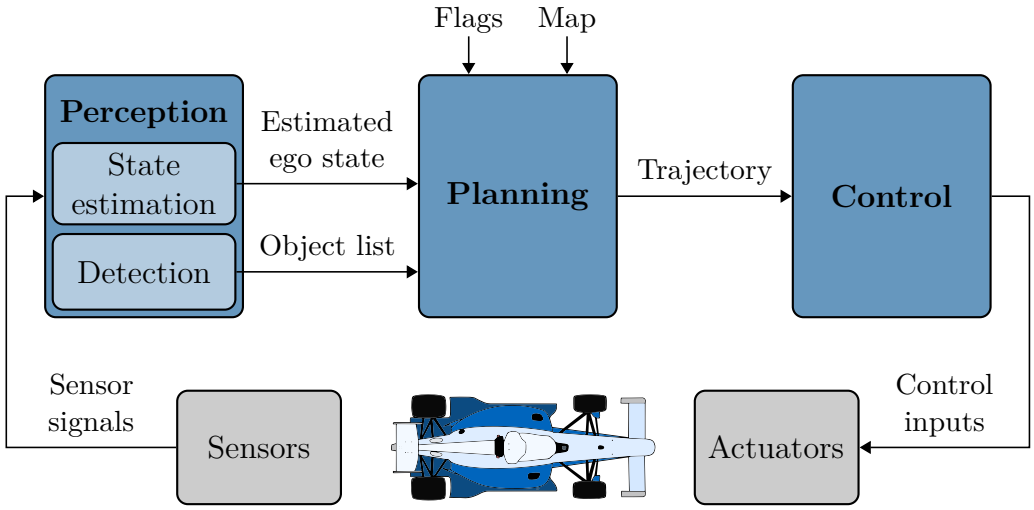


Figure 1.2: Software architecture with modules (blue) and vehicle interfaces (gray).

racing series IAC and A2RL.[2] The racing flags are external inputs sent by the race control during a race to announce the current track conditions and are, therefore, not part of the AD software. According to the race regulations, the vehicle must adhere to specific rules, such as the maximum speed, depending on the active flags. Based on the four inputs, the planning module must generate a feasible and collision-free trajectory that minimizes lap time while adhering to the racing rules specified by the racing flags. Finally, the planned trajectory is realized within the control module by the proper choice of the control inputs.

All modules are executed regularly and asynchronously with the latest input data, allowing the system to react to unexpected environmental changes. In this context, a single execution of a module is referred to as a cycle. The planning module, therefore, generates a trajectory in each planning cycle based on the latest ego vehicle state, object list, and racing flags. For efficiency, the planning horizon is finite, resulting in a receding horizon (RH) execution similar to the concept of Model Predictive Control (MPC) [35]. Execution in an RH fashion requires that a solution exists in every planning cycle. This requirement is referred to as recursive feasibility. In single-vehicle racing, it is theoretically

[2]For the sake of completeness, it should be mentioned that approaches such as [34] also exist in which local maps are regularly generated online based on the current sensor data. However, this has the disadvantage of increased computational effort and lower accuracy and is, therefore, not pursued further in this thesis.

guaranteed by a sufficiently long planning horizon so that deceleration to a valid state, e.g., standstill, is possible within the horizon. In multi-vehicle racing, however, recursive feasibility cannot be guaranteed due to the unpredictable movement of the opponent vehicles. In practice, the choice of the planning horizon is a trade-off, as increasing the horizon also increases the computational effort. However, short execution times are required to react quickly to changes in the highly dynamic environment.

1.2.3 Planning Procedure

The task of time-optimal trajectory planning, subject to internal (feasibility) and external (collision avoidance) constraints, constitutes a non-convex long-horizon optimization problem. Since solving this complex optimization problem in one step is time-consuming, it is usually divided into the subtasks of local and global planning. Additionally, local planning can be performed either in a conventional or interaction-aware manner. This subsection explores both the division into global and local planning, as well as the distinction between conventional and interaction-aware approaches. It concludes with an overview of the resulting planning structures.

Global and Local Planning

The planning module is generally divided into the subtasks of global and local planning [36]. In global planning, the racing line, i.e., the time-optimal closed trajectory for the entire race track without considering opponent vehicles, is generated. This step can be executed offline, as the race track map is known in advance, and the racing line is independent of the ego state. In local planning, a trajectory with a shorter planning horizon starting from the current ego state is generated in each cycle according to the RH concept. The trajectory is then transferred to the control module as a target. The fundamental idea of the division into global and local planning is that the trajectory generated by local planning should match the racing line whenever possible and deviate from it if it is blocked by static objects or slower opposing vehicles. The racing line generated by global planning thus specifies the long-term feasible behavior, allowing a shorter planning horizon and, therefore, reduced computation times for local planning, which is crucial for

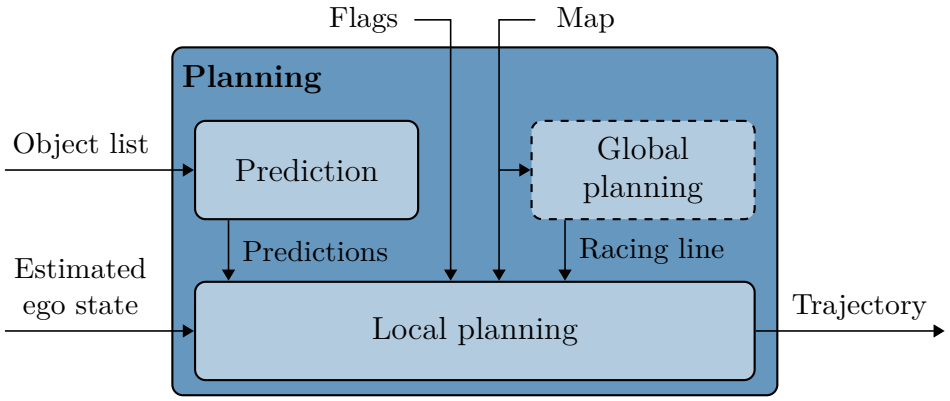
online execution. Local planning then incorporates the local environment, i.e., it plans necessary evasion and overtaking maneuvers based on the current inputs.

The division into global and local planning is used in all proposed methods throughout this thesis. For the sake of completeness, it should be noted that behavioral planning is often mentioned in the literature as a third category in addition to global and local planning. This involves planning high-level maneuvers, such as driving to the left, straight, or right, and accelerating, braking, or driving at a constant speed. However, this category is not considered in this thesis, as it is not employed in the presented planning approaches.

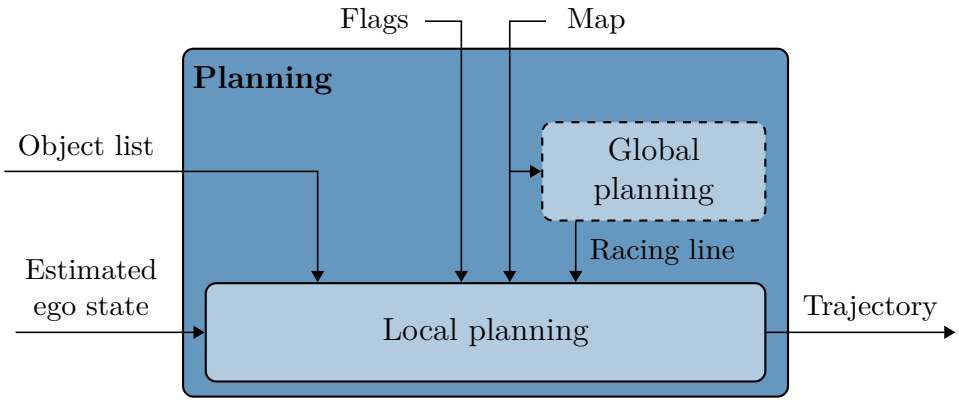
Conventional and Interaction-Aware Local Planning

Local planning can be performed either conventionally or interaction-aware. In conventional planning, the local planning task is divided into the prediction of opponent vehicles and the actual planning of a feasible and collision-free trajectory. In the prediction subtask, a trajectory is predicted for each opposing vehicle for at least the duration of the planning horizon. Subsequently, in the actual local planning task, a trajectory is planned that avoids collisions based on the predictions of other vehicles. The effectiveness of this division depends on the validity of the prediction. For example, suppose an opponent vehicle strictly follows the outer line of a turn but is predicted to pull onto the inner line. In that case, the ego vehicle may initiate an overtake on the outer line, which could result in a collision. The sequential execution of prediction and planning, therefore, is only reasonable if the opponent's movement can be well predicted, for example, by strict racing rules.

Another fundamental problem with the conventional approach is that no interaction between the ego and the opponent vehicles can be accounted for. If the opponent vehicle does not strictly follow a predefined trajectory as in the example above but interacts with the ego vehicle, an exact prediction might not be possible. An example of such an interaction is an opponent vehicle trying to block the ego vehicle behind it on a sufficiently wide straight. If the opposing vehicle is predicted to drive straight, the ego vehicle may initiate an overtaking maneuver. In response, the opponent vehicle reacts with a blocking maneuver and thus deviates from the initial prediction. In this case, the prediction of the opponent vehicle is not possible since the prediction impacts the ego vehicle's behavior, which, in turn, affects the behavior of the opponent vehicle. This



(a) Conventional planning.



(b) Interaction-aware planning.

Figure 1.3: Structures of conventional (a) and interaction-aware (b) planning approaches. The submodules framed with dashed lines run offline.

motivates interaction-aware local planning approaches that do not perform the split into prediction and planning and instead plan the trajectory in one step.

Planning Structure

With the two divisions described in this subsection, the planning module shown in Figure 1.2 can be represented by its internal submodules, as depicted in Figure 1.3. While in conventional planning, the prediction submodule generates the prediction for local planning based on the object list, in interaction-aware planning, the object list is fed directly as input for local planning.

1.3 Related Work

This section provides an overview of global planning approaches in Subsection 1.3.1, followed by conventional local planning approaches in Subsection 1.3.2 and interaction-aware local planning approaches in Subsection 1.3.3. In addition to methods designed specifically for racing, methods for road traffic are also addressed, as the fundamental concepts are similar and can be adopted. Here, some terms and concepts are introduced broadly and explained in more detail in the subsequent chapters.

This overview outlines only selected publications relevant to the context of this thesis. For a complete overview, please refer to the numerous surveys on trajectory planning for racing [36] and road traffic [37–40]. Overviews explicitly based on machine learning, particularly Reinforcement Learning (RL), are provided in [41–43]. Approaches for predicting opposing vehicles are not covered, as the subtask of prediction is not the subject of this work. For an overview and categorization of current prediction approaches, please refer to [44].

1.3.1 Global Planning

Global planning approaches calculate the time-optimal closed racing line for the entire race track. A distinction is made between fixed-path methods, where the time-optimal velocity profile is optimized for a predefined path, and free-path methods, where the path is determined as part of the optimization problem solution.

Fixed-Path Methods

These approaches are based on a fixed path, and only the velocity profile is optimized. Since the path leading to the time-optimal trajectory is not known in advance, an approximation, such as the minimum curvature path, must be selected, which may lead to suboptimality. The velocity profile is then often calculated based on the quasi-steady-state (QSS) assumption. In this, it is assumed that the racing line consists of discrete points at which the vehicle is in a state of equilibrium. This assumption neglects the transient vehicle dynamics between the discrete points, but the resulting error can be minimized with a finer spacing of the points. Based on the QSS assumption, simple gg-diagrams can be

created that limit the vehicle's combined longitudinal and lateral acceleration. The gg-diagrams were first introduced in [45] and can be generated from an arbitrarily complex vehicle model or real-world experiments.

On two-dimensional (2D) race tracks, the gg-diagrams are speed-dependent. For this case, the velocity profile of the racing line is generated in [46] by determining the local curvature maximum (apex) for each turn of the predefined path. Assuming that the longitudinal acceleration is zero at these points, the velocity profiles before and after the apex can be calculated without exceeding the limits of the gg-diagrams. The overall time-optimal velocity profile for the track is then obtained as the minimum of all partial velocity profiles of the individual turns. In [47], the time-optimal velocity profile is instead generated using a forward-backward solver based on [48, 49].

In [50], the fixed-path approach is generalized to three-dimensional (3D) race tracks, including elevation changes and inclinations. While the basic principle remains similar to the apex search in [46], the gg-diagrams are no longer dependent only on the speed but also on the vertical acceleration with which the vehicle is pushed onto the track surface.

Free-Path Methods

With the free-path methods, the path of the racing line is not predefined. In [51], the racing line is optimized by iteratively modifying the path and the corresponding velocity profile obtained by a fixed-path approach. Otherwise, free-path methods are mainly based on nonlinear optimal control problems that can be solved numerically using direct or indirect methods [52]. The cost functional to be minimized is the lap time, subject to the system dynamics of a vehicle model and the input and state constraints such as track boundaries.

Examples of lap time optimization on 2D race tracks with a dynamic single-track or double-track vehicle model are [53–56]. By incorporating these vehicle models within the optimal control problem, the transient vehicle dynamics are captured. In contrast, the authors of [57] use a simple point-mass model whose acceleration is limited by speed-dependent gg-diagrams according to the QSS assumption. Even though no transient dynamics are considered here, the racing line path is optimized, potentially yielding better results than fixed-path methods.

The described approaches can again be generalized to 3D race tracks. In [58, 59], a double-track model for a laterally flat 3D track is derived and used within the lap time optimization. The same laterally flat 3D track model is used in [60], along with a point-mass model constrained by gg-diagrams that depend on the vehicle's velocity and vertical acceleration. An extension to laterally curved 3D race tracks is provided in [61].

Comparison

Fixed-path methods are easy to implement and require short computation times. However, they rely on a predefined path, potentially leading to suboptimal solutions. In contrast, free-path methods optimize the racing line path and velocity simultaneously, generally resulting in shorter lap times. However, increased computation times are required, which depend heavily on the complexity and order of the system dynamics.

1.3.2 Conventional Local Planning

In conventional local planning, a feasible trajectory is planned online in each planning cycle. Compared to global planning, the trajectory starts in the current ego state, and the planning horizon does not cover the entire race track. In multi-vehicle racing, the collision restrictions are based on predictions of the opponent vehicles.

There are various ways to categorize conventional local planning approaches. In this thesis, the approaches are classified into optimization-based, graph-based, incremental, and sampling-based methods. At the end of this subsection, the categories are compared, and hybrid methods that cannot be uniquely categorized are presented.

Optimization-Based Methods

Optimization-based methods (also known as variational methods) are based on the numerical solution of an optimal control problem, i.e., the minimization of a cost functional subject to the vehicle dynamics and other constraints. Usually, gradient-based methods are used to solve the optimization problem, converging to a local minimum of the cost functional. However, due to the non-convex

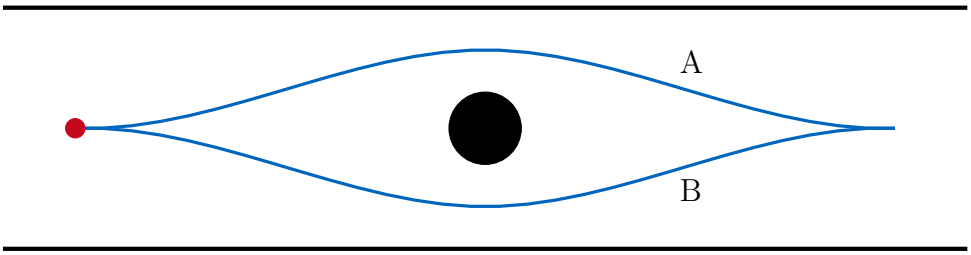


Figure 1.4: Illustration of an optimization-based method, where two different initializations, A and B, lead to distinct local minima. Track boundaries and the obstacle to be avoided are shown in black, and the vehicle position is depicted as a red circle.

nature of the local planning problem, the solution strongly depends on the initialization of the optimization problem, and finding the global minimum is not guaranteed. A simple example is a static obstacle that can be overtaken on the left or right, with both possibilities corresponding to a local minimum, as illustrated in Figure 1.4. Additionally, dynamic opponent vehicles increase the non-convexity, with local minima representing decisions such as overtaking the opponent vehicle or letting it pass. The problem's complexity and the solver's efficiency depend on the nonlinearity and order of the vehicle model, so the choice of vehicle model is a trade-off between accuracy and efficiency.

Many optimization-based methods have been applied to road traffic. In [62], an optimization problem using a kinematic single-track model with speed and acceleration constraints is formulated as part of the DARPA Grand Challenge 2005. Static and slow-moving (quasi-static) obstacles are avoided by reducing the speed limit in these areas, resulting in increased costs for these paths. The planning approach used in [63] for the Bertha Benz Memorial Route [64], in which the 100 km route from Mannheim to Pforzheim was completed, can also account for dynamic obstacles. These are not incorporated using a speed limit, as in [62], but through hard external polygonal constraints. The non-convexity is resolved by determining the overtaking side based on the opponent's driving direction and extending its shape to the opposite track boundary. In [65], a kinematic single-track model is used, but it is assumed that a drivable corridor is given. This eliminates the need to consider obstacles entirely.

For racing applications, advanced vehicle models such as the dynamic single-track model [66, 67] or, under the QSS assumption, a point-mass model constrained with gg-diagrams [68, 69] are used. These approaches either do not

account for any obstacles or only for static ones. In the case of static obstacles, it is assumed that the overtaking side is known in advance. In [69], only the velocity profile is optimized for a given path obtained from the graph-based path planning approach described in [70]. While the mentioned approaches exclusively consider 2D race tracks, a local optimization-based approach for single-vehicle racing on 3D tracks is proposed in [71]. It is based on the laterally flat race track model from [58] and the point-mass model from [60] that is constrained by gg-diagrams dependent on the vehicle's velocity and vertical acceleration.

Graph-Based Methods

In graph-based methods, the state space is discretized using a graph, and the optimal solution is determined within it. A graph consists of nodes, which usually represent discrete points in the state space, and edges linking the nodes. The edges are weighted according to a desired cost functional and checked for feasibility and collision avoidance. The optimal (cost-minimal) sequence of valid edges, starting from a start node to a target node, is searched using graph search algorithms. In contrast to optimization-based methods, the non-convexity is solved by searching the entire state space, so no prior selection of the overtaking side is necessary. However, only the discrete-optimal solution can be found, i.e., the optimal solution within the discrete graph.

The number of nodes increases exponentially with each dimension of the discretized state space. This leads to significantly increased computational effort during the search and is referred to as the *curse of dimensionality*. To keep the computation time tractable, many graph-based approaches use only a spatial graph with no temporal dimension. The velocity profile is subsequently generated based on the resulting discrete-optimal path. While this path-velocity decomposition (PVD) [72] is sufficient for static environments and low velocities, it is too conservative for highly dynamic multi-vehicle racing scenarios since most of the edges in the spatial graph must be blocked to avoid collisions with the predictions. An application for racing using a PVD can be found in [70] as part of the Roborace racing series. Nodes, including spatial information, are arranged in layers perpendicular to a reference path. Cubic polynomial edges connect the nodes of one layer to the nodes of the next layer, resulting in a closed spatial graph around the entire race track. This graph structure is visualized in

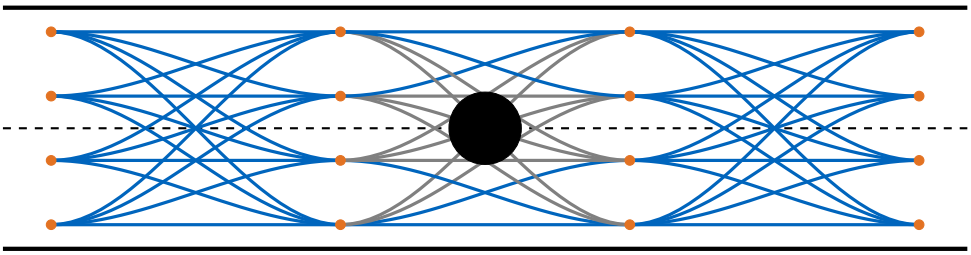


Figure 1.5: Illustration of the spatial graph structure used in the graph-based approach from [70]. The graph consists of four layers perpendicular to the reference path (dashed line), with each layer containing four nodes (orange circles) that are connected by edges (blue and gray lines). Track boundaries and the obstacle to be avoided are shown in black, with colliding edges depicted in gray.

Figure 1.5 for a straight segment of a track. The path obtained from the graph search is completed by a velocity profile generated by the forward-backward solver proposed in [47].

Approaches without PVD include the search in a spatio-temporal graph. However, due to the additional dimensions and the associated computation time demand, only coarse graph discretizations and simple edges are employed. Such approaches are often used for road traffic, as the road network allows for coarse discretizations, and the smoothness requirements for the trajectories are lower due to the reduced speeds. However, no approach with a spatio-temporal graph exists for racing, as driving at the vehicle's handling limits imposes increased requirements on the trajectory. For road traffic, a spatial graph is extended in [73] by discrete values for the time and velocity dimensions. The authors of [74] use a spatial graph that consists of nodes arranged in layers along a reference path, similar to [70]. The edges connecting the layers are cubic polynomials in curvature, based on [75]. The temporal extension, however, is not performed with discrete velocity and time values. Instead, the time and velocity dimensions are divided into intervals, resulting in a time-velocity grid. Edges that end in the same cell of the grid are classified as similar, and all edges except the edge via which the cell can be reached at the lowest cost are discarded. This limits the number of edges, counteracting the curse of dimensionality. The spatio-temporal edges are based on constant acceleration and, therefore, exhibit discontinuities in the acceleration profile. There are also approaches in which the system dynamics are utilized to generate a graph, resulting in

feasible trajectories. For instance, in [76], constant values of the input space are evaluated, linking relative equilibria of the single-track kinematic model.

If a graph exists, the optimal sequence of edges can be searched for, beginning from a start node (ego state) and ending in a goal node (usually a goal position or a minimum horizon). Besides an exhaustive search of the entire graph, there are advanced graph search algorithms that find the solution more efficiently. The best known is the Dijkstra algorithm [77], where the node that can be reached from the start node with the lowest cumulative cost (cost-to-come) is expanded. In this context, expansion refers to the process in which the child nodes of the expanded node are examined according to its outgoing edges. The A* algorithm [78] extends the Dijkstra search by a heuristic function that estimates the remaining costs to the goal node (cost-to-go) for each node. Accordingly, the node with the lowest sum of cost-to-come and the estimated cost-to-go is expanded. Further variants have been developed, such as the Weighted A* algorithm [79] featuring a run time reduction or the D* algorithm [80] for dynamically changing environments.

Incremental Methods

In incremental methods, the vehicle's state or input space is randomly sampled repeatedly. Each sampled state is connected to existing nodes, resulting in a tree-like structure consisting of sampled nodes and edges. Compared to graph-based methods, where the graph structure is fixed, the tree becomes larger or finer with each sample, providing solutions that cannot be found with a fixed graph structure.

Most approaches of this kind are variants of the Rapidly Exploring Random Tree (RRT) originally proposed in [81, 82]. In each step, a point in the state space is sampled, and the nearest node within the tree (neighbor) is selected according to a distance metric. The system's input is then determined such that forward integration of the system dynamics starting from the neighbor ends in a state that minimizes the distance to the sampled point. If the resulting trajectory is valid (collision-free), the reached state is added to the tree as a node, and the process is repeated. An example tree used to search for a path to a goal region is shown in Figure 1.6. The RRT algorithm is probabilistically complete, i.e., the probability of finding a valid solution, if one exists, approaches one as the number of samples increases. However, the authors of [83] show

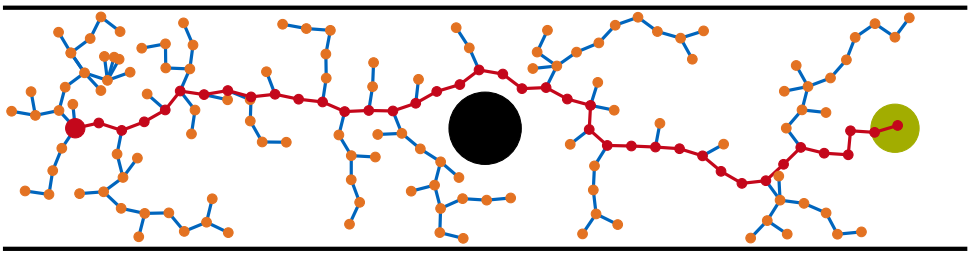


Figure 1.6: RRT for path planning of a holonomic robot, i.e., one that can move in any direction. The tree consists of nodes (orange and red circles) and edges (blue and red lines). Track boundaries and the obstacle to be avoided are shown in black, and the goal region to be reached is shown in green. The resulting solution from the start node (robot position) to the goal node is depicted in red.

that the original RRT algorithm may only converge to a suboptimal solution. Therefore, they propose the RRT* algorithm, which is asymptotically optimal, meaning that it converges to the optimal solution as the number of samples increases. The essential procedure of the RRT is retained, but the edges within the tree can be rewired. Sufficient conditions for asymptotic optimality under differential constraints are given in [84]. Another extension is RRT$^{\mathrm{X}}$ [85], which can handle dynamic environments (e.g., moving obstacles) by rewiring the tree when changes occur.

For road traffic applications, a closed-loop variant of the RRT is used in [86], in which the inputs are not generated for the open-loop system but for the stable closed-loop systems. Using this approach, speeds of up to 25 m/s were achieved in the DARPA Urban Challenge 2007. A planning approach with RRT for several vehicles simultaneously is presented in [87]. Splines are used to smooth the suboptimal RRT solution. In [88], biased sampling perpendicular to a reference path is used to reduce the computation time, whereby speeds of nearly 17 m/s were achieved in a static environment. As a racing application, in [89], RRT is used for path planning. The resulting path is smoothed using parametric cubic splines, and a speed profile is subsequently calculated within the vehicle's dynamic limits. Racing applications using RRT* are given in [90, 91].

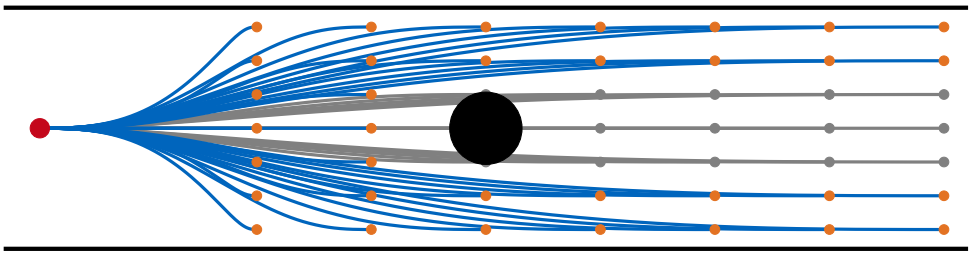


Figure 1.7: Jerk-optimal trajectory candidates (blue and gray lines) based on the approach in [92], connecting the vehicle state (red circle) with deterministically sampled end states (orange and gray circles). Track boundaries and the obstacle to be avoided are shown in black, with colliding trajectories depicted in gray. The solution would be the minimum-cost valid trajectory based on a desired cost functional.

Sampling-Based Methods

In sampling-based methods, no incrementally growing tree is constructed. Instead, they usually sample deterministically in the state space and connect the ego state with the sampled states, producing a set of trajectory candidates. Valid trajectory candidates are weighted according to a cost functional, and the minimum cost trajectory is selected. Although such methods are similar to graph-based methods, the approaches differ in two main aspects. First, the graph is usually precomputed offline in graph-based methods, while in deterministic sampling, the trajectory candidates are generated online. Secondly, in contrast to the graph-based methods, no advanced graph search must be performed since only the trajectory with the minimum cost can be selected.

Many approaches for road traffic and racing are based on the generation of jerk-optimal trajectory candidates, i.e., the minimization of the time derivative of the acceleration. Jerk optimality leads to smooth trajectories that avoid the excitation of high-frequency dynamics neglected in the controller design, thus improving vehicle stability. At the same time, an efficient calculation is possible, as quintic (fifth-order) polynomials represent the jerk-optimal solution [93]. For road traffic, the authors of [92, 94] use jerk-optimal connections in the longitudinal and lateral direction of the vehicle to connect the ego state with the sampled end states. This approach is exemplarily illustrated in Figure 1.7. In [95], snap-optimal[3] trajectories (seventh-order polynomials) are utilized for

[3]Snap denotes the time derivative of the jerk, i.e., the fourth time derivative of the position.

even smoother transitions. The jerk-optimal trajectory candidates are also used in racing applications, for example, by various teams participating in the IAC [31, 32, 96]. However, jerk-optimal trajectories are only suitable for simple oval tracks, as the simple polynomials do not reflect the complexity of the time-optimal racing line to be followed on complex road courses. There are alternative approaches to the generation of trajectory candidates in racing, such as [97, 98], but these have not yet been evaluated on full-scale vehicles.

Comparison and Hybrid Methods

The main difference between optimization-based methods, on one side, and graph-based, incremental, and sampling-based methods, on the other side, is that in the former, no discretization of the state space by a graph, a tree, or trajectory candidates is involved. As a result, the graph-based, incremental, and sampling-based methods can only find the optimal solution within the given discretization.[4] The challenge with these methods is thus to discretize the state space such that it contains solutions as close as possible to the optimal solution. However, the computation time increases with finer discretization and more dimensions. At the same time, non-convexity poses a problem for optimization-based methods since the frequently used gradient-based solvers only converge to a local optimum. In addition, there is less flexibility in the design of the cost functional, as the solver depends on it. In contrast, a more flexible cost design is possible for the graph-based, incremental, and sampling-based methods.

Since the methods have contrasting properties, they are often combined. A common approach is to first solve the non-convexity of the problem using graph-based, incremental, or sampling-based methods. Coarse discretizations are sufficient since the only objective is to find the global optimum without higher accuracy requirements. Subsequently, an optimization-based method initialized with the discrete-optimal solution refines the solution. Applications of such hybrid approaches for road traffic can be found in [99–103]. A variant for multi-vehicle racing is described in [104], where the search in a coarse spatio-temporal graph finds the global optimum, and a numerical optimization improves this solution.

[4] Although the discretization in incremental methods becomes finer over time, it is limited by the finite computation time.

1.3.3 Interaction-Aware Local Planning

Conventional planning methods are based on the prediction of opponent vehicles. Assuming that the opponent vehicles follow their predictions strictly, a trajectory is planned for the ego vehicle that does not collide with the predictions. However, even in simple scenarios, such as merging onto a high-traffic highway, this can lead to situations where no collision-free trajectory is available apart from a standstill, also known as the *freezing robot problem* [105]. This reactive and conservative behavior is particularly undesirable in racing and has led to greater attention on interaction-aware planning approaches in recent years [106]. These approaches eliminate the sequential execution of prediction and planning, allowing the incorporation of interaction between the ego and the opponent vehicles.

As surveyed in [107], numerous supervised learning approaches exist for road traffic based on data sets with real traffic scenarios. Neural networks (NNs) are trained, which, depending on the past trajectories of all participants, output the future trajectory for the desired planning horizon, imitating human behavior. However, such approaches are currently unsuitable for racing, as required large racing data sets are unavailable. Therefore, this subsection does not address supervised learning approaches in detail and instead focuses on the commonly used methods based on game theory and RL.

Game-Theoretic Methods

Game theory deals with decision-making problems (games) that involve several participants (players). The intention of each player is defined by its cost functional, which incorporates the states of the other players. Depending on the cost functional design, this results in a cooperative or competitive game. In contrast to a classic optimal control problem, a simple minimization of the cost functionals is not applicable due to the coupling within them. Instead, equilibrium states are usually determined in which, under certain assumptions, no player intends to change their strategy.

In [108], a two-vehicle non-cooperative racing game is formulated and solved, with particular attention given to the difference between the open-loop and feedback solution. In the open-loop solution, the sequence of actions is predetermined, whereas in the feedback solution, players follow a strategy that enables

them to adjust their actions based on the current state at each stage of the game. The authors of [109] discretize the joint state and input space and use dynamic programming to solve a two-vehicle merging and blocking game. They follow a hierarchical two-step approach consisting of a long-term high-level and a short-term low-level planner for more efficient computation. Despite this two-step approach, the application is limited to only coarse discretizations. A cooperative and a blocking racing game with two vehicles are analyzed in [110]. A set of trajectory candidates is generated for both players, and the costs are calculated for each trajectory combination. Within the resulting cost matrix, different equilibria are analyzed. In [111], a two-vehicle blocking scenario is investigated, in which the ego vehicle in front tries to block the car behind it. As in [110], a set of trajectory candidates is generated for both vehicles, and a corresponding cost matrix is calculated. Instead of searching for an equilibrium within this matrix, different level-K strategies are determined based on level-K reasoning. In this context, a level-K strategy is the optimal strategy when the opponent behaves according to the level-(K-1) strategy, whereas the level-0 strategy does not reflect any interaction. After determining the level-K strategies, the level of the opponent vehicle is estimated, and the ego vehicle's trajectory is chosen optimally according to the level-K reasoning. In [112, 113], a set of trajectory candidates is generated exclusively for the ego vehicle, and the other players are categorized as leaders or followers. The leaders are predicted with a behavior model independent of the ego vehicle. For each trajectory candidate of the ego vehicle, the followers' response is then simulated using a reactive behavior model, and the candidate with the best outcome is selected. An alternative game-theoretic approach follows the iterative best response scheme, which is applied to racing in [114–116]. In this scheme, the trajectory for all players is iteratively replanned while the trajectories of the other vehicles are kept fixed. If this procedure converges, an equilibrium is reached.

RL-Based Methods

RL is a subfield of machine learning and comprises several algorithms that tackle decision-making problems using the trial-and-error principle. The agent (decision instance) interacts with the environment according to a policy and receives feedback in the form of a scalar reward at each time step. The policy maps the state space (information accessible to the agent) to the action space (executable

actions). RL algorithms aim to maximize the cumulative reward over time by adapting the policy. A more detailed, formal description follows in Section 5.1.

RL can learn complex policies in highly interactive and high-dimensional environments, making it a promising approach for interaction-aware planning [43]. While it already outperforms optimal control approaches [117] and reaches the level of human world champions [118] in autonomous drone racing, comparable successes are still pending for full-scale autonomous vehicles. The following overview covers various RL-based approaches for AD on highways or race tracks and emphasizes the different options regarding the action and state space. The methods described here are not limited exclusively to trajectory planning, as many approaches directly select the vehicle's control inputs, thereby combining the planning and control tasks. Additionally, although the reward function fundamentally influences the overall performance, this aspect is not discussed further here, as its choice depends on the individual task. For a detailed discussion of the differences and characteristics in reward design, please refer to [119, 120]. Also, multi-agent RL approaches [121] that simultaneously train multiple agents are not addressed.

There are various options for defining the action space. First, there are *low-level* approaches in which the vehicle's control inputs are selected directly. Most applications focus on highway overtaking [122] or highway merging scenarios [123–125]. There are also extensive studies on single-vehicle racing, which naturally do not include any interaction with other agents. The state space consists of either ego state and track information [126, 127], point cloud data [128, 129], the camera image [130, 131], or combinations of these [132–134]. Applications for multi-vehicle racing are available for the game Gran Turismo Motorsport. In [135], built-in game agents that follow predefined trajectories are overtaken. The state space consists of state information about the ego vehicle, point cloud data, and the upcoming curvature profile of the center line. A curriculum learning scheme is used for more efficient training, i.e., the task's difficulty is increased iteratively from single-vehicle to multi-vehicle racing scenarios. The trained agent outperforms the built-in agent and performs similarly to an experienced human driver. In [136], the opponent vehicles are controlled by the built-in game agent and earlier versions of the RL agent. The state space contains track information about the upcoming section and state information about the ego and opponent vehicles. With this approach, the RL agent outperforms even human

expert drivers. Approaches with a low-level action space are not restricted in the executable maneuvers, as the control inputs are selected directly. However, training may be more complex and time-consuming, as the agent must additionally learn the vehicle dynamics. In addition, a real-world application on full-scale vehicles is safety-critical, as inaccuracies between the training model and the actual vehicle can lead to unsafe behavior, especially at the handling limits.

In contrast to low-level approaches, there are *high-level* approaches in which high-level maneuvers, such as left/straight/right and accelerating/constant speed/braking, define the action space. Based on the selected high-level maneuver, a corresponding underlying trajectory is executed. The real-world application of high-level approaches is less safety-critical, as the actual selection of the control inputs can be performed by a conventional controller tuned to the respective vehicle. However, the definition of a discrete, finite set of maneuvers is only suitable in structured environments. Therefore, high-level approaches are mainly used in highway applications, as in [137–139]. In contrast, they are rarely used in unstructured environments like racing, as representing all necessary maneuvers with a finite action space is challenging. A single simulative racing application is given in [140], where the opponent vehicle follows a fixed trajectory or behaves randomly. The state space contains the relative positions and velocities between the ego and opponent vehicle.

Besides the low-level and high-level approaches, a third option is to generate the trajectory without the additional step of high-level maneuver selection, which is referred to as *direct trajectory generation*. Compared to low-level approaches, they are superior regarding safety, as the generated trajectory can be realized by a conventional controller, as in high-level approaches. At the same time, they are not limited to predefined maneuvers, allowing for a greater diversity of driving maneuvers. The most straightforward option is to select a set of intermediate states as the action space, which together form the trajectory. However, an underlying structure is often used instead to reduce the complexity of the trajectory generation. In [141], a set of jerk-optimal trajectory candidates is generated using the sampling-based approach from [92]. The action space consists of the weighting parameters in the cost functional that is used to determine the optimal trajectory candidate. The authors show that this approach increases the success rate in the examined road traffic scenarios due to the adaptation of the cost functional. However,

the executable maneuvers are limited to the given set of trajectories candidates, which restricts more sophisticated maneuvers that would be desirable in racing scenarios. In [142] and [143], only a single trajectory is generated instead of a set of trajectory candidates. In [142], the reachable gaps on the highway represent the action space, and the selected gap is chosen as the endpoint of the jerk-optimal trajectory. A more flexible trajectory generation is provided in [143], with the endpoint of the jerk-optimal trajectory being specified by the RL agent. However, the feasibility of the trajectory is not guaranteed, and the examined scenario ends unsuccessfully in the case of an infeasible trajectory.

In addition to the three categories described for defining the action space, there are various other options that will not be discussed further. Noteworthy are emerging approaches such as [144, 145], in which the cost parameters of an MPC represent the action space of the RL agent, achieving higher success rates than classical MPC.

Comparison

Compared to conventional approaches, both game-theoretic and RL-based methods are able to incorporate interaction with opposing vehicles into the planning process. However, both approaches make certain assumptions. Game-theoretic approaches assume that all participants behave according to the defined cost functionals and the chosen equilibrium. If the actual intention of the opposing vehicles deviates from this, the success rate may decrease. Meanwhile, in RL-based approaches, the opponent vehicles are assumed to behave as modeled in the training. If the trained agent is evaluated in a different environment with behaviors not included in the training, the success rate may also decrease. However, for RL-based approaches, generalization can be achieved by including different behaviors in the training environment.

With game-theoretic approaches, a complex game must be solved in each cycle, limiting their application to coarse discretizations and usually only two vehicles. RL-based approaches, by contrast, require increased offline computation due to the training process but enable efficient online execution. Another drawback is their lack of explainability and safety, as most approaches involve an NN with no feasibility guarantees. However, RL-based methods scale to multiple vehicles without online computational losses, provided that a suitable state space, such as camera images or point cloud data, is used.

1.4 Research Contributions and Outline

The state of the art presented in Section 1.3 categorizes local planning into different methods, each with distinct strengths and limitations. This thesis focuses on graph-based, sampling-based, and RL-based methods, while optimization-based, game-theoretic, and incremental methods are not covered due to their specific challenges for real-world applications. Optimization-based and game-theoretic methods involve significant computational complexity, making them challenging for real-time planning with limited computational resources. Incremental methods are excluded due to their inherent randomness and the associated highly variable computation times. While these methods are valuable in other contexts, they are discussed as potential directions for future research in the outlook.

The remaining graph-based, sampling-based, and RL-based local planning methods exhibit the following limitations, among others:

Limitation 1 Graph-based local planning approaches suffer from the curse of dimensionality. For multi-vehicle racing, therefore, only approaches based on PVD are used, where an optimal path is first planned with a spatial graph, followed by the velocity profile. However, since the temporal component is not considered in the path planning step, large areas on the race track must be blocked to ensure collision avoidance with the prediction. PVD, therefore, leads to conservative and defensive planning behavior, which is particularly undesirable in competitive racing. Approaches with spatio-temporal graphs exist for road traffic but use coarse discretizations and a simple edge structure for efficient generation. However, this is disadvantageous regarding performance and vehicle stability, especially at the vehicle's handling limits.

Limitation 2 Current local planning methods for multi-vehicle racing assume flat race tracks and neglect 3D effects such as banked turns and elevation changes. Depending on the actual 3D track geometry, this leads to over- or underestimation of the dynamic vehicle limits, resulting in infeasibility issues or reduced performance.

Limitation 3 Many local planning methods for racing are based on the generation of a set of jerk-optimal trajectory candidates following the sampling-based approach from [92]. While this is sufficient for simple race tracks, such as

oval tracks, it is unsuitable for complex road courses since the jerk-optimal trajectories cannot represent the complexity of the time-optimal racing line.

Limitation 4 The division into global and local planning is valid if the ego vehicle follows the offline-generated racing line. However, if the ego vehicle deviates from the racing line, e.g., due to an overtaking maneuver, the racing line loses its validity, as the current vehicle state is not incorporated into the racing line.

Limitation 5 Most interaction-aware RL-based planning approaches use either low-level or high-level action spaces. However, direct trajectory generation would simultaneously lead to high maneuver diversity and potentially a safer real-world vehicle application. While RL-based direct trajectory generation is occasionally employed for road traffic, there exists no application for racing.

Limitation 6 RL-based methods have no safety guarantees due to the NNs involved, so a generated trajectory may be infeasible or not collision-free. In such a case, current approaches terminate the scenario unsuccessfully, preventing a real-world application.

The aim of this thesis is the development of local planning algorithms that address the mentioned limitations and, at the same time, are suitable for real-world application. Based on the limitations, the following contributions are provided:

Contribution 1 Limitation 1 is addressed by proposing a hybrid trajectory planning approach for oval tracks that combines the advantages of sampling-based and graph-based approaches. It utilizes the fact that only the initial segment of the trajectory is actually traversed due to repeated planning according to the RH scheme. While the beginning of the trajectory should, therefore, be sufficiently smooth, the remaining part of the trajectory can be planned more coarsely. Accordingly, a spatio-temporal graph with variable discretization and edge structure is used. The edges, which connect the ego vehicle state with the graph and thus represent the beginning of the trajectory, are generated jerk-optimal and in sufficient numbers by sampling. The remaining edges are generated with a lower discretization and exhibit a constant acceleration profile for efficient calculation. This approach enables a sufficiently long planning horizon on oval tracks while simultaneously ensuring the smoothness of the

traversed part of the trajectory. The graph used is based on the spatial graph from [70], extended with the temporal dimensions according to [74]. The sampling is based on the approach from [92].

Contribution 2 A sampling-based local trajectory planning approach for 3D race tracks is presented that overcomes limitations 2 and 3. Instead of generating jerk-optimal trajectory candidates according to [92], a modified trajectory generation method is proposed, which allows for racing line following on arbitrarily complex road courses. The 3D effects are incorporated in the feasibility checks based on [60] using gg-diagrams dependent on the speed and vertical acceleration. Furthermore, limitation 4 is addressed by examining the impact of replanning the time-optimal racing line from the current vehicle state on the performance of the sampling-based local planning approach. This is compared with the use of a racing line generated offline using global planning approaches.

Contribution 3 The last contribution deals with limitations 5 and 6 in a highly interactive overtaking scenario with an actively blocking opponent vehicle. An RL-based planning approach is proposed, which directly generates a trajectory and solves the scenario with a high success rate, even at high levels of interaction. Concerning limitation 6, a fallback instance is introduced, which intervenes in the event of an infeasible or colliding trajectory. In this case, the feasible and collision-free trajectory that is closest to the initially planned trajectory is selected from a set of trajectory candidates.

The three contributions were not developed independently but rather built on each other over time and were motivated by the participation of the TUM Autonomous Motorsport team in the IAC and A2RL. The team's previous planning approach consisted of graph-based path planning [70] with subsequent velocity generation [47], sufficient for the single-vehicle races at the Roborace competition. However, multi-vehicle racing was only possible to a limited extent for the reasons mentioned above. The hybrid planning approach from contribution 1 thus corresponds to the continuation of the previous approach and was used by the team from 2021 to early 2023 at different oval tracks. The first race on a road course, the Autodromo Nazionale di Monza, took place in June 2023. This necessitated developing the sampling-based approach from contribution 2, as the hybrid planning approach is limited to oval tracks. Since this race, the team has used the sampling-based approach with slight

modifications. The RL-based planning approach from contribution 3 was not tested on a real vehicle and is regarded as a fundamental concept to be further developed.

The remainder of this thesis is structured according to the three contributions. Chapter 2 introduces preliminaries that provide the common foundation of all contributions without addressing the planning process itself. These preliminaries include the race track representation, the vehicle model, and the specification of the planning module interfaces. The actual contributions 1 to 3, consisting of their methodology, the results, and a discussion, follow in Chapters 3 to 5. Chapter 5 additionally contains an introduction to RL theory, which is not relevant to Chapters 3 and 4. Chapter 6 concludes with a summary and an outlook for future research directions.

2 Preliminaries for Planning in Autonomous Racing

This chapter describes the common foundation of the planning approaches in the subsequent chapters. Mathematical notations and conventions are explained in Section 2.1, and the basics of rigid body kinematics are described in Section 2.2. This is followed by the 3D track model in Section 2.3 and the vehicle model in Section 2.4. Lastly, Section 2.5 provides a mathematical description of the local planning module's inputs and outputs, which have already been introduced informally in Figure 1.3.

2.1 Notations and Conventions

The mathematical notations and conventions described here are employed in the remainder of this thesis to ensure a clear and consistent description.

Coordinate Frames

All coordinate frames are right-handed and are indicated with calligraphic symbols, e.g., the inertial frame is denoted as $\mathcal{I}$.

Vectors and Matrices

Vectors and matrices are represented in bold font, while scalars are not bold. Examples are the unit matrix $\boldsymbol{I}$, the zero vector $\boldsymbol{0}$, the basis vector $\boldsymbol{e}$, and, as a scalar, the time t. Further, as primary kinematic quantities, positions are denoted by $\boldsymbol{x}$, velocities by $\boldsymbol{v}$, and accelerations by $\boldsymbol{a}$. The frame in which a vector is expressed is indicated by an upper left index, while the lower right index indicates the quantity to which the vector refers. For instance, ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{AB}}$ denotes the velocity of frame $\mathcal{B}$ relative to the frame $\mathcal{A}$, expressed in the coordinates

Table 2.1: Sets of numbers

Notation	Set	Description
$\mathbb{N}$	$\{1, 2, 3, \dots\}$	Set of natural numbers
$\mathbb{N}_0$	$\{0, 1, 2, 3, \dots\}$	Set of natural numbers including 0
$\mathbb{R}$	$(-\infty, \infty)$	Set of real numbers
$\mathbb{R}_{\geq 0}$	$[0, \infty)$	Set of non-negative real numbers
$\mathbb{R}_{>0}$	$(0, \infty)$	Set of positive real numbers
$\mathbb{R}_{\leq 0}$	$(-\infty, 0]$	Set of non-positive real numbers
$\mathbb{R}_{<0}$	$(-\infty, 0)$	Set of negative real numbers

of $\mathcal{I}$. As another example, ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{A}}$ represents the basis vector in the direction of the x-axis of frame $\mathcal{A}$, again expressed in the coordinates of $\mathcal{I}$.

Mathematical Sets

In this work, the sets of numbers listed in Table 2.1 are used. In the case of vectors, $\mathbb{R}^n$ represents the space of n-dimensional vectors with $n \in \mathbb{N}$, where each element of the vector is a real number. Similarly, $\mathbb{R}^{n \times m}$ represents the set of matrices of dimension $n \times m$ with $n, m \in \mathbb{N}$, where each element of the matrix is a real number.

Differentiation

The derivative of a scalar function $f(t) \in \mathbb{R}$ with respect to the time $t \in \mathbb{R}$ is denoted as $\dot{f}(t) = \frac{\mathrm{d}f(t)}{\mathrm{d}t}$. Similarly, the derivative of $f(s) \in \mathbb{R}$ with respect to the progress $s \in \mathbb{R}$ is denoted as $f'(s) = \frac{\mathrm{d}f(s)}{\mathrm{d}s}$, where s is introduced in Section 2.3.

Further, for the function $f(\boldsymbol{x}) \in \mathbb{R}$ with $\boldsymbol{x} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\top \in \mathbb{R}^n$, where $n \in \mathbb{N}$, the gradient is abbreviated as

$$\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} & \frac{\partial f(\boldsymbol{x})}{\partial x_2} & \dots & \frac{\partial f(\boldsymbol{x})}{\partial x_n} \end{bmatrix}^\top . \tag{2.1}$$

In the context of RL, however, the gradient is denoted using the nabla operator, $\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}$, as this is the standard notation in the corresponding literature. While mathematically equivalent, both notations are used contextually in this thesis to align with the conventions of their respective fields.

Trigonometric Functions

For the sake of brevity, trigonometric functions are sometimes abbreviated as $\sin(\square) = \mathrm{s}_\square$ and $\cos(\square) = \mathrm{c}_\square$.

2.2 Fundamentals of Rigid Body Kinematics

This section is based on the derivations from [58] and [146]. It begins with a brief review of skew-symmetric matrices and the cross product in Subsection 2.2.1, followed by an introduction to rotation matrices in Subsection 2.2.2.

2.2.1 Skew-Symmetric Matrices and Cross Products

A square matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ of order $n \in \mathbb{N}$ is called skew-symmetric if

$$\boldsymbol{A}^\top = -\boldsymbol{A}. \tag{2.2}$$

For the special case of $n = 3$, any skew-symmetric matrix $\boldsymbol{A}$ can be parameterized by a single vector $\boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^\top \in \mathbb{R}^3$ with

$$\boldsymbol{A} = \boldsymbol{S}(\boldsymbol{a}) \coloneqq \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \tag{2.3}$$

The representation in (2.3) can be used to express the cross product of two vectors $\boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^\top \in \mathbb{R}^3$ and $\boldsymbol{b} = \begin{bmatrix} b_1 & b_2 & b_3 \end{bmatrix}^\top \in \mathbb{R}^3$ as a matrix multiplication:

$$\boldsymbol{a} \times \boldsymbol{b} = \boldsymbol{S}(\boldsymbol{a})\, \boldsymbol{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}. \tag{2.4}$$

The vector resulting from the cross product $\boldsymbol{a} \times \boldsymbol{b}$ is orthogonal to $\boldsymbol{a}$ and $\boldsymbol{b}$ following the right-hand rule.

2.2.2 Rotation Matrices

A three-dimensional rotation matrix $\boldsymbol{R} \in SO(3) \subset \mathbb{R}^{3\times3}$ is an orthogonal square matrix with a determinant of 1, i.e., it belongs to the special orthogonal group defined by

$$SO(3) = \left\{ \boldsymbol{R} \in \mathbb{R}^{3\times3} \mid \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I},\ \det(\boldsymbol{R}) = 1 \right\}. \tag{2.5}$$

Basic Properties

The cross product is invariant under rotation matrices [146], i.e, for two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^3$, it holds that

$$(\boldsymbol{R}\boldsymbol{a}) \times (\boldsymbol{R}\boldsymbol{b}) = \boldsymbol{R}(\boldsymbol{a} \times \boldsymbol{b}). \tag{2.6}$$

This can be rewritten using (2.4) as

$$\boldsymbol{S}(\boldsymbol{R}\boldsymbol{a})\boldsymbol{R}\boldsymbol{b} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b}. \tag{2.7}$$

Therefore, for an arbitrary vector $\boldsymbol{b}$, the following relation applies:

$$\boldsymbol{S}(\boldsymbol{R}\boldsymbol{a}) = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{a})\boldsymbol{R}^\top. \tag{2.8}$$

Further, a time-varying rotation matrix $\boldsymbol{R}(t) \in SO(3)$ is considered, which changes dependent on the time $t \in \mathbb{R}_{\geq 0}$. As introduced in Section 2.1, all derivatives with respect to t are denoted as $\frac{\mathrm{d}\square}{\mathrm{d}t} = \dot{\square}$. Due to the orthogonality property of rotation matrices, it follows that

$$\frac{\mathrm{d}\boldsymbol{R}(t)\boldsymbol{R}^\top(t)}{\mathrm{d}t} = \dot{\boldsymbol{R}}(t)\boldsymbol{R}^\top(t) + \left(\dot{\boldsymbol{R}}(t)\boldsymbol{R}^\top(t)\right)^\top = \boldsymbol{0}. \tag{2.9}$$

By rearranging (2.9) into the form of (2.2), it becomes evident that $\dot{\boldsymbol{R}}(t)\boldsymbol{R}^\top(t)$ is skew-symmetric. Similarly, applying the same procedure with reversed matrix order, it can be concluded that $\boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t)$ is also skew-symmetric:

$$\frac{\mathrm{d}\boldsymbol{R}^\top(t)\boldsymbol{R}(t)}{\mathrm{d}t} = \left(\boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t)\right)^\top + \boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t) = \boldsymbol{0}. \tag{2.10}$$

Kinematic Relations

Consider two right-handed coordinate frames, denoted with $\mathcal{I}$ and $\mathcal{B}$, sharing the same origin. While the inertial frame $\mathcal{I}$ is fixed, the moving frame $\mathcal{B}$ rotates relative to the inertial frame. The basis vectors ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}(t), {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}}(t), {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}}(t) \in \mathbb{R}^3$ of frame $\mathcal{B}$ expressed in the inertial frame $\mathcal{I}$ form together the rotation matrix $\boldsymbol{R}_{\mathcal{I}\mathcal{B}}(t) = \begin{bmatrix} {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}(t) & {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}}(t) & {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}}(t) \end{bmatrix} \in SO(3)$. If the position of a particle expressed in frame $\mathcal{B}$ is now given by ${}^{\mathcal{B}}\boldsymbol{x}(t) = \begin{bmatrix} {}^{\mathcal{B}}x_{\mathrm{x}}(t) & {}^{\mathcal{B}}x_{\mathrm{y}}(t) & {}^{\mathcal{B}}x_{\mathrm{z}}(t) \end{bmatrix}^{\top} \in \mathbb{R}^3$, its coordinates relative to the inertial frame $\mathcal{I}$ can be computed using the rotation matrix:

$$ {}^{\mathcal{I}}\boldsymbol{x}(t) = {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}(t)\, {}^{\mathcal{B}}x_{\mathrm{x}}(t) + {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}}(t)\, {}^{\mathcal{B}}x_{\mathrm{y}}(t) + {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}}(t)\, {}^{\mathcal{B}}x_{\mathrm{z}}(t) = \boldsymbol{R}_{\mathcal{I}\mathcal{B}}(t)\, {}^{\mathcal{B}}\boldsymbol{x}(t). \tag{2.11} $$

Since rotation matrices are orthogonal, the converse relation can be stated as

$$ {}^{\mathcal{B}}\boldsymbol{x}(t) = \boldsymbol{R}_{\mathcal{I}\mathcal{B}}^{\top}(t)\, {}^{\mathcal{I}}\boldsymbol{x}(t) = \boldsymbol{R}_{\mathcal{B}\mathcal{I}}(t)\, {}^{\mathcal{I}}\boldsymbol{x}(t). \tag{2.12} $$

From (2.11), it becomes evident that even if there is no movement relative to frame $\mathcal{B}$, i.e., ${}^{\mathcal{B}}\boldsymbol{x}(t)$ is constant, a movement still exists relative to frame $\mathcal{I}$ due to the rotation. For this specific case, the velocity of the particle ${}^{\mathcal{I}}\boldsymbol{v}(t) \in \mathbb{R}^3$ expressed in frame $\mathcal{I}$ can be derived as follows (subsequently, the index $\mathcal{I}\mathcal{B}$ is omitted for the rotation matrix):

$$ {}^{\mathcal{I}}\boldsymbol{v}(t) = {}^{\mathcal{I}}\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{R}}(t)\, {}^{\mathcal{B}}\boldsymbol{x}(t) \overset{(2.12)}{=} \dot{\boldsymbol{R}}(t)\boldsymbol{R}^{\top}(t)\, {}^{\mathcal{I}}\boldsymbol{x}(t). \tag{2.13} $$

As shown using (2.9), the matrix $\dot{\boldsymbol{R}}(t)\boldsymbol{R}^{\top}(t)$ is skew-symmetric and can be parameterized according to (2.3) by a vector of dimension 3. This vector is defined as the angular velocity ${}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{I}\mathcal{B}}(t) \in \mathbb{R}^3$ of the rotating frame $\mathcal{B}$ relative to $\mathcal{I}$ and, as shown later, it is expressed in the inertial frame $\mathcal{I}$:

$$ \boldsymbol{S}\left({}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{I}\mathcal{B}}(t)\right) = \dot{\boldsymbol{R}}(t)\boldsymbol{R}^{\top}(t). \tag{2.14} $$

Using (2.4) and (2.14), the relationship between particle velocity and position from (2.13) can be expressed as a cross product:

$$ {}^{\mathcal{I}}\boldsymbol{v}(t) = {}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{I}\mathcal{B}}(t) \times {}^{\mathcal{I}}\boldsymbol{x}(t). \tag{2.15} $$

Since the cross product inherently produces a vector ${}^{\mathcal{I}}\boldsymbol{v}(t)$ perpendicular to ${}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}(t)$ and ${}^{\mathcal{I}}\boldsymbol{x}(t)$, ${}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}(t)$ must be perpendicular to the plane in which the particle rotates. Consequently, ${}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}(t)$ corresponds to the axis of rotation around which frame $\mathcal{B}$ rotates relative to $\mathcal{I}$. Additionally, the magnitude $\left\|{}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}(t)\right\|_2$ corresponds to the angular rate of rotation.

Further, (2.13) can be extended by multiplying from the left with $\boldsymbol{I} = \boldsymbol{R}(t)\boldsymbol{R}^\top(t)$ to obtain

$$ {}^{\mathcal{I}}\boldsymbol{v}(t) = \boldsymbol{R}(t)\left(\boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t)\right)\boldsymbol{R}^\top(t)\,{}^{\mathcal{I}}\boldsymbol{x}(t). \tag{2.16} $$

As shown using (2.10), also the matrix $\boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t)$ is skew-symmetric and can analogously to (2.14) be parameterized using the vector ${}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}(t)$, containing the angular velocity of the rotating frame $\mathcal{B}$ expressed in itself:

$$ \boldsymbol{S}\left({}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}(t)\right) = \boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t). \tag{2.17} $$

Finally, substituting (2.17) in (2.16) and using the relation (2.8) yields

$$ {}^{\mathcal{I}}\boldsymbol{v}(t) = \boldsymbol{S}\left(\boldsymbol{R}(t)\,{}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}(t)\right){}^{\mathcal{I}}\boldsymbol{x}(t). \tag{2.18} $$

Comparing (2.18) with (2.14) substituted in (2.13), it becomes evident that the parameterizations of the matrices $\dot{\boldsymbol{R}}(t)\boldsymbol{R}^\top(t)$ and $\boldsymbol{R}^\top(t)\dot{\boldsymbol{R}}(t)$ are indeed expressed in the $\mathcal{I}$ and $\mathcal{B}$ frames, respectively, and are correspondingly related to each other:

$$ {}^{\mathcal{I}}\boldsymbol{\omega}_{\mathcal{IB}}(t) = \boldsymbol{R}(t)\,{}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}(t). \tag{2.19} $$

Euler Angles

Due to the constraints imposed by the orthogonality property and the determinant being 1, a rotation matrix $\boldsymbol{R} \in SO(3)$ can be locally parameterized with only three parameters. One possibility for this parameterization is the use of Euler angles, which decompose the whole rotation into three elementary rotations, $\boldsymbol{R}_{\mathrm{x}}, \boldsymbol{R}_{\mathrm{y}}, \boldsymbol{R}_{\mathrm{z}} \in SO(3)$, around the x-, y-, and z-axes:

$$ \boldsymbol{R}_{\mathrm{x}}(\varphi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \mathrm{c}_\varphi & -\mathrm{s}_\varphi \\ 0 & \mathrm{s}_\varphi & \mathrm{c}_\varphi \end{bmatrix}, \; \boldsymbol{R}_{\mathrm{y}}(\mu) = \begin{bmatrix} \mathrm{c}_\mu & 0 & \mathrm{s}_\mu \\ 0 & 1 & 0 \\ -\mathrm{s}_\mu & 0 & \mathrm{c}_\mu \end{bmatrix}, \; \boldsymbol{R}_{\mathrm{z}}(\theta) = \begin{bmatrix} \mathrm{c}_\theta & -\mathrm{s}_\theta & 0 \\ \mathrm{s}_\theta & \mathrm{c}_\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.20} $$

Although different conventions can be used to order the elementary rotations, the focus here is on the intrinsic ZYX Euler angles convention. In this convention, the rotation is first performed around the z-axis by angle $\theta \in \mathbb{R}$, then around the resulting y-axis by angle $\mu \in \mathbb{R}$, and finally around the resulting x-axis by angle $\varphi \in \mathbb{R}$, yielding the following composition:

$$\boldsymbol{R}(\theta, \mu, \varphi) = \boldsymbol{R}_{\mathrm{z}}(\theta)\boldsymbol{R}_{\mathrm{y}}(\mu)\boldsymbol{R}_{\mathrm{x}}(\varphi) \tag{2.21a}$$

$$= \begin{bmatrix} \mathrm{c}_\theta \mathrm{c}_\mu & \mathrm{c}_\theta \mathrm{s}_\mu \mathrm{s}_\varphi - \mathrm{s}_\theta \mathrm{c}_\varphi & \mathrm{c}_\theta \mathrm{s}_\mu \mathrm{c}_\varphi + \mathrm{s}_\theta \mathrm{s}_\varphi \\ \mathrm{s}_\theta \mathrm{c}_\mu & \mathrm{s}_\theta \mathrm{s}_\mu \mathrm{s}_\varphi + \mathrm{c}_\theta \mathrm{c}_\varphi & \mathrm{s}_\theta \mathrm{s}_\mu \mathrm{c}_\varphi - \mathrm{c}_\theta \mathrm{s}_\varphi \\ -\mathrm{s}_\mu & \mathrm{c}_\mu \mathrm{s}_\varphi & \mathrm{c}_\mu \mathrm{c}_\varphi \end{bmatrix}. \tag{2.21b}$$

If, again, a time-varying rotation matrix $\boldsymbol{R}(t) = \boldsymbol{R}_{\mathrm{z}}(\theta(t))\,\boldsymbol{R}_{\mathrm{y}}(\mu(t))\,\boldsymbol{R}_{\mathrm{x}}(\varphi(t)) \in SO(3)$ is considered, the angular velocity ${}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}(t)$ can be expressed in terms of the change of the Euler angles $\dot{\varphi}(t)$, $\dot{\mu}(t)$, and $\dot{\theta}(t)$ using (2.17). When omitting the arguments, the relation can be derived as follows:

$$\boldsymbol{S}\left({}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{IB}}\right) = \boldsymbol{R}^\top \dot{\boldsymbol{R}} \tag{2.22a}$$

$$\overset{(2.21a)}{=} \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top \boldsymbol{R}_{\mathrm{z}}^\top \left(\dot{\boldsymbol{R}}_{\mathrm{z}}\boldsymbol{R}_{\mathrm{y}}\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{R}_{\mathrm{z}}\dot{\boldsymbol{R}}_{\mathrm{y}}\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{R}_{\mathrm{z}}\boldsymbol{R}_{\mathrm{y}}\dot{\boldsymbol{R}}_{\mathrm{x}}\right) \tag{2.22b}$$

$$= \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top \boldsymbol{R}_{\mathrm{z}}^\top \dot{\boldsymbol{R}}_{\mathrm{z}}\boldsymbol{R}_{\mathrm{y}}\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top \dot{\boldsymbol{R}}_{\mathrm{y}}\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{R}_{\mathrm{x}}^\top \dot{\boldsymbol{R}}_{\mathrm{x}} \tag{2.22c}$$

$$\overset{(2.17)}{=} \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top \boldsymbol{S}\left(\dot{\theta}\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}}\right)\boldsymbol{R}_{\mathrm{y}}\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{S}\left(\dot{\mu}\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}}\right)\boldsymbol{R}_{\mathrm{x}} + \boldsymbol{S}\left(\dot{\varphi}\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}\right) \tag{2.22d}$$

$$\overset{(2.8)}{=} \boldsymbol{S}\left(\dot{\theta} \cdot \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}}\right) + \boldsymbol{S}\left(\dot{\mu} \cdot \boldsymbol{R}_{\mathrm{x}}^\top\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}}\right) + \boldsymbol{S}\left(\dot{\varphi} \cdot {}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}\right) \tag{2.22e}$$

$$= \boldsymbol{S}\left(\dot{\theta} \cdot \boldsymbol{R}_{\mathrm{x}}^\top \boldsymbol{R}_{\mathrm{y}}^\top\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{z},\mathcal{B}} + \dot{\mu} \cdot \boldsymbol{R}_{\mathrm{x}}^\top\,{}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{y},\mathcal{B}} + \dot{\varphi} \cdot {}^{\mathcal{B}}\boldsymbol{e}_{\mathrm{x},\mathcal{B}}\right). \tag{2.22f}$$

2.3 Track Representation

A laterally flat 3D track is used to describe the race track geometry, meaning that variations along the cross section of the track are not modeled. The formal analytical description is provided in Subsection 2.3.1, while the numerical modeling is detailed in Subsection 2.3.2.

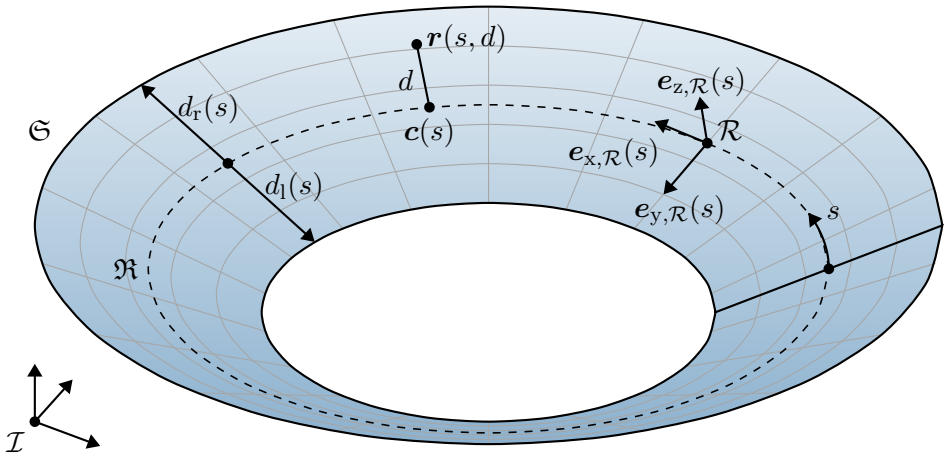


Figure 2.1: Illustration of a closed, counterclockwise orientated 3D race track with negative banking ($\varphi < 0°$) and without slope ($\mu = 0°$).

2.3.1 Analytical Model

For the description of a 3D race track, the representation introduced in [58] is used, which is based on a spine and a lateral offset to it. The spine, hereafter referred to as the reference line, is a 3D curve

$$\mathfrak{R} = \left\{ {}^{\mathcal{I}}\boldsymbol{c}(s) \in \mathbb{R}^3 \mid s \in [0, s_\mathrm{f}] \right\} \tag{2.23}$$

of length $s_\mathrm{f} \in \mathbb{R}_{>0}$, parameterized with respect to its arc length s. As introduced in Section 2.1, all derivatives with respect to this arc length are denoted as $\frac{\mathrm{d}\square(s)}{\mathrm{d}s} = \square'(s)$. Since closed race tracks are specifically considered, ${}^{\mathcal{I}}\boldsymbol{c}(0) = {}^{\mathcal{I}}\boldsymbol{c}(s_\mathrm{f})$ applies. A moving coordinate frame associated with the reference line is now introduced, as depicted in Figure 2.1. This frame, referred to as the road frame $\mathcal{R}$, is spanned by the three basis vectors ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}(s), {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s), {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}(s) \in \mathbb{R}^3$ and has its origin ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}\mathcal{R}}(s) \in \mathbb{R}^3$ on the reference line, i.e., ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}\mathcal{R}}(s) = {}^{\mathcal{I}}\boldsymbol{c}(s)$. The basis vector ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}(s) = {}^{\mathcal{I}}\boldsymbol{c}'(s)$ is tangential to the reference line $\mathfrak{R}$. The y-axis ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s)$ is orthogonal to ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}(s)$ and lies in the plane of the race track. The road frame is completed with ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}(s) = {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}(s) \times {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s)$, which thus points orthogonally out from the track surface. The orientation of the road frame $\mathcal{R}$ relative to the inertial frame $\mathcal{I}$ is described by the rotation matrix $\boldsymbol{R}_{\mathcal{I}\mathcal{R}}(s) = \begin{bmatrix} {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}(s) & {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s) & {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}(s) \end{bmatrix} \in SO(3)$. It is parameter-

ized using the intrinsic ZYX Euler angles convention introduced in Subsection 2.2.2:

- The orientation angle $\theta(s)$ corresponds to the rotation around the z-axis and specifies the orientation direction of the track from a bird's eye view.
- The slope angle $\mu(s)$ corresponds to the rotation around the resulting y-axis and indicates the inclination of the track, i.e., whether it is an uphill ($\mu < 0°$) or downhill ($\mu > 0°$) section.
- The banking angle $\varphi(s)$ corresponds to the rotation around the resulting x-axis and indicates the banking, i.e., the tilting of the track to the left ($\varphi < 0°$) or right ($\varphi > 0°$).

The change in orientation of the road frame $\mathcal{R}$ relative to $\mathcal{I}$ is described by the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s) = \begin{bmatrix}\Omega_{\mathrm{x}}(s) & \Omega_{\mathrm{y}}(s) & \Omega_{\mathrm{z}}(s)\end{bmatrix}^{\top} \in \mathbb{R}^3$. While the angular velocity ${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}}(t) \in \mathbb{R}^3$ (introduced in Subsection 2.2.2 and further described in Section 2.4) indicates the change in orientation with respect to the time t, the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s)$ corresponds to the change in orientation with respect to the arc length s and is, therefore, not explicitly dependent on time. The time-independent track geometry, taking into account all 3D effects, is therefore described by the three elements of ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s)$:

- $\Omega_{\mathrm{z}}(s)$ is the geodesic curvature, which describes the curvature around the z-axis of $\mathcal{R}$. It is the only curvature that is also present on a flat track and indicates how tight a turn is.
- $\Omega_{\mathrm{y}}(s)$ is the normal curvature, which describes the curvature around the y-axis of $\mathcal{R}$. This curvature is effective when driving over a hill or through a dip.
- $\Omega_{\mathrm{x}}(s)$ is the geodesic (or relative) torsion, which describes the curvature around the x-axis of $\mathcal{R}$. For instance, it indicates the curvature that is caused by the angular change within a banked turn.

Analogous to the angular velocity in (2.22f), the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s)$ can be expressed using the Euler angles' derivatives with respect to s:

$$
\begin{aligned}
{}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s) &= \begin{bmatrix} {}^{\mathcal{R}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}} & \boldsymbol{R}_{\mathrm{x}}^{\top}(\varphi(s))\,{}^{\mathcal{R}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}} & \boldsymbol{R}_{\mathrm{x}}^{\top}(\varphi(s))\boldsymbol{R}_{\mathrm{y}}^{\top}(\mu(s))\,{}^{\mathcal{R}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}} \end{bmatrix} \begin{bmatrix} \varphi'(s) \\ \mu'(s) \\ \theta'(s) \end{bmatrix} && \text{(2.24a)} \\
&\overset{(2.20)}{=} \begin{bmatrix} 1 & 0 & -\sin(\mu(s)) \\ 0 & \cos(\varphi(s)) & \cos(\mu(s)) \cdot \sin(\varphi(s)) \\ 0 & -\sin(\varphi(s)) & \cos(\mu(s)) \cdot \cos(\varphi(s)) \end{bmatrix} \begin{bmatrix} \varphi'(s) \\ \mu'(s) \\ \theta'(s) \end{bmatrix}. && \text{(2.24b)}
\end{aligned}
$$

Given the width between the reference line and the track boundaries, denoted as $d_{\mathrm{l}}(s) \in \mathbb{R}_{>0}$ and $d_{\mathrm{r}}(s) \in \mathbb{R}_{<0}$ for the left and right sides, respectively, the track surface $\mathfrak{S}$ can now be defined as

$$
\begin{aligned}
\mathfrak{S} = \Big\{ {}^{\mathcal{I}}\boldsymbol{r}(s,d) = {}^{\mathcal{I}}\boldsymbol{c}(s) + {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s) \cdot d \in \mathbb{R}^3 \\
\mid s \in [0, s_{\mathrm{f}}],\, d \in [d_{\mathrm{r}}(s), d_{\mathrm{l}}(s)] \Big\}.
\end{aligned} \tag{2.25}
$$

While s represents the progress along the reference line, d corresponds to the lateral offset on the race track's plane from the reference line.

A 2D track results as a special case of the 3D model described here. For this, $\varphi(s) = \mu(s) = 0°$ must apply for $s \in [0, s_{\mathrm{f}}]$, resulting in $\Omega_{\mathrm{x}}(s) = \Omega_{\mathrm{y}}(s) = 0\,\mathrm{rad/m}$ according to (2.24b). Consequently, only the orientation angle $\theta(s)$ and the geodesic curvature $\Omega_{\mathrm{z}}(s)$ are effective.

2.3.2 Numerical Model

It is assumed that discrete points of the left ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k] \in \mathbb{R}^3$ and right ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k] \in \mathbb{R}^3$ track boundary of equal number $N_{\mathrm{map}} \in \mathbb{N}$ are available, where $k \in \{0, 1, \ldots, N_{\mathrm{map}}\}$. The data points are identical for $k = 0$ and $k = N_{\mathrm{map}}$, resulting in a closed track. Furthermore, let the discrete points of the reference line ${}^{\mathcal{I}}\boldsymbol{c}[k]$ be given, lying on the respective line connecting ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k]$ and ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k]$.[1] The discrete arc lengths are obtained from the cumulative Euclidean distance

[1] If the reference line is not specified, a possible line can be determined as ${}^{\mathcal{I}}\boldsymbol{c}[k] = 1/2({}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k] + {}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k])$, ensuring that ${}^{\mathcal{I}}\boldsymbol{c}[k]$ lies on the respective line connecting ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k]$ and ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k]$. However, the point pairs must be sufficiently close to ensure that the reference line remains within the track boundaries.

between adjacent points of the reference line:

$$s[k] = \begin{cases} 0 & \text{for } k = 0, \\ s[k-1] + \left\| {}^{\mathcal{I}}\boldsymbol{c}[k] - {}^{\mathcal{I}}\boldsymbol{c}[k-1] \right\|_2 & \text{for } k \in \{1, 2, \ldots, N_{\mathrm{map}}\}. \end{cases} \tag{2.26}$$

The basis vectors ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}[k]$, ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}[k]$, and ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}[k]$ at the discrete points are determined similarly to [58] based on forward finite differentiation:

$${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}[k] = \begin{cases} \frac{{}^{\mathcal{I}}\boldsymbol{c}[k+1] - {}^{\mathcal{I}}\boldsymbol{c}[k]}{\| {}^{\mathcal{I}}\boldsymbol{c}[k+1] - {}^{\mathcal{I}}\boldsymbol{c}[k] \|_2} & \text{for } k \in \{0, 1, \ldots, N_{\mathrm{map}} - 1\}, \\ {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}[0] & \text{for } k = N_{\mathrm{map}}, \end{cases} \tag{2.27}$$

$${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}[k] = \frac{{}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k] - {}^{\mathcal{I}}\boldsymbol{c}[k]}{\left\| {}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k] - {}^{\mathcal{I}}\boldsymbol{c}[k] \right\|_2} \qquad \text{for } k \in \{0, 1, \ldots, N_{\mathrm{map}}\}, \tag{2.28}$$

$${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}[k] = {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}[k] \times {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}[k] \qquad \text{for } k \in \{0, 1, \ldots, N_{\mathrm{map}}\}. \tag{2.29}$$

The rotation matrices $\boldsymbol{R}_{\mathcal{I}\mathcal{R}}[k] = \left[{}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{x},\mathcal{R}}[k] \;\; {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}[k] \;\; {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{z},\mathcal{R}}[k] \right]$ can be constructed using the basis vectors. The corresponding Euler angles $\varphi[k]$, $\mu[k]$, and $\theta[k]$ can be computed by comparison of $\boldsymbol{R}_{\mathcal{I}\mathcal{R}}[k]$ with (2.21b). The derivatives of the Euler angles, which are needed in (2.24b) for the calculation of the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{I}\mathcal{R}}[k]$, are again obtained by forward finite differentiation:

$$\xi'[k] = \begin{cases} \frac{\xi[k+1] - \xi[k]}{s[k+1] - s[k]} & \text{for } k \in \{0, 1, \ldots, N_{\mathrm{map}} - 1\} \\ \xi'[0] & \text{for } k = N_{\mathrm{map}} \end{cases} \quad \text{with} \quad \xi \in \{\varphi, \mu, \theta\}. \tag{2.30}$$

Similarly to (2.30), the derivative of the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}'_{\mathcal{I}\mathcal{R}}[k]$ is computed using finite differencing. Finally, the track widths are obtained as:

$$d_{\mathrm{l}}[k] = \left\| {}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k] - {}^{\mathcal{I}}\boldsymbol{c}[k] \right\|_2 \text{ and } d_{\mathrm{r}}[k] = -\left\| {}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k] - {}^{\mathcal{I}}\boldsymbol{c}[k] \right\|_2. \tag{2.31}$$

Due to potentially noisy track boundary data ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{l}}[k]$ and ${}^{\mathcal{I}}\boldsymbol{b}_{\mathrm{r}}[k]$, the track smoothing procedure proposed in [58] is applied. It is based on an optimization scheme that reduces the noise in the data while preserving track closure.

2.4 Vehicle Model

In the literature, models of varying accuracy exist to describe the dynamics of a vehicle. On the one hand, these include simple models such as point-mass and single-track models, characterized by their simplicity and computational efficiency. On the other hand, higher-fidelity two-track and multi-body models involve fewer assumptions and, therefore, better represent the actual vehicle behavior. A detailed introduction to the various models can be found in existing literature, such as [147].

A further division is made into kinematic and dynamic vehicle models. Kinematic vehicle models only incorporate the kinematic relations of the vehicle quantities, such as position, velocity, and acceleration. Since these models do not explicitly account for the forces that cause acceleration, the accelerations are not inherently limited, potentially resulting in infeasibly high values. To ensure physical realism, it is thus necessary to impose external constraints on the acceleration. Dynamic vehicle models, in contrast, consider the forces acting on the vehicle, e.g., by establishing the balances of forces and moments. In addition to forces such as drag force and aerodynamic downforces, the forces acting on the tires are decisive for the vehicle's behavior. The lateral and longitudinal tire forces primarily depend on the lateral and longitudinal slip and the vertical wheel load. Various complex and accurate tire models exist for calculating tire forces based on these quantities, including linear models or the empirically determined Pacejka tire model (Magic Formula) [148].

Within this thesis, a point-mass model introduced in Subsection 2.4.1 is used for trajectory planning due to its computational efficiency. Since it is a kinematic vehicle model, constraining the vehicle's acceleration is essential. For this, gg-diagrams, which are introduced in Subsection 2.4.2, are used. These diagrams are generated with a higher-fidelity model, allowing the complex vehicle behavior to be captured despite the simplicity of the used point-mass model. For a more realistic behavior, this section also introduces a maximum curvature allowed to be driven by the point mass.

2.4.1 Point-Mass Model

The point-mass model is frequently used for planning and is characterized by its simplicity. It is based on a point mass that lies in the vehicle's center of gravity

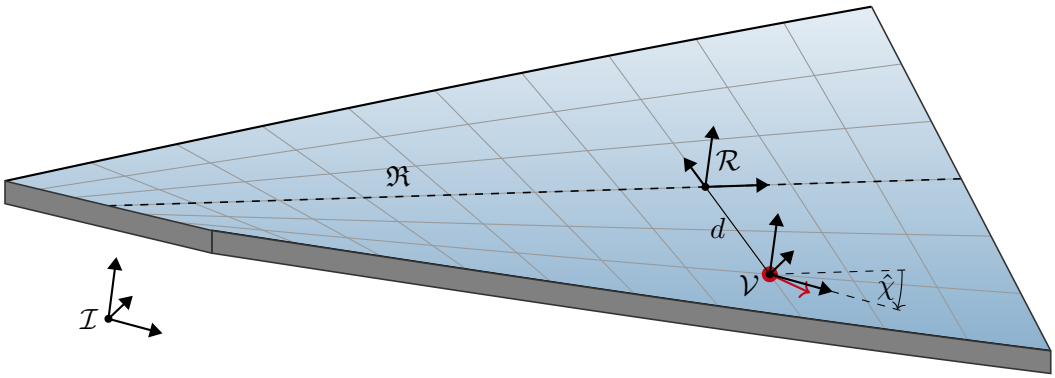


Figure 2.2: Inertial $\mathcal{I}$, road $\mathcal{R}$, and velocity $\mathcal{V}$ frames illustrated on a straight twisting track segment with $\Omega_x > 0\,\mathrm{rad/m}$. The point mass (red circle) lies in the origin of $\mathcal{V}$. Due to the twisting of the track, its total velocity (red arrow) is not fully aligned with the x-axis of $\mathcal{V}$, leading to a vertical velocity component in the velocity frame.

projected onto the laterally flat 3D track. This section introduces the relevant coordinate frames of the point-mass model and its physical quantities. However, the ordinary differential equations that describe its system dynamics are not derived, as they are not needed in the planning approaches presented in this thesis. For a derivation of the system dynamics, please refer to [60] and [61], on which the following descriptions are based.

Subsection 2.3.1 introduces the road frame $\mathcal{R}$ to represent the race track. It is a frame that is oriented with the reference line $\mathfrak{R}$ at the corresponding track progress s. However, no temporal processes that are required to describe the vehicle's movement have yet been defined. This is now accounted for by moving the road frame $\mathcal{R}$ along the reference line so that the point mass always lies on the normal vector ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}\left(s(t)\right)$. The relation between $\mathcal{R}$ and the point mass is shown exemplarily in Figure 2.2, with the point mass lying in the origin of the velocity frame $\mathcal{V}$, which is specified in more detail later. This definition of the road frame allows the position ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}\mathcal{V}}(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}^{\top} \in \mathbb{R}^3$ of the point mass (or the origin of $\mathcal{V}$) to be determined via the reference line ${}^{\mathcal{I}}\boldsymbol{c}(s(t))$ at progress $s(t)$ and a lateral offset $d(t)$ along the normal vector of $\mathcal{R}$:

$$ {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{I}\mathcal{V}}\left(s(t), d(t)\right) = {}^{\mathcal{I}}\boldsymbol{c}\left(s(t)\right) + {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}\left(s(t)\right) \cdot d(t). \tag{2.32} $$

Due to the motion of the point mass on the track, the road frame moves along the reference line with the speed $\dot{s}(t)$. Consequently, the angular velocity ${}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}}(t)$ of the road frame (change in orientation with respect to t) can be determined using the chain rule by multiplying $\dot{s}(t)$ with the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s(t))$ (change in orientation with respect to s):

$$ {}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}}(t) = \begin{bmatrix} \omega_{\mathrm{x}}(t) \\ \omega_{\mathrm{y}}(t) \\ \omega_{\mathrm{z}}(t) \end{bmatrix} = {}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}(s(t)) \cdot \dot{s}(t) = \begin{bmatrix} \Omega_{\mathrm{x}}(s(t)) \cdot \dot{s}(t) \\ \Omega_{\mathrm{y}}(s(t)) \cdot \dot{s}(t) \\ \Omega_{\mathrm{z}}(s(t)) \cdot \dot{s}(t) \end{bmatrix}. \tag{2.33} $$

The aforementioned velocity frame $\mathcal{V}$ originates in the point mass position and is partially aligned with the point mass velocity. More precisely, the orientation of $\mathcal{V}$ results from $\mathcal{R}$ by a pure rotation around its z-axis such that no lateral velocity occurs in the velocity frame. The rotation matrix $\boldsymbol{R}_{\mathcal{RV}}(t) \in SO(3)$ describes the orientation of $\mathcal{V}$ relative to $\mathcal{R}$, and the angle between the two frames is called the relative orientation $\hat{\chi}(t)$, i.e., $\boldsymbol{R}_{\mathcal{RV}}(t) = \boldsymbol{R}_{\mathrm{z}}(\hat{\chi}(t))$. With this definition of $\mathcal{V}$, the velocity ${}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}}(t) \in \mathbb{R}^3$ of the point mass (or the origin of $\mathcal{V}$) expressed in the velocity frame itself has no lateral velocity component:

$$ {}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}}(t) = \begin{bmatrix} V(t) \\ 0 \\ w(t) \end{bmatrix}. \tag{2.34} $$

$V(t)$ thus corresponds to the collective speed of the point mass in the x-y plane of $\mathcal{R}$ and $\mathcal{V}$ but not to the total speed $\sqrt{V^2(t) + w^2(t)}$ due to the existing vertical speed component $w(t)$, as illustrated in Figure 2.2. For simplicity in the calculations, the velocity frame is not defined as fully aligned with the total speed, i.e., such that the z-component in (2.34) would also be 0. Instead, as in [60], it is assumed that the vertical velocity $w(t)$ is sufficiently small so that $V(t)$ lies approximately in the road plane.

The acceleration ${}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t) \in \mathbb{R}^3$ of the point mass (or the origin of $\mathcal{V}$) resulting from the motion along the 3D track surface is denoted as

$$ {}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t) = \begin{bmatrix} \hat{a}_{\mathrm{x}}(t) \\ \hat{a}_{\mathrm{y}}(t) \\ \hat{a}_{\mathrm{z}}(t) \end{bmatrix}. \tag{2.35} $$

The accelerations introduced here are labeled with $\hat{\square}$ and are referred to as kinematic accelerations, as they result solely from the kinematics of the 3D motion. However, no gravitational acceleration components are taken into account that an accelerometer in a moving vehicle would additionally measure. The actual measured accelerations ${}^{\mathcal{V}}\tilde{\boldsymbol{a}}_{\mathcal{IV}}(t) = \begin{bmatrix}\tilde{a}_{\mathrm{x}}(t) & \tilde{a}_{\mathrm{y}}(t) & \tilde{g}(t)\end{bmatrix}^{\top} \in \mathbb{R}^3$ are labeled with $\tilde{\square}$ and referred to as apparent accelerations. They are composed of the kinematic accelerations ${}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t)$ and the components resulting from the gravitational acceleration vector ${}^{\mathcal{I}}\boldsymbol{g} = \begin{bmatrix}0 & 0 & g\end{bmatrix}^{\top} \in \mathbb{R}^3$ expressed in the velocity frame $\mathcal{V}$ with $g = 9.81\,\mathrm{m/s^2}$:

$$
\begin{aligned}
{}^{\mathcal{V}}\tilde{\boldsymbol{a}}_{\mathcal{IV}}(t) &= {}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t) + {}^{\mathcal{V}}\boldsymbol{g}(t) && \text{(2.36a)}\\
&= {}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t) + \boldsymbol{R}_{\mathcal{RV}}^{\top}(t)\boldsymbol{R}_{\mathcal{IR}}^{\top}(s(t))\,{}^{\mathcal{I}}\boldsymbol{g}. && \text{(2.36b)}
\end{aligned}
$$

Substituting the rotation matrices $\boldsymbol{R}_{\mathcal{IR}}(s(t)) = \boldsymbol{R}(\theta(s(t)), \mu(s(t)), \varphi(s(t)))$ and $\boldsymbol{R}_{\mathcal{RV}}(t) = \boldsymbol{R}_{\mathrm{z}}(\hat{\chi}(t))$ into (2.36b), the following relations follow, neglecting the arguments:

$$
{}^{\mathcal{V}}\tilde{\boldsymbol{a}}_{\mathcal{IV}} = \begin{bmatrix}\tilde{a}_{\mathrm{x}}\\ \tilde{a}_{\mathrm{y}}\\ \tilde{g}\end{bmatrix} = \begin{bmatrix}\hat{a}_{\mathrm{x}}\\ \hat{a}_{\mathrm{y}}\\ \hat{a}_{\mathrm{z}}\end{bmatrix} + \begin{bmatrix}g \cdot (\cos(\mu)\sin(\varphi)\sin(\hat{\chi}) - \sin(\mu)\cos(\hat{\chi}))\\ g \cdot (\cos(\mu)\sin(\varphi)\cos(\hat{\chi}) + \sin(\mu)\sin(\hat{\chi}))\\ g \cdot (\cos(\mu)\cos(\varphi))\end{bmatrix}. \qquad (2.37)
$$

The third component of ${}^{\mathcal{V}}\tilde{\boldsymbol{a}}_{\mathcal{IV}}(t)$ is the acceleration orthogonal to the track surface and is labeled $\tilde{g}(t)$ in analogy to the gravitational acceleration g, which acts orthogonally on 2D tracks. More generally, for 2D tracks with $\mu(s(t)) = \varphi(s(t)) = 0°$, the relations in (2.37) yield $\tilde{a}_{\mathrm{x}}(t) = \hat{a}_{\mathrm{x}}(t)$, $\tilde{a}_{\mathrm{y}}(t) = \hat{a}_{\mathrm{y}}(t)$, and $\tilde{g}(t) = g$ with $\hat{a}_{\mathrm{z}}(t) = 0\,\mathrm{m/s^2}$.

In summary, the road $\mathcal{R}$ and velocity $\mathcal{V}$ frame share the same s-coordinate. While $\mathcal{V}$ originates in the point mass, $\mathcal{R}$ originates in the reference line ${}^{\mathcal{I}}\boldsymbol{c}(s(t))$ evaluated at the progress $s(t)$. The orientation of $\mathcal{R}$ relative to $\mathcal{I}$ is determined by the Euler angles $\theta(s(t))$, $\mu(s(t))$, and $\varphi(s(t))$ introduced in Subsection 2.3.1. The orientation of $\mathcal{V}$ then results from $\mathcal{R}$ rotated around its z-axis by the relative orientation $\hat{\chi}(t)$.

2.4.2 Model Constraints

Since the point-mass model is highly simplified, the physical quantities introduced in Subsection 2.4.1 may reach values that cannot be realized in practice. It is, therefore, essential to introduce kinematic and dynamic constraints.

Kinematic Constraints

A real vehicle has a minimum realizable turning radius and, thus, a maximum drivable curvature $\hat{\kappa}_{\max} \in \mathbb{R}_{>0}$. Therefore, the curvature $\hat{\kappa}(t) \in \mathbb{R}$ of the path traversed by the point mass must be constrained:

$$|\hat{\kappa}(t)| \leq \hat{\kappa}_{\max}. \tag{2.38}$$

On a 2D track, the driven path is described by only one scalar curvature value, which corresponds to the required curvature $\hat{\kappa}(t)$. In contrast, the driven path on a 3D track is represented by the spatial curvature vector ${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{IV}} \in \mathbb{R}^3$ of the velocity frame $\mathcal{V}$. In this case, the scalar curvature of the path projected onto the x-y plane of $\mathcal{V}$, i.e., the third component of ${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{IV}}$, is utilized in (2.38).

Dynamic Constraints

As the point-mass model is a kinematic vehicle model, the vehicle accelerations must be constrained. For this purpose, gg-diagrams are used, which restrict the vehicle's combined longitudinal $\tilde{a}_{\mathrm{x}}(t)$ and lateral $\tilde{a}_{\mathrm{y}}(t)$ acceleration. Exceeding the gg-diagram limits corresponds to undesired driving states, such as excessive oversteer or understeer. Typically, the gg-diagrams are utilized for 2D tracks, but since 3D tracks are also considered in this thesis, the generalization of [60] is introduced.

Various vehicle and track quantities influence the size and shape of the diagrams. On a 2D race track, the main dependency is the vehicle's velocity $V(t)$, which enters quadratically into the drag force and aerodynamic downforces. This dependency is visualized in Figure 2.3 (left). With higher velocities $V(t)$, the combined acceleration that can be realized increases due to the higher aerodynamic downforce. At the same time, the longitudinal acceleration potential decreases due to the increased drag force. As a result, the vehicle is limited by the tire-road contact at low speeds and by the engine power at high speeds.

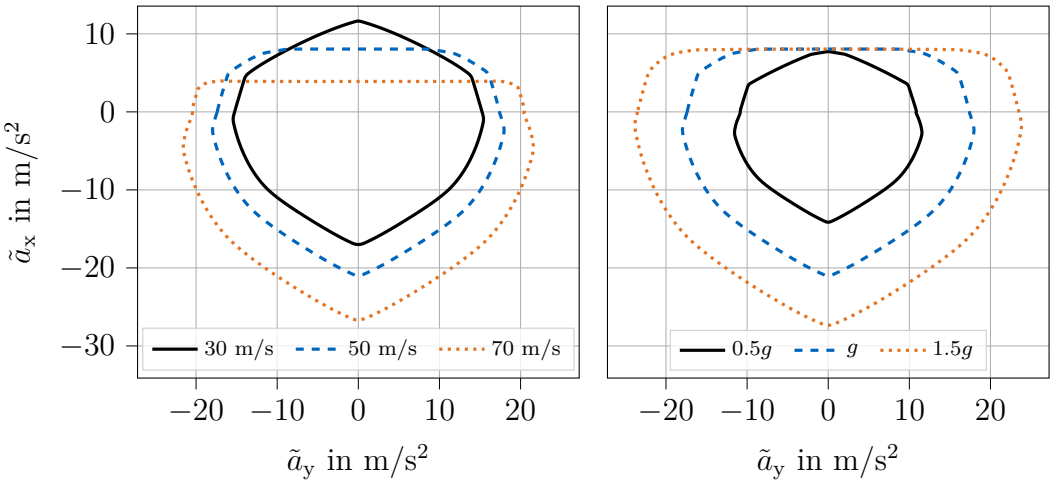


Figure 2.3: Influence of velocity V and apparent vertical acceleration $\tilde{g}$ on gg-diagrams. Left: gg-diagrams for different values of V with a fixed apparent vertical acceleration $\tilde{g} = g$. Right: gg-diagrams for different values of $\tilde{g}$ with a fixed velocity $V = 50\,\mathrm{m/s}$.

The velocity-dependent gg-diagrams are extended for 3D tracks in [60] by adding the dependence of the apparent vertical acceleration $\tilde{g}(t)$ orthogonally to the track surface. While $\tilde{g}(t)$ on 2D tracks always corresponds to the gravitational acceleration g, different track geometries result in accelerations not equal to g, which can significantly affect the gg-diagrams. One example of this is driving through a dip or over a hill, which causes $\tilde{g}(t)$ to increase or decrease. As shown in Figure 2.3 (right), for a constant velocity $V(t)$, higher vertical accelerations $\tilde{g}(t)$ allow higher combined accelerations to be realized.

The balance equations of forces of the 3D vehicle model, which is used to generate the gg-diagrams, contain the apparent accelerations $\tilde{a}_x(t)$ and $\tilde{a}_y(t)$. In the apparent accelerations, the gg-diagrams are symmetrical with respect to the $\tilde{a}_x$-axis. The influence of the 3D effects only becomes evident when the gg-diagrams are expressed in terms of the kinematic accelerations $\hat{a}_x(t)$ and $\hat{a}_y(t)$. Figure 2.4 shows gg-diagrams converted into kinematic accelerations for a flat, a downhill, and a right-banked straight. The downhill straight causes gravity to support acceleration and counteract braking. The track inclined to the right supports acceleration to the right (negative sign) and counteracts acceleration to the left (positive sign).

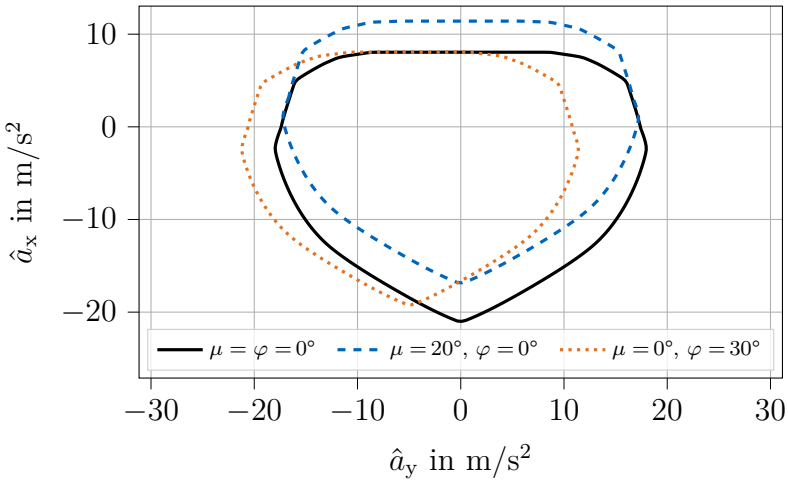


Figure 2.4: Influence of the slope $\mu(s) > 0°$ and banking angle $\varphi(s) > 0°$ on gg-diagrams expressed with kinematic accelerations for a fixed velocity $V = 50\,\mathrm{m/s}$. The orientation angle θ is 0°, and all Euler angles are constant, resulting in a flat (solid line), a downhill (dashed line), and a right-banked (dotted line) straight.

The true gg-diagrams can have arbitrarily complex shapes. However, the parameterization of such a shape is difficult, and checking a combined acceleration pair $(\hat{a}_x(t), \hat{a}_y(t))$ within the shape is computationally time-consuming. In this thesis, therefore, the diamond-shaped representation presented in [71] is used, which requires only four parameters and is faster to check. The following inequalities are used for the check, with the time argument neglected:

$$\tilde{a}_x \leq \tilde{a}_{x,\max}(V, \tilde{g}), \tag{2.39a}$$

$$|\tilde{a}_y| \leq \tilde{a}_{y,\max}(V, \tilde{g}), \tag{2.39b}$$

$$|\tilde{a}_x| \leq |\tilde{a}_{x,\min}(V, \tilde{g})| \left[1 - \left[\frac{|\tilde{a}_y|}{\tilde{a}_{y,\max}(V, \tilde{g})}\right]^{\rho(V,\tilde{g})}\right]^{\frac{1}{\rho(V,\tilde{g})}}. \tag{2.39c}$$

As depicted in Figure 2.5, the first two parameters, $\tilde{a}_{x,\min}(V, \tilde{g}) \in \mathbb{R}_{\leq 0}$ and $\tilde{a}_{y,\max}(V, \tilde{g}) \in \mathbb{R}_{\geq 0}$, correspond to the corner points of the diamond. The third parameter, $\tilde{a}_{x,\max}(V, \tilde{g}) \in \mathbb{R}_{\geq 0}$, constrains the positive acceleration and thus represents the limited motor power. The last parameter, $\rho(V, \tilde{g}) \in [1, 2]$, is a shape factor that allows the diamond shape to be varied from a rhombus with $\rho(V, \tilde{g}) = 1$ to an ellipse with $\rho(V, \tilde{g}) = 2$. Since the diamond-shaped

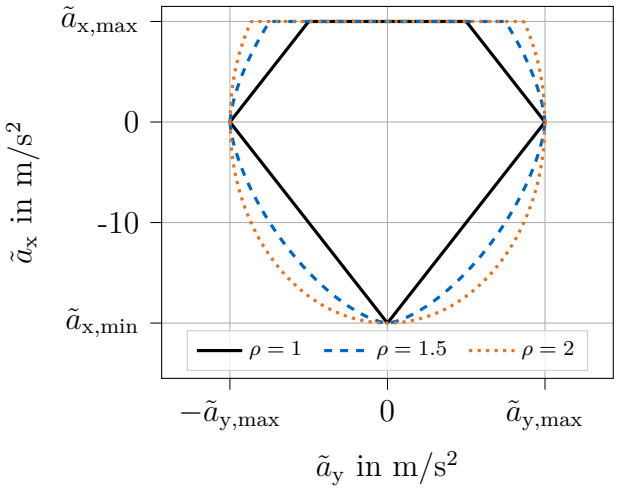


Figure 2.5: Diamond-shaped representation of the gg-diagram with the fixed parameters $\tilde{a}_{\mathrm{x,min}} = -20\,\mathrm{m/s^2}$, $\tilde{a}_{\mathrm{y,max}} = 15\,\mathrm{m/s^2}$, and $\tilde{a}_{\mathrm{x,max}} = 10\,\mathrm{m/s^2}$. Varying ρ between 1 and 2 results in shapes between a rhombus and an ellipse with the same corner points.

representation of the gg-diagrams only approximates the real shape, the four parameters must be chosen appropriately. On the one hand, the diamond shapes must underapproximate the real shape to avoid infeasible areas within the gg-diagrams. On the other hand, for better performance, the underapproximation should fill as much of the real shape as possible.

The gg-diagrams can be generated either through simulation using a high-accuracy dynamic vehicle model or experimentally with the actual vehicle. In this work, they are generated in simulation and adapted experimentally, although the generation process is not detailed here. Instead, refer to [57] or [60] as examples of simulative generation. The parameters $\tilde{a}_{\mathrm{x,min}}(V, \tilde{g})$, $\tilde{a}_{\mathrm{y,max}}(V, \tilde{g})$, $\tilde{a}_{\mathrm{x,max}}(V, \tilde{g})$, and $\rho(V, \tilde{g})$ are given for discrete value pairs of velocity V and apparent vertical acceleration $\tilde{g}$. Linear interpolation is used to evaluate the gg-diagrams between the given data points of V and $\tilde{g}$.

2.5 Local Planning Interfaces

In each planning cycle of the local planning module, the trajectory output is generated based on inputs that are informally illustrated in Figure 1.3.

With the definition of the coordinate systems in Sections 2.3 and 2.4, these quantities can now be formulated mathematically. The local trajectory planning approaches presented in this thesis operate in the Curvilinear coordinates $s(t)$ and $d(t)$. However, the model constraints in Subsection 2.4.2 are defined in Cartesian coordinates, and the adjacent modules of local planning operate in Cartesian coordinates. Therefore, the transformation from Curvilinear to Cartesian coordinates

$$\left[s, \dot{s}, \ddot{s}, d, \dot{d}, \ddot{d}\right](t) \rightarrow [x, y, z, V, w, \hat{a}_{\mathrm{x}}, \hat{a}_{\mathrm{y}}, \hat{a}_{\mathrm{z}}, \hat{\chi}, \hat{\kappa}](t) \tag{2.40}$$

and the reverse transformation are required, which are both stated in Appendix A. Since a transformation is possible in both directions, the inputs and outputs in this section are only specified in Curvilinear coordinates. Nevertheless, specific Cartesian quantities will be used in the local planning approaches presented in this thesis.

2.5.1 Inputs

The inputs of the local planning module are the offline-generated race track map, the state estimation of the ego vehicle, the racing line, and the racing flags. For multi-vehicle racing, these inputs are supplemented either by the object list (interaction-aware planning) or the prediction (conventional planning) of the opponent vehicles.

Map

The map of the race track is available as $N_{\mathrm{map}} + 1$ discrete data points, whereby the data of the first and last points are identical except for the progress s. According to the track model in Subsection 2.3.2, the data points are available in the following format:

$$\Big\{\Big(s[k], {}^{\mathcal{I}}\boldsymbol{c}[k], \boldsymbol{R}_{\mathcal{IR}}[k], {}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}[k], {}^{\mathcal{R}}\boldsymbol{\Omega}'_{\mathcal{IR}}[k], d_{\mathrm{l}}[k], d_{\mathrm{r}}[k]\Big) \mid k \in \{0, 1, \ldots, N_{\mathrm{map}}\}\Big\}. \tag{2.41}$$

A data point consists of the progress s, the reference line ${}^{\mathcal{I}}\boldsymbol{c}$, the rotation matrix $\boldsymbol{R}_{\mathcal{IR}}$, the spatial curvature vector ${}^{\mathcal{R}}\boldsymbol{\Omega}_{\mathcal{IR}}$, its derivative with respect

to the progress ${}^{\mathcal{R}}\boldsymbol{\Omega}'_{\mathcal{IR}}$, and the left and right track widths d_l and d_r. For an evaluation between the discrete data points, there are different options, as compared in [149, 150]. In this thesis, linear interpolation is used, which leads to straight-line segments connecting the discrete points while avoiding discontinuities at the segment transitions.

Estimated Ego State

The estimated state of the ego vehicle $\boldsymbol{z}_0 \in \mathbb{R}^6$ is provided as a vector:

$$\boldsymbol{z}_0 = \begin{bmatrix} s_0 & \dot{s}_0 & \ddot{s}_0 & d_0 & \dot{d}_0 & \ddot{d}_0 \end{bmatrix}^\top . \tag{2.42}$$

All Cartesian quantities associated with the estimated state $\boldsymbol{z}_0$ are also labeled with the index 0.

Racing Line

The racing line $(s_\mathrm{rl}(t), d_\mathrm{rl}(t))\,,\ t \in [0, T_\mathrm{rl}]$ is a curve parameterized with respect to time t with the temporal horizon $T_\mathrm{rl} \in \mathbb{R}_{>0}$. It is available as $N_\mathrm{rl} \in \mathbb{N}$ discrete data points:

$$\Big\{\Big(t_\mathrm{rl}[k], s_\mathrm{rl}[k], \dot{s}_\mathrm{rl}[k], \ddot{s}_\mathrm{rl}[k], d_\mathrm{rl}[k], \dot{d}_\mathrm{rl}[k], \ddot{d}_\mathrm{rl}[k]\Big) \mid k \in \{1, 2, \dots, N_\mathrm{rl}\}\Big\}. \tag{2.43}$$

All Cartesian quantities corresponding to the racing line are also indexed with rl. Linear interpolation is used to evaluate the racing line within the discrete data points. To evaluate the racing line at the ego progress s_0, the progress of its first data point must be less than or equal to s_0, i.e., $s_\mathrm{rl}[0] \leq s_0$. In addition, the horizon T_rl must be sufficiently large to allow the racing line to be evaluated at the s-coordinates of any trajectory that may be generated.

Object List / Predictions

The number of detected opponent vehicles $N_\mathrm{opp} \in \mathbb{N}_0$ may vary in each planning cycle. For interaction-aware planning, an object list containing data for each opponent vehicle indexed by l is provided:

$$\Big\{\Big(s_{\mathrm{opp},l}, \dot{s}_{\mathrm{opp},l}, d_{\mathrm{opp},l}, \dot{d}_{\mathrm{opp},l}\Big) \mid l \in \{1, 2, \dots, N_\mathrm{opp}\}\Big\}. \tag{2.44}$$

All Cartesian quantities associated with the state of the opponent vehicle l are also labeled with the index opp, l. If specifically only one opponent vehicle is considered, the index l is omitted. The opponent data is limited to the first time derivative, as higher-order derivatives are unsuitable due to the available detection sensors and the resulting estimation noise. For conventional planning, a prediction $\left(s_{\mathrm{pr},l}(t), d_{\mathrm{pr},l}(t)\right),\ t \in [0, T_{\mathrm{pr}}]$ of time horizon $T_{\mathrm{pr}} \in \mathbb{R}_{>0}$ is given for each opponent vehicle. Each prediction consists of $N_{\mathrm{pr}} \in \mathbb{N}$ discrete data points:

$$\Big\{(t_{\mathrm{pr},l}[k], s_{\mathrm{pr},l}[k], \dot{s}_{\mathrm{pr},l}[k], d_{\mathrm{pr},l}[k], \dot{d}_{\mathrm{pr},l}[k]) \\ \mid k \in \{1, \dots, N_{\mathrm{pr}}\}, l \in \{1, 2, \dots, N_{\mathrm{opp}}\}\Big\}. \tag{2.45}$$

Linear interpolation is used to evaluate the predictions between the discrete points.

Flags

The racing flags are transferred to the planning module as integer variables. According to the racing rules, the flags are translated in a dedicated state machine into corresponding planning signals, such as a speed limit $V_{\mathrm{limit}}(t) \in \mathbb{R}_{>0}$, whether overtaking is allowed, or whether the vehicle should start, stop, or pit. A detailed description of the flag logic and the conversion into the planning signals is not provided in this thesis but is available in [151].

2.5.2 Output

The sole output of the local planning module is the trajectory $\left(s(t), d(t)\right),\ t \in [0, T_{\mathrm{traj}}]$ to be followed with the temporal planning horizon $T_{\mathrm{traj}} \in \mathbb{R}_{>0}$. It is a curve parameterized with respect to time t and is represented by $N_{\mathrm{traj}} \in \mathbb{N}$ discrete data points with information up to the second time derivative:

$$\Big\{\Big(t[k], s[k], \dot{s}[k], \ddot{s}[k], d[k], \dot{d}[k], \ddot{d}[k]\Big) \mid k \in \{1, 2, \dots, N_{\mathrm{traj}}\}\Big\}. \tag{2.46}$$

Since the feasibility is only checked for the discrete points, interpolation within the points can lead to infeasibility issues. These are is addressed by using safety margins in the checks and selecting a sufficiently large N_{traj}. Additionally, the planning horizon must not exceed the prediction horizon, i.e., $T_{\mathrm{traj}} \leq T_{\mathrm{pr}}$.

3 Hybrid Planning on Two-Dimensional Oval Tracks

This chapter presents a hybrid local trajectory planning approach for 2D oval tracks, addressing limitation 1 from Section 1.4. In existing graph-based planning approaches without PVD, spatio-temporal graphs with coarse discretization and simple edge structures are used to counteract the curse of dimensionality. However, trajectories generated from such graphs are difficult for the controller to track accurately, restricting the ability to reach the vehicle's handling limits and potentially affecting stability. The edges connecting the ego vehicle state to the graph are particularly crucial, as, according to the RH execution, they contain the trajectory segment that is actually traversed by the race car. Therefore, the main idea of the hybrid approach is to use a coarse spatio-temporal graph with a small number of simple-structured edges. In contrast, the edges connecting the ego vehicle's state to the graph are generated in larger numbers and with increased smoothness through sampling. The combination of fine sampling with a coarse spatio-temporal graph allows a sufficiently long planning horizon while ensuring high smoothness of the traversed trajectory segment. The spatio-temporal graph consists of the spatial graph from [70], previously used by the TUM Autonomous Motorsport team, with the extension by the temporal dimensions of [74]. The sampling is based on the approach from [92] using jerk-optimal curves in Curvilinear coordinates.

Within the hybrid planning approach, tracks are considered in 2D, although the actual track may include 3D effects. This means that in the planning process, $\varphi(s) = \mu(s) = 0°$ and $\Omega_\mathrm{x}(s) = \Omega_\mathrm{y}(s) = 0\,\mathrm{rad/m}$ are assumed. Consequently, $\tilde{a}_\mathrm{x}(t) = \hat{a}_\mathrm{x}(t)$, $\tilde{a}_\mathrm{y}(t) = \hat{a}_\mathrm{y}(t)$, $\hat{a}_\mathrm{z}(t) = 0\,\mathrm{m/s^2}$, $\tilde{g}(t) = g$, and the gg-diagrams depend only on velocity V. In addition, the curvature of a path is characterized solely by its geodesic curvature, which is simply referred to as curvature.

This chapter is based on the author's publications [152] and [151] and additionally contains a more detailed classification and description, e.g., the selection of the initial layer. The planning concept is described in Section 3.1, and simulative as well as experimental results are presented in Section 3.2. A discussion in Section 3.3, outlining the approach's limitations, concludes this chapter.

3.1 Methodology

The spatio-temporal graph underlying the hybrid planning approach is described in Subsection 3.1.1, followed by a breakdown of the approach into subtasks in Subsection 3.1.2. The three subtasks of start state determination, initial edge generation, and graph search are detailed in Subsections 3.1.3, 3.1.4, and 3.1.5, respectively. The validity check of the edges is introduced in Subsection 3.1.6, and the cost functional used for the graph search in Subsection 3.1.7.

3.1.1 Graph Structure

The spatio-temporal graph consists of a spatial graph extended by the temporal dimensions of velocity V and time t. The spatial graph is based on the layered graph structure proposed in [70] for path planning in racing. The extension by the temporal dimensions is similar to the principle introduced in [74]. However, the hybrid concept presented in the following subsections can also be applied to other graph structures with minor modifications.

Spatial Graph

As illustrated in Figure 3.1, the spatial graph comprises $N_\mathrm{s} \in \mathbb{N}$ layers $\mathfrak{L}_i$ positioned along the reference line at certain progressions s_i, where $i \in \{1, 2, \ldots, N_\mathrm{s}\}$ denotes the index of the discrete longitudinal coordinate. Here, the choice of layer spacing $D_\mathrm{s} = s_{i+1} - s_i \in \mathbb{R}_{>0}$ is a trade-off, as larger layer spacings result in fewer layers to be searched and, thus, in shorter computation times. However, denser layer coverage is required in turns to better replicate the changing racing line in these segments.

A layer $\mathfrak{L}_i$ is a set of $N_\mathrm{d} \in \mathbb{N}$ nodes $\mathfrak{n}_{ij} \in \mathfrak{L}_i$ arranged along the normal vector ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s_i)$ of the road frame at the corresponding progress s_i, where

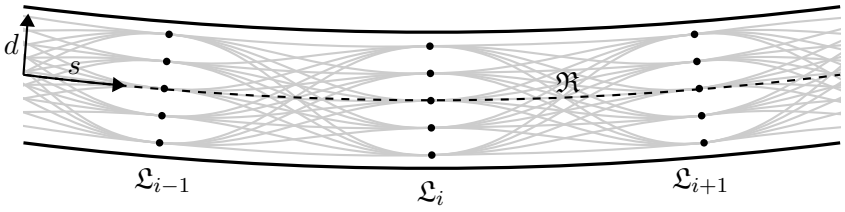


Figure 3.1: Illustration of the spatial graph for a curved track segment consisting of three layers. Each layer comprises five spatial nodes (black circles) arranged perpendicular to the reference line $\mathfrak{R}$. The spatial nodes of a layer are connected to those of the following layer by spatial edges (gray lines).

$j \in \{1, 2, \ldots, N_\mathrm{d}\}$ denotes the index of a node within its layer. A node $\mathfrak{n}_{ij}$, referred to as a spatial node, is described by the tuple $(s_i, d_{ij}, \hat{\chi}_{ij})$ or $(x_{ij}, y_{ij}, \hat{\chi}_{ij})$, consisting of its position (s_i, d_{ij}) or (x_{ij}, y_{ij}) and its relative orientation $\hat{\chi}_{ij}$. To be able to plan trajectories that match the racing line, a spatial node labeled with the index $j_\mathrm{rl} \in \{1, 2, \ldots, N_\mathrm{d}\}$ is placed on the racing line in each layer, i.e., $d_{ij_\mathrm{rl}} = d_\mathrm{rl}(s_i)$. The remaining nodes are placed to the left and right of the racing line with a fixed lateral spacing $D_\mathrm{d} \in \mathbb{R}_{>0}$ up to the track boundaries in the interval $[d_\mathrm{r}(s_i) + {}^{D_\mathrm{w}}\!/_2 + D_\mathrm{saf},\ d_\mathrm{l}(s_i) - {}^{D_\mathrm{w}}\!/_2 - D_\mathrm{saf}]$, considering the vehicle width $D_\mathrm{w} \in \mathbb{R}_{>0}$ and a safety distance $D_\mathrm{saf} \in \mathbb{R}_{\geq 0}$. Thus, the spatial nodes of a layer $\mathfrak{L}_i$ have the same longitudinal position s_i but varying lateral positions d_{ij}. The relative orientation of a node $\hat{\chi}_{ij}$ is determined depending on its lateral position by linear interpolation between the relative orientation of the according track boundary and the racing line. Consequently, the node on the racing line is oriented like the racing line itself, i.e., $\hat{\chi}_{ij_\mathrm{rl}} = \hat{\chi}_\mathrm{rl}(s_i)$, and the remaining nodes tend towards the racing line while avoiding pointing into the track boundaries.

Edges of the spatial graph, referred to as spatial edges, connect each node of a layer $\mathfrak{L}_i$ with each node of the subsequent layer $\mathfrak{L}_{i+1}$, as depicted in Figure 3.1. The last layer is connected to the first layer, creating a cyclic graph that covers the entire race track. The spatial edges are cubic polynomials in the Cartesian coordinates $x(s)$ and $y(s)$ with $s \in [s_i, s_{i+1}]$. The boundary conditions of the polynomials are selected such that the position and relative orientation of the origin and destination nodes are adhered to, resulting in a C^1-continuous

transition between two consecutive edges. However, a curvature continuity cannot be guaranteed at the transition of the edges. Using the polynomials, the corresponding relative orientation $\hat{\chi}(s)$ and curvature $\hat{\kappa}(s)$ profiles can be determined analytically.

The spatial graph can be calculated offline, as the race track and racing line are known in advance. Further, each spatial edge is checked offline to determine whether it satisfies the kinematic condition (2.38) and lies within the track boundaries. If invalid, the edge is removed from the graph. Subsequently, all edges that originate in nodes without parents or end in nodes without children are iteratively removed, avoiding an online check of the kinematic constraint.

Spatio-Temporal Graph

The temporal part of the graph results from the extension of the spatial nodes by the velocity V and time t dimensions. Following the approach in [74], these additional dimensions are segmented into $N_\mathrm{V} \in \mathbb{N}$ and $N_\mathrm{t} \in \mathbb{N}$ intervals, respectively, resulting in spatio-temporal nodes, as illustrated in Figure 3.2. Thus, a spatio-temporal node consists of an underlying spatial node $\mathfrak{n}_{ij}$ and a range of velocities $[V_p, V_{p+1})$ and times $[t_q, t_{q+1})$, which are indexed by $p \in \{1, 2, \dots, N_\mathrm{V}\}$ and $q \in \{1, 2, \dots, N_\mathrm{t}\}$, respectively. Due to the division into intervals, a spatio-temporal node does not represent one specific discrete state. Instead, infinite states with the same position and similar temporal values are assigned to a single spatio-temporal node.

In the following, the start state refers to the state from which the planning process begins for a given planning cycle. In each cycle, a trajectory composed of several spatio-temporal edges must be generated starting from this start state and having a minimum planning horizon T_traj. As illustrated in Figure 3.2, the spatio-temporal edges can be divided into two sets. The first set $\mathfrak{E}_0$ consists of the initial edges, which connect the start state to the offline-calculated graph and are required since the start state is not necessarily located on a spatial node. The second set $\mathfrak{E}_1$ comprises the remaining edges, which connect the layers of the graph. The layer chosen to generate the initial edges $\mathfrak{E}_0$ is referred to as the initial layer $\mathfrak{L}_\iota$ with the index $\iota \in \{1, 2, \dots, N_\mathrm{s}\}$ and may change at each planning cycle. Since the start state varies in each planning cycle and the spatio-temporal nodes do not represent discrete states, the spatio-temporal edges cannot be precomputed.

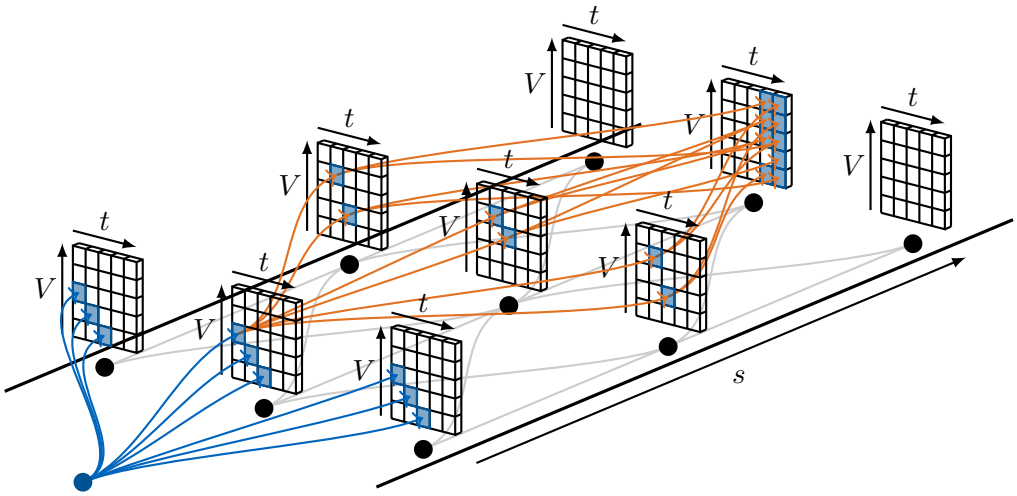


Figure 3.2: Illustration of the spatio-temporal graph for a straight track segment consisting of three layers. The spatial part of the graph consists of three spatial nodes (black circles) per layer, which are connected by spatial edges (gray lines). Each spatial node is extended by 25 spatial-temporal nodes (cells in grids). The start state $\tilde{\boldsymbol{z}}_0$ (blue circle), from which the graph search begins, is connected to the initial layer $\mathfrak{L}_\iota$ by initial edges $\mathfrak{E}_0$ (blue lines). The remaining spatio-temporal edges $\mathfrak{E}_1$ (orange lines) connect the spatio-temporal nodes, though not all are shown for clarity.

3.1.2 Planning Procedure

The hybrid local planning approach consists of the three steps shown in Figure 3.3. The first step is to determine the start state from which to start the actual planning process. Here, the start state does not necessarily correspond to the ego vehicle's state $\boldsymbol{z}_0$. In the second step, the start state is connected to the graph structure by creating initial edges $\mathfrak{E}_0$. Since the initial edges cover a longer time horizon than the computation time of a whole planning cycle, according to the RH execution, they include the trajectory segment that is actually traversed by the race car. Thus, the initial edges are elementary for vehicle stability and race performance. As a third step, the remaining edges $\mathfrak{E}_1$ are generated, and the optimal valid spatio-temporal edge sequence is determined using a suitable cost functional and an efficient graph search algorithm.

In [74], all spatio-temporal edges are generated based on constant acceleration, enabling efficient computation. In this approach, referred to as the uniform acceleration approach, additional spatial edges are generated online that connect

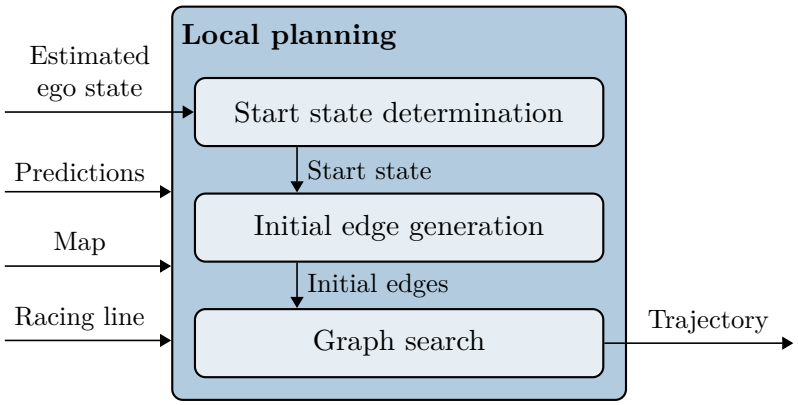


Figure 3.3: Procedure of the hybrid planning approach.

the start state to the spatial nodes in the initial layer $\mathfrak{L}_\iota$, complementing the spatial edges precomputed offline. Subsequently, several uniform acceleration profiles are applied along the fixed spatial edges, resulting in spatio-temporal edges. Thus, the initial edges from the start state to a given spatial node have the same underlying path and uniform acceleration profiles, meaning that the acceleration of the start state does not serve as the initial condition for the edges. These coarse edges in space and time cannot cover the entire state space and are difficult for the controller to track, preventing the vehicle from being pushed to its handling limits.

Instead of treating the initial edges $\mathfrak{E}_0$ the same as the remaining edges $\mathfrak{E}_1$, in the proposed hybrid planning approach, more variable initial edges are generated through a sampling-based step. However, the remaining edges $\mathfrak{E}_1$ are still based on the efficient constant acceleration calculation, allowing the minimum planning horizon to be achieved with sufficiently short computation times. In contrast to the uniform acceleration approach, the hybrid planning approach achieves smooth acceleration transitions for the initial edges, taking into account the initial conditions. Another advantage of the approach is that it generates individual paths for each initial edge. Unlike in [74], where different acceleration profiles are based on the same underlying path, each path is unique in the hybrid planning approach since the spatial and temporal information are generated simultaneously. This adjusts the path to match the temporal information, resulting in lower lateral accelerations and enabling driving closer to the handling limits.

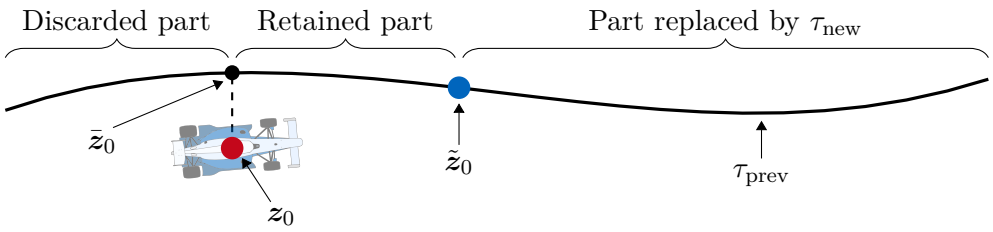


Figure 3.4: Illustration of the start state determination (not to scale). First, the estimated ego state z_0 (red circle) is projected onto the trajectory planned in the previous planning cycle τ_{prev} (black line), resulting in a projected state $\bar{z}_0$ (black circle). The start state $\tilde{z}_0$ (blue circle) is then obtained by advancing along τ_{prev} by an estimate of the required computation time beginning from $\bar{z}_0$.

The remainder of this section deals with the start state determination (Subsection 3.1.3) and, in particular, the generation of the initial edges (Subsection 3.1.4). In contrast, the graph search (Subsection 3.1.5), including the generation of the remaining edges $\mathfrak{E}_1$, is only briefly discussed, as this is not relevant to the general hybrid planning concept. The validity checks (Subsection 3.1.6) and the cost functional (Subsection 3.1.7) are presented in separate subsections, as both the initial and remaining edges are validated and assigned costs. Since both subsections relate to the initial edge generation and the graph search, they are also not depicted as individual steps in Figure 3.3.

3.1.3 Start State Determination

The selection of the start state $\tilde{z}_0 = \begin{bmatrix} \tilde{s}_0 & \dot{\tilde{s}}_0 & \ddot{\tilde{s}}_0 & \tilde{d}_0 & \dot{\tilde{d}}_0 & \ddot{\tilde{d}}_0 \end{bmatrix}^\top \in \mathbb{R}^6$ is crucial for the overall race performance. All variables associated with $\tilde{z}_0$ are indicated with a tilde next to the index 0, distinguishing them from the quantities corresponding to the estimated state of the vehicle z_0. Since the planned trajectory serves as the reference for the tracking controller, z_0 cannot be used directly as the start state $\tilde{z}_0$. This would cause the reference trajectory to follow the actual vehicle state, preventing a control error from accumulating and thus providing no incentive for the controller to follow the trajectory. Instead, as shown in Figure 3.4, the estimated state is projected onto the trajectory planned in the previous planning cycle τ_{prev}, resulting in a projected state $\bar{z}_0 \in \mathbb{R}^6$ and allowing a control error to accumulate.

In addition, the computation time of the planning cycle must be compensated for, as the vehicle follows the trajectory τ_{prev} while the new trajectory τ_{new} is being generated. If, for example, the projected state $\bar{\boldsymbol{z}}_0$ is selected as $\tilde{\boldsymbol{z}}_0$, the newly planned trajectory τ_{new} can deviate from τ_{prev} along the same track segment that lies in the vehicle's past at the end of the planning cycle. Consequently, switching the reference of the controller from τ_{prev} to τ_{new} would result in a sudden change in the control error, which could affect vehicle stability. At the same time, the start state $\tilde{\boldsymbol{z}}_0$ should not be selected too far ahead on the race track, as this would result in delayed responsiveness. Therefore, it is desirable to incorporate the actual progress of the vehicle at the end of the planning cycle when determining the start state. However, since the required computation time and possible control errors are unknown at the beginning of a cycle, an estimate is obtained by advancing along τ_{prev} by the estimated computation time beginning from $\bar{\boldsymbol{z}}_0$, as shown in Figure 3.4.

The part of τ_{prev} located to the rear of $\bar{\boldsymbol{z}}_0$ is discarded, as this segment lies in the vehicle's past. The part between $\bar{\boldsymbol{z}}_0$ and $\tilde{\boldsymbol{z}}_0$ is retained, as the actual vehicle state at the end of the planning cycle may lie in this segment. Starting from $\tilde{\boldsymbol{z}}_0$, the initial edges are generated, and the graph search is executed. The final trajectory is, therefore, a concatenation of the retained part and the new solution of the graph search.

3.1.4 Initial Edge Generation

The proposed concept for generating the initial edges utilizes a sampling-based approach that connects the start state $\tilde{\boldsymbol{z}}_0 = \begin{bmatrix} \tilde{s}_0 & \dot{\tilde{s}}_0 & \ddot{\tilde{s}}_0 & \tilde{d}_0 & \dot{\tilde{d}}_0 & \ddot{\tilde{d}}_0 \end{bmatrix}^\top$ to the spatio-temporal graph, specifically to the chosen initial layer $\mathfrak{L}_\iota$. Several end states and end times are sampled in the Curvilinear coordinates and then connected to the start state $\tilde{\boldsymbol{z}}_0$ through jerk-optimal curves, resulting in a set of lateral and longitudinal curves. Jerk optimality is employed, as jerky trajectories can excite neglected high-frequency dynamics in the tracking controller, potentially reducing vehicle stability. Once generated, the lateral and longitudinal curves are combined into pairs, forming the set of initial edges $\mathfrak{E}_0$.

Selection of the Initial Layer

The initial layer $\mathfrak{L}_\iota$ is chosen as the first layer whose distance to the start position $\Delta s_i = s_i - \tilde{s}_0 \in \mathbb{R}$ exceeds a speed-dependent minimum distance $\Delta s_{\min}(\tilde{V}_0) \in \mathbb{R}_{>0}$ in the direction of travel:

$$\iota = \arg\min_i \Delta s_i \quad \text{subject to} \quad \Delta s_i > \Delta s_{\min}(\tilde{V}_0). \tag{3.1}$$

The minimum distance $\Delta s_{\min}(\tilde{V}_0)$ increases with rising speed $\tilde{V}_0$ to prevent high curvature values and the associated infeasible lateral accelerations. The distance values are determined empirically for discrete values of $\tilde{V}_0$, and linear interpolation is used for an evaluation between the data points.

Jerk-Optimal Curves

The lateral $d(t)$ and longitudinal curves $s(t)$ are each generated based on a jerk-optimal movement, which reduces abrupt acceleration changes and has a beneficial effect on vehicle stability. Additionally, Appendix B (case 1) proves that for a third-order integrator chain with position $\xi(t) \in \mathbb{R}$, a simple quintic (fifth-order) polynomial minimizes the jerk-optimal cost functional

$$J\left(\xi(t)\right) = \frac{1}{2} \int_0^{t_\mathrm{e}} \dddot{\xi}^2(t)\, \mathrm{d}t \tag{3.2}$$

subject to the initial condition $\begin{bmatrix} \xi(0) & \dot{\xi}(0) & \ddot{\xi}(0) \end{bmatrix}^\top$ at the start time $t_0 = 0\,\mathrm{s}$ and the fixed end condition $\begin{bmatrix} \xi(t_\mathrm{e}) & \dot{\xi}(t_\mathrm{e}) & \ddot{\xi}(t_\mathrm{e}) \end{bmatrix}^\top$ at the end time $t_\mathrm{e} \in \mathbb{R}_{>0}$. The parameters of the polynomial can be efficiently computed, allowing real-world operation.

When applied to the generation of the initial edges, the start conditions of the lateral and longitudinal curves are defined by the start state $\tilde{\boldsymbol{z}}_0$:

$$\begin{bmatrix} s(0) & \dot{s}(0) & \ddot{s}(0) \end{bmatrix}^\top = \begin{bmatrix} \tilde{s}_0 & \dot{\tilde{s}}_0 & \ddot{\tilde{s}}_0 \end{bmatrix}^\top, \tag{3.3a}$$

$$\begin{bmatrix} d(0) & \dot{d}(0) & \ddot{d}(0) \end{bmatrix}^\top = \begin{bmatrix} \tilde{d}_0 & \dot{\tilde{d}}_0 & \ddot{\tilde{d}}_0 \end{bmatrix}^\top. \tag{3.3b}$$

The following paragraphs define the lateral and longitudinal end conditions and the end time to produce the set of initial edges. First, the choice of end positions

and velocities is described to integrate all initial edges into the introduced graph structure. This is followed by the selection of the end accelerations to improve the racing performance.

Integration into Graph Structure

Since the quintic polynomials must be integrated as initial edges within the graph structure, the end conditions must be sampled such that the edges terminate in the initial layer $\mathfrak{L}_\iota$ at the correct individual positions $(s_\iota, d_{\iota j})$ and headings $\hat{\chi}_{\iota j}$ of each spatial node $\mathfrak{n}_{\iota j}$. As depicted in Figure 3.1, the spatial nodes in the initial layer are arranged perpendicular to the reference line and, therefore, have the same longitudinal position s_ι. Hence, the end position of the longitudinal curve has to be chosen accordingly without sampling, i.e., $s(t_\mathrm{e}) = s_\iota$. To ensure that the edges terminate in the spatial nodes $\mathfrak{n}_{\iota j}$, the lateral end position must be sampled according to the lateral position of each spatial node, i.e., $d(t_\mathrm{e}) = d_{\iota j}$.

In addition to the position, the individual headings $\hat{\chi}_{\iota j}$ of the spatial nodes at the end of an edge must also be guaranteed. This is achieved by appropriately specifying the end conditions of the lateral and longitudinal velocity $\dot{d}(t_\mathrm{e})$ and $\dot{s}(t_\mathrm{e})$, respectively. Instead of specifying the velocities directly, the global end velocity $V(t_\mathrm{e})$ is sampled to cover the entire possible velocity range. To address the lower engine power at high speeds, finer sampling is applied in this region. Thus, the complete velocity range is divided into two fixed intervals, each containing a different number of equidistant distributed velocity samples: $[V_1, \dots, V_{N_\underline{\mathrm{V}}}, \dots, V_{N_\underline{\mathrm{V}}+N_{\bar{\mathrm{V}}}}]$. Here, $N_\underline{\mathrm{V}} \in \mathbb{N}$ represents the number of samples in the low-speed range, while $N_{\bar{\mathrm{V}}} \in \mathbb{N}$ denotes the number in the high-speed range. To ensure the individual heading of a node, each velocity sample V_m is then transformed into the corresponding lateral and longitudinal velocity $\dot{d}_{jm}$ and $\dot{s}_{jm}$, where $m \in \left\{1, 2, \dots, N_\underline{\mathrm{V}} + N_{\bar{\mathrm{V}}}\right\}$ indicates the index of the respective velocity sample. According to the relations in (A.11b), the lateral and longitudinal velocities are obtained by

$$\dot{s}(t_\mathrm{e}) = \dot{s}_{jm} = \frac{V_m \cdot \cos\left(\hat{\chi}_{\iota j}\right)}{1 - \Omega_\mathrm{z}(s_\iota) d_{\iota j}}, \tag{3.4}$$

$$\dot{d}(t_\mathrm{e}) = \dot{d}_{jm} = V_m \cdot \sin\left(\hat{\chi}_{\iota j}\right). \tag{3.5}$$

Thus, for the j-th spatial node in the initial layer $\mathfrak{L}_\iota$ and the m-th velocity sample, a unique pair of lateral and longitudinal velocities is established. With this specification, the end conditions for position and velocity are fully defined, and the end conditions for time t_e and acceleration $\ddot{s}(t_\mathrm{e})$ and $\ddot{d}(t_\mathrm{e})$ remain to be determined.

Performance Improvement for Racing

In sampling-based methods for road traffic, the lateral and longitudinal accelerations at the end time t_e are often set to $0\,\mathrm{m/s^2}$ for comfort reasons. However, this approach is unsuitable in a racing context. For instance, during an acceleration maneuver, initial edges must adapt early to meet the end conditions, which can lead to a loss of racing performance. Further, simple equidistant sampling of end acceleration and end time is impractical due to the curse of dimensionality and the associated computational time. Therefore, the goal is to keep the number of samples low without compromising race performance.

The problem is addressed by determining a suitable pair of end acceleration $\hat{a}_{\mathrm{x},jm}$ and end time $t_{\mathrm{e},jm}$ for each spatial node $\mathfrak{n}_{\iota j}$ combined with a velocity sample V_m, avoiding additional sampling in the acceleration and time dimensions. The determined accelerations in Cartesian coordinates $\hat{a}_{\mathrm{x},jm}$ are then transformed into the corresponding Curvilinear accelerations $\ddot{s}_{jm}$ and $\ddot{d}_{jm}$, which serve as the end conditions for the lateral and longitudinal curves. Rearranging (A.15) to (A.22) results in the required transformation:

$$\begin{aligned}\ddot{s}(t_\mathrm{e}) = \ddot{s}_{jm} = \frac{1}{1-\Omega_\mathrm{z}(s_\iota)d_{\iota j}}\Big(&\hat{a}_{\mathrm{x},jm}\cos(\hat{\chi}_{\iota j}) - \hat{\kappa}_{\iota j}V_m\dot{d}_{jm} \\ &+ 2\Omega_\mathrm{z}(s_\iota)\dot{s}_{jm}\dot{d}_{jm} + \Omega'_\mathrm{z}(s_\iota)d_{\iota j}\dot{s}_{jm}^2\Big),\end{aligned} \tag{3.6}$$

$$\begin{aligned}\ddot{d}(t_\mathrm{e}) = \ddot{d}_{jm} = \frac{\ddot{s}_{jm}\dot{d}_{jm}}{\dot{s}_{jm}} + \frac{V_m}{\cos(\hat{\chi}_{\iota j})}&\left(\hat{\kappa}_{\iota j}V_m - \Omega_\mathrm{z}(s_\iota)\dot{s}_{jm}\right) \\ &- \tan(\hat{\chi}_{\iota j})\left(\Omega'_\mathrm{z}(s_\iota)d_{\iota j}\dot{s}_{jm}^2 + \Omega_\mathrm{z}(s_\iota)\dot{d}_{jm}\dot{s}_{jm}\right).\end{aligned} \tag{3.7}$$

These equations indicate that a curvature $\hat{\kappa}_{\iota j}$ must be defined at the end of an edge. However, as discussed in Subsection 3.1.1, spatial edges with a C^1-continuous transition are used, meaning the curvature at a spatial node is not unique. Consequently, the curvature $\Omega_\mathrm{z}(s_\iota)$ of the reference line at progress s_ι is chosen, independent of the lateral position $d_{\iota j}$ of the node being considered.

It now remains to determine the end acceleration $\hat{a}_{\mathrm{x},jm}$ and the corresponding end time $t_{\mathrm{e},jm}$ for each combination of spatial node $\mathfrak{n}_{\iota j}$ and velocity sample V_m. To avoid fluctuations in the acceleration profile and thus further minimize jerk, a constant acceleration is preferred. Consequently, the initial edges are generated close to a uniformly accelerating edge by selecting the uniform acceleration and the corresponding end time as end conditions. In contrast to the uniform acceleration approach with fixed spatial edges and uniform acceleration profiles, this method retains the advantage of individual paths, accounts for the start acceleration, and achieves jerk optimality, reducing the load on the tracking controller.

The uniform acceleration and time duration required to traverse a path of length $l_{jm} \in \mathbb{R}_{>0}$ from the velocity $\tilde{V}_0$ of the start state to the desired velocity sample V_m can be derived using the equations of uniformly accelerated motion. Accordingly, the end acceleration and the corresponding end time are determined by

$$t_{\mathrm{e},jm} = \frac{2 \cdot l_{jm}}{V_m + \tilde{V}_0} \qquad \text{and} \qquad \hat{a}_{\mathrm{x},jm} = \frac{V_m - \tilde{V}_0}{t_{\mathrm{e},jm}}. \tag{3.8}$$

For the evaluation of $t_{\mathrm{e},jm}$, the length l_{jm} of an edge is required. However, the length of an edge is not known before its generation, requiring an approximation to be determined. Since the paths of edges ending in the same spatial node tend to be of similar length, only one length per spatial node $\mathfrak{n}_{\iota j}$ is determined and utilized for all velocity samples V_m. A single edge is generated for each spatial node $\mathfrak{n}_{\iota j}$, with its length serving as an approximate value for l_{jm}. These edges are used solely to estimate the path lengths and are discarded after their generation. The end conditions of these temporary edges are chosen so that they terminate at the pose of the corresponding spatial node $\mathfrak{n}_{\iota j}$ and with the maximum velocity $V_{N_{\mathrm{V}}+N_{\tilde{\mathrm{V}}}}$. Further, the end acceleration is set to zero, and the end time is calculated according to (3.8), with the Euclidean distance between the position of the start state $\tilde{\boldsymbol{z}}_0$ and the considered spatial node $\mathfrak{n}_{\iota j}$ as length. The exact selection of these values is not crucial, as only the length of the edge is used.

The described choice of end accelerations $\hat{a}_{\mathrm{x},jm}$ and end times $t_{\mathrm{e},jm}$ leads to initial edges exhibiting uniformly accelerated motion characteristics while adhering to the initial conditions and maintaining jerk optimality, as illustrated in Figure 3.5. The velocity profiles of edges with similar start and end accelerations

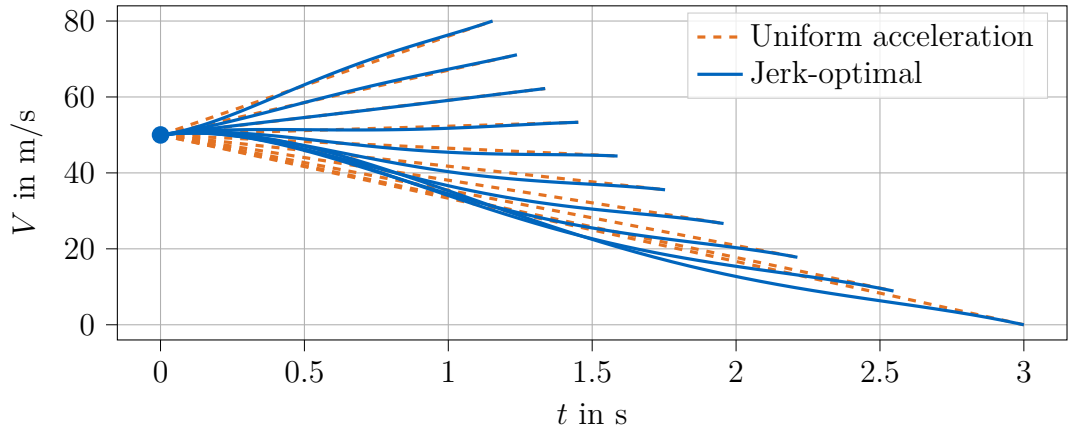


Figure 3.5: Comparison of the initial edges velocity profiles generated by the uniform acceleration approach and the hybrid planning approach, which is based on jerk-optimal curves. For illustration, ten end velocities V_m are sampled equidistantly from the interval $[0, 80]\,\mathrm{m/s}$. The start state $\tilde{\boldsymbol{z}}_0$ exhibits a velocity $\tilde{V}_0$ of $50\,\mathrm{m/s}$ (blue circle) and an acceleration of $10\,\mathrm{m/s^2}$.

closely approximate uniform acceleration. However, as the start acceleration differs from the end acceleration, the velocity profile deviates increasingly from the uniform acceleration in order to fulfill all boundary conditions.

The computation time for the proposed sampling-based initial edge generation can be reduced by precalculating the transformation of all sample points. Since the spatial nodes $\mathfrak{n}_{ij}$ of the graph and the velocity samples V_m are known offline, for each pair, the end conditions for position and velocity in Curvilinear coordinates can be precomputed using (3.4) and (3.5). In contrast, the end acceleration $\hat{a}_{\mathrm{x},jm}$ and end time $t_{\mathrm{e},jm}$ from (3.8) cannot be precomputed as the start velocity $\tilde{V}_0$ is not known offline. However, due to the computational complexity of (3.6) and (3.7), the end conditions $\ddot{s}_{jm}$ and $\ddot{d}_{jm}$ are precalculated offline for equidistantly sampled Cartesian accelerations at each pair of spatial node $\mathfrak{n}_{ij}$ and velocity sample V_m. Since the sampled acceleration values do not exactly match $\hat{a}_{\mathrm{x},jm}$ calculated online from (3.8), the end conditions of the sample that is closest to $\hat{a}_{\mathrm{x},jm}$ are used.

3.1.5 Graph Search

The third and final step, subsequent to the start state determination and the generation of the initial edges, is the graph search, in which the remaining edges $\mathfrak{E}_1$ are created, and the optimal sequence of spatio-temporal edges is searched for. As this step is not the focus of this thesis, only the core idea is explained. A detailed description and results showing the computation time improvements of the used method are provided in [153].

As illustrated in Figure 3.2, the graph search involves creating spatio-temporal edges of the set $\mathfrak{E}_1$, starting from the initial layer $\mathfrak{L}_\iota$. Following the approach in [74], these edges are generated by applying uniform acceleration profiles along the outgoing spatial edges of the corresponding spatial node $\mathfrak{n}_{\iota j}$. The start velocity and time of a spatio-temporal edge correspond to the V-t pair reached by the corresponding incoming spatio-temporal edge. The resulting discontinuity in the piecewise constant acceleration profile of the trajectory is not critical, as this trajectory segment is not traversed by the vehicle. To limit the computation time, only the zero acceleration and the feasible minimum and maximum accelerations are applied to a spatial edge, resulting in a coarse graph.

All edges, including the initial edges, are checked for validity according to Subsection 3.1.6. If an edge passes the checks, it is added to the graph and costs are assigned to it using the cost functional defined in Subsection 3.1.7. Rather than performing an exhaustive search that generates all possible spatio-temporal edges in a layer, a graph search based on Dijkstra's algorithm is employed, reducing the number of edges. Accordingly, the state with the lowest cumulative cost from the start state to itself (cost-to-come) is expanded. The search ends when the state with the lowest cost-to-come fulfills the goal condition of the minimum planning horizon T_{traj}. Furthermore, to reduce computation time, spatio-temporal edges whose end state lies within the same spatio-temporal node are considered similar. Of these edges, only the edge with the lowest cost-to-come is retained, and the rest are discarded.

3.1.6 Validity Checks

To ensure the validity of an edge, it is checked for feasibility, collision, and racing rule compliance. The checks may result in an insufficient number of valid

edges to compose a trajectory with the desired planning horizon. In such cases, a soft-checking concept becomes crucial, ensuring that a solution is available to guide the vehicle into a viable state. For instance, the trajectory with the fewest limit violations can be selected, or the violations can be penalized within the cost functional. As the detailed concept is not relevant to the results presented in this thesis, no further details are provided. For more information, refer to [151].

Feasibility Checks

As described in Subsection 3.1.1, each spatial edge within the graph is checked offline for kinematic feasibility according to (2.38). Therefore, only the initial edges require online verification. In contrast, each spatio-temporal edge is checked online for dynamic feasibility according to (2.39).

Collision Checks

Each spatio-temporal edge is checked for collision with the track boundaries and the respective temporal part of the predictions. As the spatial edges within the graph are known offline, the collision check with the track boundaries is also performed offline for these edges. Since the computation time is not decisive here, the vehicle geometry is accurately represented by oriented rectangles. To avoid a time-consuming check in Cartesian coordinates for the initial edges generated online, a simplified check is conducted in Curvilinear coordinates using the vehicle width D_w:

$$d_\mathrm{r}(s(t)) + \frac{D_\mathrm{w}}{2} + D_\mathrm{saf} \leq d(t) \leq d_\mathrm{l}(s(t)) - \frac{D_\mathrm{w}}{2} - D_\mathrm{saf}. \tag{3.9}$$

Since this check only considers the lateral position and vehicle width, it neglects the vehicle's relative orientation and the resulting change in its occupied area. Therefore, the check is only exact when the vehicle's orientation along the initial edge matches that of the reference line, i.e., $\hat{\chi}(t) = 0°$. However, this assumption can be compensated for by adding the safety distance D_saf.

In addition to the described collision check with the track boundaries, a collision check between an edge and an opponent's prediction must also be performed. This check follows a hierarchical structure that avoids an exact,

computationally expensive check of each edge. The check is divided into the following three stages, which increase in accuracy and thus computation time:

1. Spatial stage: Check whether the paths of the examined edge and the respective prediction segment collide. Each path, including the vehicle geometries along it, is overapproximated by a single oriented rectangle.

2. Coarse temporal stage: Check whether there is a collision at any point in time along the examined edge between the ego and opponent vehicle, with both being overapproximated with circles.

3. Exact temporal stage: Check whether there is a collision at any point in time along the examined edge between the ego and opponent vehicle, with both being accurately represented by oriented rectangles.

All stages are performed with the safety distance D_{saf}. If no collision is detected in one stage, the more computationally demanding subsequent stages are skipped. While this reduces the average computation time for the collision checks, the required time in a single planning cycle can vary depending on the prediction. In the worst case, all three stages must be executed for each edge, potentially leading to excessive computation times. To mitigate this, instead of checking for collisions over the entire planning horizon T_{traj}, the check is performed only for a shorter horizon $T_{\text{col}} \in (0, T_{\text{traj}}]$. Beyond this horizon, the edges solely consider the prediction through the cost functional in the following subsection, avoiding discarding an edge.

Racing Rule Compliance Checks

To ensure adherence to the rules, the edges are checked for compliance with racing regulations. These include adhering to the speed limit $V_{\text{limit}}(t)$ and avoiding overtaking in designated no-passing zones. To comply with the speed limit, only edges with a velocity profile below the current limit are retained. However, if an edge's initial velocity already exceeds the limit, the edge is also preserved, provided the average acceleration of the edge is negative, i.e., it is a decelerating edge. To prevent overtaking in no-passing zones, edges with an end progress greater than the prediction's end progress but a smaller start progress are discarded. Additionally, to maintain a target distance to the front

opponent vehicle, the target velocity $V_{\text{follow}}(t) \in \mathbb{R}_{>0}$ in the cost functional is varied depending on the actual distance, as described in Subsection 3.1.7.

3.1.7 Cost Functional Design

If a spatio-temporal edge with the start time $t_0 \in [0, T_{\text{traj}})$ and the end time $t_{\text{e}} \in (0, T_{\text{traj}}]$ is valid, it is assigned a cost that forms the foundation of the search for the optimal edge sequence. The used cost functional is introduced in detail in [153] and is briefly repeated here for the sake of completeness. The cost functional is designed so that the resulting trajectory follows the racing line and, if necessary, overtaking or evasive maneuvers are carried out. Accordingly, the cost functional $J \in \mathbb{R}_{\geq 0}$ consists of four terms, each weighted by a parameter $c_{\square} \in \mathbb{R}_{\geq 0}$, which allow for balancing between racing line following and collision avoidance:

$$J = \int_{t_0}^{t_{\text{e}}} \left(c_{\text{d}} \cdot |\Delta d(t)| + c_{\text{V}} \cdot \Delta V^2(t) + c_{\hat{\kappa}} \cdot \hat{\kappa}^2(t) + c_{\text{pr}} \cdot \sum_{l=1}^{N_{\text{opp}}} J_{\text{pr},l}(t) \right) \mathrm{d}t. \quad (3.10)$$

The first term penalizes lateral deviations $\Delta d(t) = \tilde{d}_{\text{rl}}(t) - d(t) \in \mathbb{R}$ from the racing line. Instead of $d_{\text{rl}}(t)$, $\tilde{d}_{\text{rl}}(t)$ is used, representing the racing line lateral position not at the corresponding racing line progress $s_{\text{rl}}(t)$ but at the progress $s(t)$ of the edge. This decouples the penalty for lateral position deviations from a possible velocity deviation, which is penalized in the next term. Further, while all other terms enter the cost functional quadratically, the lateral deviation only has a linear influence, as this reduces the costs for larger deviations, providing greater flexibility for overtaking maneuvers. The second term penalizes deviations $\Delta V(t) = V_{\text{target}}(t) - V(t) \in \mathbb{R}$ from the target velocity $V_{\text{target}}(t) \in \mathbb{R}_{>0}$. Here, $V_{\text{target}}(t)$ is calculated as the minimum of the racing line velocity $V_{\text{rl}}(t)$, the speed limit $V_{\text{limit}}(t)$, and, if overtaking is not permitted, a velocity $V_{\text{follow}}(t)$, which is selected such that a longitudinal target distance to the opponent vehicle in front is maintained. If the actual distance is less than the target distance, $V_{\text{follow}}(t)$ is set to be lower than the opponent's average velocity. Conversely, if the actual distance exceeds the target distance, $V_{\text{follow}}(t)$ is set to be greater. The exact calculation of $V_{\text{follow}}(t)$ is not described further in this thesis, as it is not decisive for the results. For details, please refer to [151] instead. The third term penalizes the curvature of an edge, resulting in

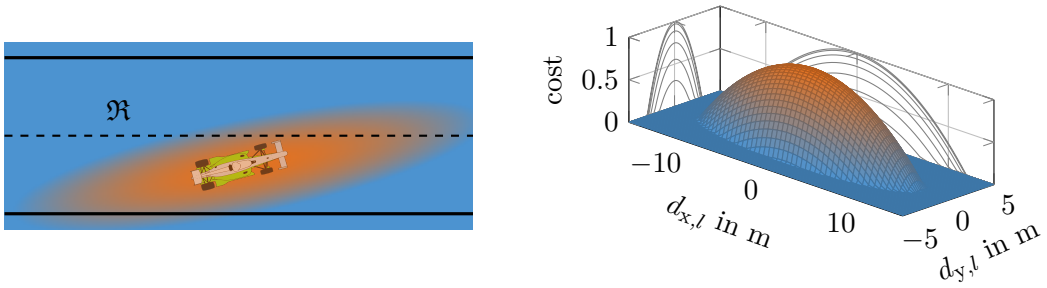


Figure 3.6: Visualization of the elliptical prediction cost distribution aligned with the heading of the opponent vehicle for a single point in time. Left: Illustration of the cost distribution on a straight. Right: Cost distribution for $p_\mathrm{x} = 12\,\mathrm{m}$ and $p_\mathrm{y} = 3\,\mathrm{m}$.

trajectories with low lateral motion and, thus, low lateral accelerations. Lastly, the fourth term becomes effective if a deviation from the racing line is required due to opposing vehicles indexed by l. It relates to an elliptical cost distribution $J_{\mathrm{pr},l}(t) \in \mathbb{R}_{\geq 0}$ that covers the prediction of the respective opponent vehicle at each point in time t, penalizing edges close to the prediction:

$$J_{\mathrm{pr},l}(t) = \max\left\{1 - \left(\frac{d_{\mathrm{x},l}^2(t)}{p_\mathrm{x}^2(t)} + \frac{d_{\mathrm{y},l}^2(t)}{p_\mathrm{y}^2(t)}\right),\ 0\right\} \cdot h(t). \tag{3.11}$$

$d_{\mathrm{x},l}(t) \in \mathbb{R}_{\geq 0}$ and $d_{\mathrm{y},l}(t) \in \mathbb{R}_{\geq 0}$ correspond to the longitudinal and lateral distance between the ego and the prediction of opponent vehicle l from the opponent vehicle's perspective. This causes the elliptical cost distribution to be aligned with the opponent, as shown in Figure 3.6. The boundary of the cost distribution is defined by the time-varying parameters $p_\mathrm{x}(t) \in \mathbb{R}_{>0}$ and $p_\mathrm{y}(t) \in \mathbb{R}_{>0}$. Outside the ellipse parameterized by these values, the costs are bounded below by 0, meaning that no costs arise in this region. Generally, $p_\mathrm{x}(t) > p_\mathrm{y}(t)$ is selected due to the higher uncertainty in the prediction of the longitudinal movement. Furthermore, the values $p_\mathrm{x}(t)$ and $p_\mathrm{y}(t)$ and, thus, the size of the ellipses increase with the planning horizon to reflect the increasing uncertainty of the prediction. However, since the elliptical distributions enlarged with the horizon can cover a large part of the track, conservative behavior arises, preventing aggressive overtaking maneuvers. To counteract this, the costs are scaled with $h(t) \in \mathbb{R}_{\geq 0}$, which decreases linearly with the horizon.

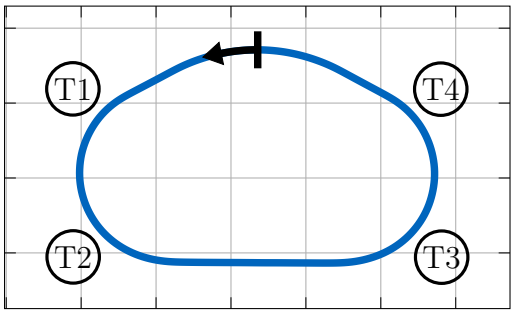


Figure 3.7: Layout of the LVMS. The arrow indicates the direction of travel and the $s = 0\,\mathrm{m}$ position. One grid cell equals 200 m in width and height.

3.2 Results

Before the actual results, Subsection 3.2.1 provides computational details on the numerical realizations of the hybrid local planning approach and the concepts of the adjacent modules. This is followed by simulative results in Subsection 3.2.2, in which the presented sampling-based initial edge generation is compared with the uniform acceleration approach regarding performance and vehicle stability. Finally, Subsection 3.2.3 gives experimental results from the final race of the IAC on 7 January 2022, demonstrating the general idea of the proposed hybrid planning approach.

All experiments are carried out on the Las Vegas Motor Speedway (LVMS), visualized in Figure 3.7. It is a 2480 m long oval track with a curved frontstretch, a straight backstretch, and four turns banked up to 20°. However, as mentioned at the beginning of this chapter, the three-dimensionality of the track is not accounted for in this approach.

3.2.1 Computational Details

The hybrid local planning approach is implemented in Python with individual C-functions of computationally time-consuming operations such as collision checks. As described in Subsection 2.5.2, each planned trajectory is represented by N_{traj} equidistantly distributed discrete points in time. These points serve as integration points for the numerical approximation of the cost functional (3.11) using the rectangle method. The validity checks are performed only for the

Table 3.1: Parameters for the results of the hybrid planning approach (descriptions are provided in the list of symbols on page xvii)

Parameter	Value	Unit
D_w	1.93	m
D_s	75	m
D_d	1.4	m
N_V	20	-
N_t	10	-
$N_\underline{\mathrm{V}}$	20	-
$N_{\bar{\mathrm{V}}}$	30	-
$V_{N_\underline{\mathrm{V}}}$	25	m/s
$V_{N_\underline{\mathrm{V}}+N_{\bar{\mathrm{V}}}}$	85	m/s
$\Delta s_\mathrm{min}(\tilde{V}_0)$	10 to 50	m
N_traj	50	-
T_traj	5	s
T_col	2	s

Parameter	Value	Unit
D_saf	0.5	m
$\hat{\kappa}_\mathrm{max}$	0.1	m^{-1}
$\tilde{a}_\mathrm{y,max}(V)$	18 to 40	$\mathrm{m/s}^2$
$\tilde{a}_\mathrm{x,min}(V)$	-13 to -7	$\mathrm{m/s}^2$
$\tilde{a}_\mathrm{x,max}(V)$	3 to 5.4	$\mathrm{m/s}^2$
$\rho(V)$	1	-
c_d	5	-
c_V	5	-
$c_{\hat{\kappa}}$	$5 \cdot 10^5$	-
c_pr	$1 \cdot 10^3$	-
$p_\mathrm{x}(t)$	30	m
$p_\mathrm{y}(t)$	4 to 5	m

discrete points, and the velocity-dependent parameters representing the gg-diagrams are determined through simulation and adjusted empirically. All parameters used for the experiments are listed in Table 3.1.

All results in this section are obtained using the sequential asynchronous software architecture shown in Figures 1.2 and 1.3a as part of the TUM Autonomous Motorsport team's software stack. The Kalman filter proposed in [154] based on a 2D point-mass model is used for state estimation. For object detection, various pipelines exist from different sensor modalities. In the races of the experimental data shown here, only LiDAR clustering and a RADAR pipeline for long-range detection were used [155]. The different object lists are merged using the multi-modal sensor fusion and tracking method from [156]. It utilizes an extended Kalman filter built on a point-mass model, assuming a constant turn rate and velocity. The prediction of the opponent vehicles is essentially based on the assumption of a constant lateral distance to the track boundaries and a constant velocity [157]. For global planning, the free-path approach based on a two-track model from [55] is used. Finally, the planned trajectory is tracked

Table 3.2: Lap time comparison for a standing start and a flying lap on the LVMS

	Lap time	
Approach	Standing start lap	Flying lap (mean / variance)
Uniform acceleration	47.814 s	30.431 s / 0.001 s^2
Hybrid planning	51.854 s	30.001 s / 0.004 s^2

by a hierarchical controller consisting of a high-level Tube-MPC and low-level PI controllers, including a feed-forward law. The Tube-MPC, described in [158], is based on a point-mass model and reoptimizes the planned trajectory within a safe[1] spatial tube. The resulting longitudinal and lateral acceleration profiles are tracked by the low-level controllers tuned to the specific race car. A more detailed description of the modules and the vehicle hardware can be found in [30].

3.2.2 Simulative Results

The proposed hybrid planning approach is compared with the uniform acceleration approach from [74], where the initial edges $\mathfrak{E}_0$ are generated identically to the other edges $\mathfrak{E}_1$. For comparability, $N_{\underline{\mathrm{V}}} + N_{\bar{\mathrm{V}}} = 50$ constant acceleration profiles are applied for each generated spatial edge in the uniform acceleration approach, resulting in the same number of initial edges as in the hybrid planning approach. The influence of both approaches on performance and vehicle stability is evaluated through experiments involving single-vehicle driving, as well as braking and evasive maneuvers.

Single-Vehicle Driving

Single-vehicle driving along the racing line without speed restrictions is a crucial element in an autonomous racing series and a suitable performance indicator. Starting from a standstill and proceeding with twenty flying laps, the resulting lap times are summarized in Table 3.2. Figure 3.8 shows the velocity and lateral

[1] The collision-free condition of the tube is ensured by artificially enlarged vehicle geometries in the collision check.

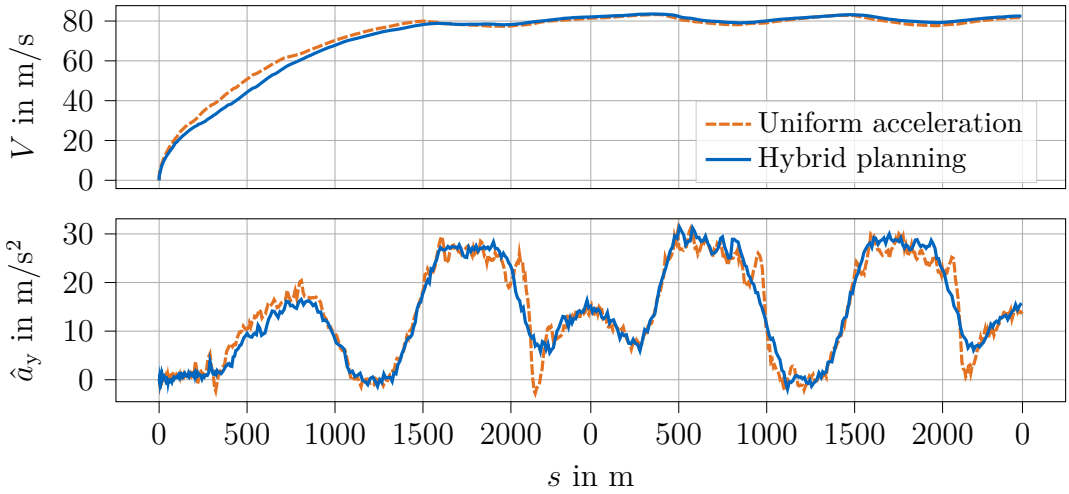


Figure 3.8: Comparison between the hybrid planning and uniform acceleration approach for a standing start and flying laps on the LVMS.

acceleration profiles for the standing start and the first flying lap (beginning at the second occurrence of $s = 0\,\mathrm{m}$). The hybrid planning approach performs worse than the uniform acceleration approach at the standing start, resulting in a higher lap time. This is due to the uniform acceleration approach's capability to immediately generate high acceleration profiles despite the initial zero acceleration. In contrast, the hybrid planning approach adheres to the initial condition and gradually transitions from zero to higher accelerations. However, as the races of the IAC only involve flying starts, this limitation in standing starts is not further addressed.

For flying laps, the hybrid planning approach achieves higher speeds, particularly in turns, leading to reduced lap times. This advantage stems from the simultaneous generation of spatial and temporal information, which allows for higher speeds for a given lateral acceleration. The lateral acceleration profile further indicates that the hybrid planning approach provides smoother vehicle handling in turns. Notably, at the exit of the fourth turn at $s = 2200\,\mathrm{m}$, the uniform acceleration approach exhibits sharp deflections, even causing negative lateral accelerations in a turn with positive curvature. Since the uniform acceleration approach leads to higher lateral accelerations for a given speed, edges of higher curvature are infeasible, leaving only slightly curved edges.

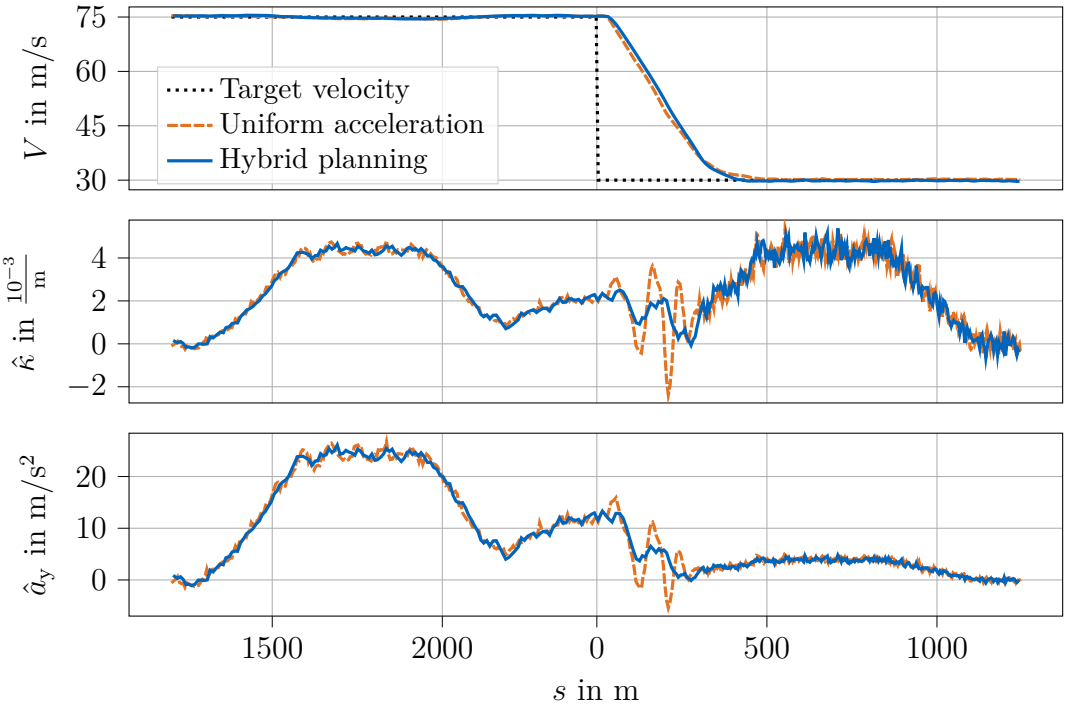


Figure 3.9: Comparison between the hybrid planning and uniform acceleration approach for a braking maneuver on the frontstretch of the LVMS.

Braking Maneuver

Decelerating to a complete stop or a lower speed is crucial to race track operations. For instance, after a high-speed lap, it is necessary to reduce speed in order to safely enter the pit lane. At the same time, controlled braking is also essential for safety reasons, e.g., to avoid collisions. Figure 3.9 illustrates the resulting velocity, curvature, and lateral acceleration profiles in an example braking maneuver, where the target speed V_{target} is abruptly reduced from 75 to 30 m/s. While the velocity profiles during deceleration appear similar for both approaches, the uniform acceleration method exhibits significant oscillations in the curvature and lateral acceleration profiles. These lateral deflections grow as the start and end speed difference increases, potentially leading to unstable vehicle behavior.

Further investigations show that the lateral oscillations are noticeably reduced when performing this maneuver with idealized tracking rather than using a

tracking controller. This indicates that the oscillations are not inherently planned by the uniform acceleration approach but instead emerge from the interference between the planned trajectories and the Tube-MPC, which has difficulty tracking the coarse trajectories even with reoptimization. In contrast, the hybrid planning approach generates smoother initial edges that are easier for the controller to track, resulting in more stable vehicle behavior.

Object Evasion

Static obstacles on the race track require rapid and precise planning of evasive maneuvers to avoid collisions. This relates not only to stopped opponent vehicles but also to objects intentionally placed on the track as part of the racing competition format. Part of the IAC race on 23 October 2021 was to navigate around two static obstacles at a minimum speed of 28 m/s. The two obstacles were positioned longitudinally at a distance of 100 m and blocked opposite sides of the track. This evasive scenario is replicated below on the straight backstretch of the LVMS using the hybrid planning and the uniform acceleration approach. The backstretch is 15 m wide, and the objects are placed 5.5 m away from the respective track boundaries. As the object evasion difficulty increases with higher ego velocities, the scenario is tested at speeds ranging from 25 to 65 m/s in increments of 5 m/s. To account for the simulation's non-determinism caused by hardware-related variations in module execution times, each run is repeated twenty times. Additionally, the detection range plays a critical role in the complexity of the maneuver. To assess this, behaviors are compared at lower and higher detection ranges of 100 m and 200 m, respectively. It is important to note that the racing line is laterally on one side of the track, requiring an outward and inward movement.

All evasive maneuvers tested were successful, meaning they were collision-free. Beyond this success rate, the average absolute control error between the reoptimized target from the Tube-MPC and the actual lateral acceleration $\hat{a}_\mathrm{y}$ is an important metric, indicating vehicle stability. Figure 3.10 shows this error for the different ego velocities and sensor ranges. Notably, the mean absolute $\hat{a}_\mathrm{y}$ error increases with higher speeds. Moreover, with a detection range of 200 m, the hybrid planning approach results in lower control errors than the uniform acceleration approach, particularly at higher speeds. However, with a detection range of 100 m, although the hybrid planning approach still leads to lower errors

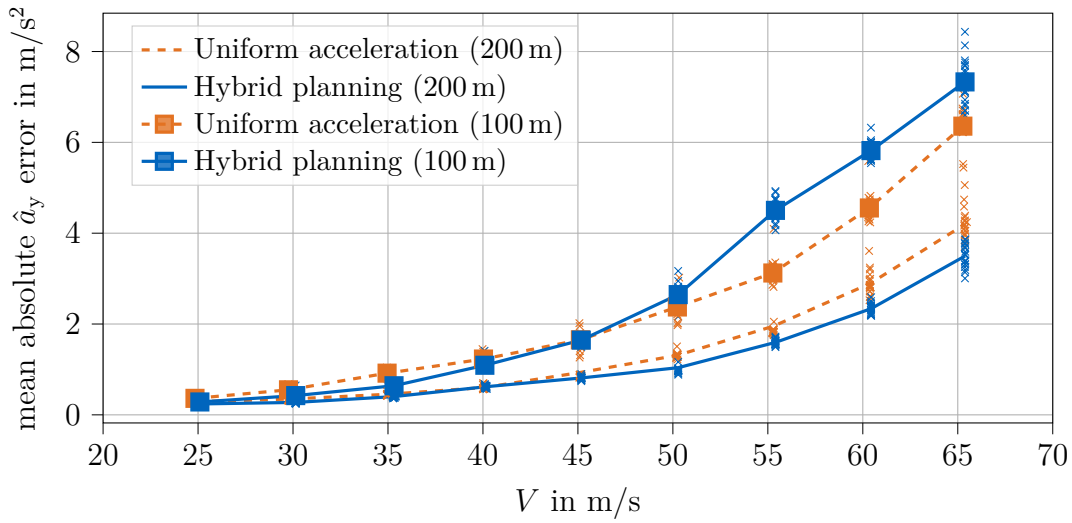


Figure 3.10: Comparison between the hybrid planning and uniform acceleration approach for an evasion maneuver on the backstretch of the LVMS.

at lower speeds, this changes at velocities above 45 m/s. This shift is due to the faster computation time of the uniform acceleration approach, enabling quicker reactions to environmental changes. In contrast, the higher computation time of the hybrid planning approach reduces the available distance for reaction, leading to higher lateral accelerations and larger control errors. It should be noted that these results are specific to the particular evasion scenario and cannot be generalized across all maneuvers. The behavior depends on the relative positions of the obstacles and the fixed layers in the spatial graph, which may influence the overall planning outcome.

3.2.3 Real-World Application

The proposed hybrid planning approach was deployed on a full-scale autonomous race car in various IAC races across different oval tracks from 2021 to early 2023. The competitions included single-vehicle and two-vehicle racing, along with showcases with up to three vehicles. Across all races, the Dallara AV-21 race car (Figure 1.1a), based on a modified Indy Lights chassis, was used. The vehicle hardware included an Intel Xeon E-2278GE CPU with 8 cores and 64 GB RAM, with the local planning approach utilizing a single core. With this setup,

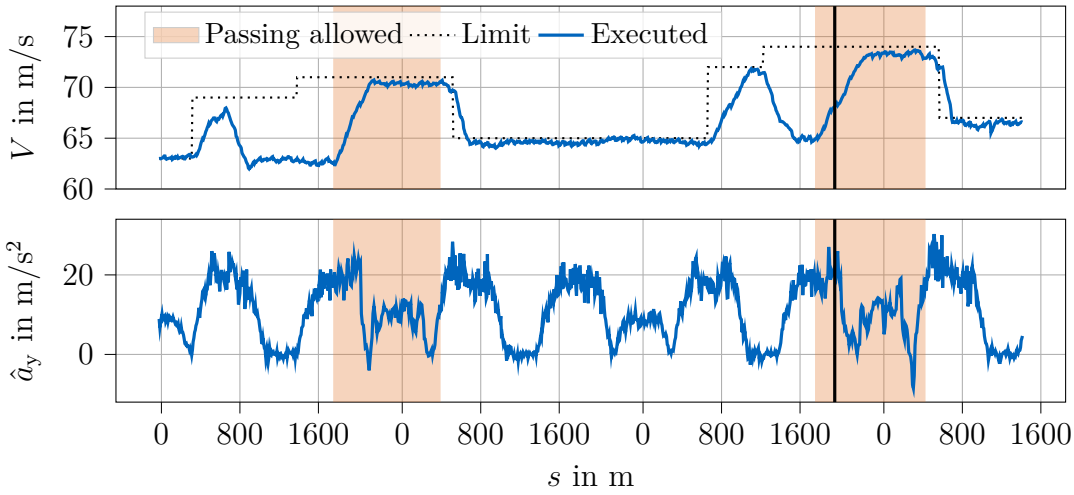


Figure 3.11: Executed velocity and lateral acceleration of the last two overtaking maneuvers during the final of the IAC competition on 7 January 2022 at the LVMS. The permitted passing period is marked in red. The black vertical line indicates the planning cycle shown in Figure 3.12.

the hybrid planning approach demonstrated an average computation time of 95 ms per planning cycle using the discretization stated in Table 3.1.

This subsection presents only results from the final race of the IAC competition on 7 January 2022 at the LVMS. In this final, two vehicles alternated in performing overtakes at progressively higher speeds. The vehicle to be overtaken, referred to as the defender, had to drive at a certain speed on the inside of the oval track. The overtaking vehicle, referred to as the attacker, had to maintain a longitudinal distance of more than 30 m from the defender. Once the designated overtaking zone has been entered, the overtaking maneuver must be completed within two laps. After the first team completed an overtake, the roles were swapped, and the second team was required to perform an overtake at the same defender speed. Once both teams completed the overtake, the defender's speed was increased until an overtaking maneuver was unsuccessful or one of the teams gave up.

Figure 3.11 shows the velocity and lateral acceleration profiles executed during the last two overtaking maneuvers by the TUM Autonomous Motorsport team. The figure also indicates the speed limit V_{limit} and the zones where overtaking is allowed. Before the two overtaking maneuvers, the defender is followed at

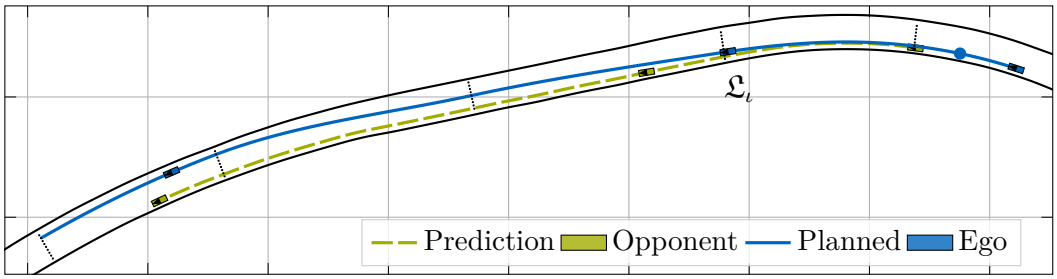
the respective speed, which prevents reaching the speed limit. When passing is permitted, the vehicle initiates the overtaking maneuver and accelerates to the speed limit V_{limit}.

Figure 3.12 presents the planned trajectory at the time step marked in Figure 3.11. The defender is predicted to have a constant velocity of approximately 65 m/s along the inner part of the track. Since the relative progress between the determined start state and the next layer is smaller than $\Delta s_{\min}(\tilde{V}_0)$, the second next layer is selected as the initial layer $\mathfrak{L}_\iota$. The initial edge connects the start state to the chosen initial layer over a horizon of 1.7 s, and the minimum planning horizon of $T_{\text{traj}} = 5\,\text{s}$ is achieved by the subsequent edges obtained from the graph search. The resulting trajectory initiates an overtaking maneuver, influenced by the cost functional, which penalizes paths close to the prediction. For the initial edge, a smooth transition of the longitudinal acceleration profile is evident, which leads to the advantageous behavior in terms of performance and vehicle stability shown in Subsection 3.2.2. In contrast, the trajectory beyond 1.7 s shows a rough course with discontinuities in the acceleration profile caused by the uniformly accelerated edges.

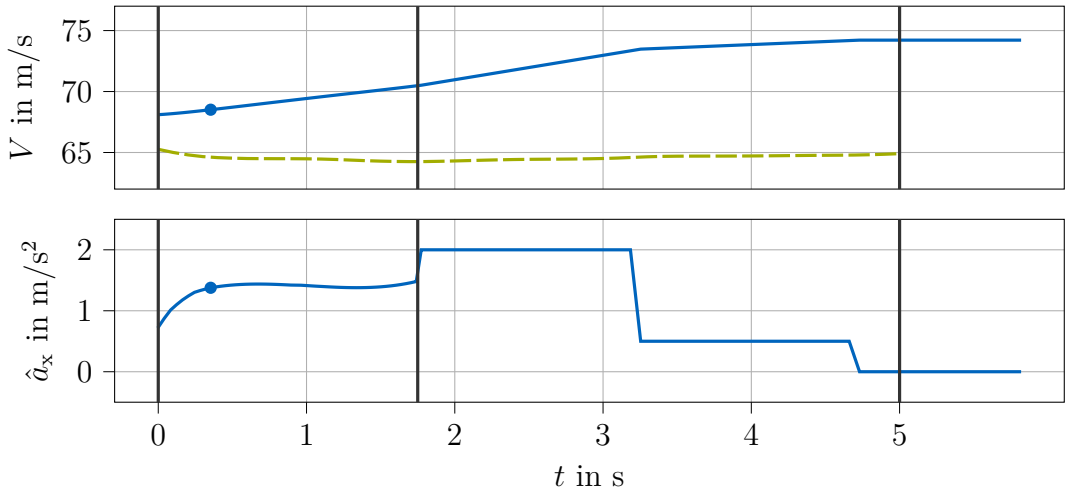
3.3 Discussion

The hybrid local trajectory planning approach represents the continuation of the previous planning approach of the TUM Autonomous Motorsport team. While the previous approach consisted of graph-based path planning [70] with a subsequent velocity profile generation [47], the proposed hybrid planning approach is the first graph-based planning approach for racing without PVD. The simultaneous consideration of the spatial and temporal domain enables planning without the predictions blocking large areas of the track, leading to more competitive behavior. The hybrid approach of sampling-based initial edge generation and a subsequent coarse graph search allows for a long planning horizon and simultaneously high smoothness of the traversed trajectory segment, improving performance and vehicle stability.

Despite the advantages offered by the hybrid planning approach, certain limitations remain. Firstly, the spatial graph is fixed, meaning the local planning depends on the vehicle's relative position to the layers. Thus, different trajectories can be planned at various locations on the race track, even though



(a) Planned and predicted path. One grid cell equals 50 m in width and height. The spatial nodes of the graph are depicted in black.



(b) Predicted opponent velocity as well as planned ego velocity and longitudinal acceleration.

Figure 3.12: Planned overtaking maneuver on the frontstretch of the LVMS. The trajectory is from the planning cycle marked in Figure 3.11. The start state $\tilde{\boldsymbol{z}}_0$ is indicated by blue circles. The poses along the prediction and the planned path are visualized in Figure 3.12a for the beginning of the trajectory, the end of the initial edge, and the minimum planning horizon of $T_{\text{traj}} = 5\,\text{s}$. The corresponding times are marked as black vertical lines in Figure 3.12b.

Figure 3.13: Spin at the final of the IAC competition on 7 January 2022 at the LVMS. *Image Credits: Indy Autonomous Challenge*

the scenario remains the same. Also, since only the end conditions of the initial edges can be specified, stopping is only possible at the spatial nodes without further modification of the concept. Secondly, the approach is not suited for road courses with sharply curved sections. These sections involve significant changes in acceleration, requiring smaller layer spacings to avoid infeasibility issues with the constant acceleration edges $\mathfrak{E}_1$. However, this would require searching through more layers for the same planning horizon, leading to insufficiently high computation times. Thirdly, as mentioned at the beginning of this chapter, the 3D effects of the track are not taken into account. The gg-diagrams are only speed-dependent and are, therefore, the same on the flat frontstretch and the 20° banked turns. To avoid any performance loss, the gg-diagrams were adjusted to the banked turns, resulting in an overestimation of the real dynamic potential on the flat sections. However, problems can arise in critical, unforeseen situations when infeasible accelerations are planned. For instance, in the final race of the IAC competition on 7 January 2022, several opponent vehicles were mistakenly detected near the ego vehicle on the frontstretch. The planning module initiated a highly dynamic evasive maneuver, resulting in the spin shown in Figure 3.13. Fourthly, the racing line does not incorporate the current state of the ego vehicle, potentially leading to suboptimal decisions. Deviating from the offline-generated racing line results in a new time-optimal trajectory, which is potentially better suited as a reference for local planning. Lastly, the vehicle interaction is not accounted for due to the conventional planning structure of

sequential prediction and planning from Figure 1.3a. As the strict racing rules in the previous IAC races allowed only limited interaction, a reliable prediction of the opponents was possible. However, interaction-aware planning may be required if the racing rules become more relaxed in future competitions.

4 Sampling-Based Planning on Three-Dimensional Road Courses

This chapter introduces a sampling-based local trajectory planning approach for 3D road courses, addressing limitations 2, 3, and 4 from Section 1.4. In existing sampling-based approaches for racing, such as [31, 32, 96], jerk-optimal trajectory candidates are generated, according to [92]. These approaches have been used successfully on simple oval tracks in the IAC. However, they are unsuitable for road courses, as the simplicity of the jerk-optimal trajectory candidates prevents the replication of the more complex racing line geometry. Therefore, the planning approach presented here includes a revised trajectory candidate generation method that enables tracking of the racing line on any road course. Also, in contrast to the hybrid planning approach, the 3D effects of the race tracks are accounted for through the gg-diagrams, preventing underestimation or overestimation of the actual dynamic limits. Furthermore, the impact of online replanning of the racing line, starting at the current ego vehicle state, on overall behavior is investigated. For this, the offline-executed global planning submodule from Figure 1.3a is replaced by an online local planning approach with a limited horizon and without accounting for opponent vehicles.

The last two mentioned contributions of 3D effect consideration and online replanning of the racing line could also be applied to the hybrid planning approach from Chapter 3. However, comparisons with the hybrid planning approach are not evaluated, as the approach is not applicable to real vehicles on road courses due to the excessive computation times that would be required. Instead, experiments are carried out using the proposed sampling-based approach with and without 3D effect consideration and online replanning of the racing

line. In addition, the sampling-based approach presented here is compared with the widely used sampling-based approach from [92].

This chapter is based on the author's publication [159]. In addition, it contains a more detailed description of the racing line adaption and the reference line choice. Moreover, it provides additional results from the A2RL competition from 27 April 2024. The sampling-based approach is explained methodologically in Section 4.1, followed by simulative and experimental results in Section 4.2 and a brief discussion in Section 4.3.

4.1 Methodology

Subsection 4.1.1 explains the underlying planning procedure of the sampling-based approach. This is followed by the individual subtasks of trajectory generation in Subsection 4.1.2, trajectory validation in Subsection 4.1.3, and trajectory selection in Subsection 4.1.4.

4.1.1 Planning Procedure

The sampling-based local trajectory planning approach consists of the subtasks depicted in Figure 4.1. The first step is to determine the start state $\tilde{\boldsymbol{z}}_0$ from the estimated ego vehicle state $\boldsymbol{z}_0$ for the current planning cycle. This step is the same as in the hybrid planning approach in Subsection 3.1.3 and is therefore not repeated here. Second, a set of trajectory candidates is generated by sampling several end states and connecting them with the determined start state. In the third and fourth steps, each trajectory candidate is checked for validity, and the optimal one is selected from the valid trajectory candidates using a cost functional.

In the subtask of trajectory generation, candidates must be generated that are valid and, ideally, minimize the cost functional used for selection. Since the cost functional penalizes deviations from the racing line $(s_{\mathrm{rl}}(t), d_{\mathrm{rl}}(t))$, trajectory candidates must be generated that at least resemble it. On oval tracks, the racing line usually has constant velocities and small curvatures, so the jerk-optimal trajectory candidates used in [31, 32, 92, 96] can sufficiently replicate it. However, performance and validity issues can arise on complex road courses since the jerk-optimal candidates do not replicate the racing line or are not even

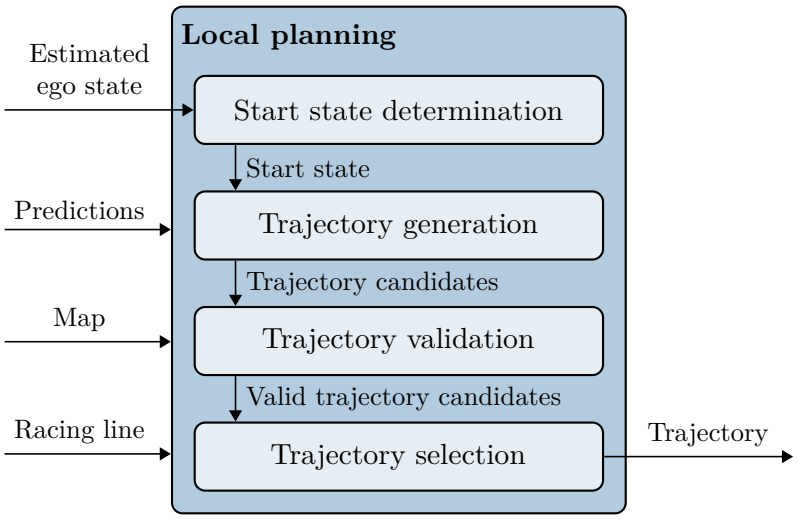


Figure 4.1: Procedure of the sampling-based planning approach.

valid. Therefore, the proposed sampling-based approach includes racing line information in the trajectory generation.

4.1.2 Trajectory Generation

Given the start state $\tilde{\boldsymbol{z}}_0 = \begin{bmatrix} \tilde{s}_0 & \dot{\tilde{s}}_0 & \ddot{\tilde{s}}_0 & \tilde{d}_0 & \dot{\tilde{d}}_0 & \ddot{\tilde{d}}_0 \end{bmatrix}^\top$ of a planning cycle, multiple end states are sampled. The start state is then connected to the sampled end states with a fixed horizon T_{traj}, resulting in a set of trajectory candidates. The generation process is divided into creating a set of longitudinal curves $\mathfrak{T}_{\text{long}}$ followed by a set of lateral curves $\mathfrak{T}_{\text{lat}}$. The respective curves are then combined in pairs, forming the set of trajectory candidates.

Longitudinal Curve Generation

The set $\mathfrak{T}_{\text{long}}$ consists of $2 \cdot N_{\dot{\text{s}}} \in \mathbb{N}$ longitudinal curves $s_i(t)$, $t \in [0, T_{\text{traj}}]$ indexed by $i \in \{1, 2, \ldots, 2 \cdot N_{\dot{\text{s}}}\}$. Since all trajectory candidates originate from the start state $\tilde{\boldsymbol{z}}_0$, they share the same initial conditions:

$$\begin{bmatrix} s_i(0) & \dot{s}_i(0) & \ddot{s}_i(0) \end{bmatrix}^\top = \begin{bmatrix} \tilde{s}_0 & \dot{\tilde{s}}_0 & \ddot{\tilde{s}}_0 \end{bmatrix}^\top. \tag{4.1}$$

Sampling the quantities at the end time T_{traj} poses a trade-off. On the one hand, sampling should be applied to each dimension so that many diverse trajectory candidates are available for trajectory selection. On the other hand, generating trajectory candidates that would not be selected should be avoided to reduce the computation time. Since the specification of an exact progress on the race track is not decisive in racing, the longitudinal end position $s(T_{\mathrm{traj}})$ is not fixed but is chosen to be free. Hence, the end state is constrained to a manifold, and the end position is set automatically, avoiding sampling in this dimension. Instead, the naturally more essential velocity dimension is sampled. For this, $N_{\dot{\mathrm{s}}}$ equidistantly distributed samples are generated in the interval $\dot{s}_{\mathrm{e},i} \in \left[0, \max\left(\dot{\tilde{s}}_0, \dot{s}_{\mathrm{rl}}\left(T_{\mathrm{traj}}\right)\right) \cdot K_{\dot{\mathrm{s}}}\right]$, additionally containing the end racing line velocity $\dot{s}_{\mathrm{rl}}\left(T_{\mathrm{traj}}\right)$. To allow for necessary deviations from the racing line, the range is expanded by the factor $K_{\dot{\mathrm{s}}} > 1$. For computation time reasons, however, no further sampling is carried out in the acceleration dimension, and instead, an acceleration value $\ddot{s}_{\mathrm{e},i}$ is determined for each velocity sample $\dot{s}_{\mathrm{e},i}$. The acceleration $\ddot{s}_{\mathrm{e},i}$ associated with the velocity sample on the racing line should correspond to the acceleration of the racing line $\ddot{s}_{\mathrm{rl}}(T_{\mathrm{traj}})$ to enable tracking of it. However, if the examined curve deviates significantly from the racing line, the racing line acceleration may be unsuitable as the end condition. To generate reasonable trajectories even for larger deviations, the end acceleration is linearly interpolated between $0\,\mathrm{m/s^2}$ and $\ddot{s}_{\mathrm{rl}}(T_{\mathrm{traj}})$ based on the proximity of $\dot{s}_0$ to $\dot{s}_{\mathrm{rl}}(0)$ and $\dot{s}_{\mathrm{e},i}$ to $\dot{s}_{\mathrm{rl}}(T_{\mathrm{traj}})$. Therefore, the end conditions for the longitudinal curves are

$$s(T_{\mathrm{traj}}) \text{ free} \quad \text{and} \quad \begin{bmatrix} \dot{s}(T_{\mathrm{traj}}) & \ddot{s}(T_{\mathrm{traj}}) \end{bmatrix}^\top = \begin{bmatrix} \dot{s}_{\mathrm{e},i} & \ddot{s}_{\mathrm{e},i} \end{bmatrix}^\top. \tag{4.2}$$

The initial condition (4.1) must now be connected to the sampled end conditions (4.2). For this, jerk-optimal curves are often used as in [31, 32, 92, 96]. As proven in Appendix B (case 2), the jerk-optimal connection corresponds to quartic (fourth-order) polynomials for the present case of a free end position. The resulting jerk-optimal curves reduce abrupt acceleration changes, positively impacting subsequent tracking control and vehicle stability. However, these curves have a simple structure and generally cannot capture the geometry of a complex racing line. As a result, tracking performance degrades on road courses, limiting the practical application of jerk-optimal trajectories to simpler tracks, such as ovals, where polynomials can closely approximate the racing line.

Therefore, an alternative method to connect the start state and the sampled end states is required.

In the proposed approach, jerk-optimal deviations to the racing line are generated and then added to the racing line. This enables exact tracking of the racing line while allowing smooth, jerk-optimal deviations, e.g., for overtaking or braking maneuvers. To achieve this, the racing line start conditions are subtracted from the actual start conditions (4.1):

$$\begin{bmatrix} \tilde{s}_0 - s_{\mathrm{rl}}(0) & \dot{\tilde{s}}_0 - \dot{s}_{\mathrm{rl}}(0) & \ddot{\tilde{s}}_0 - \ddot{s}_{\mathrm{rl}}(0) \end{bmatrix}^\top . \tag{4.3}$$

Analogously, the racing line end conditions are subtracted from the sampled end conditions (4.2):

$$\begin{bmatrix} \dot{s}_{\mathrm{e},i} - \dot{s}_{\mathrm{rl}}(T_{\mathrm{traj}}) & \ddot{s}_{\mathrm{e},i} - \ddot{s}_{\mathrm{rl}}(T_{\mathrm{traj}}) \end{bmatrix}^\top . \tag{4.4}$$

The adjusted states (4.3) and (4.4) are then connected using the quartic polynomials, representing jerk-optimal deviations from the racing line. To comply with the original boundary conditions (4.1) and (4.2), the racing line profile is finally added to the resulting polynomials. This curve generation method is referred to hereafter as relative curve generation and the resulting curves as relative curves. A comparison with the jerk-optimal curves is given in Figure 4.2. Even if the start state coincides with the racing line, the jerk-optimal edges cannot replicate its geometry. By contrast, the relative edges exhibit the racing line's fundamental geometry, allowing exact replication.

The relative curves can effectively replicate the racing line and maneuvers similar to it. Other maneuvers, such as pure acceleration maneuvers, are only possible to a limited extent, as the relative curves depend on the racing line. If, for example, the racing line exhibits a strong deceleration, as in Figure 4.2, the relative curves decelerate more strongly than the jerk-optimal curves. In such sections of the race track, this prevents reaching the racing line velocity if the start velocity is lower. Therefore, for each sampled end state (4.2), both a jerk-optimal and a relative curve are generated, resulting in $2 \cdot N_{\dot{\mathrm{s}}}$ different longitudinal edges in the set $\mathfrak{T}_{\mathrm{long}}$.

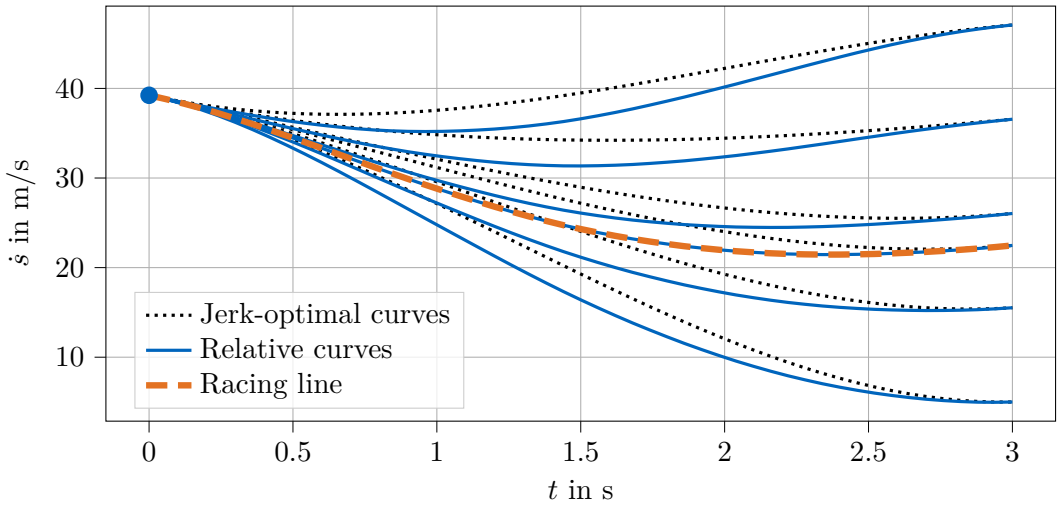


Figure 4.2: Comparison of jerk-optimal and relative trajectory generation with $N_{\dot{s}} = 6$, $K_{\dot{s}} = 1.2$, and $T_{\text{traj}} = 3\,\text{s}$. The start state $\tilde{\boldsymbol{z}}_0$ matches the racing line with $\dot{\tilde{s}}_0 = 39\,\text{m/s}$ and $\ddot{\tilde{s}}_0 = -7\,\text{m/s}^2$, and its velocity is visualized as a blue circle.

Lateral Curve Generation

The set of lateral curves $\mathfrak{T}_{\text{lat}}$ is generated analogously to $\mathfrak{T}_{\text{long}}$. However, the lateral racing line profile $d_{\text{rl}}(t)$ cannot be used directly in the relative curve generation, as the lateral profile to traverse the racing line path changes with varying longitudinal curves. An example of this for a longitudinal curve $s_i(t)$ with a lower velocity $\dot{s}_i(t)$ than the racing line $\dot{s}_{\text{rl}}(t)$ is shown in Figure 4.3. As a specific progress s is reached at a later point in time for the given longitudinal curve, the lateral racing line position profile $d_{\text{rl}}(t)$ is temporally stretched. While the same values of the lateral racing line position $d_{\text{rl}}(t)$ are reached only at a later point in time, the derivatives $\dot{d}_{\text{rl}}(t)$ and $\ddot{d}_{\text{rl}}(t)$ are entirely different, as the same path is traversed with a different time profile. Therefore, for each longitudinal curve $s_i(t)$, the corresponding adapted lateral profile $\tilde{d}_{\text{rl},i}(t)$ is calculated, which is required to traverse the racing line path. A detailed description of this racing line adaptation is given in Appendix C.[1] $\tilde{d}_{\text{rl},i}(t)$ is then used as the reference for the relative lateral curves instead of the original racing line $d_{\text{rl}}(t)$. This implies

[1] Note that the racing line adaption is not required for the hybrid planning approach from Chapter 3, as the racing line's time derivatives do not enter the edge generation or the cost functional.

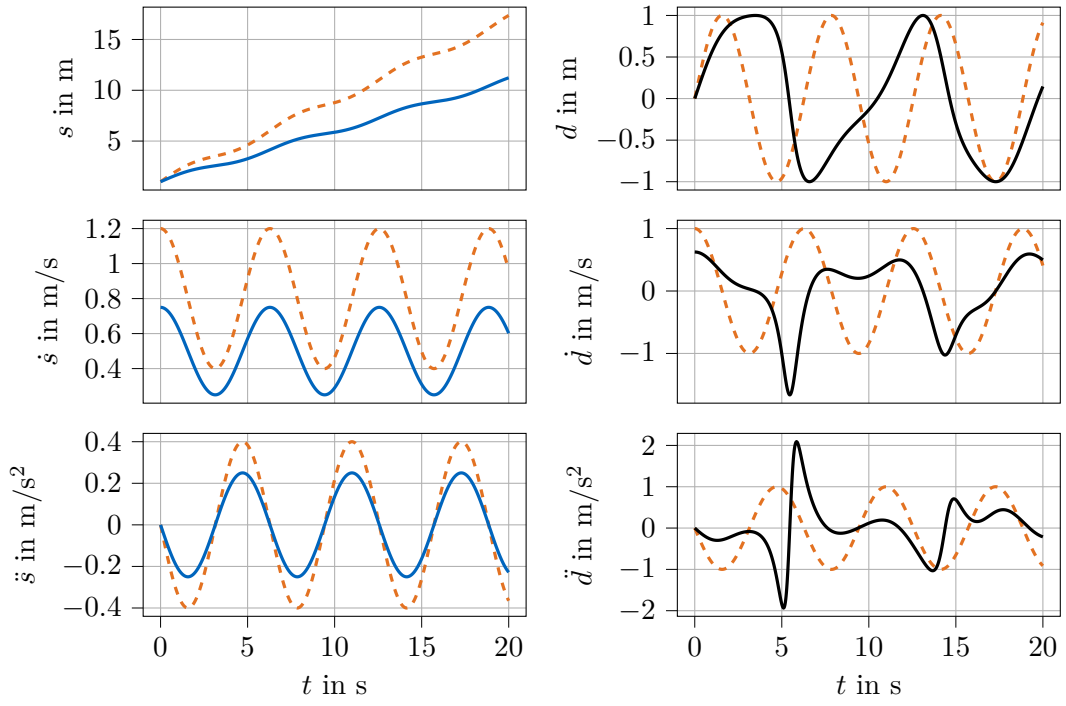


Figure 4.3: Illustration of the racing line adaptation. Left column: Longitudinal profiles of the racing line $s_{\mathrm{rl}}(t)$ (orange dashed line) and a longitudinal curve $s_i(t)$ (blue solid line). Right column: Lateral profile of the racing line $d_{\mathrm{rl}}(t)$ (orange dashed line) and the lateral profile $\tilde{d}_{\mathrm{rl}}(t)$ needed to traverse the racing line path with the given longitudinal curve (black solid line).

that $\mathfrak{T}_{\mathrm{lat}}$ needs to be created after $\mathfrak{T}_{\mathrm{long}}$. Furthermore, due to this dependency, the Cartesian product $\mathfrak{T}_{\mathrm{long}} \times \mathfrak{T}_{\mathrm{lat}}$ cannot be used to generate all trajectory candidates, as done in [92].

With this, each longitudinal curve $s_i(t)$ is completed with $2 \cdot N_{\mathrm{d}} \in \mathbb{N}$ lateral curves $d_{ij}(t)$, $t \in [0, T_{\mathrm{traj}}]$ indexed by $j \in \{1, 2, \ldots, 2 \cdot N_{\mathrm{d}}\}$. Again, each curve originates in the start state $\tilde{\boldsymbol{z}}_0$, yielding the initial conditions:

$$\begin{bmatrix} d_{ij}(0) & \dot{d}_{ij}(0) & \ddot{d}_{ij}(0) \end{bmatrix}^{\top} = \begin{bmatrix} \tilde{d}_0 & \dot{\tilde{d}}_0 & \ddot{\tilde{d}}_0 \end{bmatrix}^{\top}. \tag{4.5}$$

Since the lateral position on the race track is decisive, the end position is sampled, unlike in the generation of longitudinal curves. For this, N_{d} equidistantly distributed samples $d_{\mathrm{e},ij} \in [d_{\mathrm{r}}(s_{\mathrm{e},i}) + {}^{D_{\mathrm{w}}}\!/_2 + D_{\mathrm{saf}},\ d_{\mathrm{l}}(s_{\mathrm{e},i}) - {}^{D_{\mathrm{w}}}\!/_2 - D_{\mathrm{saf}}]$

are generated within the track boundaries, additionally including the end racing line position $\tilde{d}_{\mathrm{rl},i}(T_{\mathrm{traj}})$. Here, the sampling interval is narrowed by half of the vehicle width D_{w} and a safety distance D_{saf}, as trajectory candidates directly at the boundary would be invalid according to the trajectory validation in Subsection 4.1.3. The velocity and acceleration dimensions are not sampled for computation time reasons. Instead, the velocity $\dot{d}_{\mathrm{e},ij}$ corresponding to a position sample $d_{\mathrm{e},ij}$ is determined to align the trajectory candidates with both the racing line and track boundaries. To achieve this, linear interpolation is performed based on the lateral position $d_{\mathrm{e},ij}$ between the relative orientations $\hat{\chi}$ of the track boundaries and the racing line. The associated $\dot{d}_{\mathrm{e},ij}$ is then determined using (A.8). The corresponding acceleration $\ddot{d}_{\mathrm{e},ij}$ is selected by linear interpolation depending on the lateral position $d_{\mathrm{e},ij}$ between $0\,\mathrm{m/s^2}$ on the track boundaries and $\tilde{\ddot{d}}_{\mathrm{rl},i}(T_{\mathrm{traj}})$ on the racing line. The end conditions for the lateral curves are thus:

$$\begin{bmatrix} d(T_{\mathrm{traj}}) & \dot{d}(T_{\mathrm{traj}}) & \ddot{d}(T_{\mathrm{traj}}) \end{bmatrix}^{\top} = \begin{bmatrix} d_{\mathrm{e},ij} & \dot{d}_{\mathrm{e},ij} & \ddot{d}_{\mathrm{e},ij} \end{bmatrix}^{\top}. \tag{4.6}$$

The connection of the initial condition (4.5) with the end conditions (4.6) using jerk-optimal and relative curves is performed analogously to the procedure described for the longitudinal curves, which is not repeated here in detail. However, fifth-order instead of fourth-order polynomials are used, as these are jerk-optimal for a fixed end position, according to Appendix B (case 1). In addition, the adapted racing line profile $\tilde{d}_{\mathrm{rl},i}(t)$, instead of the original one $d_{\mathrm{rl},i}(t)$, is used for the relative curve generation. For increased maneuver diversity of the trajectory candidates, both a jerk-optimal and a relative curve are generated for each sampled end position. This results in $2 \cdot N_{\mathrm{d}}$ lateral curves per longitudinal curve, yielding a total of $4 \cdot N_{\dot{\mathrm{s}}} \cdot N_{\mathrm{d}}$ trajectory candidates.

Choice of Reference Line

After generating the trajectory candidates in Curvilinear coordinates, they are transformed into Cartesian coordinates, as described in Appendix A. Since Curvilinear coordinates are defined based on a reference line ${}^{\mathcal{I}}\boldsymbol{c}(s)$, the choice of ${}^{\mathcal{I}}\boldsymbol{c}(s)$ significantly impacts the resulting trajectories in Cartesian coordinates.

The described curve generation methods are based on either jerk-optimal transitions (jerk-optimal curves) or jerk-optimal deviations from the racing

line (relative curves). However, jerk optimality applies only in Curvilinear coordinates. According to (A.11b), larger $d(t)$ and $\hat{\chi}(t)$ values distort the velocity profile $V(t)$ of a trajectory compared to its $\dot{s}(t)$ profile. As a result, trajectories whose paths deviate from the reference line may exhibit a distorted $V(t)$ profile despite a smooth $\dot{s}(t)$ profile, potentially leading to increased accelerations and infeasibility. To address this, the path to be driven should ideally serve as the reference line, ensuring $d(t) = 0\,\mathrm{m}$, $\hat{\chi}(t) = 0°$, and $V(t) = \dot{s}(t)$. However, in multi-vehicle racing, the driven path is unknown before the race. Since the racing line is followed most of the time and overtaking maneuvers usually involve a similar orientation, the offline-generated racing line is selected as the reference line, i.e., $d_{\mathrm{rl}}(t) = 0\,\mathrm{m}$.

With the described choice of the reference line, the jerk-optimal and relative curve generation methods result in the same curves in the lateral direction when offline racing line generation is used. Consequently, relative curve generation can be skipped in the lateral direction in this case. However, it becomes necessary again when using the online racing line generation, as the online racing lines can deviate from the reference line. In contrast, relative curve generation is always required in the longitudinal direction.

4.1.3 Trajectory Validation

All trajectory candidates are subjected to a validity check of feasibility, collision, and racing rule compliance. As discussed in the validity checks for the hybrid planning approach from Subsection 3.1.6, a soft-checking concept is essential. If all trajectory candidates fail a particular check, the trajectory with the fewest limit violations is selected.

Feasibility Checks

Each generated trajectory is checked for kinematic feasibility according to (2.38) and dynamic feasibility according to (2.39). In contrast to the dynamic feasibility check of the hybrid planning approach from Subsection 3.1.6, the gg-diagrams here depend on the vertical acceleration $\tilde{g}$ to account for the 3D effects of the race tracks.

Collision Checks

The trajectory's path is validated to ensure it lies within the track boundaries, considering the vehicle's geometry. As for the initial edges in the hybrid planning approach in Subsection 3.1.6, this check is performed efficiently in Curvilinear coordinates using (3.9), incorporating the vehicle width D_{w} and the safety distance D_{saf}.

A collision check between the trajectory candidates and the prediction of opponent vehicles is not performed. Instead, collisions are avoided by the corresponding term in the cost functional, as introduced in Subsection 4.1.4. This approach is sufficient for the scenarios and races analyzed in this thesis, offering the benefits of reduced computational effort and ensuring that infeasible states caused by opponent vehicles within the safety distance are avoided.

Racing Rule Compliance Checks

Compliance with the speed limit $V_{\mathrm{limit}}(t)$ is verified as in the hybrid planning approach from Subsection 3.1.6. In contrast to the hybrid planning approach, avoiding overtaking when it is not permitted and maintaining a target distance to the vehicle in front is realized solely by varying $V_{\mathrm{follow}}(t)$ in the cost functional, as described in Subsection 4.1.4. This has proven sufficient in the IAC and A2RL races to date, offering the advantage of fewer checks and reduced computational overhead.

4.1.4 Trajectory Selection

All trajectory candidates valid according to Subsection 4.1.3 are feasible, lie within the race track, and comply with the speed limit. Among these candidates, one must be selected that follows the racing line as closely as possible, if necessary, avoids opposing vehicles, and, if overtaking is not permitted, maintains a target distance from the front opposing vehicle. For this, a cost functional assigns a scalar cost to each trajectory candidate $(s_i(t), d_{ij}(t))$, and the one with the lowest cost is selected. The cost functional comprises three terms, each weighted by a parameter $c_{\square}$:

$$J = \int_0^{T_{\mathrm{traj}}} \left(c_{\mathrm{d}} \cdot \Delta d^2(t) + c_{\mathrm{V}} \cdot \frac{\Delta V^2(t)}{V_{\mathrm{target}}^2(t)} + c_{\mathrm{pr}} \cdot \sum_{l=1}^{N_{\mathrm{opp}}} J_{\mathrm{pr},l}(t) \right) \mathrm{d}t. \tag{4.7}$$

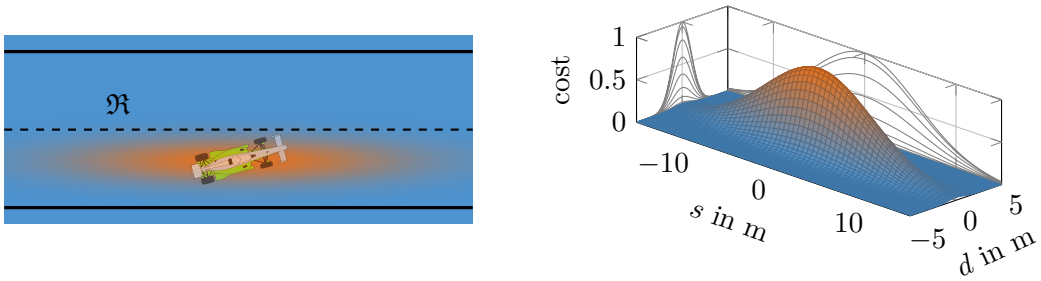


Figure 4.4: Visualization of the elliptical prediction cost distribution aligned with the reference line $\mathfrak{R}$ for a single point in time. Left: Illustration of the cost distribution on a straight. Right: Cost distribution for $p_\mathrm{s} = 0.015$, $p_\mathrm{d} = 0.5$, and $s_{\mathrm{pr},l} = d_{\mathrm{pr},l} = 0\,\mathrm{m}$.

The cost functional is structured similarly to the one used in the hybrid planning approach from Subsection 3.1.7. The first term penalizes lateral position deviations $\Delta d(t) = \tilde{d}_{\mathrm{rl},i}(t) - d_{ij}(t)$ from the racing line. While the lateral deviation enters the cost functional (3.10) of the hybrid planning approach linearly, it is included quadratically in this design. Empirical tests are still pending to determine whether a linear influence would also lead to better behavior for the sampling-based planning approach. The second term weights the deviation $\Delta V(t) = V_\mathrm{target}(t) - V_{ij}(t)$ from the target velocity V_target. By normalizing the deviation with $V^2_\mathrm{target}(t)$, the relative velocity deviation is penalized so that the costs are consistent across different velocity ranges. This becomes necessary with the transition from oval tracks to road courses, as significant velocity differences occur due to the tight turns. The target V_target is chosen as the minimum of the racing line velocity V_rl, the speed limit V_limit, and, if overtaking is not permitted, the speed V_follow to follow an opponent at a desired distance. The third term penalizes trajectories close to the prediction of the opposing vehicles to avoid collisions. The expression $J_{\mathrm{pr},l}(t)$ is formulated as

$$J_{\mathrm{pr},l}(t) = \exp\left(-p_\mathrm{s} \cdot \left[s_{\mathrm{pr},l}(t) - s_i(t)\right]^2 - p_\mathrm{d} \cdot \left[d_{\mathrm{pr},l}(t) - d_{ij}(t)\right]^2\right). \tag{4.8}$$

It corresponds to an elliptical cost distribution parameterized by $p_\mathrm{s} \in \mathbb{R}_{>0}$ and $p_\mathrm{d} \in \mathbb{R}_{>0}$ that covers the prediction of the opposing vehicle indexed by l, as visualized in Figure 4.4. Unlike the design in (3.11), the ellipse is not aligned with the heading of opposing vehicles. Instead, the ellipse is aligned with the reference line due to the formulation in Curvilinear coordinates. This prevents

the ellipse from blocking the entire width of the race track, as illustrated in Figure 3.6 (left), which could otherwise happen at sharp turn entries or due to uncertainties in the heading estimation of the opposing vehicles. Additionally, the exponential progression provides a smooth transition, avoiding sudden changes in trajectory planning between successive planning cycles. Furthermore, unlike in (3.11), the ellipses do not increase with the planning horizon and are therefore not scaled with $h(t)$. While this may be considered for future applications, it is not required for the results presented here and is omitted to simplify implementation. A fourth cost term for weighting the curvature, as formulated in (3.10), does not exist due to the necessary high curvatures on road courses.

4.2 Results

Subsection 4.2.1 contains computational details on the sampling-based local trajectory planning approach. Simulative results comparing the jerk-optimal with the proposed relative trajectory generation method and planning with and without consideration of 3D effects follow in Subsection 4.2.2. Additionally, this subsection demonstrates the impact of online racing line generation on the sampling-based planning approach compared to using an offline racing line. Finally, Subsection 4.2.3 provides experimental results from the final of the A2RL on 27 April 2024.

The simulations are performed on the LVMS and the Mount Panorama Circuit in Bathurst (MPCB). While the LVMS is an oval track (Figure 3.7), the MPCB is a complex road course, as shown in Figure 4.5 (left). The MPCB is 6250 m long, has a maximum height difference of 175 m, a maximum slope of 13°, and a maximum banking of 8°. The final race of the A2RL took place at the Yas Marina Circuit in Abu Dhabi (YMCA), a 5285 m long road course, visualized in Figure 4.5 (right). Compared to the MPCB, it has only slight height differences and low slope and banking angles, both of which are less than 3°.

4.2.1 Computational Details

The sampling-based planning approach is implemented in Python. The planned trajectories are represented by N_{traj} equidistant time points, with feasibility

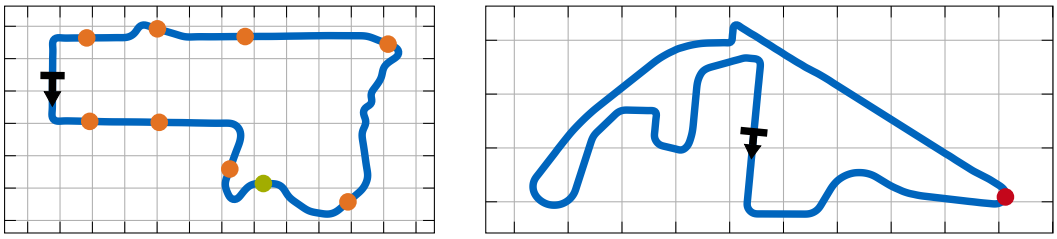


Figure 4.5: Layouts of the MPCB (left) and YMCA (right). The arrows indicate the direction of travel and the respective $s = 0\,\mathrm{m}$ position. One grid cell equals 200 m in width and height. The orange and green circles on the MPCB mark the positions of the static objects mentioned in Subsection 4.2.2. The red circle on the YMCA indicates the turn shown in Figure 4.8.

checks performed only at these points. The discrete points also serve as integration points for the numerical approximation of the cost functional (4.7) using the rectangle rule.

The procedure proposed in [71] is applied for racing line generation. It supports both offline generation of a closed racing line for the entire race track and online generation for a desired spatial horizon $H_{\mathrm{rl}} \in \mathbb{R}_{>0}$. The generated racing line is available in each planning cycle as N_{rl} discrete points that are guaranteed to be feasible. However, interpolation at the N_{traj} points required for local planning can lead to a violation of the limits. Additionally, correcting deviations from the racing line in local planning becomes challenging if the same limits are used for the feasibility checks in both local planning and racing line generation. To address these issues, the limits used in the feasibility checks of the sampling-based approach are set slightly higher than those of the racing line. This is realized by applying a larger safety margin $D_{\mathrm{saf,rl}} > D_{\mathrm{saf}}$ for the racing line in the collision checks. Furthermore, the extreme values $\tilde{a}_{\mathrm{x,min}}(V, \tilde{g})$, $\tilde{a}_{\mathrm{x,max}}(V, \tilde{g})$, and $\tilde{a}_{\mathrm{y,max}}(V, \tilde{g})$ of the gg-diagrams in (2.39) are scaled by a factor $\tilde{a}_{\mathrm{scl}} \in (0, 1]$ for the racing line. However, if the absolute acceleration limits are low, the relative margin caused by the scaling $\tilde{a}_{\mathrm{scl}}$ may not be sufficient. Therefore, an additional absolute margin $\tilde{a}_{\mathrm{mgn}} \in \mathbb{R}_{\geq 0}$ is introduced, ensuring a minimum margin regardless of the absolute limit values.

All results were generated using the sequential software architecture from Figures 1.2 and 1.3a. However, for the online racing line generation results,

Table 4.1: Parameters for the simulative results of the sampling-based planning approach (descriptions are provided in the list of symbols on page xvii)

Parameter	Value	Unit	Parameter	Value	Unit
D_w	1.93	m	$\tilde{a}_\mathrm{y,max}(V,\tilde{g})$	0 to 45	$\mathrm{m/s^2}$
$N_\dot{\mathrm{s}}$	40	-	$\tilde{a}_\mathrm{x,min}(V,\tilde{g})$	-55 to 0	$\mathrm{m/s^2}$
$K_\dot{\mathrm{s}}$	1.2	-	$\tilde{a}_\mathrm{x,max}(V,\tilde{g})$	0 to 21	$\mathrm{m/s^2}$
N_d	15	-	$\rho(V,\tilde{g})$	1 to 2	-
N_traj	30	-	$\tilde{a}_\mathrm{scl}$	0.9	-
T_traj	3	s	$\tilde{a}_\mathrm{mgn}$	0.8	$\mathrm{m/s^2}$
H_rl	500	m	c_d	0.1	-
Δt_sim	0.1	s	c_V	100	-
D_snr	200	m	c_pr	5000	-
D_saf	0.2	m	p_s	0.015	-
$D_\mathrm{saf,rl}$	0.5	m	p_d	0.5	-
$\hat{\kappa}_\mathrm{max}$	0.1	$\mathrm{m^{-1}}$			

the offline-executed global planning submodule from Figure 1.3a is replaced by an online submodule. The simulations employ ideal state estimation, detection, prediction, and control, with a simulation time step $\Delta t_\mathrm{sim} \in \mathbb{R}_{>0}$ and opposing vehicles detected within a sensor range $D_\mathrm{snr} \in \mathbb{R}_{>0}$. The modules are executed synchronously, neglecting the influence of different computation times. Accordingly, the estimated state also represents the start state of local planning, i.e., $\tilde{\boldsymbol{z}}_0 = \boldsymbol{z}_0$. The generation of the gg-diagrams follows the method presented in [57], where the actual limit shape is determined using a double-track model combined with a Pacejka tire model. To perform the computationally advantageous check (2.39), a diamond-shaped underapproximation of the actual diagram shape is derived, following the method outlined in [71]. All parameters used for the simulations are provided in Table 4.1.

For the experimental results, the software stack of the TUM Autonomous Motorsport team described in Subsection 3.2.1 was used, except for the aforementioned generation of the racing line. However, since the used Tube-MPC is designed for 2D tracks, the 3D quantities of the planned trajectory must

Table 4.2: Lap time comparison using jerk-optimal and relative curve generation

#	Mode	Lap time in s		Number of violations	
		LVMS	MPCB	LVMS	MPCB
1	Offline racing line	27.12	125.96	-	-
2	Jerk-optimal curve generation	27.12	135.70	0	58
3	Relative curve generation	27.12	125.95*	0	0

*The slightly lower lap time compared to the offline racing line is due to numerical inaccuracies in the lap time calculation.

be transformed into the x-y plane using appropriate rotation matrices. The gg-diagrams have been optimized empirically.

4.2.2 Simulative Results

The presented sampling-based planning approach is compared with the concepts commonly used in the literature. First, a comparison is made between the jerk-optimal and relative curve generation. Next, planning with and without consideration of the 3D effects of the race track is analyzed. Finally, the impact of offline versus online racing line generation on local planning is evaluated.

Jerk-Optimal and Relative Curve Generation

To highlight the necessity of relative curve generation, a flying solo lap is analyzed using both jerk-optimal and relative curve generation. Therefore, the objective of the local planning approach is to follow the offline-computed racing line without considering other vehicles. Table 4.2 presents the resulting lap times, with the lap time of the corresponding racing line serving as a baseline for comparison. Additionally, the number of planning cycles in which no valid solution could be found, triggering the soft-checking concept from Subsection 4.1.3, is specified.

On the LVMS, both curve generation methods achieve the racing line's lap time. However, on the MPCB, the lap time is significantly higher when using jerk-optimal curve generation due to the race track's more complex geometry

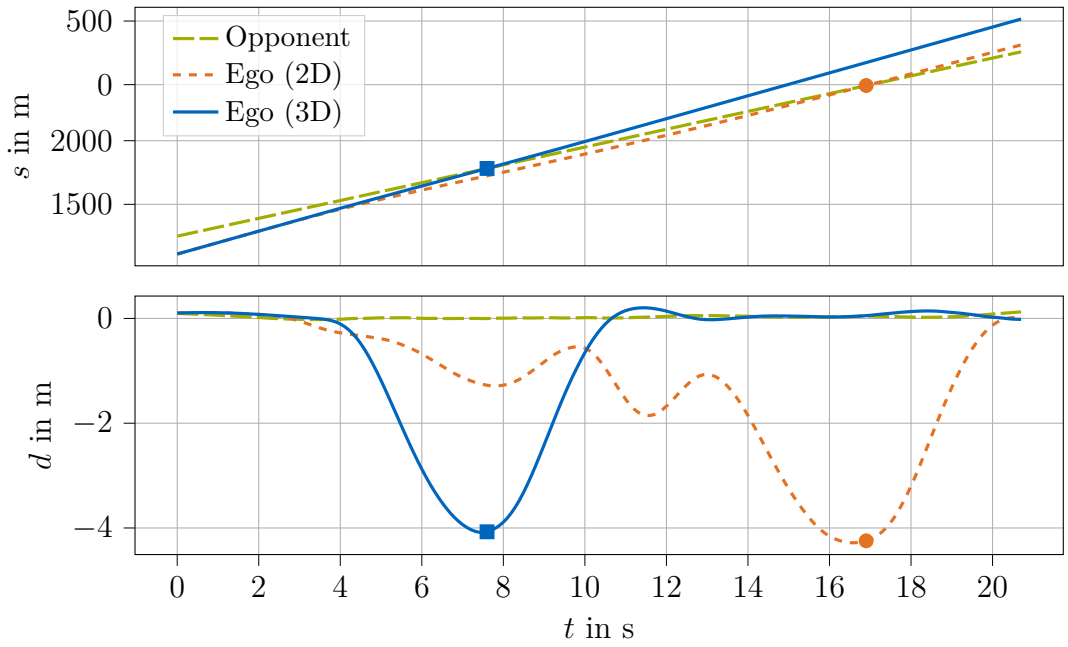


Figure 4.6: Comparison of an overtaking maneuver on the LVMS in Curvilinear coordinates using 2D and 3D friction checks. The points in time at which the ego and opponent vehicle have the same s-coordinate are indicated by squares (3D) and circles (2D).

and the associated racing line, which the jerk-optimal curves fail to replicate accurately. Additionally, completing a full lap on the MPCB without feasibility violations is not possible with jerk-optimal curves, primarily due to poor racing line tracking and the associated high speeds at turn entrances.

Planning in 2D and 3D

The 3D features of a race track described in Section 2.3 cause a variation of $\tilde{g}$ and a reduction or increase in the dynamic potential. If the 3D effects are not incorporated into planning, the actual potential is overestimated or underestimated, leading to feasibility issues or conservative behavior. The latter is demonstrated by a scenario on the LVMS involving an opposing vehicle following the racing line at a constant speed of 70 m/s and an ego vehicle behind it attempting to overtake. The scenario starts with the opponent vehicle on the backstretch at progress $s = 1250\,\mathrm{m}$. The ego vehicle starts 140 m behind the

opponent on the racing line path at the racing line speed of 90 m/s. This scenario is executed once considering the 3D effects and once assuming a 2D race track. For the 2D case, a flat track ($\mu(s) = \varphi(s) = 0°$ and $\Omega_\mathrm{x}(s) = \Omega_\mathrm{y}(s) = 0\,\mathrm{rad/m}$) is assumed in the friction checks (2.39), resulting in $\tilde{a}_\mathrm{x}(t) = \hat{a}_\mathrm{x}(t)$, $\tilde{a}_\mathrm{y}(t) = \hat{a}_\mathrm{y}(t)$, and $\tilde{g}(t) = g$.

Figure 4.6 shows the scenario for both cases. In the 3D case, an overtaking maneuver is initiated at the first opportunity in turn T3, as the negative banking of the left turn supports accelerations to the left ($\hat{a}_\mathrm{y}(t) > 0\,\mathrm{m/s^2}$), providing sufficiently high dynamic potential for the maneuver. In the 2D case, however, the accelerations to the left are underestimated, preventing an overtake in the turns. Instead, the overtaking maneuver is delayed on the frontstretch, as lower lateral accelerations are sufficient due to the lower geodesic curvature $\Omega_\mathrm{z}(s)$. The lateral oscillations observed just before the maneuver stem from planning inconsistencies between consecutive planning cycles due to the limited planning horizon. Overall, the overtaking maneuver can be achieved approximately 9 s earlier by incorporating the 3D effects into planning.

Offline and Online Racing Line Generation

In the above results, an offline-calculated racing line is used, which does not consider the ego vehicle's estimated state. This is compared to an online racing line generation, in which the time-optimal trajectory for a limited spatial horizon H_rl is replanned in each planning cycle starting from the start state $\tilde{\boldsymbol{z}}_0$. The comparison is conducted for both single- and multi-vehicle racing.

A flying lap is considered for single-vehicle racing, with the resulting lap times given in entries #2 and #3 in Table 4.3. As previously noted in Table 4.2 (entries #1 and #3), the lap time of the racing line can be reached for both the LVMS and the MPCB using offline racing line generation. In contrast, online racing line generation results in increased lap times, as the time-optimal solution for the entire race track is not sufficiently approximated by the recursive online optimization with a limited spatial horizon according to the RH principle.

For multi-vehicle racing, flying laps are conducted with opposing vehicles on the offline-generated racing line, requiring a deviation from it. On the LVMS, twelve opponent vehicles are placed with a spacing of 200 m, following the racing line at 70 % of its speed. On the MPCB, nine static objects are placed along the racing line on the progressions shown in Figure 4.5 (left). Static obstacles

Table 4.3: Lap time comparison using offline and online racing line generation

#	Mode	Lap time in s		Number of violations	
		LVMS	MPCB	LVMS	MPCB
1	Offline racing line	27.12	125.96	-	-
2	Single with offline racing line	27.12	125.95*	0	0
3	Single with online racing line	27.17	126.13	0	0
4	Multi with offline racing line	27.74	127.59	0	5
5	Multi with online racing line	27.41	126.99	0	0

*The slightly lower lap time compared to the offline racing line is due to numerical inaccuracies in the lap time calculation.

are used on the MPCB for reliable lap time comparison, as the difficulty of overtaking maneuvers on road courses depends heavily on the position of the opposing vehicles. Although the exact obstacle placement still affects the absolute lap times, this evaluation allows for a comparison between the lap times arising from offline and online racing line generation. Entries #4 and #5 in Table 4.2 present the resulting lap times. Contrary to the single-vehicle case, the application of online racing line generation yields reduced lap times on both tracks. This lap time difference is mainly due to the deviations from the racing line speed required during overtaking. As the vehicle operates close to its limits, planning a trajectory that quickly recovers the speed of the racing line is challenging, especially when the solution space is discretized using trajectory candidates. When using the offline-generated racing line, many of the finite trajectory candidates that regain the racing line speed quickly are infeasible. In contrast, when using online racing line generation, the time-optimal trajectory is replanned at each planning cycle without discretizing the solution space. As the updated racing line is used for relative curve generation, more trajectory candidates that return to maximum speed quickly are feasible, leading to lower lap times.

Figure 4.7 demonstrates the relationship between online racing line generation and local planning. In the first planning cycle, the racing line collides with the opposing vehicle, leading to an evasive maneuver being planned. In the next

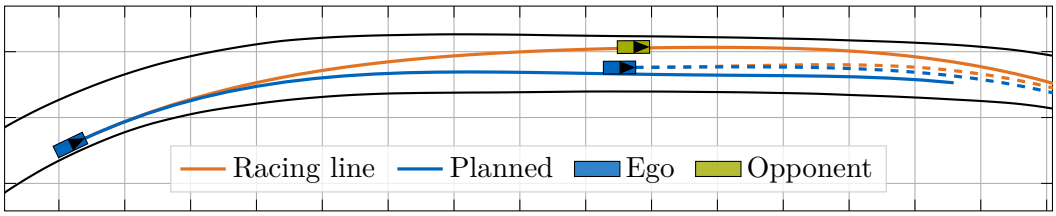


Figure 4.7: Overtaking maneuver of a static obstacle (green circle in Figure 4.5) with online racing line generation on the MPCB. The solid and dashed lines correspond to two different planning cycles over time. One grid cell equals 10 m in width and height.

planning cycle shown, the newly calculated racing line corresponds to a shorter path, which is followed by local planning. If the offline-generated racing line were used instead, local planning would result in a trajectory that deviates from the actual time-optimal solution, leading to increased lap times.

4.2.3 Real-World Application

The sampling-based local planning approach has been used on various oval tracks and road courses within the IAC and A2RL since mid-2023. Due to the high computational demands of online optimization, only offline racing line generation was employed for the races. Since the racing line also served as the reference line, the generation of jerk-optimal curves in the lateral direction was sufficient. In addition, when generating the relative curves in the longitudinal direction, finer sampling near the racing line velocity was used to better approach the racing line. There were also minor changes to the cost design, which are not detailed here. For the A2RL, the Dallara EAV24 race car (Figure 1.1b), which is an autonomous version of the Super Formula car, was used. The hardware consists of an AMD EPYC 7313P CPU with 16 cores and 256 GB RAM, with one core available for the local planning approach. Using this setup, an average computation time of 105 ms per planning cycle was achieved.

To demonstrate the application of the sampling-based planning approach, this subsection covers the final race of the A2RL competition on 27 April 2024 at the YMCA. Unlike an oval track such as the LVMS, the YMCA features various straight and curvy sections, which lead to more significant variations in speed and lateral acceleration, requiring the generation of relative curves. The final race involved four vehicles, with starting positions determined by a

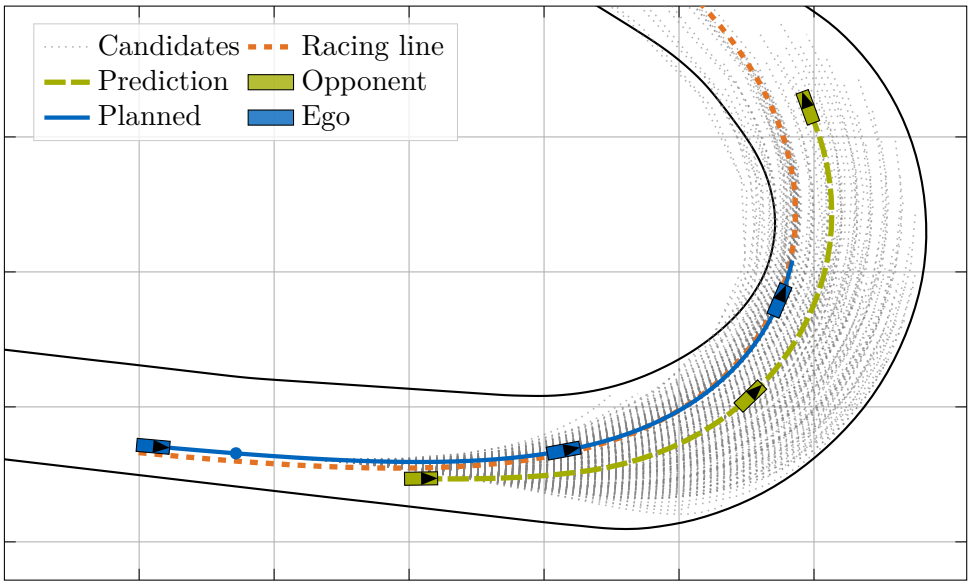


Figure 4.8: Planned and predicted path during the final overtaking maneuver of the A2RL competition on 27 April 2024 at the YMCA. The position of the turn is marked in Figure 4.5 (right). The start state $\tilde{z}_0$ is indicated by a blue circle. The poses along the prediction and the planned path are visualized for three points in time. For clarity, only 25 % of the generated trajectory candidates are depicted. One grid cell equals 20 m in width and height.

preceding time trial. The winner was the first vehicle to complete a predefined number of laps. Vehicles in trailing positions were equipped with a push-to-pass feature, temporarily unlocking increased engine power.

Figure 4.8 shows the path of the planned trajectory for the final overtaking maneuver of the TUM Autonomous Motorsport team. Additionally, a subset of the trajectory candidates generated in this planning cycle is depicted, closely resembling the racing line. Based on the opponent's deviation from the racing line, it is predicted that the opponent will not take the inside line through the turn but will leave space instead. Accordingly, the selected trajectory candidate primarily follows the racing line, enabling an inside overtaking maneuver through the turn. However, trajectory candidates with a similar path but higher speeds were not selected due to their infeasibility. While the ego vehicle completed the planned maneuver, the opposing vehicle stopped in the turn for technical issues.

4.3 Discussion

The sampling-based local trajectory planning approach overcomes many of the limitations of the hybrid planning approach mentioned in Section 3.3. First, there is no fixed graph structure, making the planning independent of the vehicle's positioning on the track. Instead, the end states of the trajectory candidates are sampled online. Second, the racing line can be followed on race tracks of arbitrarily complexity and not just on oval tracks. Third, the 3D effects of the race track are considered, providing a better representation of the vehicle's actual limits and avoiding infeasibility or poor racing performance. Fourth, the online racing line generation allows the vehicle to reach its limits earlier, as the time-optimal trajectory is replanned starting from the vehicle state without discretizing the state space. Although the last two aspects can also be incorporated into the hybrid planning approach, it remains unsuitable for many current IAC and A2RL races due to its restriction to oval tracks.

However, one disadvantage of the sampling-based approach is the limited variety of maneuvers that can be planned due to the discretization of the state space by trajectory candidates. Maneuvers that deviate from the jerk-optimal or relative curves cannot be planned, motivating improved trajectory generation methods. Furthermore, unlike the hybrid planning approach, it is not possible to plan a trajectory that initially moves away from the racing line and then returns to it, as only the end state of the trajectory candidates is sampled. Instead, such a maneuver can only be executed across several planning cycles. An improvement would be a multi-stage sampling concept, in which intermediate states are additionally sampled. This brings the concept closer to the hybrid planning approach without fixed layers but at the cost of increased computational time. Another limitation arises in the real-world application of online racing line generation. While it has proven methodologically beneficial in simulations, its lack of robustness and increased computation time have so far prevented its real-world application. These problems primarily arise in multi-vehicle racing, where the time-optimal solution between adjacent planning cycles can differ due to different start states. Accordingly, initializing the optimization problem with the solution from the previous cycle does not always lead to rapid convergence. Lastly, as in the hybrid planning approach, the interaction between the vehicles is not accounted for. Instead, the planning

approaches only react to the predicted movement of the opponent according to the conventional planning structure shown in Figure 1.3a. However, incorrect predictions already led to collisions within the more relaxed rules of the A2RL, motivating interaction-aware planning approaches.

5 Interaction-Aware Planning using Reinforcement Learning

This chapter addresses interaction-aware planning with a focus on limitations 5 and 6 outlined in Section 1.4. Most existing RL-based planning approaches for racing use either low-level control inputs from a vehicle model or high-level behaviors as action space. However, while low-level action spaces pose safety risks in real-world applications due to modeling inaccuracies, the use of discrete high-level maneuvers can result in reduced maneuver diversity. The RL-based approach presented here directly generates trajectories, avoiding both disadvantages. However, since an NN is used for action selection, no safety guarantees are given, meaning that the resulting trajectories may be invalid. To still provide a valid trajectory in such cases, a safety layer (SL) is introduced, which generates a valid trajectory closely resembling the original invalid one.

The RL-based planning approach and the SL are tested in an interactive simulative blocking scenario involving an ego vehicle and an actively blocking opponent vehicle that operates based on a control law designed explicitly for blocking. In contrast, the ego vehicle is controlled by the RL-based planning approach under the assumption of perfect tracking. To demonstrate the limitations of the conventional planning structure consisting of sequential prediction and planning, the RL-based approach is compared with the sampling-based planning approach presented in Chapter 4. In addition, a game-theoretic approach is included in the comparison, in which the exact model knowledge of the opponent vehicle is utilized to determine the individual reaction for each trajectory candidate.

This chapter is based on the author's publication [160]. It additionally includes an introduction to the necessary RL theory, the game-theoretic approach used for comparison, and more details and results on the training of the RL-based approach. The RL theory is covered in Section 5.1, while the scenario

description, including the opponent vehicle's blocking behavior, is provided in Section 5.2. Section 5.3 explains the RL-based approach with the SL as well as the conventional and game-theoretic approaches used for comparison. Evaluation results for all three approaches and the SL are presented in Section 5.4 and discussed in Section 5.5, along with further research directions.

5.1 Reinforcement Learning Theory

RL is an area of machine learning focused on decision-making. Unlike supervised or unsupervised learning, RL does not rely on pre-existing datasets. Instead, an agent learns by interacting with an environment in a trial-and-error fashion, generating its data through experience. For each decision, the agent receives a reward signal that serves as feedback for the action taken. RL algorithms aim to maximize the expected cumulative reward over time.

Conceptually, RL is closely related to optimal control, where the goal is to find a controller that minimizes or maximizes a cost functional. While optimal control typically assumes a known system model and focuses on solving mathematical optimization problems, RL extends this approach to problem formulations where the system dynamics or reward structure may be unknown, relying on data-driven exploration.

This section provides a brief introduction to the theory of RL, covering fundamental terms, concepts, and training algorithms. The content is primarily based on the textbook from Sutton and Barto [161]. Subsection 5.1.1 defines the RL problem in general and the objective of RL algorithms. Next, Subsection 5.1.2 introduces the concept of value functions, which are employed in the RL algorithms described in Subsection 5.1.3. Finally, Subsection 5.1.4 presents further RL concepts relevant in the context of this thesis.

5.1.1 Problem Formulation

The RL problem is formalized by a Markov Decision Process (MDP), which consists of an agent interacting with an environment, as shown in Figure 5.1. The agent acts as the decision-maker, while the environment encompasses everything outside the agent. In each discrete time step $t_\mathrm{d} \in \mathbb{N}_0$, the environment's

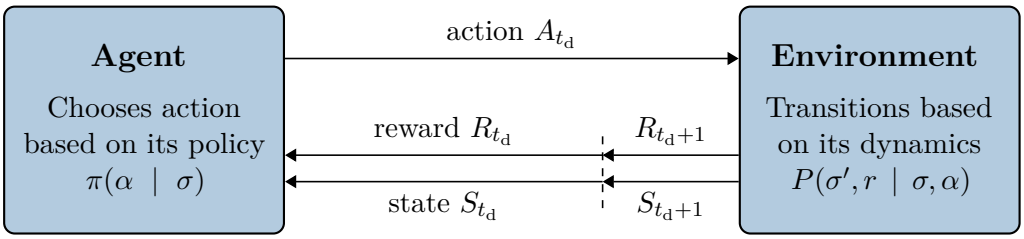


Figure 5.1: Schematic diagram of an MDP consisting of an agent acting with an environment.

configuration is represented by its state $S_{t_\mathrm{d}} \in \mathfrak{s}$.[1] Based on this state, the agent selects an action $A_{t_\mathrm{d}} \in \mathfrak{a}$ that influences the environment and thus transitions it to a new state $S_{t_\mathrm{d}+1} \in \mathfrak{s}$ in the next time step. In addition, the agent receives a scalar reward $R_{t_\mathrm{d}+1} \in \mathfrak{r} \subset \mathbb{R}$ in response to the selected action. This reward can be positive to encourage desirable behaviors or negative to punish undesirable ones. $\mathfrak{s}$, $\mathfrak{a}$, and $\mathfrak{r}$ are the respective sets of all possible states, actions, and rewards.

So far, the interface between the agent and the environment has been introduced, but the two entities themselves have not yet been detailed. On one side, the agent's probability of choosing an action $\alpha \in \mathfrak{a}$ in a given state $\sigma \in \mathfrak{s}$ is defined by its policy $\pi \in [0, 1]$:

$$\pi(\alpha \mid \sigma) = \Pr\left\{A_{t_\mathrm{d}} = \alpha \mid S_{t_\mathrm{d}} = \sigma\right\}. \tag{5.1}$$

On the other side, given a specific state $\sigma \in \mathfrak{s}$ and action $\alpha \in \mathfrak{a}$, the probability of transitioning to a new state $\sigma' \in \mathfrak{s}$ and receiving a reward $r \in \mathfrak{r}$ is described by the environment's dynamics $P \in [0, 1]$:

$$P(\sigma', r \mid \sigma, \alpha) = \Pr\left\{S_{t_\mathrm{d}} = \sigma', R_{t_\mathrm{d}} = r \mid S_{t_\mathrm{d}-1} = \sigma, A_{t_\mathrm{d}-1} = \alpha\right\}. \tag{5.2}$$

In an MDP, the transition probability (5.2) depends only on the immediately preceding state and action, $S_{t_\mathrm{d}-1}$ and $A_{t_\mathrm{d}-1}$, and not on earlier ones. Therefore, the state space must contain all information from the past that influences the transition. If this Markov property is not satisfied, the state space can be extended accordingly.

[1] Deviating from the standard notation in the literature, t_d is used to denote the discrete time steps in MDPs, distinguishing it from the continuous time t used in trajectory planning.

This thesis considers only episodic MDPs, where the number of time steps t_d is finite. Therefore, the process of the agent selecting an action according to π and the environment changing its state according to P continues until a terminal state is reached or a predefined maximum number of time steps has elapsed. Such a run starting in an initial state S_0 until termination is called an episode. It results in a sequence of state, action, and reward up to the terminal time step $N_\mathrm{ep} \in \mathbb{N}$:

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, \ldots, R_{N_\mathrm{ep}-1}, S_{N_\mathrm{ep}-1}, A_{N_\mathrm{ep}-1}, R_{N_\mathrm{ep}}, S_{N_\mathrm{ep}}. \tag{5.3}$$

The goal in RL is to find the optimal policy π^* that maximizes the return $G_{t_\mathrm{d}} \in \mathbb{R}$, which is the discounted sum of all future rewards starting from the current time step t_d up to the terminal time step N_ep of the episode:

$$\begin{aligned} G_{t_\mathrm{d}} &= R_{t_\mathrm{d}+1} + \gamma R_{t_\mathrm{d}+2} + \gamma^2 R_{t_\mathrm{d}+3} + \cdots + \gamma^{N_\mathrm{ep}-t_\mathrm{d}-1} R_{N_\mathrm{ep}} && \text{(5.4a)} \\ &= R_{t_\mathrm{d}+1} + \gamma G_{t_\mathrm{d}+1}. && \text{(5.4b)} \end{aligned}$$

With the discount factor $\gamma \in [0, 1]$, the rewards can be weighted based on their temporal distance. For $\gamma = 0$, only the current reward is considered, i.e., $G_{t_\mathrm{d}} = R_{t_\mathrm{d}+1}$, while for $\gamma = 1$, all rewards are weighted equally. Note that, despite starting from the same initial state S_0, the sequence in (5.3) and, consequently, the return (5.4) may vary across different episodes due to the stochastic nature of the policy π and dynamics P.

5.1.2 Value Functions

With the definition of the return G_{t_d} in (5.4), value functions can be introduced that indicate how beneficial it is to be in a particular state σ or to perform an action α in that state. As the return depends on the selected actions, the value functions are given for a specific policy $\pi(\alpha \mid \sigma)$.

The state-value function $v_\pi(\sigma) \in \mathbb{R}$ specifies the expected return G_{t_d} when being in state σ and then following policy π:

$$v_\pi(\sigma) = \mathbb{E}_\pi \left[G_{t_\mathrm{d}} \mid S_{t_\mathrm{d}} = \sigma \right]. \tag{5.5}$$

It is defined as an expected value, as different episodes can result in varying returns. The index π indicates the dependency on the policy. Due to the recursive nature of the return G_{t_d} in (5.4b), a recursive form for the state-value function, known as the Bellman equation, can also be established:

$$v_\pi(\sigma) = \mathbb{E}_\pi \left[R_{t_\mathrm{d}+1} + \gamma v_\pi(\sigma') \mid S_{t_\mathrm{d}} = \sigma \right]. \tag{5.6}$$

It states that, in expectation, the return G_{t_d} is composed of the immediate reward $R_{t_\mathrm{d}+1}$ and the expected return from the state σ' at the subsequent time step $t_\mathrm{d}+1$. For a formal derivation, please refer to [161, Section 3.5].

Similarly, the action-value function $q_\pi(\sigma, \alpha) \in \mathbb{R}$ specifies the expected return G_{t_d} when being in state σ, selecting action α (which may deviate from the policy), and then following policy π:

$$q_\pi(\sigma, \alpha) = \mathbb{E}_\pi \left[G_{t_\mathrm{d}} \mid S_{t_\mathrm{d}} = \sigma, A_{t_\mathrm{d}} = \alpha \right]. \tag{5.7}$$

Therefore, the two value functions differ solely in the choice of the immediate action A_{t_d}. While for $v_\pi(\sigma)$, this action is selected according to the policy, for $q_\pi(\sigma, \alpha)$, it is set to the specific action α. From time step $t_\mathrm{d}+1$ onward, the actions are chosen according to the policy for both functions.

5.1.3 Reinforcement Learning Algorithms

RL algorithms aim to find the optimal policy π^* that maximizes the return G_{t_d}. They are divided into value-based, policy-based, and actor-critic methods.

Value-Based Methods

With valued-based methods, the optimal state-value function v_{π^*} or action-value function q_{π^*} is learned first, and the optimal policy π^* is derived from it. As it is not possible to determine π^* from v_{π^*} without knowing the environment's dynamics P, most value-based algorithms learn the optimal action-value function q_{π^*}. The optimal policy π^* results directly from it:

$$\pi^*(\alpha \mid \sigma) = \begin{cases} 1 & \text{if } \alpha = \arg\max\limits_{\bar{\alpha} \in \mathfrak{a}} q_{\pi^*}(\sigma, \bar{\alpha}), \\ 0 & \text{otherwise.} \end{cases} \tag{5.8}$$

The most fundamental value-based algorithm is policy iteration, where q_{π^*} is learned through repeated policy evaluation and policy improvement. The algorithm begins with a randomly initialized policy. In the policy evaluation step, the action-value function associated with the current policy is estimated based on data collected by interacting with the environment. In the subsequent policy improvement step, a new policy is derived from the estimated action-value function by selecting the action with the highest value, as done in (5.8). These steps are repeated iteratively until the action-value function converges or a maximum number of iterations is reached. With policy iteration, one step is completed before the next step is performed. However, the two steps do not need to be executed sequentially or until full convergence. The general idea of iterative policy evaluation and improvement, regardless of how both steps are executed, is referred to as generalized policy iteration. Alternative value-based algorithms are based on this idea, with Q-learning [162] and its variants [163] being the most widely used.

For low-dimensional state and action spaces, the algorithms can be implemented using tables that store action values for each state-action pair. However, this approach is no longer feasible as the state space grows or becomes continuous. In such cases, function approximators like NNs can be used, which take the state and action space as inputs and output the corresponding action value. Although this approach sacrifices many convergence guarantees, it remains practical for real-world applications. However, value-based methods are generally unsuitable for high-dimensional action spaces since computing the maximum value over all possible actions, as must be done in (5.8), becomes computationally intractable.

Policy-Based Methods

Policy-based methods parameterize the policy π rather than deriving it from a learned value function. This approach allows for the direct optimization of the policy, even in continuous action spaces. The policy $\pi(\alpha \mid \sigma, \boldsymbol{\theta})$ is parameterized using the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^{N_\theta}$, where $N_\theta \in \mathbb{N}$ is the number of parameters. Any differentiable function approximator can be applied for this, i.e., $\nabla_{\boldsymbol{\theta}} \pi(\alpha \mid \sigma, \boldsymbol{\theta})$ must exist. However, NNs are widely used.

By adjusting the parameter vector $\boldsymbol{\theta}$ and thus the policy $\pi(\alpha \mid \sigma, \boldsymbol{\theta})$, policy-based methods aim to maximize a performance criterion $J_\pi(\boldsymbol{\theta}) \in \mathbb{R}$, typically

defined as the expected return starting from a given state σ:

$$J_\pi(\boldsymbol{\theta}) = v_\pi(\sigma) \overset{(5.5)}{=} \mathbb{E}_\pi \left[G_{t_\mathrm{d}} \mid S_{t_\mathrm{d}} = \sigma \right]. \tag{5.9}$$

To maximize $J_\pi(\boldsymbol{\theta})$, gradient ascent can be applied, i.e., the parameters are iteratively updated in the direction of the gradient $\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}_\mathrm{old})$. With the step size $\eta_\theta \in \mathbb{R}_{>0}$, the update rule is

$$\boldsymbol{\theta}_\mathrm{new} = \boldsymbol{\theta}_\mathrm{old} + \eta_\theta \cdot \nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}_\mathrm{old}). \tag{5.10}$$

However, it is not possible to determine the gradient directly, as the expected value of the return G_{t_d} depends on the state distribution, which in turn depends on the unknown dynamics P of the environment. Therefore, a reformulation of the gradient is needed, which is provided by the policy gradient theorem proved in [161, Section 13.2] (the symbol $\propto$ denotes "proportional to"):

$$\nabla_{\boldsymbol{\theta}} J_\pi(\boldsymbol{\theta}) \propto \mathbb{E}_\pi \left[\gamma^{t_\mathrm{d}} G_{t_\mathrm{d}} \frac{\nabla_{\boldsymbol{\theta}} \pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta})}{\pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta})} \right]. \tag{5.11}$$

The policy gradient theorem (5.11) leads to an expectation that cannot be determined exactly. However, it can be estimated by executing episodes and using the collected data. At the end of an episode, the return G_{t_d} can be calculated for each time step t_d, and since both the gradient of π and the policy π itself are accessible, multiple samples of the expected value in (5.11) can be generated. Averaging over all collected samples yields the REINFORCE algorithm [164] with the following update rule:

$$\boldsymbol{\theta}_\mathrm{new} = \boldsymbol{\theta}_\mathrm{old} + \eta_\theta \cdot \frac{1}{N_\mathrm{ep}} \sum_{t_\mathrm{d}=1}^{N_\mathrm{ep}} \gamma^{t_\mathrm{d}} G_{t_\mathrm{d}} \frac{\nabla_{\boldsymbol{\theta}} \pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta}_\mathrm{old})}{\pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta}_\mathrm{old})}. \tag{5.12}$$

Intuitively, the quotient in (5.12) determines the direction in the parameter space for updating $\boldsymbol{\theta}$ to increase the likelihood of selecting action A_{t_d} in state S_{t_d}. The return G_{t_d}, in contrast, defines the magnitude and sign of the update: a positive return increases the probability of selecting A_{t_d} in state S_{t_d}, while a negative return decreases it.

The return G_{t_d} can vary significantly across different episodes, a property known as high *variance*. This should generally be avoided, as it may lead to slow convergence and unstable learning. One way to reduce the variance of the gradient estimate is by introducing an appropriately chosen baseline $b(\sigma) \in \mathbb{R}$, which is subtracted from the return G_{t_d} in the update rule (5.12). As long as the baseline is not dependent on the action, it does not change the expected value in (5.11), as proven in [161, Section 13.4]. The resulting algorithm is called REINFORCE with baseline and uses a slightly modified update rule:

$$\boldsymbol{\theta}_\mathrm{new} = \boldsymbol{\theta}_\mathrm{old} + \eta_\theta \cdot \frac{1}{N_\mathrm{ep}} \sum_{t_\mathrm{d}=1}^{N_\mathrm{ep}} \gamma^{t_\mathrm{d}} \Big(G_{t_\mathrm{d}} - b(S_{t_\mathrm{d}}) \Big) \frac{\nabla_{\boldsymbol{\theta}}\, \pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta}_\mathrm{old})}{\pi(A_{t_\mathrm{d}} \mid S_{t_\mathrm{d}}, \boldsymbol{\theta}_\mathrm{old})}. \tag{5.13}$$

A commonly used baseline is an estimate $\hat{v}_\pi$ of the state-value function v_π, which can be generated using the policy evaluation techniques from the value-based methods. To handle high-dimensional state and action spaces, typically a differentiable function approximator $\hat{v}_\pi(\sigma, \boldsymbol{\phi})$ is used, where $\boldsymbol{\phi} \in \mathbb{R}^{N_\phi}$ is the parameter vector, and $N_\phi \in \mathbb{N}$ represents the number of parameters. The vector $\boldsymbol{\phi}$ is adjusted to minimize the expected squared error $\mathbb{E}_\pi \left[\frac{1}{2} \left(v_\pi(\sigma) - \hat{v}_\pi(\sigma, \boldsymbol{\phi}) \right)^2 \right]$. As with updating $\boldsymbol{\theta}$, multiple samples of the expected error can be generated at the end of an episode from the received returns G_{t_d}. These samples are then averaged to estimate the expected error. Using stochastic gradient descent with the step size $\eta_\phi \in \mathbb{R}_{>0}$ yields the following update rule:

$$\begin{aligned} \boldsymbol{\phi}_\mathrm{new} &= \boldsymbol{\phi}_\mathrm{old} - \eta_\phi \cdot \frac{1}{N_\mathrm{ep}} \sum_{t_\mathrm{d}=1}^{N_\mathrm{ep}} \frac{1}{2} \nabla_{\boldsymbol{\theta}} \Big[G_{t_\mathrm{d}} - \hat{v}_\pi(S_{t_\mathrm{d}}, \boldsymbol{\phi}_\mathrm{old}) \Big]^2 && \text{(5.14a)} \\ &= \boldsymbol{\phi}_\mathrm{old} + \eta_\phi \cdot \frac{1}{N_\mathrm{ep}} \sum_{t_\mathrm{d}=1}^{N_\mathrm{ep}} \Big[G_{t_\mathrm{d}} - \hat{v}_\pi(S_{t_\mathrm{d}}, \boldsymbol{\phi}_\mathrm{old}) \Big] \cdot \nabla_{\boldsymbol{\theta}} \hat{v}_\pi(S_{t_\mathrm{d}}, \boldsymbol{\phi}_\mathrm{old}). && \text{(5.14b)} \end{aligned}$$

For both algorithms, namely REINFORCE and REINFORCE with baseline, it is necessary to wait until the end of an episode to compute the returns G_{t_d} required for the update rules (5.12), (5.13), and (5.14b). This is data inefficient and leads to slow learning. Moreover, the use of returns, which may significantly vary between episodes, still can lead to increased variance despite the introduced baseline $b(\sigma)$. Both of these issues are addressed by the following actor-critic methods.

Actor-Critic Methods

Actor-critic methods combine aspects of value-based and policy-based methods and consist of two instances: the actor and the critic. As with the policy-based methods, the actor is a parameterized policy $\pi(\alpha \mid \sigma, \boldsymbol{\theta})$ that selects the actions. Similar to value-based methods, the critic is a parameterized estimate of a value function, usually the state-value function $\hat{v}_\pi(\sigma, \boldsymbol{\phi})$, which is used to assess the actor's actions. It may appear that the REINFORCE with baseline algorithm could also be classified as an actor-critic method, as it involves learning both a policy and a state-value function. However, this algorithm uses the state-value function solely as a baseline by evaluating the current state S_{t_d}. This contrasts with actor-critic methods, where the value function assesses the actor's chosen actions. As will be discussed shortly, this is achieved by evaluating the value function in the state at time step $t_\mathrm{d} + 1$ or later, indirectly assessing the action A_{t_d} that leads to $S_{t_\mathrm{d}+1}$.

The basic idea is that the return G_{t_d}, which is only known at the end of an episode, is replaced at each time step t_d of the episode by an estimate based on Bellmann's equation (5.6). Thus, in each time step, the return G_{t_d} can be approximated by $R_{t_\mathrm{d}+1} + \gamma \hat{v}_\pi(S_{t_\mathrm{d}+1}, \boldsymbol{\phi})$, and an update can be performed analogously to (5.13) and (5.14b). This approximation is referred to as the one-step return, and the general idea of using collected rewards together with the estimated state-value function $\hat{v}_\pi$ to approximate the return G_{t_d} is called bootstrapping. Using the one-step return instead of G_{t_d} can accelerate convergence as updates can be performed at each time step. Also, it reduces variance, as $R_{t_\mathrm{d}+1}$ is the only random variable, thus involving fewer random variables compared to using the entire return G_{t_d}. However, the reduced variance comes at the cost of high *bias*, which refers to the systematic error introduced when an estimate, such as $\hat{v}_\pi$, is used instead of the actual value G_{t_d}. A trade-off between high variance and bias can be achieved if bootstrapping is performed from a later time step. For instance, in the case of a two-step return, G_{t_d} is replaced by $R_{t_\mathrm{d}+1} + \gamma R_{t_\mathrm{d}+2} + \gamma^2 \hat{v}_\pi(S_{t_\mathrm{d}+2}, \boldsymbol{\phi})$. In a generalized form, the n-step return is defined, where bootstrapping is applied starting from the n-th step after t_d. An approach that uses the weighted average of all n-step returns, referred to as Generalized Advantage Estimation (GAE), is described in [165].

In the Asynchronous Advantage Actor-Critic (A3C) algorithm [166], multiple agents operate asynchronously, each collecting data over a specified number

of time steps while following a shared policy. Once an agent has completed the time steps, the longest possible n-step return is calculated for each time step, and these are used to update the shared policy.[2] A synchronous variant of A3C, called Advantage Actor-Critic (A2C), waits for all agents to complete their steps before applying the collected data in a joint update step.

All introduced policy-based and actor-critic methods use each collected data sample only once for an update and then discard it. While performing multiple update steps with the same data samples could accelerate training, there is a risk of excessively large policy updates that could lead to instability. The Proximal Policy Optimization (PPO) algorithm [167] addresses this by clipping the ratio $\frac{\pi(A_{t_\mathrm{d}}|S_{t_\mathrm{d}},\boldsymbol{\theta}_\mathrm{new})}{\pi(A_{t_\mathrm{d}}|S_{t_\mathrm{d}},\boldsymbol{\theta}_\mathrm{old})}$ to values near 1, ensuring that the policy changes remain sufficiently small even when updates are applied multiple times using the same data samples. Since, after the first update iteration, the policy used to collect the data and the policy to be updated no longer align, importance sampling is applied. This is a method for evaluating the expected value of a particular distribution while only samples from another distribution are available. Additionally, the GAE is utilized to approximate the return G_{t_d}. PPO is widely used due to its simplicity, efficiency, and robustness, and it will also be employed in this chapter.

5.1.4 Further Concepts

Many RL algorithms lack convergence guarantees or provide them only under assumptions that are often not met in practice. Consequently, the success of training depends on the definition of the MDP, the choice of the policy function approximator, and the selection of hyperparameters. Additionally, various techniques can be employed to make training more robust and efficient.

One challenge in RL, particularly for long episodes, is the occurrence of sparse rewards, where the agent only receives rewards at the end of an episode or sporadically. This delayed feedback makes it difficult for the agent to attribute specific actions to the outcome, known as the *credit assignment problem*, potentially slowing down or preventing successful training. Reward shaping addresses this issue by breaking the problem into sub-problems, allowing the agent to receive more immediate feedback for its actions, such as a reward

[2] Typically, actor-critic methods are implemented using gradient-based optimization techniques, which are more advanced than the gradient ascent approach in (5.10).

at each time step t_d. These direct rewards, called dense rewards, can enable faster and more robust training. However, reward shaping introduces human domain knowledge into the problem formulation, which might cause the agent to converge to a local optimum [168].

RL algorithms are also combined with curriculum learning to tackle complex tasks [169, 170]. This approach divides the learning process into multiple tasks of increasing complexity. Instead of allowing the agent to act directly in the final MDP, it is first trained in simpler MDPs that represent less challenging versions of the task. The policy learned in each stage is then transferred to the next stage, progressively guiding the agent toward the final MDP.

Lastly, when using an NN as a function approximator, the state and action spaces should be normalized to values within the interval $[-1, 1]$. One reason is that normalized data tends to lower estimation errors and reduce computation times in NNs [171]. Additionally, most RL algorithms with continuous action spaces use a Gaussian distribution from which the action is sampled. The NN outputs the mean, initialized to 0, and the standard deviation, initialized to 1. Normalization of the action space ensures that the NN operates within the desired range [172].

5.2 Blocking Scenario

The interactive blocking scenario, including the objective of the ego vehicle, is described in Subsection 5.2.1. The system dynamics of the opponent vehicle and its control law intended for blocking are covered in Subsection 5.2.2.

5.2.1 Scenario Description and Goal

The blocking scenario takes place on the straight race track illustrated in Figure 5.2, with a width of 15 m and a length of 1500 m. The center line serves as the reference line $\mathfrak{R}$, i.e., $d_\mathrm{l}(s) = 7.5\,\mathrm{m}$ and $d_\mathrm{r}(s) = -7.5\,\mathrm{m}$ hold for the distances to the track boundaries. The race track is completely flat, meaning that $\varphi(s) = \mu(s) = 0^\circ$ and $\Omega_\mathrm{x}(s) = \Omega_\mathrm{y}(s) = 0\,\mathrm{rad/m}$ apply. Additionally, this leads to $\tilde{a}_\mathrm{x}(t) = \hat{a}_\mathrm{x}(t)$, $\tilde{a}_\mathrm{y}(t) = \hat{a}_\mathrm{y}(t)$, $\hat{a}_\mathrm{z}(t) = 0\,\mathrm{m/s^2}$, and $\tilde{g}(t) = g$, implying that the gg-diagrams depend solely on the velocity V. Since the reference line is straight, $\theta(s) = 0^\circ$ and $\Omega_\mathrm{z}(s) = 0\,\mathrm{rad/m}$ also hold. Nevertheless, both quantities

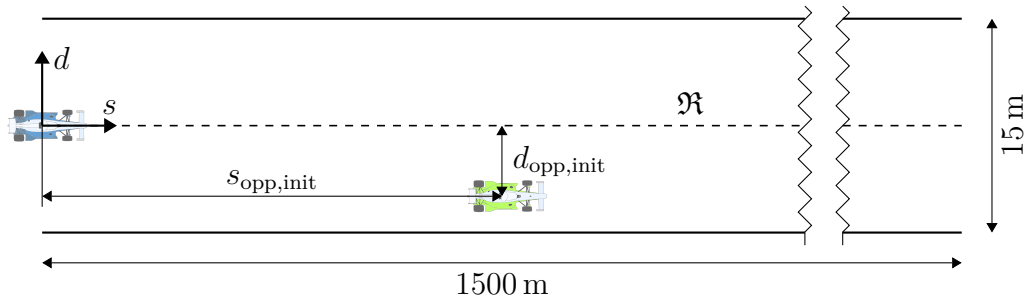


Figure 5.2: Initialization of the blocking scenario with the ego vehicle (blue) at position ($s_{\text{ego,init}} = 0\,\text{m}, d_{\text{ego,init}} = 0\,\text{m}$) and the opponent vehicle (green) at ($s_{\text{opp,init}}, d_{\text{opp,init}}$). Both vehicles are initialized with the relative orientation $\hat{\chi}_{\text{init}} = 0°$ and the velocity $V_{\text{init}} > 0\,\text{m/s}$.

are incorporated in the subsequent equations, allowing adaptation to curved tracks in future work.

The scenario consists of an ego vehicle and a blocking opponent vehicle, initialized as shown in Figure 5.2. The ego vehicle, whose variables are labeled with the index ego, begins at the start of the track with the progress $s_{\text{ego,init}} = 0\,\text{m}$ and the lateral position $d_{\text{ego,init}} = 0\,\text{m}$. The opponent vehicle, whose variables are labeled with the index opp, is positioned ahead of the ego vehicle at progress $s_{\text{opp,init}} > 0\,\text{m}$ and lateral position $d_{\text{opp,init}} \in [d_{\text{r}}, d_{\text{l}}]$. Both vehicles are initialized with the relative orientation $\hat{\chi}_{\text{init}} = 0°$ and the velocity $V_{\text{init}} > 0\,\text{m/s}$.

The objective of the ego vehicle is to overtake the blocking opposing vehicle without colliding. The scenario is considered successful if the ego vehicle has completed an overtaking maneuver without a collision by the end of the race track. An overtaking maneuver is considered complete when the ego vehicle achieves a longitudinal lead of five times the vehicle length $D_{\text{l}} \in \mathbb{R}_{>0}$, which is satisfied if

$$s_{\text{ego}}(t) > s_{\text{opp}}(t) + 5 \cdot D_{\text{l}}. \tag{5.15}$$

However, the scenario ends unsuccessfully if a collision occurs, no overtaking maneuver can be completed by the end of the race track, or a planned trajectory is infeasible.

5.2.2 Opponent Vehicle Dynamics

While the ego vehicle is controlled by the corresponding planning approach under the assumption of perfect tracking, the opponent vehicle's blocking behavior is realized by a vehicle model controlled by a PD controller. For the vehicle model, the kinematic single-track model in Curvilinear coordinates derived in Appendix D is used. Omitting the index opp for clarity, the system dynamics are given by

$$\dot{s}(t) = V(t) \cdot \cos(\hat{\chi}(t)) \cdot \frac{1}{1 - d(t) \cdot \Omega_\mathrm{z}(s(t))}, \tag{5.16a}$$

$$\dot{d}(t) = V(t) \cdot \sin(\hat{\chi}(t)), \tag{5.16b}$$

$$\dot{\hat{\chi}}(t) = \frac{V(t)}{l_\mathrm{r}} \cdot \sin(\beta(t)) - V(t) \cdot \cos(\hat{\chi}(t)) \cdot \frac{\Omega_\mathrm{z}(s(t))}{1 - d(t) \cdot \Omega_\mathrm{z}(s(t))}, \tag{5.16c}$$

$$\dot{V}(t) = \hat{a}_\mathrm{x}(t), \tag{5.16d}$$

$$\dot{\delta}(t) = \omega(t), \tag{5.16e}$$

with the body slip angle $\beta(t) = \arctan\left(\frac{l_\mathrm{r}}{l_\mathrm{r}+l_\mathrm{f}} \cdot \tan(\delta(t))\right) \in \mathbb{R}$. The state vector $\boldsymbol{z}_\mathrm{opp}(t) = \begin{bmatrix} s_\mathrm{opp}(t) & d_\mathrm{opp}(t) & \hat{\chi}_\mathrm{opp}(t) & V_\mathrm{opp}(t) & \delta_\mathrm{opp}(t) \end{bmatrix}^\top \in \mathbb{R}^5$ contains the opponent's longitudinal position $s_\mathrm{opp}(t)$, lateral position $d_\mathrm{opp}(t)$, relative orientation $\hat{\chi}_\mathrm{opp}(t)$, absolute velocity $V_\mathrm{opp}(t)$, and steering angle $\delta_\mathrm{opp}(t)$. The input vector $\boldsymbol{u}_\mathrm{opp}(t) = \begin{bmatrix} \omega_\mathrm{opp}(t) & \hat{a}_\mathrm{x,opp}(t) \end{bmatrix}^\top \in \mathbb{R}^2$ is composed of the steering rate $\omega_\mathrm{opp}(t)$ and the longitudinal acceleration $\hat{a}_\mathrm{x,opp}(t)$. The distances from the center of gravity to the rear and front axles, denoted by $l_\mathrm{r} \in \mathbb{R}_{\geq 0}$ and $l_\mathrm{f} \in \mathbb{R}_{\geq 0}$, define the vehicle's wheelbase. Additionally, the steering angle $|\delta_\mathrm{opp}(t)|$ is constrained by $\delta_\mathrm{max} \in \mathbb{R}_{>0}$, and the steering rate $|\omega_\mathrm{opp}(t)|$ is limited to $\omega_\mathrm{max} \in \mathbb{R}_{>0}$. For simplicity, an additional dynamic limitation using gg-diagrams is omitted, as the primary goal of the presented simulations is to provide a basic comparison between the different planning approaches rather than to accurately model the dynamics of blocking behavior. However, future research could incorporate more realistic dynamic vehicle models to enhance the accuracy of the simulations.

The longitudinal and lateral control of the single-track model operate independently. For the longitudinal motion, $\hat{a}_\mathrm{x,opp}(t) = 0\,\mathrm{m/s^2}$ is applied, which prevents the opponent vehicle from simply driving ahead and thereby allows

the ego vehicle to overtake. For the lateral movement, the steering rate $\omega_{\text{opp}}(t)$ is determined by an empirically designed PD controller with the proportional gain $K_{\text{P}} \in \mathbb{R}$ and the derivative gain $K_{\text{D}} \in \mathbb{R}$:

$$\omega_{\text{opp}}(t) = K_{\text{P}} \cdot e(t) + K_{\text{D}} \cdot \dot{e}(t). \tag{5.17}$$

The control error $e(t) \in \mathbb{R}$ to be minimized is defined as the difference between the desired $\hat{\chi}_{\text{opp,d}}(t) \in \mathbb{R}$ and the actual relative orientation $\hat{\chi}_{\text{opp}}(t)$ of the opponent vehicle:

$$e(t) = \hat{\chi}_{\text{opp,d}}(t) - \hat{\chi}_{\text{opp}}(t). \tag{5.18}$$

The desired relative orientation $\hat{\chi}_{\text{opp,d}}(t)$ is the heading the vehicle must have to close a lateral gap $\Delta\bar{d}(t) \in \mathbb{R}$ at a longitudinal look-ahead distance $K_{\text{L}} \in \mathbb{R}_{>0}$. These two distances form a right-angled triangle in front of the opponent vehicle, which yields an expression for the desired angle $\hat{\chi}_{\text{opp,d}}(t)$:

$$\hat{\chi}_{\text{opp,d}}(t) = \arctan\left(\frac{\Delta\bar{d}(t)}{K_{\text{L}}}\right). \tag{5.19}$$

The lateral deviation $\Delta\bar{d}(t)$ naturally includes the lateral position difference $d_{\text{ego}}(t) - d_{\text{opp}}(t)$ between the ego and opponent vehicle, as minimizing this difference aligns with the objective of a blocking maneuver. Additionally, the lateral velocity difference $\dot{d}_{\text{ego}}(t) - \dot{d}_{\text{opp}}(t)$ is incorporated to react early to any initiation of lateral movement by the ego vehicle:

$$\Delta\bar{d}(t) = d_{\text{ego}}(t) - d_{\text{opp}}(t) + \dot{d}_{\text{ego}}(t) - \dot{d}_{\text{opp}}(t). \tag{5.20}$$

In particular, the look-ahead parameter K_{L} has a decisive influence on the opponent vehicle's behavior. The step responses shown in Figure 5.3 demonstrate that smaller look-ahead distances K_{L} reduce the settling time, indicating a more aggressive blocking behavior. This is because, with smaller values of K_{L}, the desired heading $\hat{\chi}_{\text{opp,d}}(t)$ becomes larger for the same lateral deviation $\Delta\bar{d}(t)$, resulting in greater control errors $e(t)$ and, consequently, higher steering rates $\omega_{\text{opp}}(t)$.

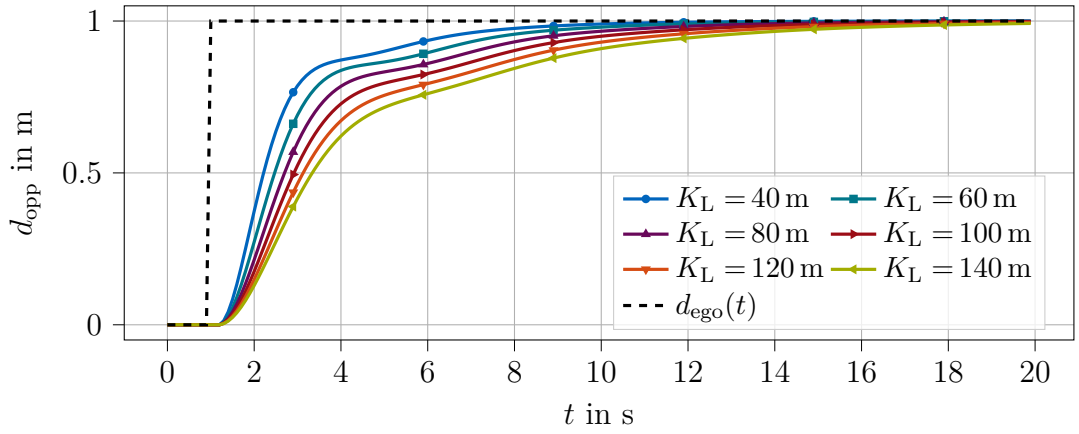


Figure 5.3: Step responses of the opponent vehicle for different look-ahead distances K_{L}. The remaining controller parameters are fixed to $K_{\text{D}} = 0.6$ and $K_{\text{P}} = 0.05\,\text{s}^{-1}$.

5.3 Methodology

While the opponent vehicle behaves according to the single-track model (5.16) with the control law (5.17) to (5.20), the ego vehicle is controlled by one of the following three planning algorithms. The conventional and game-theoretic planning approaches are described in Subsections 5.3.1 and 5.3.2, respectively. The RL-based planning approach follows in Subsection 5.3.3.

5.3.1 Conventional Planning Approach

The sampling-based approach described in Chapter 4 is used as the conventional planning approach. The racing line path is set as the center line of the straight track, and the racing line velocity is specified as the maximum vehicle speed $V_{\max} \in \mathbb{R}_{>0}$, leading to $d_{\text{rl}}(t) = 0\,\text{m}$ and $\dot{s}_{\text{rl}}(t) = V_{\text{rl}}(t) = V_{\max}$. Since these constant racing line profiles can be represented by jerk-optimal curves, jerk-optimal trajectory generation is sufficient, and relative trajectory generation can be omitted. This leads to the following differences compared to the description in Chapter 4.

Trajectory Generation

The set of longitudinal curves $\mathfrak{T}_{\text{long}}$ is generated by sampling $N_{\dot{\text{s}}}$ equidistantly distributed end velocities within the interval $\dot{s}(T_{\text{traj}}) \in [0, V_{\max}]$. The corresponding end acceleration is fixed to $\ddot{s}(T_{\text{traj}}) = 0\,\text{m/s}^2$. Since no relative curve generation is performed, and the track widths $d_{\text{l}}(s)$ and $d_{\text{r}}(s)$ are fixed, the set of lateral curves $\mathfrak{T}_{\text{lat}}$ can be generated independently of the longitudinal curves. $\mathfrak{T}_{\text{lat}}$ is created by sampling N_{d} equidistantly distributed end positions within the interval $d(T_{\text{traj}}) \in [d_{\text{r}} + {}^{D_{\text{w}}}\!/_2,\, d_{\text{l}} - {}^{D_{\text{w}}}\!/_2]$ and setting the corresponding velocity and acceleration to $\dot{d}(T_{\text{traj}}) = 0\,\text{m/s}$ and $\ddot{d}(T_{\text{traj}}) = 0\,\text{m/s}^2$, respectively. The trajectory candidates are then formed by combining each element of both sets using the Cartesian product $\mathfrak{T}_{\text{long}} \times \mathfrak{T}_{\text{lat}}$.

Trajectory Selection

With $d_{\text{rl}}(t) = 0\,\text{m}$, $V_{\text{rl}}(t) = V_{\max}$, and a single opponent, the cost functional (4.7) for assigning costs to a trajectory candidate $(s_i(t), d_j(t))$ simplifies to

$$J = \int_0^{T_{\text{traj}}} \Big(c_{\text{d}} \cdot d_j^2(t) + c_{\text{V}} \cdot \left[V_{\max} - V_{ij}(t) \right]^2 + c_{\text{pr}} \cdot J_{\text{pr}}(t) \Big)\, \text{d}t. \tag{5.21}$$

The third cost term remains with the prediction $(s_{\text{pr}}, d_{\text{pr}})$ of the opponent vehicle as

$$J_{\text{pr}}(t) = \exp\left(-p_{\text{s}} \cdot \left[s_{\text{pr}}(t) - s_i(t) \right]^2 - p_{\text{d}} \cdot \left[d_{\text{pr}}(t) - d_j(t) \right]^2 \right). \tag{5.22}$$

Prediction

The prediction of the opponent vehicle's motion is performed separately for its longitudinal $s_{\text{pr}}(t)$ and lateral movement $d_{\text{pr}}(t)$. The prediction of the longitudinal motion is based on the assumption of constant velocity, as the opponent vehicle's acceleration is $0\,\text{m/s}^2$, according to Subsection 5.2.2. In the lateral direction, the interaction between both vehicles results in an individual reaction of the opponent vehicle for each trajectory candidate, which cannot be captured with a fixed prediction. Since a reliable prediction is not possible, commonly used prediction methods are considered, assuming either a constant lateral position (CLP) or constant heading (CH), with the prediction being clipped to stay within the track boundaries.

5.3.2 Game-Theoretic Planning Approach

The game-theoretic approach is similar to the conventional approach, with the difference that the behavior of the opponent vehicle is not predicted based on any assumptions. Instead, following the leader-follower principle in [112, 113], for each trajectory candidate, the individual reaction of the opponent vehicle is determined through forward simulation of the system dynamics (5.16) using the control law (5.17) to (5.20). Finally, the valid trajectory candidate with the best outcome is selected using (5.21) and (5.22) but with the opponent's individual response instead of the fixed prediction $(s_{\mathrm{pr}}, d_{\mathrm{pr}})$.

Depending on the definition, such approaches are sometimes not considered game-theoretic in the literature, as the opponent's behavior is predefined using a controlled vehicle model, meaning the actions of both players are not optimized simultaneously. In this work, it is nevertheless classified as a game-theoretic approach as it resembles a Stackelberg game [173], where the ego vehicle acts as the leader and the opponent vehicle as the follower. This provides a clear distinction from the conventional approach, where the same fixed prediction is used for all trajectory candidates. However, it should be noted that this approach is not representative of all game-theoretic planning methods found in the literature.

5.3.3 Reinforcement Learning-Based Planning Approach

The RL-based planning approach employs the same jerk-optimal trajectory generation as the conventional and game-theoretic approach. However, instead of sampling the end state to produce a set of trajectory candidates, the end state is determined directly by the RL agent, similar to [143]. By connecting the vehicle's current state to the selected end state, only one trajectory is generated, eliminating the need for trajectory selection. After generating the trajectory, the same validity checks as in the other approaches are applied.

Once the agent has been trained, the generated trajectory is not guaranteed to satisfy the validity constraints. To prevent the episode from being terminated unsuccessfully in these cases, as done in [143], an SL is introduced. It produces a valid trajectory that closely resembles the original invalid one. Further details on the MDP formulation, training, and SL are provided below.

MDP Formulation

The state space $\mathfrak{s}$ of the MDP consists of two parts. It includes the ego vehicle's state $\begin{bmatrix} s_{\text{ego}} & \dot{s}_{\text{ego}} & \ddot{s}_{\text{ego}} & d_{\text{ego}} & \dot{d}_{\text{ego}} & \ddot{d}_{\text{ego}} & \hat{\chi}_{\text{ego}} \end{bmatrix}^\top$ and its deviation from the opponent's state $\begin{bmatrix} s_{\text{ego}} - s_{\text{opp}} & \dot{s}_{\text{ego}} - \dot{s}_{\text{opp}} & d_{\text{ego}} - d_{\text{opp}} & \dot{d}_{\text{ego}} - \dot{d}_{\text{opp}} & \hat{\chi}_{\text{ego}} - \hat{\chi}_{\text{opp}} \end{bmatrix}^\top$. The relative orientation $\hat{\chi}$ is redundantly included, as it can be derived from the Curvilinear coordinates using (A.8). However, experiments have shown that direct integration into the state space results in more efficient training. Additionally, as stated in Subsection 5.1.4, all quantities are normalized by dividing them by their respective maximum values, ensuring they lie in the range $[-1, 1]$.

The action space $\mathfrak{a}$ contains the end state $\begin{bmatrix} d(T_{\text{traj}}) & \dot{d}(T_{\text{traj}}) & \ddot{d}(T_{\text{traj}}) & \dot{s}(T_{\text{traj}}) \end{bmatrix}^\top$ of the jerk-optimal trajectory at time T_{traj}. As with the state space, all variables are normalized. The end time T_{traj} is fixed and not determined by the agent, ensuring a consistent planning horizon. Furthermore, the longitudinal end acceleration $\ddot{s}(T_{\text{traj}})$ is not included in the action space and is instead set to $0\,\text{m/s}^2$. Otherwise, negative end accelerations would be selected by the agent, which enable increased initial accelerations but result in undesirable high acceleration variations.

The dynamics P of the MDP include the state transition from the current t_d to the next discrete step $t_\text{d} + 1$ and the corresponding received reward $R_{t_\text{d}+1}$. Each step t_d represents a planning cycle, and the next state is determined under the assumption of perfect control of the planned trajectory for the ego vehicle, combined with forward integration of the system dynamics (5.16) for the opponent vehicle using the step size Δt_{sim}.

The received reward $R_{t_\text{d}+1}$ consists of sparse rewards, obtained at the end of an episode, and dense rewards $\bar{R}_{t_\text{d}+1} \in \mathbb{R}$, received at each discrete step. For sparse rewards, $R_{\text{success}} \in \mathbb{R}_{>0}$ is granted if the episode ends successfully, while $R_{\text{failure}} \in \mathbb{R}_{<0}$ is assigned if it ends unsuccessfully:

$$R_{t_\text{d}+1} = \begin{cases} R_{\text{success}} & \text{if episode ends successful,} \\ R_{\text{failure}} & \text{if episode ends unsuccessful,} \\ \bar{R}_{t_\text{d}+1} & \text{otherwise.} \end{cases} \tag{5.23}$$

The dense reward $\bar{R}_{t_\mathrm{d}+1}$ incentivizes high relative speeds and large lateral distances to guide the agent toward successful overtaking maneuvers. It is empirically designed to

$$\begin{aligned}\bar{R}_{t_\mathrm{d}+1} = &\left(\Delta\dot{s} - \Delta\dot{s}_\mathrm{max}\right) \cdot \mathbb{1}\left(\Delta\dot{s} - \Delta\dot{s}_\mathrm{max} > 0\right) \\ &+ \frac{1}{2}\left(\left|d_\mathrm{ego} - d_\mathrm{opp}\right| - k_\mathrm{scl} D_\mathrm{w}\right) \cdot \mathbb{1}\left(s_\mathrm{ego} \in \left[s_\mathrm{opp} - D_\mathrm{l}, s_\mathrm{opp} + D_\mathrm{l}\right]\right),\end{aligned} \tag{5.24}$$

where the ego and opponent vehicle variables correspond to the quantities from step $t_\mathrm{d}+1$, whose dependence is omitted here for clarity. Further, the indicator $\mathbb{1}(\square)$ specifies whether the condition $\square$ is fulfilled:

$$\mathbb{1}\left(\square\right) = \begin{cases} 1 & \text{if } \square \text{ is True,} \\ 0 & \text{if } \square \text{ is False.} \end{cases} \tag{5.25}$$

The first term in (5.24) rewards high relative speeds $\Delta\dot{s} = \dot{s}_\mathrm{ego} - \dot{s}_\mathrm{opp} \in \mathbb{R}$. To continuously encourage higher speeds, the maximum speed difference $\Delta\dot{s}_\mathrm{max} \in \mathbb{R}$ achieved up to the current step is subtracted. Also, to ensure the term does not act as a penalty, e.g., during braking maneuvers, it is only active when its value is positive. The second term rewards large lateral distances $|d_\mathrm{ego} - d_\mathrm{opp}|$ and is only active during an overtaking maneuver, i.e., when there is an overlap in the longitudinal direction between the two vehicles of length D_l. Since $|d_\mathrm{ego} - d_\mathrm{opp}|$ represents the distance between the centers of the vehicles, the vehicle width D_w is subtracted to determine the effective lateral distance. This ensures that a lateral overlap, indicating a potential collision, results in negative rewards, while positive rewards are given only when the vehicles are sufficiently separated laterally. Additionally, since the vehicle size is varied in the curriculum learning procedure described below, the vehicle width D_w is scaled by the factor $k_\mathrm{scl} \in (0, 1]$.

Training

The PPO algorithm is used for training due to its simplicity and ability to effectively handle continuous state and action spaces. It is an actor-critic method, as detailed in Subsection 5.1.3, and both the policy and state-value functions are approximated using fully connected NNs. The policy network maps the

state space to the action space, with an input layer dimension of 12 and an output layer dimension of 4. The four outputs of the policy network correspond to the means of individual Gaussian distributions, and the respective standard deviations are treated as learnable parameters. As in [167, 174], these standard deviations are initially set to 1 and are optimized together with the parameters of the NN. During training, actions are sampled from these distributions to facilitate exploration. After training, the mean of each distribution is selected as the action to ensure deterministic evaluation. Similarly, the state-value function network maps the state space to the state value, with an input layer dimension of 12 and an output layer dimension of 1. Both networks have two hidden layers, each containing 256 neurons. Each neuron uses a tanh activation function.

Additionally, curriculum learning with six progressively more complex stages is used for more efficient training. In the first stage, the agent is trained in single-vehicle driving to generate valid trajectories. Accordingly, collision checks are omitted, and the lateral distance term in the reward function (5.24) is excluded. In the subsequent five stages, the opponent vehicle is introduced, and the agent learns to overtake. The width and length of the vehicles are scaled by a factor $k_{\mathrm{scl}} \in \{0.2, 0.4, \ldots, 1.0\}$, which increases with each stage. This progressive scaling raises the difficulty of the overtaking task until the actual vehicle dimensions are used in the final stage.

Safety Layer

The trajectories generated by a trained RL agent are not guaranteed to be valid, meaning they might be infeasible or collide with the track boundaries. Particularly in situations not encountered during training, there is an increased probability of this. To prevent the episode from being terminated unsuccessfully due to an invalid trajectory $(s_{\mathrm{invalid}}(t), d_{\mathrm{invalid}}(t))$, the SL intervenes in the corresponding planning cycle by generating a valid trajectory that resembles the invalid one. For this, the same set of jerk-optimal trajectory candidates as used in the conventional and game-theoretic approaches is constructed. Each candidate $(s_i(t), d_j(t))$ is checked for validity, and the valid one minimizing the cost functional

$$J = \int_0^{T_{\mathrm{traj}}} \left(\left[s_{\mathrm{invalid}}(t) - s_i(t) \right]^2 + \left[d_{\mathrm{invalid}}(t) - d_j(t) \right]^2 \right) \mathrm{d}t \tag{5.26}$$

Table 5.1: Parameters for the results of the planning approaches (descriptions are provided in the list of symbols on page xvii)

Parameter	Value	Unit
D_w	1.93	m
D_l	4.9	m
l_r	1.72	m
l_f	1.25	m
$N_\dot{\mathrm{s}}$	40	-
N_d	20	-
N_traj	51	-
T_traj	2.5	s
R_success	10	-
R_failure	-1	-

Parameter	Value	Unit
Δt_sim	0.1	s
V_max	85	m/s
$\hat{\kappa}_\mathrm{max}$	0.2	m^{-1}
δ_max	0.43	rad
ω_max	0.39	rad/s
$\tilde{a}_\mathrm{y,max}(V)$	13 to 39	$\mathrm{m/s}^2$
$\tilde{a}_\mathrm{x,min}(V)$	-12 to -5	$\mathrm{m/s}^2$
$\tilde{a}_\mathrm{x,max}(V)$	3 to 5	$\mathrm{m/s}^2$
$\rho(V)$	1.5	-

is selected. This provides a valid trajectory resembling the invalid one, allowing the intended maneuver to be carried out. With this, an episode is only terminated if all trajectory candidates are invalid. Note that the SL is inactive during training, as the agent is supposed to independently learn the generation of valid trajectories. Therefore, the SL is only active during the evaluation of a fully trained agent, improving robustness and increasing the success rate.

5.4 Results

All results presented here were generated through simulation. Details on the implementation of the three planning approaches are given in Subsection 5.4.1, and the evaluation method of the approaches is described in Subsection 5.4.2. The actual evaluation results of the conventional, game-theoretic, and RL-based approaches are presented in Subsections 5.4.3, 5.4.4, and 5.4.5, respectively. Additionally, Subsection 5.4.5 presents the training results of the RL-based approach before the evaluation results.

Table 5.2: Environment parameters for the evaluation and training (descriptions are provided in the list of symbols on page xvii)

Parameter	Evaluation	Training #1	Training #2	Unit
$s_{\text{opp,init}}$	$\{20, 22, \dots, 100\}$	$[20, 100]$	$[20, 100]$	m
$d_{\text{opp,init}}$	$\{-6, -4, \dots, 6\}$	$[-6, 6]$	$[-6, 6]$	m
V_{init}	50	50	50	m/s
K_{L}	$\{40, 60, \dots, 140\}$	80	$\{40, 80, 120\}$	m
K_{P}	0.05	0.05	0.05	s^{-1}
K_{D}	0.6	0.6	0.6	-

5.4.1 Computational Details

All three planning approaches are implemented in Python, and the planned trajectories are represented by N_{traj} points distributed equidistantly in time. Feasibility checks are performed only at these discrete points, which are also used for the numerical integration of the cost functional (5.21) using the rectangle rule. For the RL-based approach, the Gymnasium library [175] is used to implement the environment, while the PPO implementation from Stable-Baselines3 [176] is employed with default parameters.

While the conventional approach uses the conventional planning structure shown in Figure 1.3a, the game-theoretic and RL-based approaches rely on the interaction-aware planning structure depicted in Figure 1.3b. In both cases, all modules are executed synchronously, allowing differences in computation times to be neglected and aligning the estimated ego state with the start state of the planning process, i.e., $\tilde{\boldsymbol{z}}_0 = \boldsymbol{z}_0$. Further, ideal state estimation and opponent detection are assumed. The forward Euler method is used with a step size of Δt_{sim} to integrate the opponent vehicle dynamics (5.16). The ego vehicle is propagated forward using the same step size, assuming perfect tracking of the planned trajectory. Most parameters of the simulation are listed in Table 5.1. Additional parameters for the environment and the cost functional are provided in separate tables below, as they are varied for the evaluation.

Table 5.3: Cost functional parameters of the conventional and game-theoretic approaches (descriptions are provided in the list of symbols on page xvii)

#	Approach	Ellipse	Prediction	c_d	c_V	c_pr	p_s	p_d
1	Conventional	Small	CH	0.08	0.28	5000	0.08	0.5
2	Conventional	Small	CLP	0.0	0.04	5000	0.08	0.5
3	Conventional	Medium	CH	0.0	0.08	5000	0.02	0.18
4	Conventional	Medium	CLP	0.72	1.0	5000	0.02	0.18
5	Conventional	Large	CH	0.36	0.24	5000	0.01	0.1
6	Conventional	Large	CLP	0.8	0.28	5000	0.01	0.1
7	Game-theoretic	Small	-	0.0	0.24	5000	0.08	0.5
8	Game-theoretic	Medium	-	0.0	0.96	5000	0.02	0.18
9	Game-theoretic	Large	-	0.0	1.0	5000	0.01	0.1

5.4.2 Evaluation Method

The three planning approaches are evaluated in the blocking scenario described in Section 5.2, using the parameters listed in the second column of Table 5.2. To investigate the impact of the opponent vehicle's behavior on the approaches, each approach is examined for the six look-ahead distances K_L shown in Figure 5.3, ranging from $K_\mathrm{L} = 40\,\mathrm{m}$ to $K_\mathrm{L} = 140\,\mathrm{m}$. While a high K_L corresponds to a conservative blocking behavior, implying less interaction, a low K_L represents an aggressive blocking behavior, resulting in higher interaction.

For each interaction level, i.e., look-ahead distance K_L, the opponent vehicle's initial position $(s_\mathrm{opp,init}, d_\mathrm{opp,init})$ is varied according to Table 5.2. With 41 options for $s_\mathrm{opp,init}$ and 7 options for $d_\mathrm{opp,init}$, a total of 287 episodes result per look-ahead distance. The percentage of successful episodes out of these 287 is referred to as the success rate and is used to assess the impact of interaction and to compare the approaches.

5.4.3 Conventional Planning Approach

The behavior of the conventional approach depends on the prediction method and the parameters of the cost functional (5.21) and (5.22). While CH and CLP predictions are employed, identifying the parameters c_d, c_V, c_pr, p_s, and p_d,

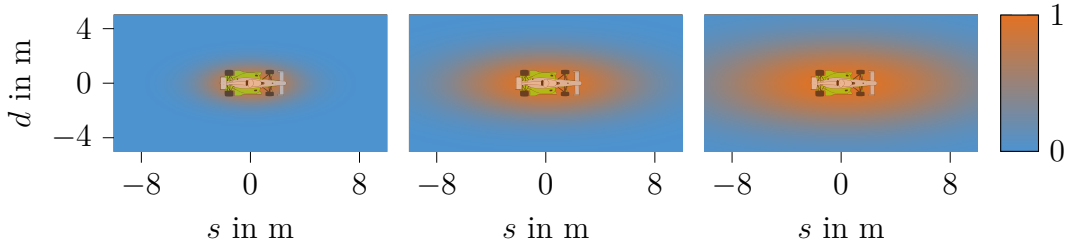


Figure 5.4: Elliptical prediction cost distribution J_{pr} for different parameterizations. Left: Small ellipse with $p_{\mathrm{s}} = 0.08$ and $p_{\mathrm{d}} = 0.5$. Center: Medium ellipse with $p_{\mathrm{s}} = 0.02$ and $p_{\mathrm{d}} = 0.18$. Right: Large ellipse with $p_{\mathrm{s}} = 0.01$ and $p_{\mathrm{d}} = 0.1$.

which maximize the mean success rate, is challenging. A full factorial search is impractical due to the number of parameters, so the six variants listed in Table 5.3 (#1 to #6) are evaluated. These variants were derived using the two predictions and the three ellipse sizes shown in Figure 5.4, defining p_{s} and p_{d}. Since the remaining parameters c_{d}, c_{V}, and c_{pr} serve as cost term weights, only their relative ratio is relevant, and c_{pr} is fixed across all variants. For the other two weights, a full factorial search is performed within the interval $[0, 1]$ using a step size of 0.04.

The success rates for the six variants are shown in Figure 5.5. Higher success rates are observed for larger values of K_{L}, as the prediction reflects the actual behavior of the opponent vehicle more accurately with less interaction. Overall, the highest success rates occur with the small ellipse, as fewer areas of the track have high costs, allowing overtaking maneuvers to be initiated immediately. Additionally, the CH prediction mostly results in higher success rates than the CLP prediction, as it captures the opponent vehicle's short-term behavior more accurately, ensuring that a larger lateral distance is maintained. Therefore, variant #1, which uses the small ellipse and CH prediction, achieves the highest average success rate, reaching nearly 100 % for all values of $K_{\mathrm{L}} \geq 60\,\mathrm{m}$. However, for the highest interaction level with $K_{\mathrm{L}} = 40\,\mathrm{m}$, the success rate drops to 0 %, as the simple strategy of immediately overtaking leads to collisions due to the aggressive blocking behavior. For other variants, the success rate at $K_{\mathrm{L}} = 40\,\mathrm{m}$ is also low, reflecting the challenges posed by high interaction.

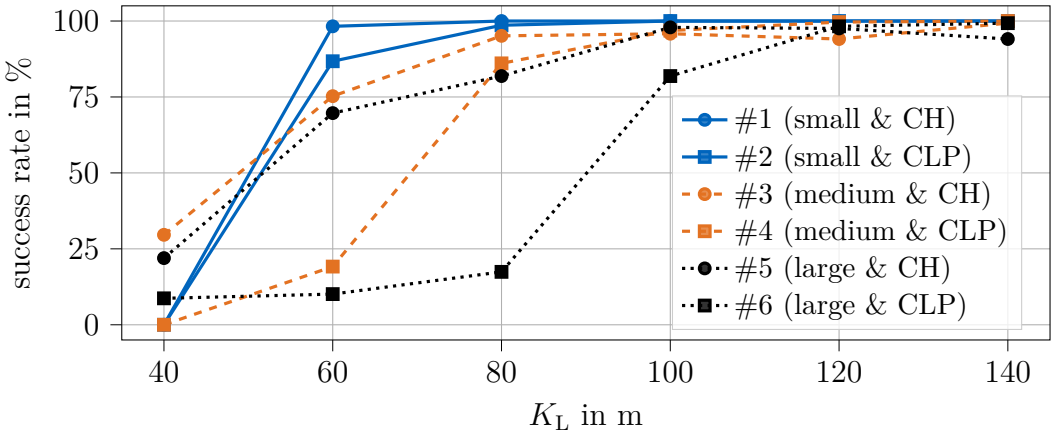


Figure 5.5: Success rates of the conventional approach (variants #1 to #6 from Table 5.3) for different look-ahead distances K_L. The size of the elliptical prediction cost distribution and the prediction method used are stated in brackets.

5.4.4 Game-Theoretic Planning Approach

As with the conventional approach, the game-theoretic approach's behavior depends on the parameters c_d, c_V, c_{pr}, p_s, and p_d of the cost functional (5.21) and (5.22). Therefore, the three variants listed in Table 5.3 (#7 to #9) are evaluated, which differ in the size of the elliptical prediction cost defined by p_s and p_d. The remaining three parameters are determined following the same procedure as for the conventional approach. However, due to significantly longer execution times, c_d is set to 0 for the full factorial search.

One might expect a success rate of 100 % for each variant of the game-theoretic approach, as the exact response of the opposing vehicle is exploited. However, this is not the case due to the limited maneuver options among the available trajectory candidates. The jerk-optimal candidates only represent simple left, straight, or right movements, which can be counteracted by aggressive blocking of the opponent vehicle. Instead of the simple trajectory candidates, candidates that correspond to a feinting maneuver would be necessary, where the opponent is first pulled to one side of the track before being overtaken on the other. With the jerk-optimal candidates, however, such a maneuver can only be executed across multiple planning cycles, which is not utilized in either the conventional or game-theoretic approach.

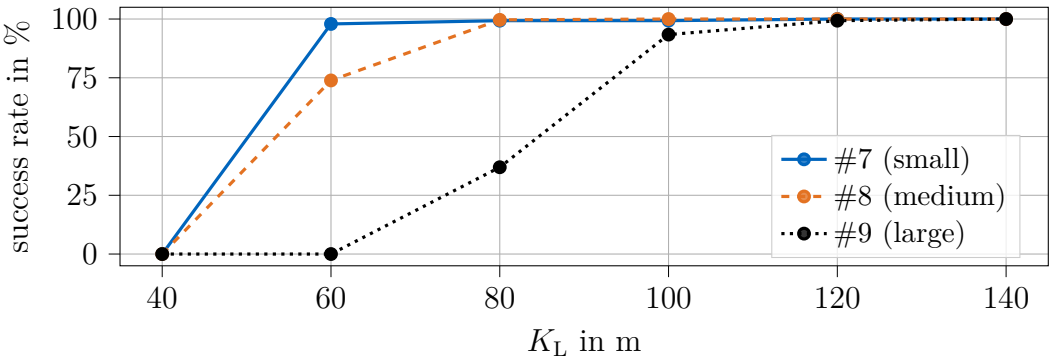


Figure 5.6: Success rates of the game-theoretic approach (variants #7 to #9 from Table 5.3) for different look-ahead distances K_L. The size of the elliptical prediction cost distribution used is stated in brackets.

Figure 5.6 shows the success rates of the three game-theoretic variants. Compared to the conventional approach shown in Figure 5.5, the success rates do not improve, and in some cases, the success rate is even lower. This is because, in the conventional approach with the CH prediction, there is typically a free side of the track, allowing a trajectory to be planned for overtaking. In contrast, the game-theoretic approach assigns high costs to all trajectories, as it accounts for the opponent's specific reactions. As a result, with the conventional approach, the ego vehicle adopts a more offensive overtaking behavior, which can lead to collisions when faced with aggressive blocking. In contrast, the game-theoretic approach leads to more conservative behavior, with the ego vehicle sometimes not initiating an overtaking maneuver to avoid a collision.

5.4.5 Reinforcement Learning-Based Planning Approach

This subsection first presents the training of the RL-based approach, followed by its evaluation. Finally, the SL is examined.

Training

To avoid overfitting, the scenario described in Section 5.2 is stochastically initialized at the beginning of an episode during training. Further, to investigate the influence of different environment initializations on the RL-based approach, the two training variants in the third and fourth columns of Table 5.2 are

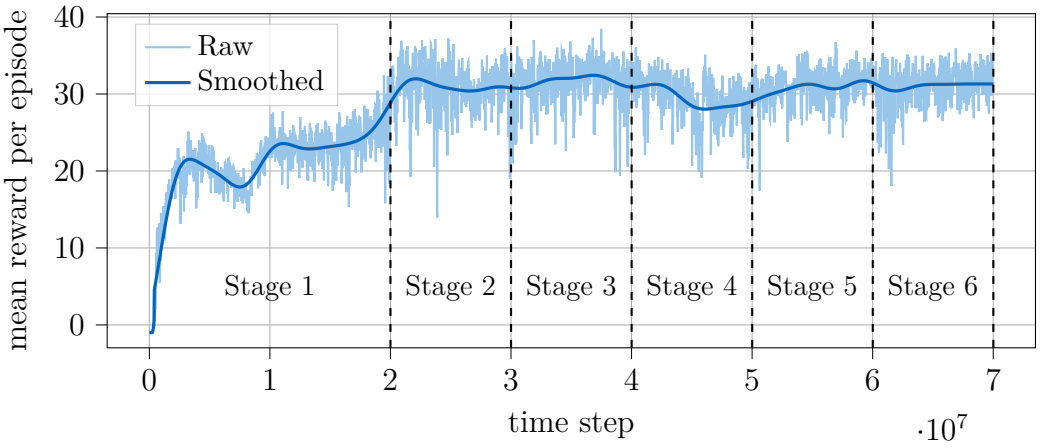


Figure 5.7: Mean cumulative reward per episode during training of the RL-based approach, averaged over the last 100 completed episodes every 4096 time steps (variant #2 from Table 5.2). The black dashed lines separate the six curriculum learning stages.

evaluated. In both variants, the opponent vehicle's position $(s_{\text{opp,init}}, d_{\text{opp,init}})$ is randomly initialized within the intervals specified in the table. The difference between both variants lies in the involved look-ahead distances K_{L}, i.e., opponent behaviors. In training variant #1, only a single K_{L} value is included, requiring generalization to the remaining blocking behaviors during evaluation, which comprises six K_{L} values. In contrast, in training variant #2, the opponent is randomly initialized with one of three different K_{L} values, reducing the need for generalization to only three additional behaviors.

The average cumulative reward per episode during training is exemplarily shown for training variant #2 in Figure 5.7. At the beginning of the training, no valid trajectories could be generated in the first time step of an episode, resulting in negative average rewards of $R_{\text{failure}} = -1$. In the first stage of training, the agent learned to create valid and accelerating trajectories. In the subsequent stages, it learned to overtake the opponent with progressively larger vehicle sizes. However, training exclusively in stage 6 without curriculum learning or transitioning directly from stage 1 to stage 6 proved unsuccessful.

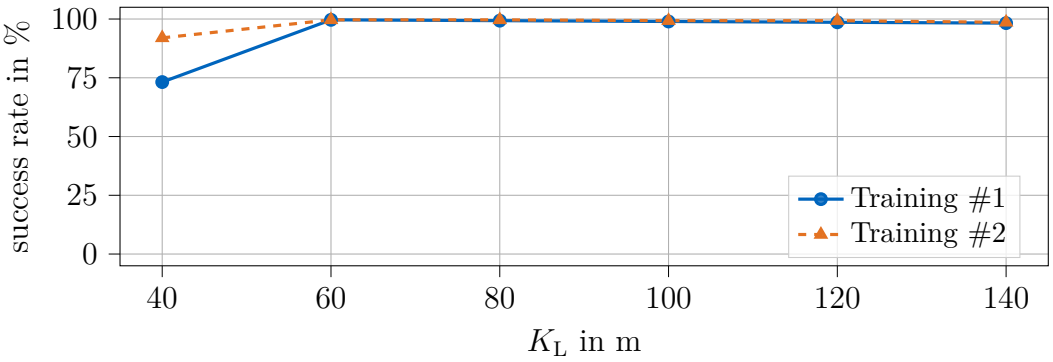


Figure 5.8: Success rates of the RL-based approach (training variants #1 to #2 from Table 5.2) for different look-ahead distances K_L.

Evaluation

Figure 5.8 shows the success rates for both training variants. For all values of K_L, except $K_L = 40\,\text{m}$, the success rates are nearly 100 %. For $K_L = 40\,\text{m}$, representing the most aggressive blocking behavior, the success rates are 73 % for training variant #1 and 92 % for training variant #2, highlighting the importance of the variety of opponent behavior included in the training.

Compared to the conventional and game-theoretic approaches, similar success rates are achieved for higher K_L values. However, for $K_L = 40\,\text{m}$, the RL-based approach yields significantly higher success rates. This can be attributed to the following two key factors:

1. More complex trajectories can be generated with the RL-based approach as the agent's action space is continuous, and the end conditions $\dot{d}(T_{\text{traj}})$ and $\ddot{d}(T_{\text{traj}})$ can be freely chosen. In contrast, due to computation time constraints, $\dot{d}(T_{\text{traj}})$ and $\ddot{d}(T_{\text{traj}})$ are fixed to $0\,\text{m/s}$ and $0\,\text{m/s}^2$, respectively, in the conventional and game-theoretic approaches. Additionally, the remaining end conditions can only take discrete values through sampling.

2. The RL-based approach can exploit the interaction with the opponent vehicle across multiple planning cycles, allowing the execution of feinting maneuvers, as shown in Figure 5.9. The blocking behavior is exploited by first driving to the right side of the track, causing the opponent vehicle to move laterally. The freed-up left side can then be used for an overtaking

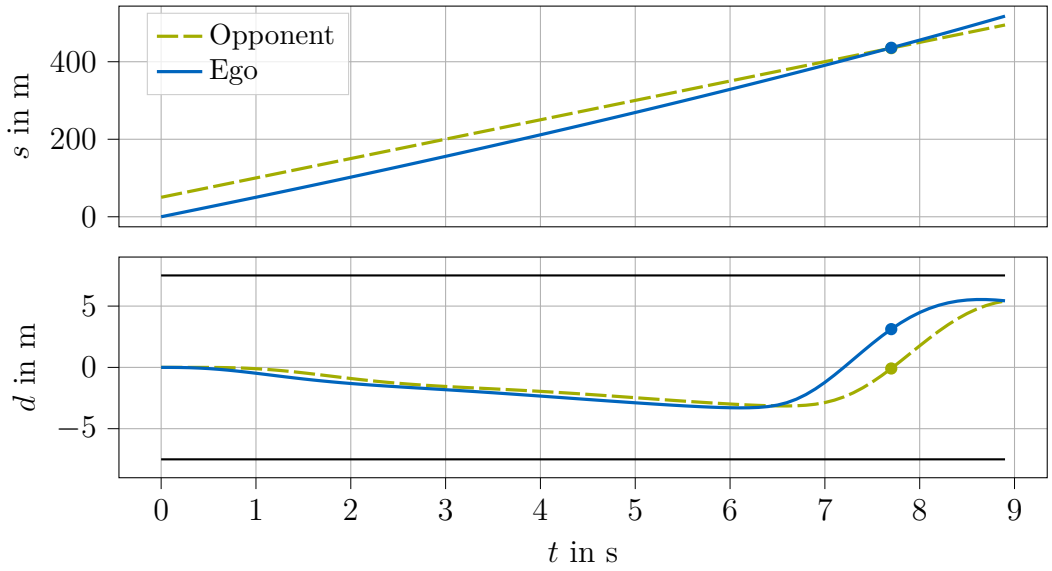


Figure 5.9: Overtaking maneuver in Curvilinear coordinates using the RL-based approach (variant #2 from Table 5.2). The opponent vehicle is initialized at $s_{\text{opp,init}} = 50\,\text{m}$ and $d_{\text{opp,init}} = 0\,\text{m}$, with a look-ahead distance of $K_{\text{L}} = 40\,\text{m}$. The point in time at which the ego and opponent vehicle have the same s-coordinate is indicated by circles. In the lower plot, the track boundaries are depicted as black lines.

maneuver. In contrast, the conventional and game-theoretic approaches select the optimal trajectory candidate within a single planning cycle and neglect that more complex maneuvers could be executed by composing the trajectories across multiple cycles.

The individual impacts of the two factors on the results are not investigated further in this thesis. Future studies could analyze this by evaluating the RL-based approach with an action space that is restricted to the sampled end states used in the conventional and game-theoretic approaches.

Safety Layer

The SL intervenes when the trajectory planned by the RL agent is invalid, generating a valid trajectory similar to the original. In the evaluation above, the RL agent did not generate an invalid trajectory in any planning cycle, so the SL was not triggered. This is because the states occurring in the evaluation are similar to those during training. To still examine the SL, the evaluation is

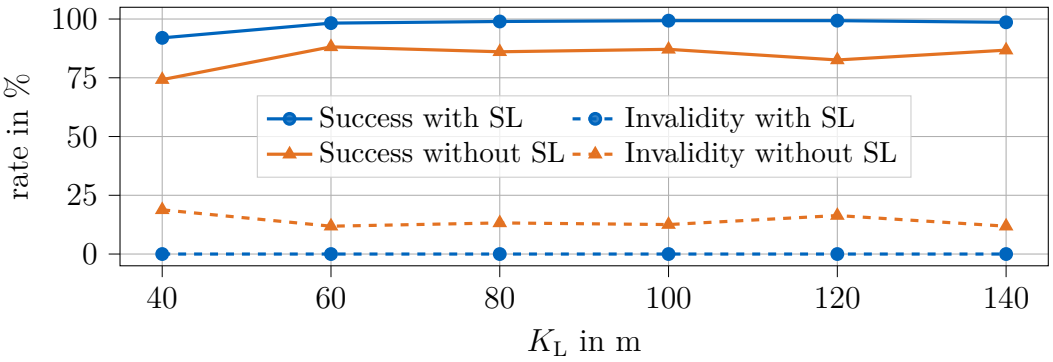


Figure 5.10: Success and invalidity rates of the RL-based approach (variant #2 from Table 5.2) with and without SL for different look-ahead distances K_L. Additive Gaussian noise with a mean of 0 m/s and a standard deviation of 0.7 m/s is applied to the opponent's longitudinal velocity $\dot{s}_\mathrm{opp}$.

repeated with additive Gaussian noise applied to the opponent's longitudinal velocity $\dot{s}_\mathrm{opp}$. Such noise could, for example, arise in real-world applications due to sensor uncertainties. By varying the opponent's velocity, states not seen during training occur, increasing the probability of invalid trajectories.

Figure 5.10 shows the success and invalidity rates for the RL-based approach with training variant #2, both with and without the SL. Analogous to the success rate, the invalidity rate indicates the percentage of episodes that ended unsuccessfully due to an invalid trajectory. Without the SL, the invalidity rate ranges between 10 % and 20 %, causing a corresponding drop in the success rate. With the SL, however, all invalid trajectories are replaced, reducing the invalidity rate to 0 %. Moreover, the trajectories generated by the SL approximate the invalid ones closely enough to achieve the same success rates as in evaluations without added noise.

In the analysis conducted here, the noise is chosen such that the trajectories planned by the agent are invalid but still represent meaningful motion. However, if the discrepancy between the states encountered during training and evaluation becomes too large, there is a risk that the agent plans trajectories that are fundamentally unrealistic or ineffective. In such cases, the SL could reduce the invalidity rate, but the success rate would likely remain similarly low.

5.5 Discussion

The conventional, game-theoretic, and RL-based approaches each exhibit different characteristics. The conventional approach relies heavily on cost parameterization and the selected prediction method. It enables aggressive overtaking behavior when well-configured, yielding high success rates in scenarios with limited interaction, i.e., at high look-ahead distances K_L. However, the conventional approach is unsuitable for high levels of interaction, as the prediction deviates more from the actual opponent behavior, leading to an increased collision rate. Generally, the conventional approach is suitable for a large variety of opponent behaviors but results in a conservative behavior or an increased collision rate for high-interaction scenarios.

With the game-theoretic approach, there is no mismatch between the opponent behavior incorporated in the planning process and the actual opponent behavior, as the individual opponent reaction is determined for each trajectory candidate. Therefore, when dealing with aggressive blocking behavior, the ego vehicle tends to follow the opponent vehicle instead of trying to overtake it. This reduces the collision rate compared to the conventional approach, but the success rate remains comparable. Both approaches only consider the trajectory candidates within the respective planning cycle and cannot perform more complex maneuvers by composing the simple trajectories across several time steps. Therefore, incorporating more sophisticated trajectory candidates or using a more advanced game-theoretic approach without discretizing the state space could improve the success rate. However, key limitations of the game-theoretic approach considered here are the need for precise knowledge of the opponent's behavior and the increased computational demand, as the dynamics of the controlled opponent vehicle must be simulated for each trajectory candidate. If the opposing vehicle behaves differently than simulated, however, higher collision rates may result than with the conventional approach. This fundamental problem also exists with alternative game-theoretic approaches, in which the behavior of both vehicles is optimized simultaneously based on a specific equilibrium. If the opposing vehicle does not behave according to the assumed equilibrium, performance can decrease significantly.

The RL-based approach presented here yields the highest success rates. Especially with the most aggressive blocking behavior, significantly higher success

rates can be achieved compared to the other two approaches, as the long-term behavior is exploited across several planning cycles. Unlike other RL-based approaches, the trajectory is generated directly by selecting the end state, simplifying real-world application compared to low-level action spaces and enabling more complex maneuvers compared to high-level action spaces. The proposed SL ensures a valid solution even when an invalid trajectory is generated. Additionally, the computation times are significantly reduced compared to the other two planning approaches - at least in the planning cycles without an active SL - as only one trajectory is generated and validated. However, before evaluation, the approach requires time-intensive offline training, often involving empirical tuning of the reward function and training parameters. Another fundamental challenge of RL-based methods is the generalization to unseen opponent behavior. If the opponent vehicle behavior in the evaluation differs significantly from the behavior contained in the training, the success rate can decrease accordingly. Therefore, as demonstrated by the two training variants examined in this chapter, diverse opponent behavior during training is essential for the success of RL-based approaches in the real world. Generalization improvements could involve adapting the state space or employing a more advanced NN architecture to incorporate opponent history.

The RL-based approach is limited to a straight, flat race track with a single opponent and is, therefore, not a fully developed planning approach for real-world use. Nonetheless, the comparison carried out here highlights the potential of RL in highly interactive scenarios if the associated challenges are addressed. Extending the state space could enable applications on closed race tracks or with multiple opponents. Furthermore, multi-agent RL is conceivable, involving simultaneous training of multiple agents in a shared environment. This allows game-theoretic equilibria to be reached by learning instead of solving a complex game online.

6 Conclusion and Outlook

Local trajectory planning for AD has been an active research topic for decades. To address edge cases on the progression from SAE level 4 to 5, supported by the rise of autonomous racing series, a dedicated research direction focusing on trajectory planning for autonomous racing has emerged.

This thesis presents three approaches that tackle different challenges of local trajectory planning for autonomous racing. In Section 6.1, these approaches are summarized, contextualized, and compared. Building on these, Section 6.2 explores potential extensions and identifies promising new research ideas. While detailed discussions of the three approaches are provided in Sections 3.3, 4.3, and 5.5, the key aspects are reiterated in this chapter.

6.1 Conclusion

Based on the related work in Section 1.3, six limitations of existing planning approaches were derived in Section 1.4. Accordingly, this thesis aimed to develop planning approaches that address these limitations. Since an essential aspect of this work is the participation in the IAC and A2RL racing series, the focus was also on real-world application. With this aim, three approaches were presented, treating the six limitations separately. The hybrid planning approach from Chapter 3 and the sampling-based approach from Chapter 4 have been applied to a full-scale race car in various IAC and A2RL races. Meanwhile, the RL-based approach from Chapter 5 is not yet suitable for real-world application and requires further work.

The hybrid planning approach from Chapter 3 addresses limitation 1, dealing with the curse of dimensionality in graph-based planning approaches. To keep the computation time low, existing approaches use a PVD, i.e., perform the planning of path and temporal profile separately, leading to a conservative multi-vehicle behavior. Alternative approaches without PVD use a spatio-temporal graph

but employ coarse discretization and simple edge structures, preventing driving at the vehicle's handling limits. In contrast, the presented hybrid approach uses a spatio-temporal graph with variable discretization and edge structure. The beginning of the trajectory, which is traversed according to the RH principle, is generated by sampling with increased smoothness. The remaining part of the trajectory is generated by a graph search with low discretization and simple edge structure, allowing short computation times despite a sufficiently long planning horizon. Various experiments have demonstrated increased vehicle stability and performance. However, the hybrid planning approach is only applicable for oval tracks as sufficiently low computation times are only achieved with high layer distances, which is ineffective with high curvatures. In addition, no 3D effects of the race track are considered, and the offline-generated racing line used does not incorporate the current ego vehicle state.

The sampling-based planning approach from Chapter 4 tackles limitations 2, 3, and 4, i.e., the missing consideration of 3D race track effects, the insufficient jerk-optimal trajectory candidates for road courses, and the use of an offline-generated racing line. The introduced relative trajectory generation allows the planning of more complex trajectories, enabling the following of the racing line on arbitrarily complex race tracks. Further, by extending the gg-diagrams with the vertical acceleration $\tilde{g}$, the race track's banking and slope can be accounted for, thus avoiding infeasibility and improving performance. Finally, it has been demonstrated that the online generation of the racing line in each planning cycle slightly degrades performance in single-vehicle racing due to the limited horizon. However, in multi-vehicle racing, it enhances performance by guiding the local planning using discrete trajectory candidates faster to the vehicle's limit.

The hybrid and sampling-based approaches are based on the conventional planning architecture consisting of sequential prediction and planning. While this has been sufficient in the previous IAC and A2RL races due to the strict racing rules, with future more relaxed rules, the vehicles will be able to interact more with each other. This may result in a mismatch between the prediction and actual opponent behavior, potentially leading to collisions and requiring the development of interaction-aware planning approaches.

In Chapter 5, the sampling-based approach from Chapter 4 is evaluated as a representative of conventional planning approaches in an interactive blocking scenario. It is compared with two interaction-aware approaches: a simple

game-theoretic approach and the RL-based one proposed here. The evaluation demonstrates an increase in the collision rate of the conventional approach with more aggressive blocking behavior of the opposing vehicle, i.e., increasing interaction. The game-theoretic approach, in which the individual opponent reaction is simulated for each trajectory candidate, reduces the collision rate. However, the success rate remains roughly the same, as in some cases, no attempt of overtaking is made. This is because only the available trajectory candidates are considered as possible trajectories, and none of them represents a successful overtaking maneuver in the case of aggressive blocking. Meanwhile, the presented RL-based planning approach achieves high success rates even for the most aggressive blocking behavior by performing more sophisticated maneuvers by composing the trajectories across several planning cycles. Also, the presented RL-based approach addresses limitations 5 and 6, which existing RL-based approaches for racing exhibit. Instead of using a low-level or high-level action space, as described in limitation 5, the action space in the proposed approach is designed to generate the trajectory directly. This increases maneuver diversity and enables a safer real-world application. Limitation 6 is the lack of safety guarantees of NNs, potentially resulting in invalid trajectories generated by the RL agent. In these cases, the introduced SL generates a valid trajectory similar to the original one, thus improving robustness.

6.2 Outlook

This section discusses the three approaches presented and offers suggestions for future work. As the approaches deal with the six limitations separately, concepts that combine the techniques are also explored. Furthermore, the potential of planning methods mentioned in the related work but not covered in this thesis is examined.

Efficient Implementation and Multi-Stage Sampling

The hybrid and sampling-based approaches discretize the state space, making the maneuver diversity and computation time significantly dependent on the generated edges or trajectory candidates. A more efficient implementation could, therefore, enable finer discretization and new capabilities for generating these

edges or candidates. In particular, the maneuver diversity of the sampling-based approach is currently limited, as the employed trajectory candidates only represent an outward or inward movement from the racing line. More complex maneuvers, such as overtaking an opponent vehicle and returning to the racing line, must instead be executed across multiple planning cycles. However, since only the current trajectory candidates are considered in each cycle, planning over multiple cycles is not utilized effectively. A multi-stage sampling concept, in which intermediate states are sampled in addition to the trajectory's end state, could enhance maneuver diversity. However, this would require not only a more efficient implementation but also a concept for the effective generation of these stages. For instance, inspired by the hybrid planning approach, a lower discretization could be used after the first stage. This would align with the fundamental idea of the hybrid approach but without a fixed layer structure or the assumption of constant acceleration for the subsequent layers.

Relative Trajectory Generation in the Spatial Domain

In the relative trajectory generation method of the sampling-based approach, jerk-optimal deviations are created, and the racing line profile is then added to them at the corresponding points in time. As a result, the generated trajectory candidates behave similarly to the racing line at each time point. For instance, when approaching a turn, the trajectory candidates start decelerating at nearly the same point in time as the racing line. However, if the speed profile of a trajectory candidate differs from that of the racing line, it may reach certain locations earlier or later. Consequently, the braking point location will be shifted compared to the braking point of the racing line, potentially leading to infeasibility issues.

Instead of generating the trajectory candidates in the temporal domain, using the spatial domain would be more effective. This approach captures braking points and other critical maneuvers more accurately by aligning the trajectories with specific locations on the track rather than time points of the racing line.

Real-World Application of Online Racing Line Generation

Online racing line generation guides the sampling-based local planning approach back to the vehicle limit faster after a deviation. In addition, the replanning

would simplify the race track testing process, as the acceleration limits could be adjusted online to efficiently approach the actual vehicle limit. However, the online racing line generation [71] used here still exhibits robustness issues and increased computational demands, preventing operation in the real world. A more advanced initialization of the optimization problem could address these issues. Alternatively, as an intermediate step toward the final solution, a simpler fixed-path optimization might be performed, where only the temporal profile of the racing line is replanned.

Additionally, an analysis of the impact of online racing line generation with a continuous local planning approach would be valuable. Without discretizing the state space, local planning is expected to return more quickly to the offline-generated racing line, potentially reducing the demonstrated performance benefits of online racing line generation.

Continuous Local Planning

With the hybrid and sampling-based approaches, only local planning approaches that discretize the state space have been considered in this thesis. Instead of further reducing the computation time to refine the discretization and generating suitable edges or trajectory candidates for each possible scenario, a continuous local planning approach would be a promising alternative. The main challenges lie in robust implementation and resolving non-convexity, especially in multi-vehicle racing. A potential strategy for the latter, as utilized by certain hybrid local planning approaches mentioned in Subsection 1.3.2, could involve a coarse graph search with simplified validity checks, whose solution initializes the optimization.

Interaction-Aware Planning

Considering vehicle interaction is a fundamental step toward human-like racing. While conventional planning methods reach their limits even in simple interactive scenarios, game-theoretic and RL-based approaches offer promising solutions. However, the demonstrated superiority of the RL-based approach over the game-theoretic approach discussed in Section 5.4 does not lead to a universally valid conclusion. A game-theoretic approach without state space discretization

could achieve comparable or superior success rates to the RL-based approach, motivating the following broader outlook.

The primary limitation of game-theoretic approaches is their rapidly increasing computational demand with each additional player, restricting most methods to two vehicles or requiring prioritization mechanisms. In contrast, RL-based approaches exhibit better scalability and, with appropriate state space design, remain independent of the number of opponents. However, this advantage comes at the cost of time-intensive and potentially sensitive training requiring manual tuning.

A significant drawback of RL-based methods is the lack of safety guarantees and explainability. This can be mitigated by techniques such as the SL introduced in this work or hybrid approaches where the RL agent parameterizes conventional methods, as explored in [141] and [144].

Further, both approaches face challenges regarding assumptions about opponent behavior. Game-theoretic approaches assume that the opponents act according to a specific equilibrium, while RL-based methods rely on the assumption that opponents behave similarly to the training data. These assumptions may not always hold, especially in the early development stages of many teams competing in racing series. However, RL-based methods possess the potential for generalization. Therefore, incorporating diverse opponent behaviors during training represents a key challenge for advancing RL-based approaches.

Given the high computational cost of solving games with more than two players, game-theoretic approaches are unlikely to be viable for real-world applications in the near future, leaving RL-based methods as the more practical alternative. Extending the RL-based approach presented in this thesis to road courses and multiple opponent vehicles could be a potential next step. Additionally, as more data becomes available, supervised learning approaches, commonly used for trajectory planning in road traffic, could become feasible for autonomous racing.

Optimization of Planning Parameters

The cost functionals in the hybrid and sampling-based planning approaches and the reward function in the RL-based approach contain tuning parameters that significantly influence the planning behavior. In this thesis, these parameters were tuned manually based on empirical results. However, manual tuning is time-consuming, relies on trial and error, and is subjective, as it depends on the

intuition and experience of the person performing the tuning. These drawbacks motivate a detailed parameter analysis and the application of systematic parameter optimization methods. However, optimization brings the challenge of defining a suitable objective that leads to the desired planning behavior. One approach is to formulate the optimization problem based on a set of predefined scenarios that the planning algorithm must handle effectively. By evaluating the planning performance in these scenarios, the optimization process aims to find parameter values that perform best across all situations. Nevertheless, covering all relevant situations with a finite set of scenarios remains challenging.

Opponent Behavior Identification

Vehicles on the race track can exhibit varying behaviors, especially when racing rules are relaxed. Accordingly, regardless of the chosen planning method, identifying the opponents' individual behavior has the potential to reduce collisions and enhance performance. For instance, in conventional methods, a basic prediction model could be adapted for each opponent vehicle based on its observed past movements. Similarly, in game-theoretic approaches, the cost functional guiding the behavior of opponent vehicles could be estimated. In RL-based approaches, the state space could be augmented to include either the past motions of opponent vehicles or dedicated estimated behavioral parameters.

Enhanced Degree of Realism

With more efficient implementations and algorithms, the approaches can be extended to better reflect reality. For a more accurate consideration of the vehicle dynamics, the actual geometry of the gg-diagrams could be incorporated into the planning process rather than relying on the computationally efficient underapproximation used here. Also, laterally curved race tracks, as shown in [61], could be used instead of the laterally flat tracks employed in this work. Furthermore, instead of using a point-mass model with the QSS assumption, more complex dynamic vehicle models could be employed, capturing transient vehicle behavior. However, the benefits should always be carefully weighed against the increased computational demands for higher levels of detail.

Adaption to Road Traffic

The methods developed for racing can be adapted for road traffic to enhance road safety. By considering 3D effects, particularly the slope angle, the vehicle's dynamic limits could be better exploited, which is especially important in edge cases such as emergency braking or evasive maneuvers. Furthermore, following the principles of hybrid planning or multi-stage sampling, a planning approach could be designed to meet high smoothness requirements and a long planning horizon, enhancing vehicle stability while simultaneously enabling far-sighted decision-making. Additionally, the RL-based approach incorporating the SL could be utilized to address interaction dynamics in road traffic.

Bibliography

[1] N. Thomopoulos and M. Givoni. "The autonomous car—a blessing or a curse for the future of low carbon mobility? An exploration of likely vs. desirable outcomes". In: *European Journal of Futures Research* 3.1, Article 14 (2015), pp. 1–14. DOI: 10.1007/s40309-015-0071-z (cit. on p. 1).

[2] T. J. Crayton and B. M. Meier. "Autonomous vehicles: Developing a public health research agenda to frame the future of transportation policy". In: *Journal of Transport & Health* 6 (2017), pp. 245–252. DOI: 10.1016/j.jth.2017.04.004 (cit. on p. 1).

[3] S. Singh. *Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey*. Tech. rep. DOT HS 812 506. Washington, DC, USA: National Highway Traffic Safety Administration, 2018. URL: https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812506 (cit. on p. 1).

[4] J. B. Greenblatt and S. Shaheen. "Automated Vehicles, On-Demand Mobility, and Environmental Impacts". In: *Current Sustainable/Renewable Energy Reports* 2.3 (2015), pp. 74–81. DOI: 10.1007/s40518-015-0038-5 (cit. on p. 1).

[5] R. Mudge, D. Montgomery, E. Groshen, J. P. Macduffie, S. Helper, and C. Carson. *America's Workforce and the Self-Driving Future: Realizing Productivity Gains and Spurring Economic Growth*. Tech. rep. Washington, DC, USA: Securing America's Future Energy, 2018. URL: https://avworkforce.secureenergy.org/ (cit. on p. 1).

[6] McKinsey & Company. *Autonomous driving's future: Convenient and connected*. 2023. URL: https://mckinsey.com/industries/automotive-and-assembly/our-insights/autonomous-drivings-future-convenient-and-connected (visited on 03/01/2025) (cit. on p. 1).

[7] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE Standard J3016_202104. 2021. DOI: 10.4271/J3016_202104 (cit. on p. 2).

[8] Honda Motor Co., Ltd. *Honda to Begin Sales of Legend with New Honda SENSING Elite*. 2021. URL: https://global.honda/en/newsroom/news/2021/4210304eng-legend.html (visited on 03/01/2025) (cit. on p. 2).

[9] Mercedes-Benz Group AG. *Certification for SAE Level 3 system for U.S. market*. 2023. URL: https://group.mercedes-benz.com/innovation/product-innovation/autonomous-driving/drive-pilot-nevada.html (visited on 03/01/2025) (cit. on p. 2).

[10] McKinsey & Company. *Change vehicles: How robo-taxis and shuttles will reinvent mobility*. 2019. URL: https://mckinsey.com/industries/automotive-and-assembly/our-insights/change-vehicles-how-robo-taxis-and-shuttles-will-reinvent-mobility (visited on 03/01/2025) (cit. on p. 2).

[11] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, T. Stahl, L. Hermansdorfer, B. Lohmann, and M. Lienkamp. "What can we learn from autonomous level-5 motorsport?" In: *9th International Munich Chassis Symposium 2018*. Ed. by P. Pfeffer. Wiesbaden: Springer Vieweg, 2019, pp. 123–146. DOI: 10.1007/978-3-658-22050-1_12 (cit. on p. 3).

[12] A. Jiang. "Research on the Development of Autonomous Race Cars And Impact on Self-driving Cars". In: *Journal of Physics: Conference Series* 1824.1, Article 012009 (2021), pp. 1–7. DOI: 10.1088/1742-6596/1824/1/012009 (cit. on p. 3).

[13] R. Behringer, S. Sundareswaran, B. Gregory, R. Elsley, B. Addison, W. Guthmiller, R. Daily, and D. Bevly. "The DARPA grand challenge - development of an autonomous vehicle". In: *IEEE Intelligent Vehicles Symposium, 2004*. 2004, pp. 226–231. DOI: 10.1109/IVS.2004.1336386 (cit. on p. 3).

[14] S. Thrun et al. "Stanley: The robot that won the DARPA Grand Challenge". In: *Journal of Field Robotics* 23.9 (2006), pp. 661–692. DOI: 10.1002/rob.20147 (cit. on p. 3).

[15] C. Urmson et al. "A robust approach to high–speed navigation for unrehearsed desert terrain". In: *Journal of Field Robotics* 23.8 (2006), pp. 467–508. DOI: 10.1002/rob.20126 (cit. on p. 3).

[16] P. G. Trepagnier, J. Nagel, P. M. Kinney, C. Koutsougeras, and M. Dooner. "KAT–5: Robust systems for autonomous vehicle navigation in challenging and unknown terrain". In: *Journal of Field Robotics* 23.8 (2006), pp. 509–526. DOI: 10.1002/rob.20128 (cit. on p. 3).

[17] D. Braid, A. Broggi, and G. Schmiedel. "The TerraMax autonomous vehicle". In: *Journal of Field Robotics* 23.9 (2006), pp. 693–708. DOI: 10.1002/rob.20140 (cit. on p. 3).

[18] C. Urmson et al. "Autonomous driving in urban environments: Boss and the Urban Challenge". In: *Journal of Field Robotics* 25.8 (2008), pp. 425–466. DOI: 10.1002/rob.20255 (cit. on p. 3).

[19] M. Montemerlo et al. "Junior: The Stanford entry in the Urban Challenge". In: *Journal of Field Robotics* 25.9 (2008), pp. 569–597. DOI: 10.1002/rob.20258 (cit. on p. 3).

[20] A. Bacha et al. "Odin: Team VictorTango's entry in the DARPA Urban Challenge". In: *Journal of Field Robotics* 25.8 (2008), pp. 467–492. DOI: 10.1002/rob.20248 (cit. on p. 3).

[21] J. Leonard et al. "A perception–driven autonomous urban vehicle". In: *Journal of Field Robotics* 25.10 (2008), pp. 727–774. DOI: 10.1002/rob.20262 (cit. on p. 3).

[22] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield. "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge". In: *Journal of Field Robotics* 25.9 (2008), pp. 598–614. DOI: 10.1002/rob.20260 (cit. on p. 3).

[23] I. Miller et al. "Team Cornell's Skynet: Robust perception and planning in an urban environment". In: *Journal of Field Robotics* 25.8 (2008), pp. 493–527. DOI: 10.1002/rob.20253 (cit. on p. 3).

[24] M. Nolte, T. Form, S. Ernst, R. Graubohm, and M. Maurer. "The Carolo-Cup Student Competition: Involving Students with Automated Driving". In: *2018 12th European Workshop on Microelectronics Education (EWME)*. 2018, pp. 95–99. DOI: 10.1109/EWME.2018.8629462 (cit. on p. 3).

[25] A. Agnihotri, M. O'Kelly, R. Mangharam, and H. Abbas. "Teaching Autonomous Systems at 1/10th-scale: Design of the F1/10 Racecar, Simulators and Curriculum". In: *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 2020, pp. 657–663. DOI: 10.1145/3328778.3366796 (cit. on p. 3).

[26] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, L. Hermansdorfer, T. Stahl, T. Herrmann, and M. Lienkamp. "A Software Architecture for the Dynamic Path Planning of an Autonomous Racecar at the Limits of Handling". In: *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*. 2019, pp. 1–8. DOI: 10.1109/ICCVE45908.2019.8965238 (cit. on p. 3).

[27] J. Betz, A. Wischnewski, A. Heilmeier, F. Nobis, T. Stahl, L. Hermansdorfer, and M. Lienkamp. "A Software Architecture for an Autonomous Racecar". In: *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*. 2019, pp. 1–6. DOI: 10.1109/VTCSpring.2019.8746367 (cit. on p. 3).

[28] J. Kabzan et al. "AMZ Driverless: The full autonomous racing system". In: *Journal of Field Robotics* 37.7 (2020), pp. 1267–1294. DOI: 10.1002/rob.21977 (cit. on p. 3).

[29] S. Nekkah et al. "The Autonomous Racing Software Stack of the KIT19d". In: *SAE International Journal of Connected and Automated Vehicles* 5.1 (2022), pp. 73–86. DOI: 10.4271/12-05-01-0007 (cit. on p. 3).

[30] J. Betz et al. "TUM autonomous motorsport: An autonomous racing software for the Indy Autonomous Challenge". In: *Journal of Field Robotics* 40.4 (2023), pp. 783–809. DOI: 10.1002/rob.22153 (cit. on pp. 3, 73).

[31] C. Chanyoung Chung, A. Finazzi, H. Seong, D. Lee, S. Lee, B. Kim, G. Gang, and D. Hyunchul Shim. "Autonomous System for Head-to-Head Race: Design, Implementation, and Analysis; Team KAIST at the Indy Autonomous Challenge". In: *IEEE Transactions on Field Robotics* 2 (2025), pp. 574–600. DOI: 10.1109/TFR.2024.3497922 (cit. on pp. 3, 20, 83, 84, 86).

[32] A. Raji et al. "er.autopilot 1.1: A Software Stack for Autonomous Racing on Oval and Road Course Tracks". In: *IEEE Transactions on Field Robotics* 1 (2024), pp. 332–359. DOI: 10.1109/TFR.2024.3501252 (cit. on pp. 3, 20, 83, 84, 86).

[33] A. Saba et al. "Fast and Modular Autonomy Software for Autonomous Racing Vehicles". In: *Field Robotics* 4 (2024), pp. 1–45. DOI: 10.55417/fr.2024001 (cit. on p. 3).

[34] B. D. Evans, H. W. Jordaan, and H. A. Engelbrecht. "High-performance Racing on Unmapped Tracks using Local Maps". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 851–856. DOI: 10.1109/IV55156.2024.10588861 (cit. on p. 7).

[35] D. Q. Mayne. "Model predictive control: Recent developments and future promise". In: *Automatica* 50.12 (2014), pp. 2967–2986. DOI: 10.1016/j.automatica.2014.10.128 (cit. on p. 7).

[36] J. Betz, H. Zheng, A. Liniger, U. Rosolia, P. Karle, M. Behl, V. Krovi, and R. Mangharam. "Autonomous Vehicles on the Edge: A Survey on Autonomous Vehicle Racing". In: *IEEE Open Journal of Intelligent Transportation Systems* 3 (2022), pp. 458–488. DOI: 10.1109/OJITS.2022.3181510 (cit. on pp. 8, 11).

[37] C. Katrakazas, M. Quddus, W.-H. Chen, and L. Deka. "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions". In: *Transportation Research Part C: Emerging Technologies* 60 (2015), pp. 416–442. DOI: 10.1016/j.trc.2015.09.011 (cit. on p. 11).

[38] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli. "A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles". In: *IEEE Transactions on Intelligent Vehicles* 1.1 (2016), pp. 33–55. DOI: 10.1109/TIV.2016.2578706 (cit. on p. 11).

[39] D. González, J. Pérez, V. Milanés, and F. Nashashibi. "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 (cit. on p. 11).

[40] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser. "A Review of Motion Planning for Highway Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 21.5 (2020), pp. 1826–1848. DOI: 10.1109/TITS.2019.2913998 (cit. on p. 11).

[41] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. "A survey of deep learning techniques for autonomous driving". In: *Journal of Field Robotics* 37.3 (2020), pp. 362–386. DOI: 10.1002/rob.21918 (cit. on p. 11).

[42] S. Aradi. "Survey of Deep Reinforcement Learning for Motion Planning of Autonomous Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 23.2 (2022), pp. 740–759. DOI: 10.1109/TITS.2020.3024655 (cit. on p. 11).

[43] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Pérez. "Deep Reinforcement Learning for Autonomous Driving: A Survey". In: *IEEE Transactions on Intelligent Transportation Systems* 23.6 (2022), pp. 4909–4926. DOI: 10.1109/TITS.2021.3054625 (cit. on pp. 11, 23).

[44] P. Karle, M. Geisslinger, J. Betz, and M. Lienkamp. "Scenario Understanding and Motion Prediction for Autonomous Vehicles—Review and Comparison". In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 16962–16982. DOI: 10.1109/TITS.2022.3156011 (cit. on p. 11).

[45] R. S. Rice. "Measuring Car-Driver Interaction with the g-g Diagram". In: *SAE Technical Paper Series*. SAE Technical Paper 730018. 1973. DOI: 10.4271/730018 (cit. on p. 12).

[46] D. L. Brayshaw and M. F. Harrison. "A quasi steady state approach to race car lap simulation in order to understand the effects of racing line and centre of gravity location". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 219.6 (2005), pp. 725–739. DOI: 10.1243/095440705X11211 (cit. on p. 12).

[47] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann. "Minimum curvature trajectory planning and control for an autonomous race car". In: *Vehicle System Dynamics* 58.10 (2020), pp. 1497–1527. DOI: 10.1080/00423114.2019.1631455 (cit. on pp. 12, 16, 28, 79).

[48] E. Velenis and P. Tsiotras. "Minimum-Time Travel for a Vehicle with Acceleration Limits: Theoretical Analysis and Receding-Horizon Implementation". In: *Journal of Optimization Theory and Applications* 138.2 (2008), pp. 275–296. DOI: 10.1007/s10957-008-9381-7 (cit. on p. 12).

[49] J. Subosits and J. C. Gerdes. "Autonomous vehicle control for emergency maneuvers: The effect of topography". In: *2015 American Control Conference (ACC)*. 2015, pp. 1405–1410. DOI: 10.1109/ACC.2015.7170930 (cit. on p. 12).

[50] S. Lovato and M. Massaro. "Three-dimensional fixed-trajectory approaches to the minimum-lap time of road vehicles". In: *Vehicle System Dynamics* 60.11 (2022), pp. 3650–3667. DOI: 10.1080/00423114.2021.1969024 (cit. on p. 12).

[51] N. R. Kapania, J. Subosits, and J. Christian Gerdes. "A Sequential Two-Step Algorithm for Fast Generation of Vehicle Racing Trajectories". In: *Journal of Dynamic Systems, Measurement, and Control* 138.9, Article 091005 (2016), pp. 1–10. DOI: 10.1115/1.4033311 (cit. on p. 12).

[52] N. Dal Bianco, E. Bertolazzi, F. Biral, and M. Massaro. "Comparison of direct and indirect methods for minimum lap time optimal control problems". In: *Vehicle System Dynamics* 57.5 (2019), pp. 665–696. DOI: 10.1080/00423114.2018.1480048 (cit. on p. 12).

[53] A. Rucco, G. Notarstefano, and J. Hauser. "An Efficient Minimum-Time Trajectory Generation Strategy for Two-Track Car Vehicles". In: *IEEE Transactions on Control Systems Technology* 23.4 (2015), pp. 1505–1519. DOI: 10.1109/TCST.2014.2377777 (cit. on p. 12).

[54] D. Casanova. "On Minimum Time Vehicle Manoeuvring: The Theoretical Optimal Lap". PhD thesis. Cranfield University, 2000. URL: https://dspace.lib.cranfield.ac.uk/items/c5c32bad-c3b1-4c06-b358-3053aba0699d (cit. on p. 12).

[55] F. Christ, A. Wischnewski, A. Heilmeier, and B. Lohmann. "Time-optimal trajectory planning for a race car considering variable tyre-road friction coefficients". In: *Vehicle System Dynamics* 59.4 (2021), pp. 588–612. DOI: 10.1080/00423114.2019.1704804 (cit. on pp. 12, 72).

[56] T. Herrmann, F. Christ, J. Betz, and M. Lienkamp. "Energy Management Strategy for an Autonomous Electric Racecar using Optimal Control". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 720–725. DOI: 10.1109/ITSC.2019.8917154 (cit. on p. 12).

[57] M. Veneri and M. Massaro. "A free-trajectory quasi-steady-state optimal-control method for minimum lap-time of race vehicles". In: *Vehicle System Dynamics* 58.6 (2020), pp. 933–954. DOI: 10.1080/00423114.2019.1608364 (cit. on pp. 12, 49, 96).

[58] G. Perantoni and D. J. N. Limebeer. "Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 1: Track Modeling and Identification". In: *Journal of Dynamic Systems, Measurement, and Control* 137.5, Article 051018 (2015), pp. 1–11. DOI: 10.1115/1.4028253 (cit. on pp. 13, 15, 33, 38, 41).

[59] D. J. N. Limebeer and G. Perantoni. "Optimal Control of a Formula One Car on a Three-Dimensional Track—Part 2: Optimal Control". In: *Journal of Dynamic Systems, Measurement, and Control* 137.5, Article 051019 (2015), pp. 1–13. DOI: 10.1115/1.4029466 (cit. on p. 13).

[60] S. Lovato and M. Massaro. "A three-dimensional free-trajectory quasi-steady-state optimal-control method for minimum-lap-time of race vehicles". In: *Vehicle System Dynamics* 60.5 (2022), pp. 1512–1530. DOI: 10.1080/00423114.2021.1878242 (cit. on pp. 13, 15, 28, 43, 44, 46, 47, 49).

[61] S. Lovato, M. Massaro, and D. J. N. Limebeer. "Curved-ribbon-based track modelling for minimum lap-time optimisation". In: *Meccanica* 56.8 (2021), pp. 2139–2152. DOI: 10.1007/s11012-021-01387-3 (cit. on pp. 13, 43, 143).

[62] L. B. Cremean et al. "Alice: An information–rich autonomous vehicle for high–speed desert navigation". In: *Journal of Field Robotics* 23.9 (2006), pp. 777–810. DOI: 10.1002/rob.20135 (cit. on p. 14).

[63] J. Ziegler, P. Bender, T. Dang, and C. Stiller. "Trajectory planning for Bertha — A local, continuous method". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 450–457. DOI: 10.1109/IVS.2014.6856581 (cit. on p. 14).

[64] J. Ziegler et al. "Making Bertha Drive—An Autonomous Journey on a Historic Route". In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), pp. 8–20. DOI: 10.1109/MITS.2014.2306552 (cit. on p. 14).

[65] M. G. Plessen, P. F. Lima, J. Mårtensson, A. Bemporad, and B. Wahlberg. "Trajectory planning under vehicle dimension constraints using sequential linear programming". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–6. DOI: 10.1109/ITSC.2017.8317665 (cit. on p. 14).

[66] I. Gundlach and U. Konigorski. "Modellbasierte Online-Trajektorienplanung für zeitoptimale Rennlinien". In: *at - Automatisierungstechnik* 67.9 (2019), pp. 799–813. DOI: 10.1515/auto-2019-0032 (cit. on p. 14).

[67] P. Scheffe, T. M. Henneken, M. Kloock, and B. Alrifaee. "Sequential Convex Programming Methods for Real-Time Optimal Trajectory Planning in Autonomous Vehicle Racing". In: *IEEE Transactions on Intelligent Vehicles* 8.1 (2023), pp. 661–672. DOI: 10.1109/TIV.2022.3168130 (cit. on p. 14).

[68] J. K. Subosits and J. C. Gerdes. "From the Racetrack to the Road: Real-Time Trajectory Replanning for Autonomous Driving". In: *IEEE Transactions on Intelligent Vehicles* 4.2 (2019), pp. 309–320. DOI: 10.1109/TIV.2019.2904390 (cit. on p. 14).

[69] T. Herrmann, A. Wischnewski, L. Hermansdorfer, J. Betz, and M. Lienkamp. "Real-Time Adaptive Velocity Optimization for Autonomous Electric Cars at the Limits of Handling". In: *IEEE Transactions on Intelligent Vehicles* 6.4 (2021), pp. 665–677. DOI: 10.1109/TIV.2020.3047858 (cit. on pp. 14, 15).

[70] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp. "Multilayer Graph-Based Trajectory Planning for Race Vehicles in Dynamic Scenarios". In: *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. 2019, pp. 3149–3154. DOI: 10.1109/ITSC.2019.8917032 (cit. on pp. 15, 16, 28, 53, 54, 79).

[71] M. Rowold, L. Ögretmen, U. Kasolowsky, and B. Lohmann. "Online Time-Optimal Trajectory Planning on Three-Dimensional Race Tracks". In: *2023 IEEE Intelligent Vehicles Symposium (IV)*. 2023, pp. 1–8. DOI: 10.1109/IV55152.2023.10186701 (cit. on pp. 15, 48, 95, 96, 141).

[72] K. Kant and S. W. Zucker. "Toward Efficient Trajectory Planning: The Path-Velocity Decomposition". In: *The International Journal of Robotics Research* 5.3 (1986), pp. 72–89. DOI: 10.1177/027836498600500304 (cit. on p. 15).

[73] J. Ziegler and C. Stiller. "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2009, pp. 1879–1884. DOI: 10.1109/IROS.2009.5354448 (cit. on p. 16).

[74] M. McNaughton, C. Urmson, J. M. Dolan, and J.-W. Lee. "Motion planning for autonomous driving with a conformal spatiotemporal lattice". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 4889–4895. DOI: 10.1109/ICRA.2011.5980223 (cit. on pp. 16, 28, 53, 54, 56–58, 66, 73).

[75] A. Kelly and B. Nagy. "Reactive Nonholonomic Trajectory Generation via Parametric Optimal Control". In: *The International Journal of Robotics Research* 22.7-8 (2003), pp. 583–601. DOI: 10.1177/02783649030227008 (cit. on p. 16).

[76] P. Scheffe, M. V. A. Pedrosa, K. Flaßkamp, and B. Alrifaee. "Receding Horizon Control Using Graph Search for Multi-Agent Trajectory Planning". In: *IEEE Transactions on Control Systems Technology* 31.3 (2023), pp. 1092–1105. DOI: 10.1109/TCST.2022.3214718 (cit. on p. 17).

[77] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/BF01386390 (cit. on p. 17).

[78] P. Hart, N. J. Nilsson, and B. Raphael. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths". In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136 (cit. on p. 17).

[79] R. Ebendt and R. Drechsler. "Weighted A* search – unifying view and application". In: *Artificial Intelligence* 173.14 (2009), pp. 1310–1342. DOI: 10.1016/j.artint.2009.06.004 (cit. on p. 17).

[80] A. Stentz. "Optimal and efficient path planning for partially-known environments". In: *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*. 1994, pp. 3310–3317. DOI: 10.1109/ROBOT.1994.351061 (cit. on p. 17).

[81] S. M. LaValle. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*. Tech. rep. 98-11. Ames, IA, USA: Department of Computer Science, Iowa State University, 1998. URL: https://lavalle.pl/papers/Lav98c.pdf (cit. on p. 17).

[82] S. M. LaValle and J. J. Kuffner. "Randomized Kinodynamic Planning". In: *The International Journal of Robotics Research* 20.5 (2001), pp. 378–400. DOI: 10.1177/02783640122067453 (cit. on p. 17).

[83] S. Karaman and E. Frazzoli. "Incremental Sampling-based Algorithms for Optimal Motion Planning". In: *Robotics: Science and Systems VI*. Ed. by Y. Matsuoka, H. Durrant-Whyte, and J. Neira. Cambridge, MA, USA: The MIT Press, 2011, pp. 267–274. DOI: 10.7551/mitpress/9123.003.0038 (cit. on p. 17).

[84] S. Karaman and E. 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 (cit. on p. 18).

[85] M. Otte and E. Frazzoli. "RRT$^{\mathrm{X}}$: Real-Time Motion Planning/Replanning for Environments with Unpredictable Obstacles". In: *Algorithmic Foundations of Robotics XI*. Ed. by H. L. Akin, N. M. Amato, V. Isler, and A. F. van der Stappen. Vol. 107. Springer Tracts in Advanced Robotics. Cham: Springer, 2015, pp. 461–478. DOI: 10.1007/978-3-319-16595-0_27 (cit. on p. 18).

[86] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How. "Motion Planning in Complex Environments Using Closed-loop Prediction". In: *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2008, pp. 1–22. DOI: 10.2514/6.2008-7166 (cit. on p. 18).

[87] R. Kala and K. Warwick. "Planning of multiple autonomous vehicles using RRT". In: *2011 IEEE 10th International Conference on Cybernetic Intelligent Systems (CIS)*. 2011, pp. 20–25. DOI: 10.1109/CIS.2011.6169129 (cit. on p. 18).

[88] J.-H. Ryu, D. Ogay, S. Bulavintsev, H. Kim, and J.-S. Park. "Development and Experiences of an Autonomous Vehicle for High-Speed Navigation and Obstacle Avoidance". In: *Frontiers of Intelligent Autonomous Systems*. Ed. by S. Lee, K.-J. Yoon, and J. Lee. Vol. 466. Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2013, pp. 105–116. DOI: 10.1007/978-3-642-35485-4_8 (cit. on p. 18).

[89] S. Feraco, S. Luciani, A. Bonfitto, N. Amati, and A. Tonoli. "A local trajectory planning and control method for autonomous vehicles based on the RRT algorithm". In: *2020 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE)*. 2020, pp. 1–6. DOI: 10.23919/AEITAUTOMOTIVE50086.2020.9307439 (cit. on p. 18).

[90] J. h. Jeon, S. Karaman, and E. Frazzoli. "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*". In: *2011 50th IEEE Conference on Decision and Control and European Control Conference*. 2011, pp. 3276–3282. DOI: 10.1109/CDC.2011.6161521 (cit. on p. 18).

[91] J. h. Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma. "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 (cit. on p. 18).

[92] M. Werling, J. Ziegler, S. Kammel, and S. Thrun. "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame". In: *2010 IEEE International Conference on Robotics and Automation*. 2010, pp. 987–993. DOI: 10.1109/ROBOT.2010.5509799 (cit. on pp. 19, 24, 26, 28, 53, 83, 84, 86, 89).

[93] A. Takahashi, T. Hongo, Y. Ninomiya, and G. Sugimoto. "Local Path Planning And Motion Control For Agv In Positioning". In: *Proceedings. IEEE/RSJ International Workshop on Intelligent Robots and Systems '.*

(IROS '89) 'The Autonomous Mobile Robots and Its Applications. 1989, pp. 392–397. DOI: 10.1109/IROS.1989.637936 (cit. on p. 19).

[94] M. Werling, S. Kammel, J. Ziegler, and L. Gröll. "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds". In: *The International Journal of Robotics Research* 31.3 (2012), pp. 346–359. DOI: 10.1177/0278364911423042 (cit. on p. 19).

[95] C. Rathgeber, F. Winkler, and S. Müller. "Kollisionsfreie Längs- und Quertrajektorienplanung unter Berücksichtigung fahrzeugspezifischer Potenziale". In: *at - Automatisierungstechnik* 64.1 (2016), pp. 61–76. DOI: 10.1515/auto-2015-0052 (cit. on p. 19).

[96] A. Raji, A. Liniger, A. Giove, A. Toschi, N. Musiu, D. Morra, M. Verucchi, D. Caporale, and M. Bertogna. "Motion Planning and Control for Multi Vehicle Autonomous Racing at High Speeds". In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*. 2022, pp. 2775–2782. DOI: 10.1109/ITSC55140.2022.9922239 (cit. on pp. 20, 83, 84, 86).

[97] A. Liniger, A. Domahidi, and M. Morari. "Optimization-based autonomous racing of 1:43 scale RC cars". In: *Optimal Control Applications and Methods* 36.5 (2015), pp. 628–647. DOI: 10.1002/oca.2123 (cit. on p. 20).

[98] M. O'Kelly, H. Zheng, A. Jain, J. Auckley, K. Luong, and R. Mangharam. "TUNERCAR: A Superoptimization Toolchain for Autonomous Racing". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5356–5362. DOI: 10.1109/ICRA40945.2020.9197080 (cit. on p. 20).

[99] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. "Path Planning for Autonomous Vehicles in Unknown Semi-structured Environments". In: *The International Journal of Robotics Research* 29.5 (2010), pp. 485–501. DOI: 10.1177/0278364909359210 (cit. on p. 20).

[100] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha. "A real-time motion planner with trajectory optimization for autonomous vehicles". In: *2012 IEEE International Conference on Robotics and Automation*. 2012, pp. 2061–2067. DOI: 10.1109/ICRA.2012.6225063 (cit. on p. 20).

[101] W. Lim, S. Lee, M. Sunwoo, and K. Jo. "Hierarchical Trajectory Planning of an Autonomous Car Based on the Integration of a Sampling and an Optimization Method". In: *IEEE Transactions on Intelligent Transportation*

Systems 19.2 (2018), pp. 613–626. DOI: 10.1109/TITS.2017.2756099 (cit. on p. 20).

[102] Y. Meng, Y. Wu, Q. Gu, and L. Liu. "A Decoupled Trajectory Planning Framework Based on the Integration of Lattice Searching and Convex Optimization". In: *IEEE Access* 7 (2019), pp. 130530–130551. DOI: 10.1109/ACCESS.2019.2940271 (cit. on p. 20).

[103] L. Xin, Y. Kong, S. E. Li, J. Chen, Y. Guan, M. Tomizuka, and B. Cheng. "Enable faster and smoother spatio-temporal trajectory planning for autonomous vehicles in constrained dynamic environment". In: *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering* 235.4 (2021), pp. 1101–1112. DOI: 10.1177/0954407020906627 (cit. on p. 20).

[104] G. Jank, M. Rowold, and B. Lohmann. "Hierarchical Time-Optimal Planning for Multi-Vehicle Racing". In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 2023, pp. 2064–2069. DOI: 10.1109/ITSC57777.2023.10422566 (cit. on p. 20).

[105] P. Trautman and A. Krause. "Unfreezing the robot: Navigation in dense, interacting crowds". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2010, pp. 797–803. DOI: 10.1109/IROS.2010.5654369 (cit. on p. 21).

[106] W. Schwarting, J. Alonso-Mora, and D. Rus. "Planning and Decision-Making for Autonomous Vehicles". In: *Annual Review of Control, Robotics, and Autonomous Systems* 1 (2018), pp. 187–210. DOI: 10.1146/annurev-control-060117-105157 (cit. on p. 21).

[107] S. Hagedorn, M. Hallgarten, M. Stoll, and A. P. Condurache. "The Integration of Prediction and Planning in Deep Learning Automated Driving Systems: A Review". In: *IEEE Transactions on Intelligent Vehicles* 10.5 (2025), pp. 3626–3643. DOI: 10.1109/TIV.2024.3459071 (cit. on p. 21).

[108] M. Rowold, A. Langmann, B. Lohmann, and J. Betz. "Open-Loop and Feedback Nash Trajectories for Competitive Racing with iLQGames". In: *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*. 2024, pp. 1827–1834. DOI: 10.1109/ITSC58415.2024.10920189 (cit. on p. 21).

[109] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan. "Hierarchical Game-Theoretic Planning for Autonomous Vehicles". In: *2019 International Conference on Robotics and Automation*

(ICRA). 2019, pp. 9590–9596. DOI: 10.1109/ICRA.2019.8794007 (cit. on p. 22).

[110] A. Liniger and J. Lygeros. "A Noncooperative Game Approach to Autonomous Racing". In: *IEEE Transactions on Control Systems Technology* 28.3 (2020), pp. 884–897. DOI: 10.1109/TCST.2019.2895282 (cit. on p. 22).

[111] K. Ji, S. Bae, N. Li, and K. Han. "Competition-Aware Decision-Making Approach for Mobile Robots in Racing Scenarios". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 878–884. DOI: 10.1109/IV55156.2024.10588596 (cit. on p. 22).

[112] N. Evestedt, E. Ward, J. Folkesson, and D. Axehill. "Interaction aware trajectory planning for merge scenarios in congested traffic situations". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 465–472. DOI: 10.1109/ITSC.2016.7795596 (cit. on pp. 22, 121).

[113] Y. Wang, Y. Lin, and M. Althoff. "Interaction-Aware Trajectory Repair in Compliance with Formalized Traffic Rules". In: *2024 IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*. 2024, pp. 1850–1857. DOI: 10.1109/ITSC58415.2024.10920252 (cit. on pp. 22, 121).

[114] Z. Wang, R. Spica, and M. Schwager. "Game Theoretic Motion Planning for Multi-robot Racing". In: *Distributed Autonomous Robotic Systems*. Ed. by N. Correll, M. Schwager, and M. Otte. Vol. 9. Springer Proceedings in Advanced Robotics. Cham: Springer International Publishing, 2019, pp. 225–238. DOI: 10.1007/978-3-030-05816-6_16 (cit. on p. 22).

[115] G. Notomista, M. Wang, M. Schwager, and M. Egerstedt. "Enhancing Game-Theoretic Autonomous Car Racing Using Control Barrier Functions". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5393–5399. DOI: 10.1109/ICRA40945.2020.9196757 (cit. on p. 22).

[116] M. Wang, Z. Wang, J. Talbot, J. C. Gerdes, and M. Schwager. "Game-Theoretic Planning for Self-Driving Cars in Multivehicle Competitive Scenarios". In: *IEEE Transactions on Robotics* 37.4 (2021), pp. 1313–1325. DOI: 10.1109/TRO.2020.3047521 (cit. on p. 22).

[117] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. "Reaching the limit in autonomous racing: Optimal control versus reinforcement learning". In: *Science Robotics* 8.82, Article eadg1462 (2023), pp. 1–14. DOI: 10.1126/scirobotics.adg1462 (cit. on p. 23).

[118] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. "Champion-level drone racing using deep reinforcement learning". In: *Nature* 620.7976 (2023), pp. 982–987. DOI: 10.1038/s41586-023-06419-4 (cit. on p. 23).

[119] B. Evans, H. A. Engelbrecht, and H. W. Jordaan. "Reward Signal Design for Autonomous Racing". In: *2021 20th International Conference on Advanced Robotics (ICAR)*. 2021, pp. 455–460. DOI: 10.1109/ICAR53236.2021.9659438 (cit. on p. 23).

[120] A. Abouelazm, J. Michel, and J. M. Zöllner. "A Review of Reward Functions for Reinforcement Learning in the context of Autonomous Driving". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 156–163. DOI: 10.1109/IV55156.2024.10588385 (cit. on p. 23).

[121] S. Gronauer and K. Diepold. "Multi-agent deep reinforcement learning: a survey". In: *Artificial Intelligence Review* 55.2 (2022), pp. 895–943. DOI: 10.1007/s10462-021-09996-w (cit. on p. 23).

[122] M. Kaushik, V. Prasad, K. M. Krishna, and B. Ravindran. "Overtaking Maneuvers in Simulated Highway Driving using Deep Reinforcement Learning". In: *2018 IEEE Intelligent Vehicles Symposium (IV)*. 2018, pp. 1885–1890. DOI: 10.1109/IVS.2018.8500718 (cit. on p. 23).

[123] Y. Tang. "Towards Learning Multi-Agent Negotiations via Self-Play". In: *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*. 2019, pp. 2427–2435. DOI: 10.1109/ICCVW.2019.00297 (cit. on p. 23).

[124] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev. "Driving in Dense Traffic with Model-Free Reinforcement Learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 5385–5392. DOI: 10.1109/ICRA40945.2020.9197132 (cit. on p. 23).

[125] M. Bouton, A. Nakhaei, D. Isele, K. Fujimura, and M. J. Kochenderfer. "Reinforcement Learning with Iterative Reasoning for Merging in Dense Traffic". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294338 (cit. on p. 23).

[126] P. Cai, X. Mei, L. Tai, Y. Sun, and M. Liu. "High-Speed Autonomous Drifting With Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 1247–1254. DOI: 10.1109/LRA.2020.2967299 (cit. on p. 23).

[127] E. Chisari, A. Liniger, A. Rupenyan, L. van Gool, and J. Lygeros. "Learning from Simulation, Racing in Reality". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 8046–8052. DOI: 10.1109/ICRA48506.2021.9562079 (cit. on p. 23).

[128] B. D. Evans, H. A. Engelbrecht, and H. W. Jordaan. "High-Speed Autonomous Racing Using Trajectory-Aided Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 8.9 (2023), pp. 5353–5359. DOI: 10.1109/LRA.2023.3295252 (cit. on p. 23).

[129] M. Hell, G. Hajgató, Á. Bogár-Németh, and G. Bári. "A LiDAR-based approach to autonomous racing with model-free reinforcement learning". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 258–263. DOI: 10.1109/IV55156.2024.10588613 (cit. on p. 23).

[130] E. Perot, M. Jaritz, M. Toromanoff, and R. de Charette. "End-to-End Driving in a Realistic Racing Game with Deep Reinforcement Learning". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2017, pp. 474–475. DOI: 10.1109/CVPRW.2017.64 (cit. on p. 23).

[131] M. Jaritz, R. de Charette, M. Toromanoff, E. Perot, and F. Nashashibi. "End-to-End Race Driving with Deep Reinforcement Learning". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 2070–2075. DOI: 10.1109/ICRA.2018.8460934 (cit. on p. 23).

[132] K. Güçkıran and B. Bolat. "Autonomous Car Racing in Simulation Environment Using Deep Reinforcement Learning". In: *2019 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2019, pp. 1–6. DOI: 10.1109/ASYU48272.2019.8946332 (cit. on p. 23).

[133] J. Niu, Y. Hu, B. Jin, Y. Han, and X. Li. "Two-Stage Safe Reinforcement Learning for High-Speed Autonomous Racing". In: *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2020, pp. 3934–3941. DOI: 10.1109/SMC42975.2020.9283053 (cit. on p. 23).

[134] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr. "Super-Human Performance in Gran Turismo Sport Using Deep Reinforcement Learning". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4257–4264. DOI: 10.1109/LRA.2021.3064284 (cit. on p. 23).

[135] Y. Song, H. Lin, E. Kaufmann, P. Dürr, and D. Scaramuzza. "Autonomous Overtaking in Gran Turismo Sport Using Curriculum Reinforcement Learning". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 9403–9409. DOI: 10.1109/ICRA48506.2021.9561049 (cit. on p. 23).

[136] P. R. Wurman et al. "Outracing champion Gran Turismo drivers with deep reinforcement learning". In: *Nature* 602.7896 (2022), pp. 223–228. DOI: 10.1038/s41586-021-04357-7 (cit. on p. 23).

[137] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker. "High-level Decision Making for Safe and Reasonable Autonomous Lane Changing using Reinforcement Learning". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2156–2162. DOI: 10.1109/ITSC.2018.8569448 (cit. on p. 24).

[138] M. Huegle, G. Kalweit, B. Mirchevska, M. Werling, and J. Boedecker. "Dynamic Input for Deep Reinforcement Learning in Autonomous Driving". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 7566–7573. DOI: 10.1109/IROS40897.2019.8968560 (cit. on p. 24).

[139] J. Wang, Q. Zhang, D. Zhao, and Y. Chen. "Lane Change Decision-making through Deep Reinforcement Learning with Rule-based Constraints". In: *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019, pp. 1–6. DOI: 10.1109/IJCNN.2019.8852110 (cit. on p. 24).

[140] D. Loiacono, A. Prete, P. L. Lanzi, and L. Cardamone. "Learning to overtake in TORCS using simple reinforcement learning". In: *IEEE Congress on Evolutionary Computation*. 2010, pp. 1–8. DOI: 10.1109/CEC.2010.5586191 (cit. on p. 24).

[141] R. Trauth, A. Hobmeier, and J. Betz. "A Reinforcement Learning-Boosted Motion Planning Framework: Comprehensive Generalization Performance in Autonomous Driving". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 2413–2420. DOI: 10.1109/IV55156.2024.10588750 (cit. on pp. 24, 142).

[142] B. Mirchevska, M. Hügle, G. Kalweit, M. Werling, and J. Boedecker. "Amortized Q-learning with Model-based Action Proposals for Autonomous Driving on Highways". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1028–1035. DOI: 10.1109/ICRA48506.2021.9560777 (cit. on p. 25).

[143] B. Mirchevska, M. Werling, and J. Boedecker. "Optimizing trajectories for highway driving with offline reinforcement learning". In: *Frontiers in Future Transportation* 4, Article 1076439 (2023), pp. 1–14. DOI: 10.3389/ffutr.2023.1076439 (cit. on pp. 25, 121).

[144] B. Zarrouki, V. Klös, N. Heppner, S. Schwan, R. Ritschel, and R. Voßwinkel. "Weights-varying MPC for Autonomous Vehicle Guidance: a Deep Reinforcement Learning Approach". In: *2021 European Control Conference (ECC)*. 2021, pp. 119–125. DOI: 10.23919/ECC54610.2021.9655042 (cit. on pp. 25, 142).

[145] B. Brito, A. Agarwal, and J. Alonso-Mora. "Learning Interaction-Aware Guidance for Trajectory Optimization in Dense Traffic Scenarios". In: *IEEE Transactions on Intelligent Transportation Systems* 23.10 (2022), pp. 18808–18821. DOI: 10.1109/TITS.2022.3160936 (cit. on p. 25).

[146] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. 1st ed. Boca Raton, FL, USA: CRC Press, 1994. ISBN: 978-1-315-13637-0. DOI: 10.1201/9781315136370 (cit. on pp. 33, 34).

[147] D. Schramm, M. Hiller, and R. Bardini. *Vehicle Dynamics: Modeling and Simulation*. 2nd ed. Berlin, Heidelberg: Springer, 2018. ISBN: 978-3-662-54482-2. DOI: 10.1007/978-3-662-54483-9 (cit. on p. 42).

[148] H. B. Pacejka. *Tire and Vehicle Dynamics*. 3rd ed. Oxford: Butterworth-Heinemann, 2012. ISBN: 978-0-08-097016-5. DOI: 10.1016/C2010-0-68548-8 (cit. on p. 42).

[149] E. Héry, S. Masi, P. Xu, and P. Bonnifait. "Map-based curvilinear coordinates for autonomous vehicles". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. 2017, pp. 1–7. DOI: 10.1109/ITSC.2017.8317775 (cit. on p. 51).

[150] P. Bender, J. Ziegler, and C. Stiller. "Lanelets: Efficient map representation for autonomous driving". In: *2014 IEEE Intelligent Vehicles Symposium Proceedings*. 2014, pp. 420–425. DOI: 10.1109/IVS.2014.6856487 (cit. on p. 51).

[151] L. Ögretmen, M. Rowold, T. Betz, A. Langmann, and B. Lohmann. "A Hybrid Trajectory Planning Approach for Autonomous Rule-Compliant Multi-Vehicle Oval Racing". In: *SAE International Journal of Connected and Automated Vehicles* 7.1 (2024), pp. 95–112. DOI: 10.4271/12-07-01-0007 (cit. on pp. 52, 54, 67, 69).

[152] L. Ögretmen, M. Rowold, M. Ochsenius, and B. Lohmann. "Smooth Trajectory Planning at the Handling Limits for Oval Racing". In: *Actuators* 11.11, Article 318 (2022), pp. 1–17. DOI: 10.3390/act11110318 (cit. on p. 54).

[153] M. Rowold, L. Ögretmen, T. Kerbl, and B. Lohmann. "Efficient Spatiotemporal Graph Search for Local Trajectory Planning on Oval Race Tracks". In: *Actuators* 11.11, Article 319 (2022), pp. 1–18. DOI: 10.3390/act11110319 (cit. on pp. 66, 69).

[154] A. Wischnewski, T. Stahl, J. Betz, and B. Lohmann. "Vehicle Dynamics State Estimation and Localization for High Performance Race Cars". In: *IFAC-PapersOnLine* 52.8 (2019), pp. 154–161. DOI: 10.1016/j.ifacol.2019.08.064 (cit. on p. 72).

[155] F. Sauerbeck, S. Huch, F. Fent, P. Karle, D. Kulmer, and J. Betz. "Learn to See Fast: Lessons Learned From Autonomous Racing on How to Develop Perception Systems". In: *IEEE Access* 11 (2023), pp. 44034–44050. DOI: 10.1109/ACCESS.2023.3272750 (cit. on p. 72).

[156] P. Karle, F. Fent, S. Huch, F. Sauerbeck, and M. Lienkamp. "Multi-Modal Sensor Fusion and Object Tracking for Autonomous Racing". In: *IEEE Transactions on Intelligent Vehicles* 8.7 (2023), pp. 3871–3883. DOI: 10.1109/TIV.2023.3271624 (cit. on p. 72).

[157] P. Karle, F. Török, M. Geisslinger, and M. Lienkamp. "MixNet: Physics Constrained Deep Neural Motion Prediction for Autonomous Racing". In: *IEEE Access* 11 (2023), pp. 85914–85926. DOI: 10.1109/ACCESS.2023.3303841 (cit. on p. 72).

[158] A. Wischnewski, T. Herrmann, F. Werner, and B. Lohmann. "A Tube-MPC Approach to Autonomous Multi-Vehicle Racing on High-Speed Ovals". In: *IEEE Transactions on Intelligent Vehicles* 8.1 (2023), pp. 368–378. DOI: 10.1109/TIV.2022.3169986 (cit. on p. 73).

[159] L. Ögretmen, M. Rowold, A. Langmann, and B. Lohmann. "Sampling-Based Motion Planning with Online Racing Line Generation for Autonomous Driving on Three-Dimensional Race Tracks". In: *2024 IEEE Intelligent Vehicles Symposium (IV)*. 2024, pp. 811–818. DOI: 10.1109/IV55156.2024.10588726 (cit. on p. 84).

[160] L. Ögretmen, M. Chen, P. Pitschi, and B. Lohmann. "Trajectory Planning Using Reinforcement Learning for Interactive Overtaking Maneuvers in Autonomous Racing Scenarios". In: *2024 IEEE 27th International*

Conference on Intelligent Transportation Systems (ITSC). 2024, pp. 1082–1089. DOI: 10.1109/ITSC58415.2024.10919549 (cit. on p. 105).

[161] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd ed. Cambridge, MA, USA: The MIT Press, 2018. ISBN: 978-0-262-03924-6 (cit. on pp. 106, 109, 111, 112).

[162] C. J. C. H. Watkins and P. Dayan. "Q-learning". In: *Machine Learning* 8.3-4 (1992), pp. 279–292. DOI: 10.1007/BF00992698 (cit. on p. 110).

[163] B. Jang, M. Kim, G. Harerimana, and J. W. Kim. "Q-Learning Algorithms: A Comprehensive Classification and Applications". In: *IEEE Access* 7 (2019), pp. 133653–133667. DOI: 10.1109/ACCESS.2019.2941229 (cit. on p. 110).

[164] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3-4 (1992), pp. 229–256. DOI: 10.1007/BF00992696 (cit. on p. 111).

[165] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2015. arXiv: 1506.02438 [cs.LG] (cit. on p. 113).

[166] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. 2016, pp. 1928–1937. URL: https://proceedings.mlr.press/v48/mniha16.html (cit. on p. 113).

[167] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG] (cit. on pp. 114, 124).

[168] J. Randløv and P. Alstrøm. "Learning to Drive a Bicycle Using Reinforcement Learning and Shaping". In: *Proceedings of the Fifteenth International Conference on Machine Learning*. 1998, pp. 463–471. URL: https://dl.acm.org/doi/10.5555/645527.757766 (cit. on p. 115).

[169] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. "Curriculum learning". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. 2009, pp. 41–48. DOI: 10.1145/1553374.1553380 (cit. on p. 115).

[170] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *Journal of Machine Learning Research* 21, Article 181 (2020), pp. 1–50. URL: https://www.jmlr.org/papers/v21/20-212.html (cit. on p. 115).

[171] J. Sola and J. Sevilla. "Importance of input data normalization for the application of neural networks to complex industrial problems". In: *IEEE Transactions on Nuclear Science* 44.3 (1997), pp. 1464–1468. DOI: 10.1109/23.589532 (cit. on p. 115).

[172] Stable Baselines3. *Reinforcement Learning Tips and Tricks*. 2025. URL: https://stable-baselines3.readthedocs.io/en/master/guide/rl_tips.html (visited on 03/01/2025) (cit. on p. 115).

[173] M. Breton, A. Alj, and A. Haurie. "Sequential Stackelberg equilibria in two-person games". In: *Journal of Optimization Theory and Applications* 59.1 (1988), pp. 71–97. DOI: 10.1007/BF00939867 (cit. on p. 121).

[174] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning*. 2015, pp. 1889–1897. URL: https://proceedings.mlr.press/v37/schulman15.html (cit. on p. 124).

[175] M. Towers et al. *Gymnasium: A Standard Interface for Reinforcement Learning Environments*. Zenodo. 2025. DOI: 10.5281/zenodo.8127025 (cit. on p. 126).

[176] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann. "Stable-Baselines3: Reliable Reinforcement Learning Implementations". In: *Journal of Machine Learning Research* 22, Article 268 (2021), pp. 1–8. URL: http://jmlr.org/papers/v22/20-1364.html (cit. on p. 126).

[177] O. Föllinger. *Optimale Regelung und Steuerung*. 3rd ed. Berlin, Boston: Oldenbourg Wissenschaftsverlag, 1994. ISBN: 978-3-486-23116-8. DOI: 10.1515/9783486787306 (cit. on p. 173).

[178] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos. *Optimal Control*. 3rd ed. Hoboken, NJ, USA: John Wiley & Sons, Ltd, 2012. ISBN: 978-0-470-63349-6. DOI: 10.1002/9781118122631 (cit. on p. 173).

[179] J. E. Marsden and A. Weinstein. *Calculus unlimited*. Menlo Park, CA, USA: Benjamin/Cummings Publishing Company, Inc., 1981. ISBN: 0-8053-6932-5 (cit. on p. 180).

Appendix A

Coordinate Transformations

This chapter contains the transformation from Curvilinear to Cartesian coordinates in Section A.1 and the inverse transformation in Section A.2.

A.1 Transformation from Curvilinear to Cartesian Coordinates

For the derivation of the transformation

$$\left[s, \dot{s}, \ddot{s}, d, \dot{d}, \ddot{d}\right](t) \rightarrow [x, y, z, V, w, \hat{a}_{\mathrm{x}}, \hat{a}_{\mathrm{y}}, \hat{a}_{\mathrm{z}}, \hat{\chi}, \hat{\kappa}](t), \tag{A.1}$$

the position ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}}(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}^{\top}$ of the point mass (or the origin of $\mathcal{V}$) is first expressed in the inertial frame $\mathcal{I}$, and its velocity ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{IV}}(t)$ and acceleration ${}^{\mathcal{I}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t)$ are determined by repeated differentiation. Both variables are then expressed by corresponding rotations in the desired velocity frame $\mathcal{V}$, resulting in the desired quantities ${}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}}(t) = \begin{bmatrix} V(t) & 0 & w(t) \end{bmatrix}^{\top}$ and ${}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}}(t) = \begin{bmatrix} \hat{a}_{\mathrm{x}}(t) & \hat{a}_{\mathrm{y}}(t) & \hat{a}_{\mathrm{z}}(t) \end{bmatrix}^{\top}$. Afterward, the angular velocity of $\mathcal{V}$ is used to determine the curvature $\hat{\kappa}(t)$ of the path traversed by the point mass projected onto the road plane. Finally, the singularities of the transformation are analyzed. For better readability, the arguments are neglected in the remainder of this chapter.

For the derivation, some physical quantities must be expressed as a function of the desired Curvilinear coordinates, which is particularly easy to do in the road frame $\mathcal{R}$. Since $\mathcal{R}$ and $\mathcal{V}$ share the same s-coordinate by definition, the distance between their origins is solely d in the direction of the normal vector of $\mathcal{R}$. Furthermore, $\mathcal{R}$ is defined such that the x-axis is tangential to the reference

line. Therefore, the velocity of $\mathcal{R}$ relative to the inertial frame only includes $\dot{s}$ in the direction of the tangential vector of $\mathcal{R}$. The following expressions can, therefore, be stated and must be integrated into the subsequent equations:

$$ {}^{\mathcal{R}}\boldsymbol{x}_{\mathcal{RV}} = \begin{bmatrix} 0 \\ d \\ 0 \end{bmatrix}, \qquad {}^{\mathcal{R}}\boldsymbol{v}_{\mathcal{IR}} = \begin{bmatrix} \dot{s} \\ 0 \\ 0 \end{bmatrix}. \tag{A.2} $$

Position

The first desired relationship is between the Curvilinear coordinates s, d and the position ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}}$ of the point mass. As introduced in (2.32), the position of $\mathcal{V}$ can be calculated by the reference line ${}^{\mathcal{I}}\boldsymbol{c}(s)$ evaluated at progress s and the displacement d in the normal direction ${}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s)$ of the road frame $\mathcal{R}$:

$$ {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}} = {}^{\mathcal{I}}\boldsymbol{c}(s) + {}^{\mathcal{I}}\boldsymbol{e}_{\mathrm{y},\mathcal{R}}(s) \cdot d. \tag{A.3} $$

Velocity and Relative Orientation

For the derivation of the velocity relations, a more general representation of (A.3) is beneficial. The position ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}}$ of the point mass can be described as the sum of the position of $\mathcal{R}$ and the position of $\mathcal{V}$ relative to $\mathcal{R}$:

$$ \begin{aligned} {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}} &= {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IR}} + {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{RV}} && \text{(A.4a)} \\ &= {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IR}} + \boldsymbol{R}_{\mathcal{IR}}\, {}^{\mathcal{R}}\boldsymbol{x}_{\mathcal{RV}}. && \text{(A.4b)} \end{aligned} $$

With that, the velocity of $\mathcal{V}$ expressed in the inertial frame $\mathcal{I}$ can be derived by the differentiation of (A.4b):

$$ \begin{aligned} {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{IV}} &= {}^{\mathcal{I}}\dot{\boldsymbol{x}}_{\mathcal{IV}} && \text{(A.5a)} \\ &= {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{IR}} + \dot{\boldsymbol{R}}_{\mathcal{IR}}\, {}^{\mathcal{R}}\boldsymbol{x}_{\mathcal{RV}} + \boldsymbol{R}_{\mathcal{IR}}\, {}^{\mathcal{R}}\dot{\boldsymbol{x}}_{\mathcal{RV}} && \text{(A.5b)} \\ &\overset{(2.17)}{=} \boldsymbol{R}_{\mathcal{IR}}\, {}^{\mathcal{R}}\boldsymbol{v}_{\mathcal{IR}} + \boldsymbol{R}_{\mathcal{IR}}\boldsymbol{S}\left({}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}}\right) {}^{\mathcal{R}}\boldsymbol{x}_{\mathcal{RV}} + \boldsymbol{R}_{\mathcal{IR}}\, {}^{\mathcal{R}}\dot{\boldsymbol{x}}_{\mathcal{RV}}. && \text{(A.5c)} \end{aligned} $$

To express the velocity in $\mathcal{V}$, a corresponding rotation is needed:

$$
\begin{aligned}
{}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}} &= \boldsymbol{R}_{\mathcal{RV}}^{\top}\boldsymbol{R}_{\mathcal{IR}}^{\top}\,{}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{IV}} && \text{(A.6a)}\\
&= \boldsymbol{R}_{\mathcal{RV}}^{\top}\,{}^{\mathcal{R}}\boldsymbol{v}_{\mathcal{IR}} + \boldsymbol{R}_{\mathcal{RV}}^{\top}\boldsymbol{S}\left({}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}}\right){}^{\mathcal{R}}\boldsymbol{x}_{\mathcal{RV}} + \boldsymbol{R}_{\mathcal{RV}}^{\top}\,{}^{\mathcal{R}}\dot{\boldsymbol{x}}_{\mathcal{RV}}. && \text{(A.6b)}
\end{aligned}
$$

Inserting (A.2), (2.33), and $\boldsymbol{R}_{\mathcal{RV}} = \boldsymbol{R}_{\mathrm{z}}(\hat{\chi})$ into (A.6b) gives the following expressions for the velocity:

$$
{}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}} = \begin{bmatrix} V \\ 0 \\ w \end{bmatrix} = \begin{bmatrix} \dot{d}\sin(\hat{\chi}) + \dot{s}\cos(\hat{\chi})\,(1-\Omega_{\mathrm{z}}d) \\ \dot{d}\cos(\hat{\chi}) - \dot{s}\sin(\hat{\chi})\,(1-\Omega_{\mathrm{z}}d) \\ \Omega_{\mathrm{x}}d\dot{s} \end{bmatrix}. \tag{A.7}
$$

Since the lateral velocity must be zero by definition of $\mathcal{V}$, an expression for $\hat{\chi}$ can be derived by setting the second line of (A.7) to zero:

$$
\hat{\chi} = \arctan\left(\frac{\dot{d}}{\dot{s}\,(1-\Omega_{\mathrm{z}}d)}\right). \tag{A.8}
$$

Further, using $\cos(\arctan(x)) = \frac{1}{\sqrt{x^2+1}}$ and $\sin(\arctan(x)) = \frac{x}{\sqrt{x^2+1}}$ yields:

$$
\cos(\hat{\chi}) = \frac{\dot{s}\,(1-\Omega_{\mathrm{z}}d)}{\sqrt{\dot{d}^2 + \dot{s}^2\,(1-\Omega_{\mathrm{z}}d)^2}}, \tag{A.9}
$$

$$
\sin(\hat{\chi}) = \frac{\dot{d}}{\sqrt{\dot{d}^2 + \dot{s}^2\,(1-\Omega_{\mathrm{z}}d)^2}}. \tag{A.10}
$$

Finally, by combining (A.9) and (A.10) together with the first line in (A.7), multiple expressions for the velocity V can be stated:

$$
\begin{aligned}
V &= \sqrt{\dot{d}^2 + \dot{s}^2\,(1-\Omega_{\mathrm{z}}d)^2} && \text{(A.11a)}\\
&= \frac{\dot{s}\,(1-\Omega_{\mathrm{z}}d)}{\cos(\hat{\chi})} = \frac{\dot{d}}{\sin(\hat{\chi})}. && \text{(A.11b)}
\end{aligned}
$$

Acceleration

A further differentiation of the velocity ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{IV}}$ results in the acceleration of $\mathcal{V}$ expressed in the inertial frame $\mathcal{I}$:

$$
\begin{aligned}
{}^{\mathcal{I}}\hat{\boldsymbol{a}}_{\mathcal{IV}} = {}^{\mathcal{I}}\dot{\boldsymbol{v}}_{\mathcal{IV}} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(\boldsymbol{R}_{\mathcal{IV}}\,{}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}}\right) && \text{(A.12a)}\\
&= \dot{\boldsymbol{R}}_{\mathcal{IV}}\,{}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}} + \boldsymbol{R}_{\mathcal{IV}}\,{}^{\mathcal{V}}\dot{\boldsymbol{v}}_{\mathcal{IV}} && \text{(A.12b)}\\
&\overset{(2.17)}{=} \boldsymbol{R}_{\mathcal{IV}}\boldsymbol{S}\left({}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IV}}\right){}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}} + \boldsymbol{R}_{\mathcal{IV}}\,{}^{\mathcal{V}}\dot{\boldsymbol{v}}_{\mathcal{IV}}. && \text{(A.12c)}
\end{aligned}
$$

Again, a rotation needs to be performed to express the acceleration in the velocity frame $\mathcal{V}$:

$$
\begin{aligned}
{}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}} &= \boldsymbol{R}_{\mathcal{IV}}^{\top}\,{}^{\mathcal{I}}\hat{\boldsymbol{a}}_{\mathcal{IV}} && \text{(A.13a)}\\
&= {}^{\mathcal{V}}\dot{\boldsymbol{v}}_{\mathcal{IV}} + \boldsymbol{S}\left({}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IV}}\right){}^{\mathcal{V}}\boldsymbol{v}_{\mathcal{IV}}. && \text{(A.13b)}
\end{aligned}
$$

Here, the angular velocity ${}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IV}} \in \mathbb{R}^3$ of the velocity frame $\mathcal{V}$ appears. It is composed of the angular velocity of the road frame $\mathcal{R}$ and the angular velocity of the velocity frame $\mathcal{V}$ relative to the road frame $\mathcal{R}$:

$$
\begin{aligned}
{}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IV}} &= {}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IR}} + {}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{RV}} && \text{(A.14a)}\\
&= \boldsymbol{R}_{\mathcal{RV}}^{\top}\,{}^{\mathcal{R}}\boldsymbol{\omega}_{\mathcal{IR}} + {}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{RV}}. && \text{(A.14b)}
\end{aligned}
$$

Since the orientation of $\mathcal{V}$ differs from the orientation of $\mathcal{R}$ only by a rotation around the z-axis by the angle $\hat{\chi}$, it follows that the relative angular velocity is ${}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{RV}} = \begin{bmatrix}0 & 0 & \dot{\hat{\chi}}\end{bmatrix}^{\top}$. Plugging this, (2.33), and $\boldsymbol{R}_{\mathcal{RV}} = \boldsymbol{R}_{\mathrm{z}}(\hat{\chi})$ into (A.14b) yields:

$$
{}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{IV}} = \begin{bmatrix}\hat{\omega}_{\mathrm{x}}\\ \hat{\omega}_{\mathrm{y}}\\ \hat{\omega}_{\mathrm{z}}\end{bmatrix} = \begin{bmatrix}\left(\cos\left(\hat{\chi}\right)\Omega_{\mathrm{x}} + \sin\left(\hat{\chi}\right)\Omega_{\mathrm{y}}\right)\cdot\dot{s}\\ \left(\cos\left(\hat{\chi}\right)\Omega_{\mathrm{y}} - \sin\left(\hat{\chi}\right)\Omega_{\mathrm{x}}\right)\cdot\dot{s}\\ \Omega_{\mathrm{z}}\dot{s} + \dot{\hat{\chi}}\end{bmatrix}. \tag{A.15}
$$

Evaluating (A.13b) now gives the acceleration as a function of the angular velocities $\hat{\omega}_{\mathrm{x}}$, $\hat{\omega}_{\mathrm{y}}$, and $\hat{\omega}_{\mathrm{z}}$ of $\mathcal{V}$, as well as the point mass velocities V and w and their derivatives:

$$
{}^{\mathcal{V}}\hat{\boldsymbol{a}}_{\mathcal{IV}} = \begin{bmatrix}\hat{a}_{\mathrm{x}}\\ \hat{a}_{\mathrm{y}}\\ \hat{a}_{\mathrm{z}}\end{bmatrix} = \begin{bmatrix}\dot{V} + \hat{\omega}_{\mathrm{y}}w\\ \hat{\omega}_{\mathrm{z}}V - \hat{\omega}_{\mathrm{x}}w\\ \dot{w} - \hat{\omega}_{\mathrm{y}}V\end{bmatrix}. \tag{A.16}
$$

The angular velocities $\hat{\omega}_\mathrm{x}$, $\hat{\omega}_\mathrm{y}$, and $\hat{\omega}_\mathrm{z}$ can be specified as a function of the Curvilinear coordinates using (A.15) together with (A.8), (A.9), and (A.10). The needed expressions for V and w are provided by (A.11a) and the third line of (A.7). The occurring derivatives $\dot{\hat{\chi}}$, $\dot{V}$, and $\dot{w}$ can be specified analytically by differentiation of (A.8), (A.11a), and the third line of (A.7), whereby $\dot{\Omega}_\mathrm{z} = \Omega_\mathrm{z}' \cdot \dot{s}$ is used according to the chain rule:

$$\dot{\hat{\chi}} = \frac{\ddot{d}\dot{s} - \dot{d}\ddot{s} + \Omega_\mathrm{z}\dot{s}\dot{d}^2 + \Omega_\mathrm{z}'d\dot{d}\dot{s}^2 + \Omega_\mathrm{z}d\dot{d}\ddot{s} - \Omega_\mathrm{z}d\ddot{d}\dot{s}}{\dot{d}^2 + \dot{s}^2\left(1 - \Omega_\mathrm{z}d\right)^2}, \tag{A.17}$$

$$\dot{V} = \frac{\ddot{d}\dot{d} + \ddot{s}\dot{s} - \Omega_\mathrm{z}\dot{s}^2\dot{d} - d\dot{s}^3\Omega_\mathrm{z}' - 2\Omega_\mathrm{z}d\ddot{s}\dot{s} + \Omega_\mathrm{z}d^2\dot{s}^3\Omega_\mathrm{z}' + \Omega_\mathrm{z}^2d\dot{s}^2\dot{d} + \Omega_\mathrm{z}^2d^2\ddot{s}\dot{s}}{\sqrt{\dot{d}^2 + \dot{s}^2\left(1 - \Omega_\mathrm{z}d\right)^2}}, \tag{A.18}$$

$$\dot{w} = \Omega_\mathrm{x}'d\dot{s}^2 + \Omega_\mathrm{x}\dot{d}\dot{s} + \Omega_\mathrm{x}d\ddot{s}. \tag{A.19}$$

Curvature

The last relationship to be derived is the curvature $\hat{\kappa}$ of the path traversed by the point mass projected on the road plane. This corresponds to the z-component of the spatial curvature vector ${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{I}\mathcal{V}}$ of $\mathcal{V}$, which represents the change in orientation with respect to the driven arc length s_x.[1] For this, the angular velocity ${}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{I}\mathcal{V}}$, which represents the change in orientation with respect to time t, is used and converted to the spatial curvature vector ${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{I}\mathcal{V}}$. To transform a change in orientation with respect to the time t to a change with respect to the driven arc length s_x, the chain rule can be applied:

$$\frac{\mathrm{d}}{\mathrm{d}s_\mathrm{x}} = \frac{\mathrm{d}t}{\mathrm{d}s_\mathrm{x}}\frac{\mathrm{d}}{\mathrm{d}t} = \frac{1}{\sqrt{V^2 + w^2}}\frac{\mathrm{d}}{\mathrm{d}t}. \tag{A.20}$$

Therefore, for the spatial curvature vector, it follows that

$${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{I}\mathcal{V}} = \frac{1}{\sqrt{V^2 + w^2}}\,{}^{\mathcal{V}}\boldsymbol{\omega}_{\mathcal{I}\mathcal{V}}. \tag{A.21}$$

The third component of ${}^{\mathcal{V}}\boldsymbol{\Omega}_{\mathcal{I}\mathcal{V}}$ corresponds to the curvature on the road plane $\hat{\kappa}$ and results by substituting (A.11a), (A.17), and the third lines of (A.7) as

[1] s_x is different from s, as s represents the progress along the reference line, which does not necessarily correspond to the driven path.

well as (A.15):

$$\hat{\kappa} = \frac{1}{\sqrt{V^2 + w^2}} \left(\Omega_z \dot{s} + \dot{\hat{\chi}} \right). \tag{A.22}$$

Singularities of the Transformation

Since the vehicle remains close to the reference line due to the track boundaries, $1 - \Omega_z d > 0$ holds. Consequently, singularities arise only for the relative orientation in (A.8) when $\dot{s} = 0\,\mathrm{m/s}$ and the curvature in (A.22) when $V = w = 0\,\mathrm{m/s}$. These can be avoided by imposing a lower limit on $\dot{s}$.

A.2 Transformation from Cartesian to Curvilinear Coordinates

The reverse transformation

$$[x, y, z, V, \hat{a}_\mathrm{x}, \hat{a}_\mathrm{y}, \hat{\chi}](t) \rightarrow \left[s, \dot{s}, \ddot{s}, d, \dot{d}, \ddot{d} \right](t) \tag{A.23}$$

can be stated in closed form by inverting the equations from Section A.1 except for the progress s. Assuming s is known, the lateral position d is obtained by inverting (A.3), and the velocities $\dot{d}$ and $\dot{s}$ are determined from (A.11b). As intermediate variables, w can be calculated from the third line of (A.7), and $\dot{\hat{\chi}}$ from the second line of (A.16) and the first and third lines of (A.15). Additionally, $\dot{V}$ can be derived from the first line of (A.16) and the second line of (A.15). With these, $\ddot{d}$ and $\ddot{s}$ are obtained by solving (A.17) and (A.18).

No analytical expression can be stated for s, since the reference line ${}^{\mathcal{I}}\boldsymbol{c}(s)$ is only available as a set of discrete points, as described in Subsection 2.3.2. Instead, s is obtained by numerically solving the following optimization problem:

$$s = \arg\min_{s'} \left\| {}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}} - {}^{\mathcal{I}}\boldsymbol{c}\left(s'\right) \right\|_2. \tag{A.24}$$

Determining the progress s can be ambiguous, as multiple points on the reference line ${}^{\mathcal{I}}\boldsymbol{c}(s)$ can have the same distance to the vehicle's position ${}^{\mathcal{I}}\boldsymbol{x}_{\mathcal{IV}}$. However, for the race tracks considered here, the curvatures of the reference line and the track widths are sufficiently small, causing the normal vectors of the reference line to intersect outside the track, thereby avoiding ambiguities.

Appendix B

Jerk-Optimal Curve Generation

Section B.1 presents the general solution of an optimal control problem based on Föllinger [177]. In Section B.2, this solution is then applied to an integrator chain, proving that quintic (fifth-order) or quartic (fourth-order) polynomials are jerk-optimal when the end state is fixed or constrained to a manifold. For an introduction to optimal control theory, please refer to [178].

B.1 General Optimal Control Solution

Consider a system $\dot{\boldsymbol{\xi}}(t) = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}(t))$ with the state vector $\boldsymbol{\xi}(t) \in \mathbb{R}^n$ and the input vector $\boldsymbol{u}(t) \in \mathbb{R}^p$. The optimal solution of the system in terms of the cost functional

$$J = h(\boldsymbol{\xi}(t), t)|_{t=t_\mathrm{e}} + \int_{t_0}^{t_\mathrm{e}} g(\boldsymbol{\xi}(t), \boldsymbol{u}(t), t)\,\mathrm{d}t \tag{B.1}$$

within the fixed time interval $[t_0, t_\mathrm{e}]$ with $\boldsymbol{\xi}(t_0) = \boldsymbol{\xi}_0$ must satisfy the following Hamilton equations (the time dependency is neglected):

$$H(\boldsymbol{\xi}, \boldsymbol{\psi}, \boldsymbol{u}, t) = g(\boldsymbol{\xi}, \boldsymbol{u}, t) + \boldsymbol{\psi}^\top \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{u}), \tag{B.2}$$

$$\boldsymbol{0} = \frac{\partial}{\partial \boldsymbol{u}} H(\boldsymbol{\xi}, \boldsymbol{\psi}, \boldsymbol{u}, t), \tag{B.3}$$

$$\dot{\boldsymbol{\psi}} = -\frac{\partial}{\partial \boldsymbol{\xi}} H(\boldsymbol{\xi}, \boldsymbol{\psi}, \boldsymbol{u}, t), \tag{B.4}$$

$$\dot{\boldsymbol{\xi}} = \frac{\partial}{\partial \boldsymbol{\psi}} H(\boldsymbol{\xi}, \boldsymbol{\psi}, \boldsymbol{u}, t) = \boldsymbol{f}(\boldsymbol{\xi}, \boldsymbol{u}). \tag{B.5}$$

The costate vector $\boldsymbol{\psi}(t) \in \mathbb{R}^n$ is a Lagrange multiplier required to account for the system dynamics. If the end state $\boldsymbol{\xi}(t_\mathrm{e})$ is fixed to $\boldsymbol{\xi}_\mathrm{e}$, the end condition $\boldsymbol{\xi}(t_\mathrm{e}) = \boldsymbol{\xi}_\mathrm{e}$ also applies. However, if the end state $\boldsymbol{\xi}(t_\mathrm{e})$ is constrained to a manifold described by the equations $\mathbf{0} = \boldsymbol{m}(\boldsymbol{\xi}(t))|_{t=t_\mathrm{e}} \in \mathbb{R}^m$ with $0 \leq m < n$, the following transversality condition must apply instead (again, the time dependency is neglected):

$$\left.\frac{\partial h(\boldsymbol{\xi}, t)}{\partial \boldsymbol{\xi}}\right|_{t=t_\mathrm{e}} + \boldsymbol{\psi}(t_\mathrm{e}) - \left.\left(\frac{\partial \boldsymbol{m}(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}\right)^\top\right|_{t=t_\mathrm{e}} \boldsymbol{\mu} = \mathbf{0}. \tag{B.6}$$

The vector $\boldsymbol{\mu} \in \mathbb{R}^m$, again, is a Lagrange multiplier required to account for the manifold.

B.2 Jerk-Optimal Solution for Integrator Chain

The third-order integrator chain

$$\dot{\boldsymbol{\xi}}(t) = \begin{bmatrix} \dot{\xi}_1(t) \\ \dot{\xi}_2(t) \\ \dot{\xi}_3(t) \end{bmatrix} = \begin{bmatrix} \xi_2(t) \\ \xi_3(t) \\ u(t) \end{bmatrix} \tag{B.7}$$

is now given with the position $\xi_1(t)$, the velocity $\xi_2(t)$, the acceleration $\xi_3(t)$, and the jerk $u(t)$ as input. The jerk-optimal solution is sought, i.e., the solution that minimizes the cost functional

$$J = \frac{1}{2}\int_0^{t_\mathrm{e}} u^2(t)\,\mathrm{d}t \tag{B.8}$$

in the fixed time interval $[0, t_\mathrm{e}]$ with the initial state $\boldsymbol{\xi}(0) = \boldsymbol{\xi}_0$. Two cases are considered for the end state $\boldsymbol{\xi}(t_\mathrm{e})$. In the first case, the end state is fixed, i.e., $\boldsymbol{\xi}(t_\mathrm{e}) = \boldsymbol{\xi}_\mathrm{e}$, and in the second case, the end position $\xi_1(t_\mathrm{e})$ is free, while $\xi_2(t_\mathrm{e})$ and $\xi_3(t_\mathrm{e})$ are fixed. With this problem definition and the costate vector $\boldsymbol{\psi}(t) = \begin{bmatrix}\psi_1(t) & \psi_2(t) & \psi_3(t)\end{bmatrix}^\top$, the Hamilton function follows according to (B.2):

$$H(\boldsymbol{\xi}, \boldsymbol{\psi}, u) = \frac{1}{2}u^2(t) + \psi_1(t)\cdot\xi_2(t) + \psi_2(t)\cdot\xi_3(t) + \psi_3(t)\cdot u(t). \tag{B.9}$$

Evaluation of (B.3) results in an expression for the optimal input:

$$u(t) = -\psi_3(t). \tag{B.10}$$

Further, the evaluation of (B.4) and (B.5) gives a coupled ordinary differential equation in $\boldsymbol{\psi}(t)$ and $\boldsymbol{\xi}(t)$:

$$\dot{\boldsymbol{\psi}} = \begin{bmatrix} \dot{\psi}_1(t) \\ \dot{\psi}_2(t) \\ \dot{\psi}_3(t) \end{bmatrix} = \begin{bmatrix} 0 \\ -\psi_1(t) \\ -\psi_2(t) \end{bmatrix}, \tag{B.11}$$

$$\dot{\boldsymbol{\xi}}(t) = \begin{bmatrix} \dot{\xi}_1(t) \\ \dot{\xi}_2(t) \\ \dot{\xi}_3(t) \end{bmatrix} = \begin{bmatrix} \xi_2(t) \\ \xi_3(t) \\ u(t) \end{bmatrix} \overset{\text{(B.10)}}{=} \begin{bmatrix} \xi_2(t) \\ \xi_3(t) \\ -\psi_3(t) \end{bmatrix}. \tag{B.12}$$

By integrating the first line from (B.11), it follows that $\psi_1(t)$ is constant. Inserting this solution into the next line and integrating it gives the solution for $\psi_2(t)$. Repeating this process yields all solutions with the integration constants c_1 to c_6:

$$\psi_1(t) = -c_6, \tag{B.13a}$$
$$\psi_2(t) = c_6 t + c_5, \tag{B.13b}$$
$$\psi_3(t) = -\frac{1}{2}c_6 t^2 - c_5 t - c_4, \tag{B.13c}$$
$$\xi_3(t) = \frac{1}{6}c_6 t^3 + \frac{1}{2}c_5 t^2 + c_4 t + c_3, \tag{B.13d}$$
$$\xi_2(t) = \frac{1}{24}c_6 t^4 + \frac{1}{6}c_5 t^3 + \frac{1}{2}c_4 t^2 + c_3 t + c_2, \tag{B.13e}$$
$$\xi_1(t) = \frac{1}{120}c_6 t^5 + \frac{1}{24}c_5 t^4 + \frac{1}{6}c_4 t^3 + \frac{1}{2}c_3 t^2 + c_2 t + c_1. \tag{B.13f}$$

The unknown integration constants c_1 to c_6 remain to be determined by utilizing the boundary conditions. Subsequently, two cases are considered: first, for a fixed end state $\boldsymbol{\xi}(t_\mathrm{e}) = \boldsymbol{\xi}_\mathrm{e}$ (case 1), and second, for a free end position $\xi_1(t_\mathrm{e})$, such that the end state is constrained to a manifold (case 2).

Case 1: Fixed end state

If the end state is fixed, there are three conditions for the initial state $\boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 = \begin{bmatrix}\xi_{1,0} & \xi_{2,0} & \xi_{3,0}\end{bmatrix}^\top$ and three for the end state $\boldsymbol{\xi}(t_\mathrm{e}) = \boldsymbol{\xi}_\mathrm{e} = \begin{bmatrix}\xi_{1,\mathrm{e}} & \xi_{2,\mathrm{e}} & \xi_{3,\mathrm{e}}\end{bmatrix}^\top$. Applying these conditions to (B.13d) to (B.13f) gives the following system of equations:

$$\begin{bmatrix}\xi_{1,0}\\ \xi_{2,0}\\ \xi_{3,0}\\ \xi_{1,\mathrm{e}}\\ \xi_{2,\mathrm{e}}\\ \xi_{3,\mathrm{e}}\end{bmatrix} = \underbrace{\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 1 & t_\mathrm{e} & \frac{1}{2}t_\mathrm{e}^2 & \frac{1}{6}t_\mathrm{e}^3 & \frac{1}{24}t_\mathrm{e}^4 & \frac{1}{120}t_\mathrm{e}^5\\ 0 & 1 & t_\mathrm{e} & \frac{1}{2}t_\mathrm{e}^2 & \frac{1}{6}t_\mathrm{e}^3 & \frac{1}{24}t_\mathrm{e}^4\\ 0 & 0 & 1 & t_\mathrm{e} & \frac{1}{2}t_\mathrm{e}^2 & \frac{1}{6}t_\mathrm{e}^3\end{bmatrix}}_{\boldsymbol{M}_1(t_\mathrm{e})} \cdot \begin{bmatrix}c_1\\ c_2\\ c_3\\ c_4\\ c_5\\ c_6\end{bmatrix}. \tag{B.14}$$

Solving the system of equations (B.14) by inversion of $\boldsymbol{M}_1(t_\mathrm{e})$ yields the unknown constants c_1 to c_6:[1]

$$\begin{bmatrix}c_1\\ c_2\\ c_3\\ c_4\\ c_5\\ c_6\end{bmatrix} = \begin{bmatrix}\xi_{1,0}\\ \xi_{2,0}\\ \xi_{3,0}\\ -\frac{3}{t_\mathrm{e}^3}(20\xi_{1,0} + 12t_\mathrm{e}\xi_{2,0} + 3t_\mathrm{e}^2\xi_{3,0} - 20\xi_{1,\mathrm{e}} + 8t_\mathrm{e}\xi_{2,\mathrm{e}} - t_\mathrm{e}^2\xi_{3,\mathrm{e}})\\ \frac{12}{t_\mathrm{e}^4}(30\xi_{1,0} + 16t_\mathrm{e}\xi_{2,0} + 3t_\mathrm{e}^2\xi_{3,0} - 30\xi_{1,\mathrm{e}} + 14t_\mathrm{e}\xi_{2,\mathrm{e}} - 2t_\mathrm{e}^2\xi_{3,\mathrm{e}})\\ -\frac{60}{t_\mathrm{e}^5}(12\xi_{1,0} + 6t_\mathrm{e}\xi_{2,0} + t_\mathrm{e}^2\xi_{3,0} - 12\xi_{1,\mathrm{e}} + 6t_\mathrm{e}\xi_{2,\mathrm{e}} - t_\mathrm{e}^2\xi_{3,\mathrm{e}})\end{bmatrix}. \tag{B.15}$$

As the constants generally do not equal 0, the jerk-optimal solution for the position $\xi_1(t)$ is a quintic (fifth-order) polynomial according to (B.13f).

[1] The matrix $\boldsymbol{M}_1(t_\mathrm{e})$ is regular for $t_\mathrm{e} \neq 0$, as $\det(\boldsymbol{M}_1(t_\mathrm{e})) = {t_\mathrm{e}^9}/{8640}$.

Case 2: End state is constrained to a manifold

If the end position $\xi_1(t_\mathrm{e})$ is free, the end state $\boldsymbol{\xi}(t_\mathrm{e})$ is constrained to a manifold specified by

$$\mathbf{0} = \boldsymbol{m}(\boldsymbol{\xi}(t))|_{t=t_\mathrm{e}} = \begin{bmatrix} \xi_2(t) - \xi_{2,\mathrm{e}} \\ \xi_3(t) - \xi_{3,\mathrm{e}} \end{bmatrix}\Bigg|_{t=t_\mathrm{e}} . \tag{B.16}$$

Since $h(\boldsymbol{\xi}(t), t) = 0$, it follows from the transversality condition (B.6) that

$$\begin{bmatrix} \psi_1(t_\mathrm{e}) \\ \psi_2(t_\mathrm{e}) \\ \psi_3(t_\mathrm{e}) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = \begin{bmatrix} 0 \\ \mu_1 \\ \mu_2 \end{bmatrix} . \tag{B.17}$$

The first line of (B.17), combined with (B.13a), shows that $c_6 = 0$. Using this with the three conditions for the initial state $\boldsymbol{\xi}(0) = \boldsymbol{\xi}_0 = \begin{bmatrix} \xi_{1,0} & \xi_{2,0} & \xi_{3,0} \end{bmatrix}^\top$ and the two conditions for the end state, $\xi_2(t_\mathrm{e}) = \xi_{2,\mathrm{e}}$ and $\xi_3(t_\mathrm{e}) = \xi_{3,\mathrm{e}}$, the following system of equations is derived from (B.13d) to (B.13f):

$$\begin{bmatrix} \xi_{1,0} \\ \xi_{2,0} \\ \xi_{3,0} \\ \xi_{2,\mathrm{e}} \\ \xi_{3,\mathrm{e}} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & t_\mathrm{e} & \frac{1}{2}t_\mathrm{e}^2 & \frac{1}{6}t_\mathrm{e}^3 \\ 0 & 0 & 1 & t_\mathrm{e} & \frac{1}{2}t_\mathrm{e}^2 \end{bmatrix}}_{\boldsymbol{M}_2(t_\mathrm{e})} \cdot \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} . \tag{B.18}$$

Solving (B.18) by inversion of $\boldsymbol{M}_2(t_\mathrm{e})$ gives the unknown constants c_1 to c_5:[2]

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \end{bmatrix} = \begin{bmatrix} \xi_{1,0} \\ \xi_{2,0} \\ \xi_{3,0} \\ -\frac{2}{t_\mathrm{e}^2}(3\xi_{2,0} + 2t_\mathrm{e}\xi_{3,0} - 3\xi_{2,\mathrm{e}} + t_\mathrm{e}\xi_{3,\mathrm{e}}) \\ \frac{6}{t_\mathrm{e}^3}(2\xi_{2,0} + t_\mathrm{e}\xi_{3,0} - 2\xi_{2,\mathrm{e}} + t_\mathrm{e}\xi_{3,\mathrm{e}}) \end{bmatrix} . \tag{B.19}$$

Since $c_6 = 0$ holds, the jerk-optimal solution for the position $\xi_1(t)$ is a quartic (fourth-order) polynomial according to (B.13f).

[2]The matrix $\boldsymbol{M}_2(t_\mathrm{e})$ is regular for $t_\mathrm{e} \neq 0$, as $\det(\boldsymbol{M}_1(t_\mathrm{e})) = t_\mathrm{e}^4/12$.

Appendix C

Racing Line Adaption for Sampling-Based Planning

This chapter describes the determination of the lateral profile $\tilde{d}_{\mathrm{rl}}(t)$ and its derivatives required to replicate the racing line path for a given longitudinal curve $s(t)$. As illustrated in Figure 4.3, the same progress is reached by $s_{\mathrm{rl}}(t)$ and $s(t)$ at different points in time. The time $\tilde{t}$ at which $s_{\mathrm{rl}}(t)$ reaches the corresponding progress $s(t)$ can be formulated as $\tilde{t} = s_{\mathrm{rl}}^{-1}(s(t))$. The adapted racing line curve $\tilde{d}_{\mathrm{rl}}(t)$ is finally obtained by evaluating $d_{\mathrm{rl}}(t)$ at $\tilde{t}$:

$$\tilde{d}_{\mathrm{rl}}(t) = d_{\mathrm{rl}}(\tilde{t}) = d_{\mathrm{rl}}(s_{\mathrm{rl}}^{-1}(s(t))). \tag{C.1}$$

The first time derivative is obtained by applying the chain rule:

$$\dot{\tilde{d}}_{\mathrm{rl}}(t) = \frac{\mathrm{d}}{\mathrm{d}\tilde{t}} d_{\mathrm{rl}}(\tilde{t}) \cdot \left(s_{\mathrm{rl}}^{-1}\right)'(s(t)) \cdot \dot{s}(t). \tag{C.2}$$

Remind that, following the convention in this thesis, $\dot{\square} = \frac{\mathrm{d}\square}{\mathrm{d}t}$ denotes the derivative with respect to time t and $\square' = \frac{\mathrm{d}\square}{\mathrm{d}s}$ the derivative with respect to progress s. By applying the product rule and chain rule, the second derivative is given by

$$\begin{aligned}\ddot{\tilde{d}}_{\mathrm{rl}}(t) =& \frac{\mathrm{d}^2}{\mathrm{d}\tilde{t}^2} d_{\mathrm{rl}}(\tilde{t}) \cdot \left[\left(s_{\mathrm{rl}}^{-1}\right)'(s(t))\right]^2 \cdot \dot{s}^2(t) \\ &+ \frac{\mathrm{d}}{\mathrm{d}\tilde{t}} d_{\mathrm{rl}}(\tilde{t}) \cdot \left(s_{\mathrm{rl}}^{-1}\right)''(s(t)) \cdot \dot{s}^2(t) \\ &+ \frac{\mathrm{d}}{\mathrm{d}\tilde{t}} d_{\mathrm{rl}}(\tilde{t}) \cdot \left(s_{\mathrm{rl}}^{-1}\right)'(s(t)) \cdot \ddot{s}(t).\end{aligned} \tag{C.3}$$

Since the derivatives of the inverse of the longitudinal racing line profile are not known, these terms have to be reformulated. As the longitudinal racing line velocity $\dot{s}_{\mathrm{rl}}(t)$ is naturally positive, $s_{\mathrm{rl}}(t)$ is a monotonically increasing and thus bijective function, allowing the application of the inverse function rule [179]:

$$\left(s_{\mathrm{rl}}^{-1}\right)'(s(t)) = \frac{1}{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})}. \tag{C.4}$$

Furthermore, the application of the chain rule leads to

$$\left(s_{\mathrm{rl}}^{-1}\right)''(s(t)) = -\frac{1}{\left[\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})\right]^2} \cdot \frac{\mathrm{d}^2}{\mathrm{d}\tilde{t}^2}\, s_{\mathrm{rl}}(\tilde{t}) \cdot \left(s_{\mathrm{rl}}^{-1}\right)'(s(t)) \tag{C.5a}$$

$$\overset{(C.4)}{=} -\frac{1}{\left[\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})\right]^3} \cdot \frac{\mathrm{d}^2}{\mathrm{d}\tilde{t}^2}\, s_{\mathrm{rl}}(\tilde{t}). \tag{C.5b}$$

Substituting (C.4) and (C.5b) into (C.2) and (C.3) finally leads to the desired expressions:

$$\tilde{d}_{\mathrm{rl}}(t) = d_{\mathrm{rl}}(\tilde{t}), \tag{C.6}$$

$$\dot{\tilde{d}}_{\mathrm{rl}}(t) = \frac{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, d_{\mathrm{rl}}(\tilde{t})}{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})} \cdot \dot{s}(t), \tag{C.7}$$

$$\ddot{\tilde{d}}_{\mathrm{rl}}(t) = \frac{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t}) \frac{\mathrm{d}^2}{\mathrm{d}\tilde{t}^2}\, d_{\mathrm{rl}}(\tilde{t}) - \frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, d_{\mathrm{rl}}(\tilde{t}) \frac{\mathrm{d}^2}{\mathrm{d}\tilde{t}^2}\, s_{\mathrm{rl}}(\tilde{t})}{\left[\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})\right]^3} \cdot \dot{s}^2(t) + \frac{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, d_{\mathrm{rl}}(\tilde{t})}{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}\, s_{\mathrm{rl}}(\tilde{t})} \cdot \ddot{s}(t). \tag{C.8}$$

$\dot{s}(t)$ and $\ddot{s}(t)$ are the derivatives of the longitudinal curve $s(t)$ and are thus known. The expressions of the racing line, depending on the time $\tilde{t}$, represent the original racing line quantities at the longitudinal curve positions $s(t)$. They can be obtained by interpolating the given racing line data (2.43) at the desired positions $s(t)$.

Appendix D

Kinematic Single-Track Model in Curvilinear Coordinates

The following assumptions are made for the derivation of the kinematic single-track model introduced in (5.16):

- The vehicle travels on the x-y plane. The center of gravity is at road level.
- Both wheels of an axle are combined into one wheel, and only the front wheel is steered.
- Only the kinematic relationships are considered. Thus, there is no side slip, i.e., the velocities of the rear and front tires are aligned with the tires.

Figure D.1 illustrates the resulting vehicle model. The input vector $\boldsymbol{u}(t) = \begin{bmatrix} \omega(t) & \hat{a}_{\mathrm{x}}(t) \end{bmatrix}^{\top}$ comprises the steering rate $\omega(t)$ and the longitudinal acceleration $\hat{a}_{\mathrm{x}}(t)$. The state vector $\boldsymbol{z}(t) = \begin{bmatrix} s(t) & d(t) & \hat{\chi}(t) & V(t) & \delta(t) \end{bmatrix}^{\top}$ consists of the longitudinal position $s(t)$, the lateral position $d(t)$, the relative orientation $\hat{\chi}(t)$ to the reference line $\mathfrak{R}$, the absolute velocity $V(t)$, and the steering angle $\delta(t)$.

The first two lines of the system dynamics, i.e., (5.16a) and (5.16b), result directly by rearranging (A.11b). The fourth line, i.e., (5.16d), follows from (A.16) for the 2D case. The fifth line, i.e., (5.16e), results from the integral relation between the steering angle $\delta(t)$ and steering rate $\omega(t)$. Thus, only the third line, i.e., (5.16c), remains, requiring an expression for $\dot{\hat{\chi}}(t)$.

The relative orientation $\hat{\chi}(t)$ is defined as the difference between the car orientation $\theta_{\mathrm{C}}(t)$ and the reference line orientation $\theta(s(t))$ at progress s:

$$\hat{\chi}(t) = \theta_{\mathrm{C}}(t) - \theta(s(t)). \tag{D.1}$$

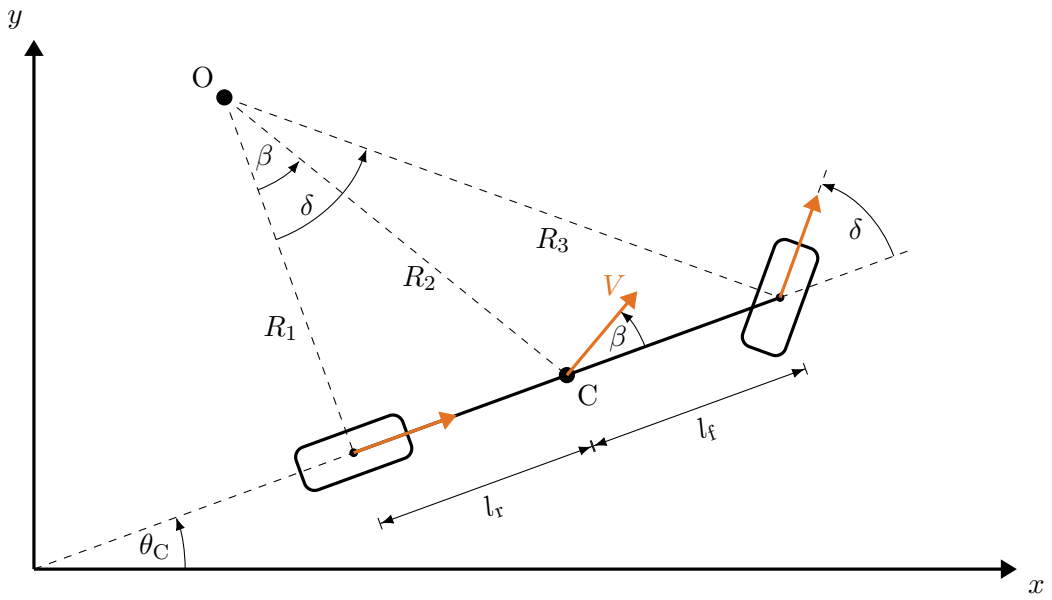


Figure D.1: Illustration of the kinematic single-track model. The dashed lines perpendicular to the velocity vectors (orange arrows) intersect at the instant center of rotation O.

Differentiating with respect to time and applying the chain rule gives

$$\dot{\hat{\chi}}(t) = \dot{\theta}_{\mathrm{C}}(t) - \theta'(s(t)) \cdot \dot{s}(t) \tag{D.2a}$$

$$\overset{(2.24b)}{=} \dot{\theta}_{\mathrm{C}}(t) - \Omega_{\mathrm{z}}(s(t)) \cdot \dot{s}(t). \tag{D.2b}$$

Note that according to (2.24b), the relationship $\theta'(s) = \Omega_{\mathrm{z}}(s)$ holds only in the 2D case. An expression for the longitudinal velocity $\dot{s}(t)$ is given with (5.16a). The rate of absolute vehicle orientation $\dot{\theta}_{\mathrm{C}}(t)$ can be expressed as

$$\dot{\theta}_{\mathrm{C}}(t) = \frac{V(t)}{R_2(t)}. \tag{D.3}$$

Furthermore, $\sin(\beta(t)) = {}^{l_{\mathrm{r}}}/_{R_2(t)}$ applies in the right-angled triangle with the hypotenuse $R_2(t)$ from Figure D.1. Inserting this into (D.3) yields an expression for $\dot{\theta}_{\mathrm{C}}(t)$ in dependence on the body slip angle $\beta(t)$:

$$\dot{\theta}_{\mathrm{C}}(t) = \frac{V(t)}{l_{\mathrm{r}}} \cdot \sin(\beta(t)). \tag{D.4}$$

Inserting (5.16a) and (D.4) into (D.2b) gives the third line of the system dynamics, i.e., (5.16c):

$$\dot{\hat{\chi}}(t) = \frac{V(t)}{l_\mathrm{r}} \cdot \sin(\beta(t)) - V(t) \cdot \cos(\hat{\chi}(t)) \cdot \frac{\Omega_\mathrm{z}(s(t))}{1 - d(t) \cdot \Omega_\mathrm{z}(s(t))}. \tag{D.5}$$

However, an expression for the body slip angle $\beta(t)$ must still be determined. From the right-angled triangle with the hypotenuse $R_3(t)$, it follows that

$$\tan(\delta(t)) = \frac{l_\mathrm{r} + l_\mathrm{f}}{R_1(t)}. \tag{D.6}$$

Finally, the desired expression for the body slip angle is obtained from the right-angled triangle with hypotenuse $R_2(t)$:

$$\beta(t) = \arctan\left(\frac{l_\mathrm{r}}{R_1(t)}\right) \tag{D.7a}$$

$$\overset{(D.6)}{=} \arctan\left(\frac{l_\mathrm{r}}{l_\mathrm{r} + l_\mathrm{f}} \cdot \tan(\delta(t))\right). \tag{D.7b}$$