Two-Stage Trajectory Planning for Automated Highway Driving

Hierarchically Combined Trajectory Sampling and Numerical Optimization Subject to Semi-Infinite Constraints

A thesis approved for the academic degree of

Doktor der Ingenieurswissenschaften (Dr.-Ing.)

at the

Faculty of Electrical Engineering and Information Technology

TU Dortmund University

by

Philip Dorpmüller, M.Sc.

Unna, Germany

Supervisor: Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram,
TU Dortmund University
Co-Advisor: Univ.-Prof. Dr.-Ing. Steffen Müller,
TU Berlin University

Day of Oral Examination: September 18, 2025

Acknowledgement

My work at the Institute of Control Theory and Systems Engineering, part of the Faculty of Electrical Engineering and Information Technology at the Technical University of Dortmund, enabled this thesis. Also, several people have contributed to this thesis. I would like to express my gratitude for their support.

I would like to thank Univ.-Prof. Dr.-Ing. Prof. h.c. Dr. h.c. Torsten Bertram for the opportunity to write my doctoral thesis, the scientific and personal advice, and the necessary resources. I am grateful for the freedom he provided me in pursuing my scientific interests and for growing a supportive and motivating work environment.

Also, I would like to thank Univ.-Prof. Dr.-Ing. Steffen Müller for his interest in my work and for reviewing my thesis as the second examiner. Likewise, I would like to thank Univ.-Prof. Dr.-Ing. Peter Krummrich for his engagement as the third examiner, and Univ.-Prof. Dr.-Ing. Christian Rehtanz for chairing the examination committee.

I would like to express my gratitude to the current and former members of the Institute of Control Theory and Systems Engineering team for providing an inspiring atmosphere. It was a great pleasure to learn and work with highly motivated and supportive colleagues. Many thanks to Dr.-Ing. Christian Lienke, who encouraged me to start my journey as a doctoral candidate. I also thank him for the time spent with scientific discussions in extended weekly meetings, and for sharing his experience.

I would also like to thank Dr.-Ing. Manuel Schmidt, Martin Krüger, Dr.-Ing. Andreas Homann, Niklas Stannartz, Christopher Diehl, and Dr.-Ing. Artemi Makarow for their scientific and personal advice. I am very grateful for the valuable ideas and the feedback from Dr.-Ing. Daniel Schauten and apl. Prof. Dr. rer. nat. Frank Hoffmann, who have guided my research and the development of this thesis.

Also, many thanks to Dr.-Ing. Khazar Dargahi Nobari and Heiko Renz for contributing to this thesis as proofreaders.

Sascha Kersting and Halit Cicek always provided excellent assistance in solving all technical problems. Likewise, I could count on Nicole Czerwinski for her support in organizational and administrative issues. Therefore, I would like to express my gratitude.

This work would not have been possible without the support of the ZF Group and the collaborative research projects initiated by Prof. Dr.-Ing. Martin Keller. Also, I am grateful for the technical discussions and the scientific reviews of Prof. Dr.-Ing. Martin Keller, Naveen Bejagam, and Dr. rer. nat. Thomas Schmitz over the course of the projects.

Special thanks go to my parents for their unconditional support at any time. I had to approach problems and questions in my free time and beyond my time at the Institute of Control Theory and Systems Engineering. Thus, I am grateful to my family and friends, who offered understanding ears and the sometimes necessary distraction.

Überlingen, October 2025 Philip Dorpmüller

Abstract

Although automated driving is often viewed as a future technology, it is already part of many peoples' everyday lives. Driver assistance systems can support the driver in various tasks, such as lane keeping or parking. Moreover, commercially available systems can relieve the driver from the driving task under certain conditions. Many practical problems have been solved, but there are still open questions and room for improvement in safety and performance. Still, computationally efficient implementations are sought that can achieve consistent behavior and far-sighted decisions. The underlying control problem is structured hierarchically into modules to reduce the computational complexity. The behavior and motion planning modules select the best maneuver and trajectory for the controlled vehicle to achieve safety, progress, and comfort.

This work formalizes the task of both modules in a single continuous-time optimal control problem. The problem incorporates preferences via the cost functional and allows the consideration of constraints due to environmental limitations, targets, and vehicle dynamics. In addition, an integer variable encodes the different maneuver options. Requirements on the optimal control problem elements are derived to ensure compliance with driving objectives and to enable stability for the given target. Stability is provided by a terminal set, designed along safe and desirable regions of the state space. The terminal set is combined with the terminal cost for far-sighted maneuver decisions. The vehicle's future trajectory is described by polynomial splines, leveraging the differential flatness property of the applied vehicle model. The spline is contained in the convex hull of its basis forms' coefficients, which are used to find a feasible solution subject to continuous-time constraints. A two-stage solution algorithm is proposed to solve the resulting mixed-integer problem efficiently. The first stage comprises a sampling-based algorithm that discretizes the time and state space to form a graph encoding nonuniform candidate splines. A shortest-path search algorithm selects the optimal candidate and the corresponding maneuver. While it finds a continuous-time feasible solution considering the integer variable, the computational complexity grows excessively with the required accuracy. Thus, a numerical optimization-based algorithm is introduced in the second stage. It solves a nonlinear program, incorporating the first stage's cost and constraints. A shrinking horizon approach based on nonuniform splines is proposed to enable recursive feasibility and convergence to the terminal set. Consequently, the first stage can always supply a feasible initial guess. This property increases the algorithm's robustness to disturbances, especially if the solution from the previous time step is infeasible.

The two algorithms are evaluated in multiple simulated highway scenarios. It is shown that the vehicle can be controlled to a distant target while successfully circumnavigating multiple other vehicles. Simultaneously, the driving objectives are mostly fulfilled. While reducing the trajectories' degrees of freedom impedes the progression toward the target, it does not compromise stability and safety.

Contents

Nomenclature

The following introduces the symbols, acronyms and abbreviations used in this work. Generally, all symbols are introduced in their respective context.
Polygons $\square, \triangle, \diamond$ denote placeholder variables and are substitutes for context-dependent single and multiple variables. The placeholders $\boxplus, \boxtimes, \triangledown$ are bound to specific contexts introduced below. In contrast, the running indices j, n, l, ι, μ are elements of the natural numbers.
Variables and functions are distinguished into scalar-valued and vector-valued. Lower or uppercase, italic, bold, or Greek letters $a, A, \mathbf{a}, \mathbf{A}, \lambda, \Lambda$ are used for scalar variables and functions. Vector variables and functions are underlined $\underline{a}, \underline{A}, \underline{\mathbf{a}}, \underline{\mathbf{A}}, \underline{\lambda}, \underline{\Lambda}$. If not omitted for brevity, functions $\square(\cdot)$ are distinguished with an argument from variables $\square$. All functions $s_{\square}(\cdot)$ denote polynomial splines in basis form. The coefficients $\underline{\mathbf{c}}_{\square}$, breakpoints $\underline{\mathbf{k}}_{\square}$, knots $\underline{k}_{\square}$, order $\rho_{\square}$, continuities $\underline{w}_{\square}$, and basis splines $\underline{b}_{\square}$ of the corresponding $s_{\square}(\cdot)$ share the same subscript. Sets and spaces are denoted by blackboard bold or Zapf Chancery letters $\mathbb{A}, \mathcal{A}$. Accents usually indicate a variation of a variable or function without accents. The following accents are reserved for special purposes:

- $\check{\square}$ variable or function lower bound,
- $\hat{\square}$ variable or function upper bound,
- $\bar{\square}$ variable or function symmetric upper bound,
- $\breve{\square}$ number of vector elements, set cardinality, or space dimensionality.

Placeholders

$\boxplus$	Placeholder indicating a spline resulting from a sum
$\boxtimes$	Placeholder indicating a spline resulting from a product
$\square, \triangle, \diamond$	General placeholder variables
$\triangledown$	Placeholder for $\triangledown = x, y$ direction indices

Notation

$\square'$	First position derivative $\square' \coloneqq \frac{\mathrm{d}\square}{\mathrm{d}z_x(\cdot)}$
$\dddot{\square}$	Third order time derivative $\dddot{\square} \coloneqq \frac{\mathrm{d}^3\square}{\mathrm{d}t^3}$
$\ddot{\square}$	Second order time derivative $\ddot{\square} \coloneqq \frac{\mathrm{d}^2\square}{\mathrm{d}t^2}$
$\dot{\square}$	First order time derivative $\dot{\square} \coloneqq \frac{\mathrm{d}\square}{\mathrm{d}t}$

Reference Frames

${}^{\mathfrak{p}}\square$	Frénet coordinate frame
${}^{\mathfrak{v}}\square$	Vehicle coordinate frame
${}^{\mathfrak{w}}\square$	World coordinate frame

Sets

$\mathbb{B}$	Basis spline evaluations resulting from all breakpoint combinations of the global planner
$\mathcal{C}$	State space nonlinear system
$\mathcal{C}_0$	Feasible state space nonlinear system
$\mathbb{D}_j$	Neighboring lane identifiers at the position of node $\underline{\mathbf{N}}_j$
$\mathcal{D}_{\tilde{v},j}$	Successor state candidates of node $\underline{\mathbf{N}}_j$ in x-direction at target velocity $\tilde{v}$
$\mathcal{D}_x, \mathcal{D}_y$	Directional sets of discrete successor states in x- and y-direction
$\mathcal{D}_{\dot{x},j}$	Successor state candidates of node $\underline{\mathbf{N}}_j$ in x-direction at velocity candidates $\mathcal{Z}_{\dot{x}}$
$\mathcal{D}_{\tilde{x},j}$	Successor state candidates of node $\underline{\mathbf{N}}_j$ in x-direction along the target trajectory behind the other vehicles
$\mathcal{D}_{y,j}$	Successor state candidates of node $\underline{\mathbf{N}}_j$ in y-direction in the neighboring lane centers
$\mathbb{H}$	Constraint splines
$\tilde{\mathbb{H}}$	Current constraint splines
$\mathcal{I}_h$	Indices of the lower bounded spline coefficients
${}_{j,n}\mathcal{I}_b$	Indices corresponding to nonzero basis splines along a spline segment between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$
$\mathcal{I}_{\check{\mathbf{k}},x}, \mathcal{I}_{\check{\mathbf{k}},y}$	Feasible breakpoint number in x- and y-direction
$\mathcal{I}_{\mathbf{N},\mathrm{o}}$	Open node indices
$\mathcal{I}_{\mathbf{N},\mathrm{s}}$	Successor node indices
$\mathcal{I}_{\mathrm{T}}$	Local target indices
$\mathcal{I}_\chi$	Feasible time intervals for maximum breakpoints χ
$\tilde{\mathcal{I}}_\square$	Indices of $s_\square(\cdot)$ coefficients in the optimization variables
$\mathcal{K}_{j,x}, \mathcal{K}_{j,y}$	Successor breakpoint candidates of node $\underline{\mathbf{N}}_j$ in x- and y-direction
$\mathcal{K}_x, \mathcal{K}_y$	Time samples in x- and y-direction
$\mathcal{L}$	Transformed state space
$\mathbb{N}$	Natural numbers
$\mathbb{N}_j$	Lower bounded natural numbers $\mathbb{N}_j = \{n \in \mathbb{N} \mid n \geq j \in \mathbb{N}\}$
$\mathbb{R}$	Real numbers
$\mathbb{R}^+$	Positive real numbers
${}_j\mathcal{S}_\square$	$\mathcal{S}_{\underline{\mathbf{k}}_\square, \underline{w}_\square, \rho_\square}$ at time step t_j
$\mathcal{S}_{\underline{\mathbf{k}}_\square, w_\square, \rho_\square}$	Set of polynomial splines $s_{\rho_\square, \underline{w}_\square}(\cdot, \cdot, \underline{\mathbf{k}}_\square)$
$\mathbb{T}$	Coefficient transformations for all breakpoint combinations
$\mathcal{T}$	Nonlinear system terminal set
$\mathcal{T}_{\mathrm{d}}$	Global target terminal equilibrium states
$\mathcal{T}_j$	Local targets $j \in \mathcal{I}_{\mathrm{T}}$ in nonlinear system terminal set
$\tilde{\mathcal{T}}$	Transformed terminal set
$\tilde{\mathcal{T}}_{x,\mathrm{d}}, \tilde{\mathcal{T}}_{y,\mathrm{d}}$	Transformed states representing the global target in x- and y-direction
$\tilde{\mathcal{T}}_{x,m}, \tilde{\mathcal{T}}_{y,m}$	$m = -2, -1, \ldots, \eta_{\mathrm{ov}}$ transformed local targets in x- and y-direction
$\mathcal{U}$	Input space nonlinear system
$\mathcal{U}_0$	Feasible input space nonlinear system

$\mathcal{X}$	Breakpoint sequence lengths
$\mathcal{Z}$	Output space nonlinear system
$\mathcal{Z}_{v,x,m}$	Initial x-velocities of the $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles
$\mathcal{Z}_{\dot{x}}$	x-velocity samples

Scalar Valued Variables

$\bar{a}_y$	Symmetric bound on vehicle y-acceleration
$\check{a}_{\text{idm},m}, \hat{a}_{\text{idm},m}$	Lower and upper bounds on acceleration of the $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles according to intelligent driver model
$\tilde{a}$	Desired acceleration of the intelligent driver model
$\tilde{a}_{\text{d}}$	Desired deceleration of the intelligent driver model
$\bar{\bar{a}}_x, \bar{\bar{a}}_y$	Symmetric bounds on velocity induced x- and y-acceleration
$c_{\text{f}}, c_{\text{r}}$	Tire cornering stiffnesses at vehicle front and rear tires
$\breve{\mathbf{c}}_{\square}$	Coefficient number of spline $s_{\square}(\cdot)$
$\bar{\xi}_{\psi}$	Symmetric bound on vehicle yaw angle
$\check{\xi}_v, \hat{\xi}_v$	Lower and upper bounds on velocity
d_{lc}	Left-most lane center's y-position in Frénet frame
d_{mc}	Middle lane center's y-position in Frénet frame
d_{rc}	Right-most lane center's y-position in Frénet frame
δ	Distance bound on initial state
δ_j	Controllability index of the nonlinear system output $z_j(\cdot)$
$\Delta\mathbf{k}$	Breakpoint interval
$\Delta\check{\mathbf{k}}$	Minimum breakpoint interval
Δt	Simulation time increment
Δ_x, Δ_y	Semi-axis length for the ego vehicle in x- and y-direction
$\Delta_{x,m}, \Delta_{y,m}$	Semi-axis lengths of the $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles in x- and y-direction
$\Delta\tilde{z}_x$	Jam distance to the leading vehicle of the intelligent driver model
$\mathring{\Delta}_y$	Semi-axis length for the prediction uncertainty in y-direction
e	Acceleration exponent of the intelligent driver model
ϵ	Distance bound on the autonomous system's state trajectory
ϵ_x, ϵ_y	Dual mode control activation threshold in x- and y-direction
$\hat{\hat{F}}_x$	Transformed terminal cost upper bound in x-direction
g	Gravitational acceleration
γ	Arc length
$\breve{h}$	Number of inequality constraints
$\tilde{h}$	Current edge feasibility
H	Prediction horizon
η_{ov}	Number of other vehicles
η_x, η_y	Number of state samples in x- and y-direction
i_{ter}	Maximum number of iterations
ι	General running index
j	General running index
J_{v}	Vehicle moment of inertia
$\breve{k}_{\square}$	Number of knots along spline $s_{\square}(\cdot)$

$\breve{\mathbf{k}}_{xy}$	Number of combined breakpoints
$\breve{\mathbf{k}}_{\square}$	Breakpoint number along spline $s_{\square}(\cdot)$
$\breve{K}_x, \breve{K}_y$	Cardinality of time samples $\mathcal{K}_x$ and $\mathcal{K}_y$
$\bar{\kappa}_{\mathrm{p}}$	Reference path curvature symmetric bound
$\bar{\kappa}'_{\mathrm{p}}$	Reference path curvature rate symmetric bound
l	General running index
$l_{\mathrm{f}}, l_{\mathrm{r}}$	Distances from vehicle center of gravity to the front and rear tire
l_{v}	Distances between front and rear wheel
$\tilde{l}$	Current edge cost
${}_{j,n}\tilde{l}$	Cost for transition between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$
${}_{j,n}\tilde{l}_x, {}_{j,n}\tilde{l}_y$	Directional costs for transition between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$ in x- and y-direction
λ_0	Minimum cost descent factor
λ_1	Terminal cost upper bound factor
m	Terminal set and vehicle index
m_{v}	Vehicle mass
μ	General running index
μ_{ad}	Friction coefficient for adhesion
n	General running index
$\breve{\mathbf{N}}$	Number of nodes
$\bar{r}_v$	Symmetric bound on velocity ratio
$\rho_{\square}$	Order of spline $s_{\square}(\cdot)$
$\check{s}_{\square}, \hat{s}_{\square}$	Lower and upper bounds on $s_{\square}(\cdot)$
t, τ	Time
t_{hw}	Desired time headway
$\breve{t}$	Number of time steps in $\underline{t}$
$\overset{\circ}{t}_{\mathrm{hw}}$	Time headway for the prediction uncertainty in x-direction
$\tilde{t}_{\mathrm{hw}}$	Desired time headway
T_x, T_y	Control horizons in x- and y-direction
$\check{T}_x, \check{T}_y$	Lower bounds on control horizons in x- and y-direction
$\tilde{T}_x, \tilde{T}_y$	Maximum segment lengths of a splines in x- and y-direction with three breakpoints
$\check{u}_a, \hat{u}_a$	Lower and upper bounds on acceleration input
v_{ch}	Characteristic velocity
$\tilde{v}$	Desired velocity of the ego vehicle
$\tilde{v}_m$	Desired velocity of the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' intelligent driver model
$w_{\mathrm{j},x}, w_{\mathrm{j},y}$	Cost weights on squared jerk in x- and y-direction
$w_{T,x}, w_{T,y}$	Cost weights on control horizons in x- and y-direction
ω_u	Order of nonlinear system input time derivatives
ω_z	Order of system ouput time derivatives
x	Abscissa
χ	Maximum number of breakpoints along a breakpoint sequence
y	Ordinate
ψ_{ov}	Symmetric bound on other vehicle heading angle

$z_{y,\mathrm{l}}, z_{y,\mathrm{r}}$	Left and right lane bounds' y-position in Frénet frame
$\bar{z}_y$	Symmetric bound on nonlinear system output in y-direction
$\breve{z}$	Dimensionality of nonlinear system output space
$\check{\dot{z}}_x, \hat{\dot{z}}_x$	Lower and upper bounds on nonlinear system output time derivative in x-direction
$\bar{\dot{z}}_y$	Symmetric bound on nonlinear system output time derivative in y-direction
$\tilde{\dot{\mathbf{z}}}_x$	x-velocity sample

Vector Valued Variables

$\underline{A}_{\mathbf{z}}$	Transformed system state matrix
${}_{j,n}B_x, {}_{j,n}B_y$	Basis spline evaluations along segment between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$ in x- and y-direction
${}^{j}_{j,n}B_x, {}^{j}_{j,n}B_y$	Basis spline evaluations at current breakpoint $\tilde{\mathbf{k}}_{1,x,j}$ along the segment between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$ in x- and y-direction
${}^{n}_{j,n}B_x, {}^{n}_{j,n}B_y$	Basis spline evaluations at current breakpoint $\tilde{\mathbf{k}}_{1,x,n}$ along the segment between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$ in x- and y-direction
$\underline{B}_{\mathbf{v}}$	Transformed system input matrix
${}^{\triangle}\underline{B}_{\square}$	Basis spline evaluations $\underline{b}_{\square}(\underline{\tau}_{\square})$
$\underline{\mathbf{c}}_{\square}$	Coefficients of spline $s_{\square}(\cdot)$
${}_{j,n}\tilde{\underline{\mathbf{c}}}_x, {}_{j,n}\tilde{\underline{\mathbf{c}}}_y$	Spline coefficients between nodes $\underline{\mathbf{N}}_j$ to $\underline{\mathbf{N}}_n$ in x- and y-direction
${}_{j}\tilde{\underline{\mathbf{c}}}_x, {}_{j}\tilde{\underline{\mathbf{c}}}_y$	Coefficients of node $\underline{\mathbf{N}}_j$ in x- and y-direction
$\tilde{\underline{\mathbf{c}}}_x, \tilde{\underline{\mathbf{c}}}_y$	Spline coefficients of current edge in x- and y-direction
$\underline{\xi}_0$	Initial nonlinear system state
$\underline{\xi}'$	Nonlinear system state at next time step
$\Delta\underline{\mathbf{k}}$	Interval between the nodes' current and next breakpoints
$\underline{k}_{\square}$	Knots of spline $s_{\square}(\cdot)$
$\tilde{\underline{k}}_{\square}$	Knot averages of spline $s_{\square}(\cdot)$
$\underline{k}_{\square\triangle}$	Combined knots of splines $s_{\square}(\cdot)$ and $s_{\triangle}(\cdot)$
$\underline{\mathbf{k}}_{0H}$	Single segment between initial breakpoint and prediction horizon
${}_{j}\underline{\mathbf{k}}_{\square}$	Breakpoints of spline ${}_{j}s_{\square}(\cdot)$ at time step t_j
${}_{j}\underline{\mathbf{k}}_{xy}$	Combined breakpoints of splines ${}_{j}s_x(\cdot)$ and ${}_{j}s_y(\cdot)$ at time step t_j
$\underline{\mathbf{k}}_{\square}$	Breakpoints of spline $s_{\square}(\cdot)$
$\underline{\mathbf{k}}_{\square\triangle}$	Combined breakpoints of splines $s_{\square}(\cdot)$ and $s_{\triangle}(\cdot)$
$\tilde{\underline{\mathbf{k}}}_{x,j}, \tilde{\underline{\mathbf{k}}}_{y,j}$	Breakpoint sequences at node $\underline{\mathbf{N}}_j$ in x- and y-direction
$\underline{\Lambda}_{T_x}$	Lagrange multipliers terminal manifold in x-direction
$\underline{\mathbf{N}}$	Graph nodes
$\underline{o}$	Optimization variables
$\underline{p}$	Nodes' predecessor indices
$\underline{q}_x, \underline{q}_y$	Polynomial coefficients in x- and y-direction
$\underline{t}$	Monotonously increasing time sequence
$\underline{T}^{\underline{b}_{\square}}_{\underline{b}_{\triangle}}$	Coefficient transformation from basis of spline $s_{\square}(\cdot)$ to basis of spline $s_{\triangle}(\cdot)$

${}_{j,n}\tilde{\underline{T}}^{\underline{b}_\square}_{\underline{b}_\triangle}$	Coefficient transformation from basis of spline $s_\square(\cdot)$ to basis of spline $s_\triangle(\cdot)$ between nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$
$\underline{\tau}_\square$	Monotonously increasing time sequence along spline $s_\square(\cdot)$
$\underline{V}_{\mathbf{N}}$	Costs at nodes $\underline{\mathbf{N}}$
${}_j\underline{w}_\square$	Continuities along breakpoints of spline ${}_j s_\square(\cdot)$ at current time step t_j
$\underline{w}_\square$	Continuities at breakpoints of spline $s_\square(\cdot)$
$\underline{w}_{\square\triangle}$	Combined continuities of splines $s_\square(\cdot)$ and $s_\triangle(\cdot)$
$\underline{\chi}_x, \underline{\chi}_y$	Nodes' placements along the breakpoint sequence of a candidate spline in x- and y-direction
$\underline{\mathbf{z}}_{x,0}, \underline{\mathbf{z}}_{y,0}$	Transformed system state initial boundary values in x- and y-direction
$\underline{\mathbf{z}}_{x,\mathrm{T}}, \underline{\mathbf{z}}_{y,\mathrm{T}}$	Transformed system state terminal boundary values in x- and y-direction
$\tilde{\underline{\mathbf{z}}}_j$	Transformed state prediction beyond the prediction horizon
$\tilde{\underline{\mathbf{z}}}_{m,j}$	Other vehicle prediction beyond the prediction horizon
$\tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{\underline{\mathbf{z}}}_{y,j}$	States corresponding to node $\underline{\mathbf{N}}_j$ in x- and y-direction

Scalar Valued Functions

$a_{\mathrm{idm},m}(\cdot)$	Acceleration of the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles according to intelligent driver model
$a_x(\cdot), a_y(\cdot)$	Vehicle accelerations in x- and y-direction
$\tilde{a}_x(\cdot), \tilde{a}_y(\cdot)$	Accelerations from constant velocity in Frénet frame in x- and y-direction
$\alpha_{\mathrm{f}}(\cdot), \alpha_{\mathrm{r}}(\cdot)$	Front and rear tire slip angles
$\alpha_j(\cdot)$	Monotonously increasing functions $j = 1, 2, 3, 4$
$\beta_1(\cdot)$	Positive definite function
$\beta_{\mathrm{v}}(\cdot)$	Sideslip angle
$\xi_v(\cdot)$	Nonlinear system velocity
$\xi_x(\cdot), \xi_y(\cdot)$	Nonlinear system positions in x- and y-direction
$\xi_\psi(\cdot)$	Nonlinear system yaw angle
$d_m(\cdot)$	y-position of $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' associated lane center in Frénet frame
$\delta_{\mathrm{v}}(\cdot)$	Steering angle
${}_\epsilon\Delta_{x,m}(\cdot), {}_\epsilon\Delta_{y,m}(\cdot)$	Prediction uncertainty semi-axis lengths for $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles in x- and y-direction
$\Delta z_{x,m}$	Gap between the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles and their leading vehicle
$\Delta\tilde{z}_{x,m}$	Desired gap between the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles and their leading vehicle
$\tilde{\Delta}_{x,m}(\cdot), \tilde{\Delta}_{y,m}(\cdot)$	Inverse squared sum of semi-axis lengths for $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles in x- and y-direction
$F(\cdot)$	Terminal cost
${}_{\mathrm{c}}F_x(\cdot), {}_{\mathrm{c}}F_y(\cdot)$	Forces at center of gravity in x- and y-direction
${}_{\mathrm{f}}F_x(\cdot), {}_{\mathrm{f}}F_y(\cdot)$	Forces at front tire in x- and y-direction

${}_{\mathrm{r}}F_x(\cdot), {}_{\mathrm{r}}F_y(\cdot)$	Forces at rear tire in x- and y-direction
$\tilde{F}(\cdot)$	Transformed terminal cost
$\tilde{F}_x(\cdot), \tilde{F}_y(\cdot)$	Transformed terminal cost in x- and y-direction
$\gamma^*(\cdot)$	Arc length along reference path at closest point to vehicle
$\kappa_{\mathrm{p}}(\cdot)$	Reference path curvature
$\kappa_{\mathrm{v}}(\cdot)$	Trajectory curvature
$l(\cdot)$	Running cost
$\tilde{l}_x(\cdot), \tilde{l}_y(\cdot)$	Transformed running costs in x- and y-direction
$L_x(\cdot), L_y(\cdot)$	Lagrange functions in x- and y-direction
$\tilde{\lambda}(\cdot)$	Current cost descent factor
$p_4(\cdot), p_5(\cdot)$	Polynomials of degree four and five
${}_{\mathrm{v}}p_x(\cdot), {}_{\mathrm{v}}p_y(\cdot)$	Vehicle positions in x- and y-direction
$r_v(\cdot)$	Ratio between nonlinear system output time derivatives $\dot{z}_x(t)$ and $\dot{z}_y(t)$
$s_{\check{a},x,\mathrm{n}}(\cdot), s_{\check{a},y,\mathrm{n}}(\cdot)$	Splines for acceleration lower bound in x- and y-direction negative case
$s_{\check{a},x,\mathrm{p}}(\cdot), s_{\check{a},y,\mathrm{p}}(\cdot)$	Splines for acceleration lower bound in x- and y-direction positive case
$s_{\hat{a},x,\mathrm{n}}(\cdot), s_{\hat{a},y,\mathrm{n}}(\cdot)$	Splines for acceleration upper bound in x- and y-direction negative case
$s_{\hat{a},x,\mathrm{p}}(\cdot), s_{\hat{a},y,\mathrm{p}}(\cdot)$	Splines for acceleration upper bound in x- and y-direction positive case
$s_{\hat{a},\tilde{y}}(\cdot), s_{\check{a},\tilde{y}}(\cdot)$	Intermediate splines for acceleration lower and upper bounds in y-direction
$s_{\Delta x^2,m}(\cdot), s_{\Delta y^2,m}(\cdot)$	Splines for squared differences with $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' positions in x- and y-direction
$s_{\Delta x,m}(\cdot), s_{\Delta y,m}(\cdot)$	Splines for differences with $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' positions in x- and y-direction
$s_{\tilde{\Delta},x,m}(\cdot)$	Splines for squared sum of semi-axis lengths of $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles in x- and y-direction
$s_{\tilde{\Delta},xy,m}(\cdot)$	Splines for product of squared semi-axis lengths of $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles
$s_{\tilde{\epsilon},x,m}(\cdot)$	Splines for the root of squared sums of semi-axis lengths in x-direction of $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles
$s_{\mathrm{I},\dddot{x}^2}(\cdot), s_{\mathrm{I},\dddot{y}^2}(\cdot)$	Splines for squared jerk integral in x- and y-direction
${}_j s_x(\cdot), {}_j s_y(\cdot)$	Splines for position at time step t_j in x- and y-direction
$s_{\mathrm{ov,f}}(\cdot), s_{\mathrm{ov,r}}(\cdot)$	Splines for front and rear vehicle collision constraint along terminal segment
$s_{\mathrm{ov},m}(\cdot)$	Spline for collision constraints for $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles
$s_{\check{\psi}}(\cdot), s_{\hat{\psi}}(\cdot)$	Splines for heading angle lower and upper bound constraints
$s_{\rho_\square,\underline{w}_\square}(\cdot)$	Spline of order $\rho_\square$ and with continuities $\underline{w}_\square$
$s_{\dddot{x}^2}(\cdot), s_{\dddot{y}^2}(\cdot)$	Splines for squared jerk in x- and y-direction
$s_{x,m}(\cdot), s_{y,m}(\cdot)$	Splines for $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' x- and y-positions
$s_{\tilde{x},m}(\cdot), s_{\tilde{y},m}(\cdot)$	Splines for $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ leading vehicles reference positions in x- and y-direction

$s_{xy,m}(\cdot)$	Splines for sum of squared and scaled position differences with $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles
$s_{\dot{x},m}(\cdot), s_{\dot{y},m}(\cdot)$	Splines for squared and scaled position differences with $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles in x- and y-direction
$s_{x^{(j)}}(\cdot), s_{y^{(j)}}(\cdot)$	Splines for order $j = 1, 2, 3$ time derivatives in x- and y-direction
$s_{\square}(\cdot)$	Shorthand for spline $s_{\rho_{\square}, \underline{w}_{\square}}(\cdot)$
$u_a(\cdot)$	Nonlinear system acceleration input
$u_{\delta_{\text{v}}}(\cdot)$	Nonlinear system steering input
$v_{\text{f}}(\cdot), v_{\text{r}}(\cdot)$	Front and rear tire velocities
$v_{\text{v}}(\cdot)$	Vehicle velocity
$\mathbf{v}_x(\cdot), \mathbf{v}_y(\cdot)$	Directional transformed system input trajectories in x- and y-direction
$V(\cdot)$	Minimium cost of optimal control problem
$\tilde{V}_3^*(\cdot)$	Cost between local targets assuming three breakpoints
$\hat{\tilde{V}}_x(\cdot), \hat{\tilde{V}}_y(\cdot)$	Directional costs upper bounds between local targets in x- and y-direction
${}_{\text{p}}\psi(\cdot)$	Reference path yaw angle
${}_{\text{v}}\psi(\cdot)$	Vehicle yaw angle
$z_x(\cdot), z_y(\cdot)$	Nonlinear system outputs in x- and y-direction
$z_{x,m}(\cdot), z_{y,m}(\cdot)$	Nonlinear system outputs of $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles in x- and y-direction
$z_{\tilde{x},m}(\cdot)$	Reference output trajectories of $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles

Vector Valued Functions

$\underline{b}_{\rho_{\square}, \underline{k}_{\square}}(\cdot)$	Basis spline of order $\rho_{\square}$ and with knots $\underline{k}_{\square}$
$\underline{b}_{\square}(\cdot)$	Shorthand for basis spline $\underline{b}_{\rho_{\square}, \underline{k}_{\square}}(\cdot)$
$\underline{c}_j(\cdot)$	Control law at time step t_j
$\underline{\xi}(\cdot)$	Nonlinear system state
$\underline{\xi}_c(\cdot)$	Closed-loop nonlinear system state
$\underline{\xi}_j^*(\cdot)$	Optimal open-loop nonlinear system state at time step t_j
$\underline{\xi}^*(\cdot)$	Optimal open-loop nonlinear system state
$\underline{f}_{\xi}(\cdot)$	Nonlinear system vectorfield
$\underline{f}_z(\cdot)$	Nonlinear system output vectorfield
$\underline{\Phi}_u(\cdot)$	Mapping from flat output time derivatives to nonlinear system input
$\underline{\Phi}_{\xi}(\cdot)$	Mapping from flat output time derivatives to nonlinear system state
$\underline{\Phi}_z(\cdot)$	Mapping from nonlinear system state and input to flat output
$\underline{g}_{\text{b}}(\cdot)$	Equality constraint functions for coefficient transformation
$\underline{g}_{\text{T},x,m}(\cdot), \underline{g}_{\text{T},y,m}(\cdot)$	Terminal equality constraints for $m = -2, -1, \ldots, \eta_{\text{ov}}$ local targets in x- and y-direction
$\underline{g}_{x^{(j)}}(\cdot), \underline{g}_{y^{(j)}}(\cdot)$	Equality constraint functions for $j = 1, 2, 3$ derivative splines in x- and y-direction
$\underline{h}(\cdot)$	Inequality constraints nonlinear program
$\underline{h}_{\xi}(\cdot)$	Inequality constraints nonlinear system optimal control problem

$\underline{\Lambda}_x(\cdot), \underline{\Lambda}_y(\cdot)$	Lagrange multipliers in x- and y-direction
${}_{\mathrm{p}}\underline{\mathbf{n}}(\cdot)$	Reference path normal vector
${}_{\mathrm{p}}\underline{p}(\cdot)$	Reference path position
${}_{\mathrm{v}}\underline{p}(\cdot)$	Vehicle position
$\underline{\Psi}(\cdot)$	Diffeomorphism
$\underline{s}_x(\cdot), \underline{s}_y(\cdot)$	Splines for transformed system state in x- and y-direction
${}_{\mathrm{p}}\underline{\mathbf{t}}(\cdot)$	Reference path tangent vector
$\underline{u}(\cdot)$	Nonlinear system input
$\underline{u}_c(\cdot)$	Closed-loop nonlinear system input
$\underline{u}^*_{c,j}(\cdot)$	Optimal open-loop nonlinear system input at time step t_j
$\underline{u}^*(\cdot)$	Optimal open-loop nonlinear sytem input
$\underline{\mathbf{v}}(\cdot)$	Transformed system input
$\underline{z}(\cdot)$	Nonlinear system output
$\underline{\mathbf{z}}(\cdot)$	Transformed system state
$\underline{\mathbf{z}}_m(\cdot)$	Transformed system state of the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles
$\underline{\mathbf{z}}_x(\cdot), \underline{\mathbf{z}}_y(\cdot)$	Transformed system states in x- and y-direction

Acronyms

B-spline	**B**asis **spline**
BABRiGeschwV	**B**undes**A**uto**B**ahn-**Ri**cht**Geschw**indigkeits-**V**erordnung
BKatV	**B**ußgeld**Kat**alog-**V**erordnung
BVP	**B**oundary **V**alue **P**roblem
COG	**C**enter **O**f **G**ravity
DP	**D**ynamic **P**rogramming
EV	**E**go **V**ehicle
GP	**G**lobal **P**lanner
GT	**G**lobal **T**arget
HJB	**H**amilton-**J**acobi-**B**ellman
IDM	**I**ntelligent **D**river **M**odel
IPOPT	**I**nterior **P**oint **Opt**imizer
LP	**L**ocal **P**lanner
LT	**L**ocal **T**arget
MINVO	**MIN**imum **VO**lume
MIQP	**M**ixed-**I**nteger **Q**uadratic **P**rogram
MPC	**M**odel **P**redictive **C**ontrol
MUMPS	**MU**ltifrontal **M**assively **P**arallel **S**olver
NLP	**N**on**L**inear **P**rogram
OCP	**O**ptimal **C**ontrol **P**roblem
ODD	**O**perational **D**esign **D**omain
QCQP	**Q**uadratically **C**onstrained **Q**uadratic **P**rogram
QP	**Q**uadratic **P**rogram
RRT	**R**apidly exploring **R**andom **T**ree
SIP	**S**emi-**I**nfinite **P**rogram
SOS	**S**ums-**O**f-**S**quares
SQP	**S**equential **Q**uadratic **P**rogram

StVO	**St**raßen**V**erkehrs-**O**rdnung
UAV	**U**nmanned **A**erial **V**ehicle

Abbreviations

abs.	absolute
acc.	acceptable
config.	configuration
constr.	constraint
cont.	continuous
eq.	equation
reach.	reachability
rel.	relative
scn.	scenario
tol.	tolerance

1

Introduction to Optimal Control for Automated Driving

1.1. Motivation

The possibility of automating the driving task promises to improve the issues faced in today's traffic. In Germany, 291 890 personal injuries have been reported in road traffic in 2023 [Fed24]. Another issue is the time lost due to traffic congestion on highways and in cities. The highest time lost per driver in traffic congestions, with 148 h on average in 2021, is observed in London [Pis21]. Shared automated vehicles are expected to reduce the demand for individual mobility [FK14], resulting in lower traffic densities and air pollution. Furthermore, shared vehicles require less parking space, potentially reducing the issue of cruising for parking [Sho06].

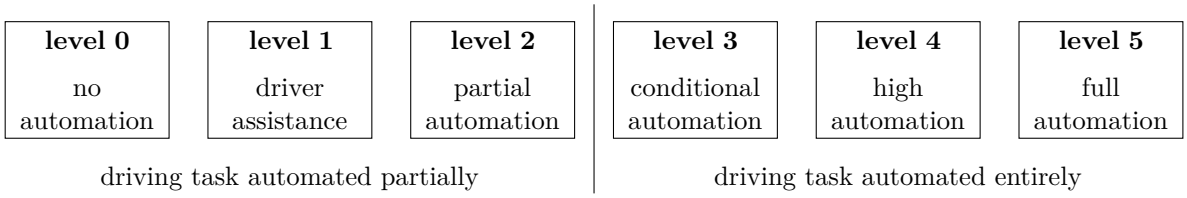


Figure 1.1.: Overview on automated driving levels [SAE21].

The degree of automation is categorized into six levels [SAE21], shown in figure 1.1. The adaptive cruise control and the lane-keeping assistant are examples of level 1 automations. When engaged in combination, they may fall into level 2. Also, parking assistants can perform the control task in longitudinal and lateral directions, enabling the vehicle to drive into a parking spot. However, the driver still has to monitor level 1 and 2 automations. Starting at level 3, the driver is relieved from the task for a limited time. A certified level 3 automation is available to consumers already [Mer23]. A level 4 automation operates in an operational design domain (ODD) without driver intervention. Serveral companies, like Waymo [Lie25] and Cruise [Bel21], are offering level 4 automated taxi services, providing evidence of the technology's applicability on a large scale. Because of the large scale, incidents involving the automated vehicles occur in public [She24]. However, companies only grant limited access to performance data and technology, so the limitations cannot

be fully assessed independently. Thus, research addressing open questions in the field of automated driving is still required at public institutions, like universities. A level 5 automation has to perform the driving task safely in any ODD without any driver intervention, which has yet to be achieved.

1.2. Automated Driving as Optimal Control Problem

The problem of automated driving can be understood as a control problem. Based on the estimated environment state and the target, the steering, throttle, and brake commands are determined repeatedly to stabilize the automated vehicle in the given target. Therefore, it is necessary to know the road network, stay on the road, and avoid collisions with vehicles and obstacles. Simultaneously, traffic rules have to be considered while implementing a compromise between progress toward the target and the passengers' comfort perception. An optimal control problem (OCP), like problem 1.2.1, is one way to formalize the control problem.

Problem 1.2.1 (Optimal control problem)**:** The OCP seeks a function of the control inputs $\underline{u} : \mathbb{R} \mapsto \mathcal{U}$ in the control input space $\mathcal{U}$ and the vehicle state $\underline{\xi} : \mathbb{R} \mapsto \mathcal{C}$ in the state space $\mathcal{C}$ as solutions to

$$\min_{\underline{\xi}(t),\underline{u}(t),T} F\Big(\underline{\xi}(T)\Big) + \int_0^T l\Big(\underline{\xi}(\tau), \underline{u}(\tau)\Big)\, \mathrm{d}\tau, \tag{1.2.1}$$

subject to

$$\underline{h}_{\xi}\Big(\underline{\xi}(t), \underline{u}(t)\Big) \geq 0, \quad \dot{\underline{\xi}}(t) = \underline{f}_{\xi}\Big(\underline{\xi}(t), \underline{u}(t)\Big), \quad t \in [0, T], \tag{1.2.2}$$
$$\underline{\xi}(T) \in \mathcal{T}, \quad \underline{\xi}(0) = \underline{\xi}_0.$$

The constraints (1.2.2) usually consider a model of the control system, which is frequently described by an ordinary nonlinear and continuous differential equation in the automated driving context. The function $\underline{f}_{\xi} : \mathcal{C} \times \mathcal{U} \mapsto \mathcal{C}$ maps the state and input to the state's time derivative. In addition, the constraints consider the current vehicle state $\underline{\xi}_0 \in \mathcal{C}$ and ensure the state at the trajectory end is located in the compact terminal set $\mathcal{T} \subset \mathcal{C}$. The terminal set also includes the vehicle's target. The inequality constraint function $\underline{h}_{\xi} : \mathcal{C} \times \mathcal{U} \mapsto \mathbb{R}^{\breve{h}}$ with $\breve{h} \in \mathbb{N}_0$ constraints keeps the trajectory collision-free, enforces limitations on the control input and states, and fulfills further design-specific criteria. The cost functional comprises the continuous running cost $l : \mathcal{C} \times \mathcal{U} \mapsto \mathbb{R}^+$ and the continuous terminal cost $F : \mathcal{C} \mapsto \mathbb{R}^+$. The running cost implements the passengers' preferences for comfort and progress. The terminal cost guides the control algorithm through the terminal set to the target by providing a cost-to-go estimate beyond the control horizon $T \in \mathbb{R}^+$. The shapes of other vehicles and obstacles considered in the constraints separate the free space, so multiple options exist to traverse the state space toward the destination. The options introduce a combinatorial aspect to the control problem. A controller that solves the OCP in each time step, allowing a reaction to environmental changes, is called model predictive control (MPC).
Due to its complexity, the control problem is usually simplified by decomposing it into subproblems. The subproblems are solved by specialized algorithms. An unidirectional

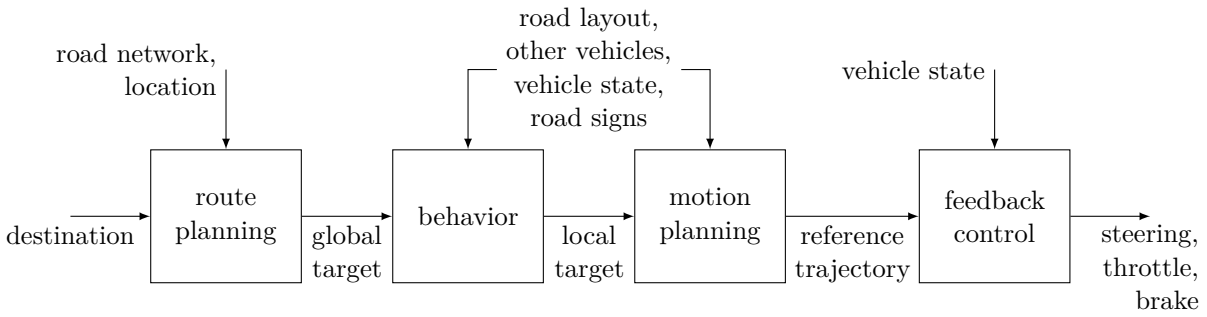


Figure 1.2.: Hierarchy for automated vehicle control.

information flow results in a modular, hierarchical algorithm structure. Figure 1.2 shows a possible structure of the control hierarchy, based on [Pad+16]. Each module acquires a less abstract representation of the automated vehicle's next action from the left to the right. The route planning module determines a path through the road network, passing targets along the path sequentially to the behavior module. The targets are called global targets (GTs). The behavior module's task is to specify a maneuver, encoded as an local target (LT). A sequence of LTs guides the vehicle between steady behaviors toward the current GT. Because the GT is also an LT, the GT resembles the last element along the sequence. The targets are composed of a lateral and a longitudinal component. The lateral component specifies a lane to drive in. The longitudinal one specifies the velocity to drive or the vehicle to follow. The motion planning module determines the concrete execution of the maneuver in terms of the future trajectory. Finally, the feedback control module provides control inputs, so the automated vehicle tracks the given solution trajectory. In this work, the automated vehicle controlled by the developed MPC is referred to as the ego vehicle (EV).
The requirements each module has to fulfill, as well as their interfaces, are specific to the control hierarchy implementation. Thus, the hierarchical framework should be understood as a guideline to structure the overall control problem. While the modules operate with a top-down information flow, each one has to be designed considering the requirements and limitations of the subsequent module. The behavior module should not command a maneuver the motion planning module cannot execute. Conversely, the behavior module should not reject a maneuver in the motion planner's capabilities.

1.3. Research Questions and Contributions

This work aims to design and implement an MPC for solving the tasks of the behavior and motion planning modules formalized in an OCP. The problems considered in the MPC design are condensed into the following overarching research questions:

1. How can an OCP be designed to consider the objectives and rules of automated highway driving while stabilizing the EV in a target not reachable within the planning horizon?

2. How can an OCP that includes discrete and continuous decision variables be solved efficiently?

3. How does the control performance scale with the algorithm complexity?

In answering the research questions, this work describes contributions in the following areas:

Spline-Based problem formulation

The solution trajectory and all inequality constraint functions are parameterized by polynomial splines in basis form. Since this work only refers to polynomial splines, such functions are called splines for brevity. The parameterization leverages the differential flatness property of a dynamic vehicle model to ensure a dynamically feasible trajectory while avoiding errors due to numerical integration. Since the splines always stay in the convex hull spanned by their control points, the constraints are fulfilled at all times if the splines' control points are feasible. However, the spline parameterization limits the OCP to a polynomial constraint and cost formulation in dependence on the trajectory. This work proposes a spline-based formulation of the OCP, considering the objectives and constraints required for automated driving in normal highway scenarios. The trajectory is described in a road-aligned reference frame, introducing nonpolynomial functions to the constraint formulation. A simplification of the constraints to polynomial functions is proposed while preserving feasibility.

Stabilizing terminal set and cost

The GT might not be reachable in the control horizon if other vehicles are present. A disconnected terminal set is designed to satisfy the need for progress and comfort equally under the given environment restrictions. Each subset resembles an LT in case the current GT is out of reach. The terminal cost is designed under simplifying assumptions to guide the decision on the current LT to progress toward the GT.

Recursive feasibility

The solution trajectory is planned in a shrinking horizon manner. It is truncated at each time step to preserve the remaining part of the previous solution trajectory using a nonuniform spline parameterization. Thus, the MPC is guaranteed to return a feasible trajectory under nominal conditions if a feasible solution exists at the initial time step. Still, the algorithm can improve the trajectory if a better solution exists. In addition, a strategy for removing the splines' breakpoints is proposed accounting for a minimum distance between breakpoints.

Hierarchical combination

The MPC has to efficiently solve an OCP for discrete and continuous decision variables while achieving a high control performance. Therefore, two solution algorithms are implemented and combined hierarchically. At the first stage of the hierarchy, the global

planner (GP) decomposes the state space and time into a discrete set. Subsequently, the solution trajectory and the LT are found in a graph of candidate trajectories. The GP can efficiently solve the combinatorial task. However, the GP's computational complexity is only tractable with a low-density discretization of a low-dimensional state space. In the second stage, the local planner (LP) provides a more accurate solution to a given LT. The LP solves a nonlinear program (NLP) that is subject to the same cost and constraints as the GP. Therefore, the GP can always provide a feasible initial guess to the NLP, allowing an efficient and reliable optimization. A solution trajectory selection combines the advantages of both stages by assessing the solution trajectories' cost and feasibility. The selection maintains a feasible solution even if one of the stages fails to find a feasible trajectory.

1.4. Outline

The following outline provides an overview of the individual chapters and their contributions to this work:

Chapter 2 provides an overview of the literature addressing the problem of trajectory planning for automated highway driving. However, the mentioned literature does not consider the semi-infinite constraints arising in a continuous-time OCP. Solution methods to semi-infinite programs (SIPs) are introduced, focussing on works utilizing the convex hull property of a spline trajectory parameterization and its application to robotics and automated driving.

Chapter 3 introduces the simulation environment and the coordinate frames used in the subsequent chapters. It also provides the required background on system stability. The notion of differential flatness is introduced and used to invert a dynamic vehicle model. Afterward, an introduction to splines in basis form is provided. The splines enable a feasible solution to SIPs with a finite constraint set.

Chapter 4 transforms the general OCP into a SIP, describing the optimal EV trajectory. Initially, the development criteria for the trajectory planning algorithm are introduced. The criteria include driving objectives, stability conditions, and complexity requirements. The general OCP is concretized by defining the terminal set, the running cost, and the incorporation of other vehicles. The trajectories in the OCP are parameterized with splines, allowing it to be transformed into a SIP.

Chapter 5 defines the semi-infinite state and input inequality constraints in the SIP. Nonlinear equations due to the vehicle model are conservatively approximated by polynomial functions, aiming at a spline-based constraint parameterization. The approximation compromises the feasible space and the complexity of the constraint function.

Chapter 6 describes the numerical optimization-based LP and the transformation of the SIP into an NLP. The state and input constraints are parameterized with splines that

are coupled with additional equality constraints. A breakpoint adaption strategy in a shrinking horizon control scheme is introduced to ensure the planner's recursive feasibility during the convergence into the current LT. Finally, the chapter evaluates the LP's control performance and stability properties in two highway scenarios.

Chapter 7 describes the sampling-based GP, which decomposes the state space and the time into discrete sets. The sets implicitly define a graph of candidate trajectories, which are explored by a shortest-path search algorithm considering the same cost and constraints as the LP. The graph search enables the selection of the current LT by integrating the guiding terminal cost into the node expansion process.

Chapter 8 introduces the terminal cost using simplifying assumptions on the future motion of the other vehicles to converge to the GT likely. The cost definition is based on stabilizing control sequences, determined for four scenes that terminate in the GT. The GP's ability to select reasonable LTs and its complexity and performance are analyzed in open-loop control on several highway scenario variations.

Chapter 9 combines the LP and the GP in a two-stage algorithm. The LP's recursive feasibility is retained, and the GP's solution trajectory and LT are leveraged for initialization. The two-stage algorithm's control performance, system stability, and computational complexity are analyzed in closed-loop control in several highway scenario variations. Based on the results, the algorithm's compliance with the development criteria is discussed.

Chapter 10 concludes this work with a summary of the concepts and results presented in the previous chapters and suggests potential future works.

2

Related Work

This work formalizes the tasks of the behavioral layer and the motion planner as OCPs, while other works take different approaches. Some authors propose defining the behavioral layer with hand-crafted rules. Rules have the advantage of allowing the direct implementation of the expert's desired behavior. The rules are implemented in a state machine [Zie+14b] or in dependence on the vehicle state [Nil+16]. Still, the expert might be uncertain about the desired behavior and its quantification. Thus, learning the rules from demonstrations by approximating the solution to a classification problem [Val+17] can be more intuitive.

The task of the motion planner can also be understood as a classification task. Do et al. [Do+13] select spline control points from obstacle boundaries of maximum distance from a separating plane. Alternatively, a regression problem is solved to approximate human demonstrations. Therefore, the parameters of a function are determined that maps from the vehicle environment to the output trajectory [Ban+19].

The mentioned approaches may be intuitive when formalizing the driving task. However, a generalization beyond the considered scenarios that ensures safety at sufficient performance is challenging. Thus, several contributions in the behavior and motion planning literature propose algorithms to solve an OCP formalization. The first section 2.1 provides an overview of the related work proposing OCPs and online solution algorithms in the context of automated driving. In line with this work's contributions, the focus is placed on contributions in the used solution algorithm and the integration of the environment model, the vehicle model, and the passenger preferences in the OCP. Conversely, contributions focusing on replicating human driving behavior [Nau+20], distributed control [Xie+22], interaction-aware planning [Sad+16], path planning [Zie+08], and robust or stochastic MPC [Brü+18] are out of scope. Also, contributions generating a solution trajectory or target based on offline approximations to OCP solutions without an additional selection algorithm, like in [Mog+21], are not considered.

Ensuring a collision-free and dynamically feasible solution trajectory is a challenge in MPC design. Such constraints are generally time-continuous, requiring the solution to a SIP. Section 2.2 provides an overview of methods for solving SIPs. A special focus is placed on applications in robotics and automated driving that employ the convex hull property of a spline-parameterized trajectory.

2.1. Trajectory Planning for Automated Driving

Solution methods for the OCP described in problem 1.2.1 are categorized into three groups [Die+06]: direct methods, indirect methods, and dynamic programming (DP).
The direct methods transcribe the OCP into a finite-dimensional optimization problem by discretizing the state and input trajectories over time. A numerical optimization algorithm solves the problem. Established direct methods include direct single shooting, direct multiple shooting, and direct collocation. The resulting optimization problems differ in sparsity, the number of constraints, and optimization variables.
Indirect methods solve the OCP by deriving a boundary value problem (BVP) from the necessary optimality conditions [Föl94]. Pontryagin's maximum principle [Pon+62] allows the incorporation of inequality constraints. While the indirect methods may achieve a more accurate solution than the direct methods, changes in the active constraints require reformulating the BVP. In addition, numerically solving BVPs with strong nonlinear dependencies can be a challenging.
DP relies on a recursive solution of subproblems to the OCP, following the principle of optimality [Bel54]. The Hamilton-Jacobi-Bellman (HJB) equation provides the necessary condition in the continuous-time case, resulting in a partial differential equation. However, the computational complexity of numerical solution algorithms grows exponentially with the state space dimensionality.
The trajectory planning literature for automated driving frequently implements a solution algorithm by combining the three methods. This review categorizes the literature based on the solution algorithms' characteristics into numerical optimization and sampling, similar to the taxonomy proposed in [Pad+16; Gon+16; Cla+20]. The additional category hierarchical algorithms combines the algorithms from the previous categories hierarchically.

2.1.1. Numerical Optimization

The solution algorithms of the first category employ numerical optimization, which resembles the application of the direct methods. A challenge in automated driving is designing differentiable cost and constraint functions that accurately model vehicle dynamics, objectives, and the environment. The choice of optimization algorithm depends, among other things, on the function types. The increasing function complexity improves accuracy but raises computational effort.
Nonlinear constraints and cost permit a nonlinear vehicle model, like a four-wheel dynamic model [Car+13] or the dynamic single-track model [Sch+18], which is required at high vehicle speeds and accelerations. Nonlinear functions also enable nonlinear transformations, like the projection of the vehicle configuration on a reference path [WL12]. Instead of a nonlinear differential equation, one can consider a nonlinear model via its differential flatness property, which many vehicle models exhibit. Leveraging the property reduces the number of optimization variables and the effort for numerical integration. A linear model is chosen in [Zie+14a], and the vehicle's nonholonomic properties are enforced with nonlinear constraints. The environment is modeled accurately with polygonal chains. The resulting NLP is generally solved using one of the free available sequential quadratic programming (SQP) [BW12] or interior point implementations [WB05]. Some works pro-

pose an optimization algorithm tailored to their problem, like an adapted SQP algorithm [Car+13].
The previously mentioned works consider strictly enforced hard constraints. Alternatively, constraints can be considered soft by introducing them as penalty terms into the cost function. Thus, they are not guaranteed to hold but allow solving the NLP efficiently. Also, they cannot fail to converge due to an infeasible problem formulation. Some works formulate an unconstrained NLP by encoding the environment model as a potential field, which is integrated with a dynamic double-track model into the cost function [Göt+15]. Other works [Kel+14; Ulb+17] assume a differentially flat vehicle model, similar to [Zie+14a]. The potential field is also combined with a spline parameterization to reduce the number of optimization variables [Göt+17].
Although NLPs can be solved efficiently they might be too computationally expensive. Alternatively, the less complex quadratic program (QP) uses a quadratic cost function and linear constraints. One possibility for an environment design is the direct formulation with linear constraints [Nil+15]. Planning in the Frénet frame [Ple17] reduces the environmental complexity on curved roads. Another approach decomposes the motion into two sequentially solved QPs [Nil+17]. Successive linearization utilizes the optimal trajectory from the previous time step to locally linearizing nonlinear constraints [Yi+16; Don+20].
All the mentioned works find the local optimal solution with gradient-based optimization algorithms. However, the discrete decisions necessary for driving introduce multiple locally optimal solutions. Integer variables can encode the decisions to activate piecewise linear constraints in a mixed-integer quadratic program (MIQP). The integer variables decide on which side to overtake other vehicles [Par+15] or to incorporate traffic rules [Qia+16a]. However, the number of possible QPs grows exponentially with the number of integer variables. Consequently, only problems with few integer variables are tractable for online solutions.

2.1.2. Sampling

Algorithms from the second category sample states or inputs from probability density functions. A graph of trajectory segments is constructed to find the optimal solution with a shortest-path search algorithm. While the solution is the global optimal one among the candidates, the computational effort scales poorly with state space dimensionality. The following sections distinguish between graph search and incremental search approaches.

Graph Search

The graph search approaches draw a finite set of samples, resembling nodes in a graph of candidate trajectories. In the most basic form, the initial state is connected to a set of terminal states, resulting in a graph of depth one. Some works discretize the input acceleration and steering angle to form tentacle-shaped trajectories composed of circular paths and linear velocities [Hun+08]. Others propose discretizing the average tire slip and the steering angle to generate trajectory candidates by simulating a dynamic vehicle model [Kel+15]. If the state space is discretized, the trajectory candidates can be steered toward specific regions in the state space. Werling et al. [Wer+12] connect

the vehicle state to a discretized terminal manifold in the Frénet frame with polynomials of degree five. The polynomials are determined efficiently by solving a linear equation system and are optimal in a squared-jerk manner [Tak+89]. The latter fact ensures time-consistent planning results, while terminal cost supports convergence into a given target. Rathgeber et al. [Rat+15] use polynomials of degree seven similarly. In contrast, the trajectories are evaluated for feasibility in the Frénet frame, assuming a low road curvature. The generation of more complex trajectory parameterizations with additional degrees of freedom involves solving QPs. Multiple QPs can be solved for optimal spline candidates, which are checked for feasibility using the differential flatness of the kinematic bicycle model [Sch+16]. Lateral polynomial candidates are generated by [Lim+21], similar to [Wer+12]. Conditioned on the respective lateral candidates, QPs yield the longitudinal candidates constrained to a longitudinal driving corridor. Several publications solve NLPs to find solution candidates. Multiple collision-free and dynamically feasible driving corridors are identified in [Sch+23]. An NLP is solved for each corridor to find the optimal trajectory. Adajania et al. [Ada+22] simultaneously solve for multiple splines, subject to dynamic and collision constraints via a tailored parallelized optimization algorithm.

Planning algorithms may connect the samples to a more complex graph, frequently resulting in a lattice shape. The most basic shortest-path search explores the graph exhaustively in a breath-first manner. State samples are connected offline by polynomial segments to form a lattice on a reduced state space in [ZS09], which is searched online. In [Lie22], the lateral states form a lattice, while the longitudinal ones form a tree. QPs are solved for spline candidate trajectories, considering the differential flatness of a dynamic vehicle model. The shortest-path search can be focussed on promising parts of the graph via a priority queue of nodes to process. The direct implementation of the DP principle prioritizes the nodes by the cost from the start [Qia+16b]. An additional cost-to-go estimate is leveraged by the A* search algorithm [Har+68] and its variants. Likhachev and Ferguson [LF09] propose an anytime variant of A* to find a feasible solution quickly. The search is guided by two cost-to-go heuristics, which are chosen depending on the obstacle density. A Hybrid A* algorithm [Dol+08] is applied by [Aja+18], guided by a cost-to-go estimate based on the shortest path in a simplified graph.

Incremental Search

Graph search approaches may produce suboptimal solutions due to their resolution completeness. Incremental search algorithms, which are frequently probabilistically complete, refine the graph with new state samples. Thus, they find a solution if one exists, given sufficient time, which is limited in practice. In works for on-road trajectory planning, variants of the rapidly exploring random tree (RRT) algorithm [LaV98] are frequently applied. Contributions in this category focus on integrating the vehicle model and the probability density function design to guide the search in structured environments. Kuwata et al. [Kuw+09] propose the closed-loop RRT. Samples are drawn from Gaussian distributions, adapted to the environment model and the intended maneuver. A linear segment connects the tree's next node to the state sample. The line is turned into a dynamically feasible trajectory by simulating a controlled kinematic vehicle model. L. Ma et al. [Ma+15] extend the ideas, and reduce the running time by connecting the tree directly with the goal

state. In addition, the algorithm is initialized with maneuver-specific offline-generated trees. A different incremental approach is proposed by [Die+23]. They sample actions from a stochastic policy learned on human demonstrations, which are applied to a world model to generate the next state. In addition, the control horizon is extended by a learned terminal cost function.

2.1.3. Hierarchical Algorithms

Combining sampling and numerical optimization algorithms hierarchically takes advantage of their opposite properties. The solution of a planning algorithm is passed to the next one for improvement and to simplify the problem to solve. This section reviews contributions that combine sampling and numerical optimization hierarchically.
Some approaches simplify the trajectory planning task by finding the optimal path first and assigning time information later. A two-step path planning algorithm is used by [Qia+16b]. First, the best piecewise linear collision-free path is found in a lattice. The path initializes a soft-constrained NLP to determine the optimal spline parameters via derivative-free local optimization. Finally, the optimal velocity is determined by solving another NLP subject to dynamic constraints. The path and velocity optimization is also designed hierarchically by [Fan+18]. First, the optimal path is found in a lattice of polynomial path segments via a DP search, which is refined by solving a QP considering a piecewise linear driving corridor. Then, the optimal velocity trajectory is found in a lattice on discrete times and positions, and improved similarly by solving a QP. Luo et al. [Luo+22] decompose the environment into trapezoidal cells. The cell edges form a graph of piecewise linear paths. The optimal path and the corresponding corridor act as an initial guess to a QP to find the optimal spline path. The velocity optimization follows the two-stage approach in [Fan+18]. In contrast, the trajectory is fully spline parameterized in [Luo+22].
While the path-velocity decomposition can achieve low computational complexity, the accuracy of the solution is limited. Therefore, approaches are proposed that efficiently solve for the trajectory in a combined manner. In the first stage proposed in [Lim+18], the optimal sequence of linear path and velocity segments is found in a lattice. The second stage solves an NLP to find the optimal trajectory with additional degrees of freedom in a collision-free corridor. Yi et al. [Yi+19] propose first exploring all homotopy classes in dynamic traffic in the sense of [Ben+15]. All trajectories connecting the same start and end states, which can be continuously deformed into each other, are homotopic and belong to the same homotopy class [Hat01, pp. 25–26]. Then, the optimal trajectory in each homotopy class is obtained with numerical optimization. The optimal trajectory among all homotopy classes is selected. An NLP is solved if a critical situation occurs, considering a dynamic vehicle model and more accurate vehicle shapes. In contrast to other contributions, a selection mechanism is proposed by [KD16] to decide on the initial guess for the second stage. Either the result of the first stage or the previous optimal trajectory is passed to the second stage. The decision is based on the solution trajectory cost in the current scene. The first stage uses the algorithm from [Wer+12], while the second stage employs nonlinear numerical optimization to find the optimal polynomial, considering the same cost and similar constraints. However, the terminal cost is only

designed as a short-term cost estimate beyond the horizon and does not allow far-sighted maneuver decisions. Also, the degrees of freedom are limited to polynomial trajectories. Some contributions propose applying an incremental search algorithm in the first stage. The RRT* algorithm [KF10] is applied to find the optimal path based on the previously optimal one [Le+19] . An NLP is solved to refine the initial path and to obtain the optimal velocity. A suboptimal and collision-free trajectory, which is composed of piecewise linear segments, is found by [Li+20] with RRT*. A QP is solved by considering a collision-free corridor, to achieve a dynamically feasible trajectory.

In summary, many efficient trajectory planning algorithms for automated driving are proposed. The hierarchical combination of algorithms can reduce the computational effort of the distinct algorithms while improving the robustness. The combination of algorithms from different categories aiming at a low computational complexity has received much attention. However, the proposed stages consider different constraints and, with the exception of [KD16], different cost. Thus, the initialization of the second stage is likely suboptimal or even infeasible. Additionally, the potential lack of a solution at any stage and recursive feasibility are not addressed. Maneuver decisions of global optimal solution algorithms are usually based on the running cost, but do not leverage terminal cost. To the best knowledge of the author, no approach guides the maneuver decisions by systematically derived terminal cost in a hierarchical algorithm.

2.2. Semi-Infinite Programming for Spline-Based Trajectory Planning

The majority of the trajectory planning literature for automated driving does not address the continuous-time nature of the constraints in equation (1.2.2). Inequality constraints are enforced at discrete time steps, but the feasibility between them is not guaranteed. The OCP becomes a SIP if it is transcribed to an optimization problem with a finite number of variables, while retaining the continuous-time constraints. However, the problem cannot be solved directly by an optimization algorithm. This section reviews numerical methods for solving SIPs, focusing on their application in optimal control for robotics and automated driving using the spline convex hull property. Reemtsen and Görner [RG98] distinguish between discretization, local reduction, semi-continuous, and hybrid methods. An overview of methods considered in this review is provided in figure 2.1.
The discretization methods solve a sequence of finite-dimensional optimization problems with an evolving time grid. The grids are pre-determined or adapted based on previous solutions, converging to the SIP solution as the grid density increases. Exchange methods [HK93, sec. 7.1] are a type of discretization method where grid points are added and removed in each iteration. Discretization methods typically achieve an approximate solution outside the feasible set with a given accuracy. Also, they are applicable to a wide range of SIPs without the need to modify the directly transcribed optimization problem. However, solving a SIP becomes computationally expensive if a high accuracy is required due to the large constraint set. Two algorithms, [Het86] and [Ree91; Vaz+01], employing the discretization method are applied in [Vaz+04] to control a robotic manipulator in

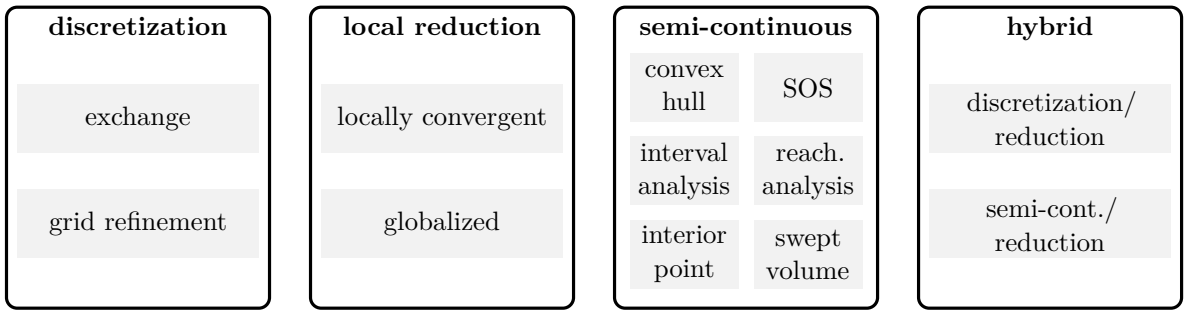


Figure 2.1.: Overview on solution methods for SIPs.

minimum time. Both algorithms are used to solve nonlinear SIPs. The algorithms are based on a sequence of increasingly refined time grids defined a-prior. Semi-infinite linear constraints, resulting from an unmanned aerial vehicle (UAV) flight corridor, are enforced by [Che+16] on an iteratively refined grid. The grid points are located at the constraints' extrema. In this case, a zero constraint violation is achieved for a sufficiently dense grid.

The local reduction methods rely on the principle of local reduction [HJ78]. The principle states the necessary conditions for locally approximating the SIP active constraint set with a finite set of constraints. The finite set is given by the global minimizers of the semi-infinite inequality constraints with respect to time. The solution to the reduced problem coincides with the solution to the SIP. While the local reduction methods can achieve a more accurate solution than the discretization methods, solving for the global minimizers of the constraint functions may be intractable. One can distinguish between locally convergent and globalized methods. Locally convergent methods require an initial guess near a SIP local minimizer, so local optimization is sufficient for convergence. New constraint evaluation points are determined in each iteration after solving the reduced problem. Globalized methods determine a new reduced problem in each iteration after searching for all local constraint minima. The current global constraint maximizers are adapted locally in an inner loop while solving the reduced problem. The work [Due+17] implements the globalized reduction to plan a collision-free trajectory among circular obstacles. Polynomials approximate the constraints on a finite set of time intervals.

The semi-continuous methods include all methods that neither rely on local reduction nor on a time grid. Generally, they achieve an approximate solution inside the feasible set by solving a time integral or bounding the constraint minimizers. Sums-of-squares (SOS) decomposition is applied to enforce the positivity of linear constraints that bound a polynomial trajectory and its time derivatives [HL06]. The decomposition results in a semi-definite program subject to linear matrix inequalities. The approach is generalized to splines by [Lou+10]. Interior point methods introduce the time integral over a barrier function into the cost, which diverges as the constraints become violated. This approach is applied in [Zha+24] to plan collision-free trajectories for a robotic manipulator. Schoels et al. [Sch+20] apply forward reachability analysis to inflate a circular robot shape at each time interval on a grid. The inflated shape considers all reachable robot positions, assuming a bounded velocity and acceleration. Formulating collision constraints on a swept volume can also be considered a semi-continuous method. The linear interpolation between convex shapes performed by [Sch+13] prevents the collision of a robot manipulator

with the environment. Interval analysis [Moo66; AM00] is used to bound the constraint functions' extrema on a discrete set of time intervals [Len+11]. The convex hull property of a spline trajectory in basis form [Boo01, B-spline prop. (ix)] is used for solving SIPs by formulating linear constraints on the spline coefficients [Sur+10]. The authors also propose a basis function segmentation to express the splines' time derivatives in terms of the spline coefficients. Thus, bounding linear transformations of the spline coefficients result in a QP to find an inner approximation to the SIP solution. The approach is extended to polynomial constraints, so a QP is retained at the cost of solution accuracy. A spline interpolation is proposed by [Loo+15b] to describe polynomial constraints with equivalent splines in basis form, which may yield an NLP. Spline derivatives are computed via a linear transformation resulting from the differentiation property [Boo01, B-spline prop. (viii)]. They also introduce degree elevation and knot insertion to increase the solution accuracy. While not exhaustive, the overview highlights methods commonly used in robotics and automated driving.
Since reduction-based methods are most efficient near the optimal solution, combining them with one of the other methods to a hybrid one seems reasonable. The combination is typically performed in multiple phases. The results of the earlier phases are used in the locally convergent methods in the later phases.
The following sections review recent works on applying the convex hull property to solve SIPs resulting from OCPs in the context of robotics and automated driving. The works are organized in the solution algorithm categories introduced in the preceding section 2.1.

2.2.1. Numerical Optimization

The methods for polynomial constraint formulation proposed in [Loo+15b] are applied to the parameterization of a holonomic mobile robot's position [Mer+17]. They contribute a minimum-time OCP, transcribed into an NLP, considering semi-infinite constraints on the position's time derivatives and the robot shape. The latter constraints are implemented with convex obstacle shapes via a spline parameterized separating hyperplane. Time scaling is applied to pre-compute the coefficient transformations in the constraint formulations. A trajectory parameterization designed for nonholonomic vehicles retains polynomial collision constraints while considering the robot's orientation [Mer+18b]. The application is extended to minimum time control of a robotic manipulator and a UAV in a constrained three-dimensional environment [PP17]. The work [Sto+16] also applies separating hyperplanes to find the optimal collision-free trajectory for a UAV. In contrast to the work of [Mer+17], each spline segment is assigned a separating hyperplane subject to optimization. The solution to an NLP is compared with the less accurate solution to a MIQP [Sto+15]. Separating hyperplanes are also used to enforce non-convex input constraints to a UAV in combination with a piecewise linear approximation to polynomial constraints [Sto+17].
Restricting the trajectory to pre-computed corridors is frequently used to implement collision avoidance computationally efficiently. The following approaches assign a corridor to each spline segment in advance. Rousseau et al. [Rou+19] constraint the trajectory to a sequence of convex corridors, connecting different kinds of waypoints for UAV cinematographic flight. The minimum-time OCP is transcribed into an NLP, where each

breakpoint is subject to optimization. The work of [FX23] uses quadratic input constraints and piecewise linear corridors, resulting in a quadratically constrained quadratic program (QCQP). A QP describes the optimal spline trajectory of a mobile robot in a piecewise linear corridor in [KF18]. The problem is extended to a QCQP with semi-infinite quadratic state and input constraints [KF17].
The spline convex hull property is also applied to solve SIPs in the context of automated driving. A corridor of connected cubes in a voxel-based spatio-temporal semantic environment representation is built by [Din+19b]. A Bézier polynomial is assigned to each cube. Their coefficients are constrained inside their respective cubes and consider kinematic semi-infinite constraints, resulting in a QP. The idea of spatio-temporal corridors is extended to more accurate prism shapes [Deo+23]. The nonholonomic vehicles' spline parameterization [Mer+18b] is applied to automated highway driving in [Dor+21], resulting in an NLP.

2.2.2. Sampling

Several works propose solving OCPs in the robotic domain, subject to semi-infinite constraints on a discretized state space. The work [Nat+24] extends the ideas in [Nat+21] to an efficient parallelized trajectory planning algorithm for a robotic manipulator. The Hybrid-A* search is performed on the state space's subspace. An NLP with a minimum-time quadratic cost is solved for the optimal spline using the time scaling approach each time a new state is visited during the search. In [Tan+19], the optimal trajectory for a UAV is found on a voxel grid with a shortest-path search. Each node expansion selects a voxel and adds the corresponding coefficient to a uniform spline. The distance between neighboring voxels shrinks with decreasing distance to obstacles to reduce the UAV's velocity. Collision and kinematic constraints are checked on the Bézier coefficients of each segment. A RRT-based search for spline coefficients of a mobile robot trajectory is proposed in [Sun+21]. Each node expansion adds a new coefficient to a spline candidate, starting with the determination of the kinematically feasible set for the next coefficient. The next coefficient is chosen as the closest feasible one to the sampled position, which exceeds a minimum distance to obstacles. In addition, the node expansion can steer toward the goal state by choosing the coefficient with minimum cost-to-go to the goal state.

2.2.3. Hierarchical Algorithms

This section reviews hierarchical trajectory planning algorithms, their solution algorithms, and approaches to reducing the distinct algorithms' complexity. In [Mer+18a], a path is planned through a warehouse to define a local corridor, which includes the relevant obstacles. The path is also used as an initial guess for the minimum-time NLP described in [Mer+18b], which is solved in the second stage. A path planning problem is solved by [Tor+19] to decompose the environment into polygonal collision-free corridors for a UAV. A MIQP is solved to find the optimal allocation of Bézier coefficients to the polygonal regions. A three-stage approach is proposed in [Li+23], addressing a UAV flight in unknown environments. The first stage plans a path through a voxel grid to

initialize the second stage, which solves an NLP considering soft collision, velocity, and acceleration constraints. The third stage solves another NLP to ensure dynamic feasibility with nonuniform breakpoints. While segments can be retained to enable time consistent planning results, this may reduce the MPC's control performance and responsiveness to disturbances.
If the first stage solves a path planning problem, the result is likely far from optimal or even infeasible for the second stage. Thus, solving a trajectory planning problem at the first stage can reduce the computational effort of the second stage even further. A nonlinear optimization algorithm especially benefits from a good initial guess. In the second stage of the algorithm proposed in [TH22a], an NLP is solved for the optimal trajectory of a UAV, initialized with the result of the first stage. In the first stage, the authors apply an A* shortest-path search to find the optimal spline coefficients, considering semi-infinite dynamic and collision constraints. A separating hyperplane between each obstacle and each spline segment guarantees a collision-free trajectory, as in [Sto+16]. The solution accuracy is improved with the minimum volume (MINVO) basis [TH22b], which reduces the distance between the splines and their convex hull. A three-stage algorithm is proposed in [Zho+19], similar to [Li+23]. In contrast, they solve for a dynamically feasible and collision-free trajectory with Hybrid A* in the first stage. Subsequently, the initial trajectory is refined by solving an NLP, subject to soft constraints, followed by a time adjustment in the third stage. A QP with constraints on Bézier polynomial coefficients in the second stage is solved in [Gao+18], which is based on a corridor of connected cubes. They propose a fast marching method [Set99] in the first stage to find the minimum-time UAV trajectory in a voxel grid. The work is extended in [Din+19a]. The first stage is replaced by an A* search, where each node represents a sequence of spline coefficients. A corridor is found by inflating a sequence of balls to maximum size. The second stage solves a more complex QCQP to refine the initial trajectory. Since no Bézier coefficients are used, additional coefficients are added iteratively to ensure a collision-free trajectory. The sampling algorithm from [Tan+19] resembles the first stage in the hierarchical algorithm of [Tan+21]. They include the corridor representation from [Din+19a] and solve a QCQP. In contrast to [Din+19a], they enforce constraints on Bézier polynomial coefficients. A four-stage planning algorithm is proposed in [WG23]. The first stage finds the shortest path toward the goal in a voxel grid, which is used to identify keypoints to guide the search for feasible spline coefficients in the second stage. The third stage solves a QCQP with constraints on the splines' MINVO basis coefficients. In line with [Zho+19], the final stage's time adjustment guarantees a dynamically feasible connection to the goal state.

Compared to the number of contributions in the robotic domain, only a few exist in the context of automated driving. No contribution approximates the nonlinear constraints on the state and input in the vehicle's Frénet frame with polynomial functions, so feasibility is preserved. Consistent result trajectories are only ensured by fixing entire spline segments, like [Li+23]. No publication explicitly addresses recursive feasibility in a shrinking horizon manner, which requires a nonuniform spline parameterization. A large portion of the related work considers a hierarchical algorithm. Some solve the same OCP in each stage, like [TH22a]. Still, no cost- and feasibility-based solution selection strategy is known that achieves consistent planning results and is robust to planning algorithm failure.

3
Fundamentals

This chapter introduces the required coordinate frames and describes the simulation environment. Then, the notion of Lyapunov stability is introduced in the context of this work. The following section specifies the vehicle model considered in the OCP while showing its differential flatness property in the lane-center-aligned Frénet coordinate frame. Finally, the required background on splines in basis form is provided.

3.1. Introduction of Coordinate Frames

This section introduces the three used coordinate frames to represent the environment and the planned vehicle motion, which are shown in figure 3.1.

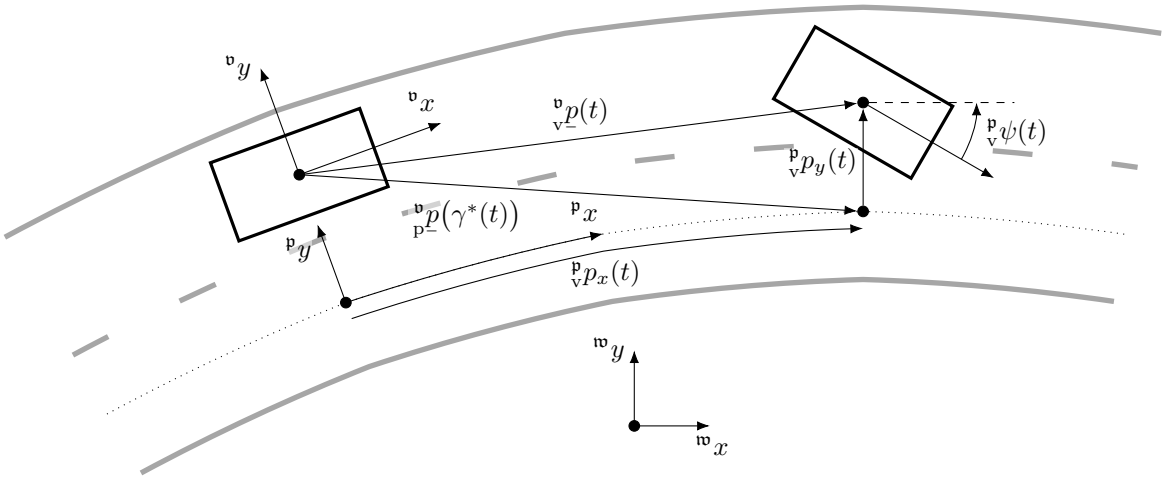


Figure 3.1.: Visualization of the coordinate frames used in this work.

Cartesian world coordinate frame: The Cartesian world coordinate frame originates with a fixed configuration in the world. The elevation in the world is assumed negligible for the trajectory planning task. Thus, the vehicles move in a plane with the abscissa $^{\mathfrak{w}}x \in \mathbb{R}$ and the ordinate $^{\mathfrak{w}}y \in \mathbb{R}$.

Cartesian vehicle coordinate frame: $^{\mathfrak{v}}x \in \mathbb{R}$ and $^{\mathfrak{v}}y \in \mathbb{R}$ denote the axes of the Cartesian vehicle coordinate frame. The origin is located in the controlled vehicle's center of gravity

(COG), and the abscissa is aligned with its longitudinal direction. The leading superscript ${}^{\mathfrak{v}}\square$ indicates the definition of a variable or function in the vehicle coordinates.

Frénet lane center coordinate frame: The vehicle motion in lane-structured environments is usually oriented along the lane center paths. Curved roads increase the complexity of the OCP when defined in a fixed Cartesian frame, as the allowed driving area is described by a non-convex shape. The distance along the center of a reference lane ${}^{\mathfrak{v}}_{\mathrm{p}}\underline{p} : \mathbb{R} \mapsto \mathbb{R}^2$ is denoted by $\gamma \in \mathbb{R}$. By defining the OCP in a coordinate frame ${}^{\mathfrak{p}}x \in \mathbb{R}$ and ${}^{\mathfrak{p}}y \in \mathbb{R}$ aligned to the reference path, the targets, and the driving area are expressed tangential and normal to a lane center. The leading superscript ${}^{\mathfrak{p}}\square$ indicates the definition of a variable or function in the Frénet coordinate frame. The position trajectory ${}^{\mathfrak{v}}_{\mathrm{v}}\underline{p}(t)$ is transformed from the vehicle frame to the Frénet frame by a shortest distance projection onto the reference lane center. The function $\gamma^* : \mathbb{R} \mapsto \mathbb{R}$ denotes the arc length of the vehicle position in dependence on time:

$$\gamma^*(t) \coloneqq \arg\min_{\gamma} \left\| {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma) - {}^{\mathfrak{v}}_{\mathrm{v}}\underline{p}(t) \right\|_2^2. \tag{3.1.1}$$

The origin of the Frénet frame is located next to the origin of the EV frame. The function ${}^{\mathfrak{p}}_{\mathrm{v}}\underline{p}(t) = \begin{bmatrix} {}^{\mathfrak{p}}_{\mathrm{v}}p_x(t) & {}^{\mathfrak{p}}_{\mathrm{v}}p_y(t) \end{bmatrix}^{\mathsf{T}} : \mathbb{R} \mapsto \mathbb{R}^2$ denotes the projection of the vehicle's position on the reference path:

$${}^{\mathfrak{p}}_{\mathrm{v}}p_x(t) = \gamma^*(t) - \arg\min_{\gamma} \left\| {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma) \right\|_2^2, \tag{3.1.2}$$

$${}^{\mathfrak{p}}_{\mathrm{v}}p_y(t) = {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^{\mathsf{T}}\Big({}^{\mathfrak{p}}_{\mathrm{v}}p_x(t) \Big) \left[{}^{\mathfrak{v}}_{\mathrm{v}}\underline{p}(t) - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\Big(\gamma^*(t)\Big) \right]. \tag{3.1.3}$$

Functions involving the reference path's gradient are defined in dependence on the Frénet frame x-position. ${}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}} : \mathbb{R} \mapsto \mathbb{R}^2$ maps the arc length to the path's normal. The lane-relative orientation ${}^{\mathfrak{p}}_{\mathrm{v}}\psi : \mathbb{R} \mapsto \mathbb{R}$ equals the difference between the vehicle orientation and the reference path orientation ${}^{\mathfrak{v}}_{\mathrm{p}}\psi : \mathbb{R} \mapsto \mathbb{R}$:

$${}^{\mathfrak{p}}_{\mathrm{v}}\psi(t) = {}^{\mathfrak{v}}_{\mathrm{v}}\psi(t) - {}^{\mathfrak{v}}_{\mathrm{p}}\psi\Big({}^{\mathfrak{p}}_{\mathrm{v}}p_x(t) \Big). \tag{3.1.4}$$

3.2. Simulation Environment

The control algorithm interacts with the CARLA simulator [Dos+17]. All experiments are carried out on a custom map, which includes a section of the A45 highway between interchange Dortmund-West and Dortmund-Eichlinghofen [Aze+22]. The EV travels in the southern direction along the three-lane road. An aerial view of the section is provided in figure 3.2. Two starting points *s1* and *s2* are used, shown in the magnified regions. The considered highway section is composed of three lanes with lateral positions $d_\square \in \mathbb{R}$ for the left-most, the middle, and the right-most center $\square = \mathrm{lc}, \mathrm{mc}, \mathrm{rc}$ relative to the center of the right-most target lane.
The other vehicles' velocities are controlled by the intelligent driver model (IDM) [Tre+00]. The used parameters are specified in table A.1. In addition to the original model, upper and lower bounds on the acceleration are introduced. The EV and all other vehicles

Figure 3.2.: Section of the A45 used for simulation with magnified starting areas, indications of the EV's initial pose, and labels for the distinct lanes.

are represented by an *Audi A2*, which is one of the models provided by the CARLA simulation. Polygonal chains represent lane boundaries and lane centers. The positions along the chains are queried from the simulator at fixed distances in front and behind the EV. As the EV moves along the road, the positions are not necessarily aligned with the ones from the previous time step, introducing uncertainty in the other vehicles' Frénet frame position estimate. The lane center paths are assumed parallel. Although the assumption excludes highway entries and exits, it simplifies the otherwise curved lane boundaries and centers to lateral offsets from the reference path in the Frénet frame.
Robustness of driving automations is necessary for a successful application and is a research topic in itself. The simulation and the environment representation are idealized, neglecting the uncertainty due to, for example, the environment sensing and perception. Thus, this work focuses on the control algorithm's performance in reaction to the road structure and other vehicles' behavior. Still, the transformation between the Cartesian and the Frenét frames is approximated due to the polygonal lane representation and can, foremost, introduce a longitudinal error. The returned reference trajectory is executed precisely, so the vehicle state always coincides with the next state along the planned trajectory.

3.3. System Stability

The next GT along the route to the destination is passed from the route planning to the behavior module. The vehicle is stabilized in the GT using the implicit control law $\underline{c}_j : [t_j, t_{j+1}) \times \mathcal{C}_0 \mapsto \mathcal{U}_0$. The MPC approximates the solution to problem 1.2.1 at each time step of a monotonously increasing time sequence $\underline{t} \in \mathbb{R}^{\breve{t}}$ of length $\breve{t} \in \mathbb{N}_0$. The time sequence is growing with the time increment $\Delta t \in \mathbb{R}^+$, so $t_{j+1} = t_j + \Delta t$ for $j = 0, 1, \ldots, \breve{t} - 1$. The solution is the optimal state and input open-loop trajectory $\underline{\xi}^* : [0, T] \times \mathcal{C}_0 \mapsto \mathcal{C}_0$ and $\underline{u}^* : [0, T] \times \mathcal{C}_0 \mapsto \mathcal{U}_0$. The inequality constraints $\underline{h}_\xi\big(\underline{\xi}(t), \underline{u}(t)\big) \geq 0$ implicitly define the feasible state and input sets $\mathcal{C}_0 \subset \mathcal{C}$ and $\mathcal{U}_0 \subset \mathcal{U}$. The control law returns the segment $t \in [t_j, t_{j+1})$ along the open-loop input trajectory $\underline{c}_j\left(t, \underline{\xi}_0\right) \coloneqq \underline{u}^*\left(t - t_j, \underline{\xi}_0\right)$. Starting from a feasible initial state $\underline{\xi}_0 \in \mathcal{C}_0$, the EV's closed-loop state and input trajectories

$\underline{\xi}_c : \mathbb{N}_0 \mapsto \mathcal{C}_0$ and $\underline{u}_c : \mathbb{N}_0 \mapsto \mathcal{U}_0$ have to stay feasible in each time step with

$$\underline{\xi}_c(j) \coloneqq \underline{\xi}_0 + \sum_{n=0}^{j} \int_{t_n}^{t_{n+1}} \underline{f}_\xi\Big(\underline{\xi}(\tau), \underline{c}_n\Big(\tau, \underline{\xi}(\tau)\Big)\Big)\, \mathrm{d}\tau, \qquad j = 0, 1, \ldots, \breve{t} - 1, \tag{3.3.1}$$

$$\underline{u}_c(j) \coloneqq \underline{c}_j\Big(t_j, \underline{\xi}_c(j)\Big), \qquad j = 0, 1, \ldots, \breve{t} - 1. \tag{3.3.2}$$

3.3.1. Asymptotic Stability

In addition to staying in the feasible state space, the controller has to render the GT an asymptotically stable equilibrium. The GT is generally defined as a closed subset $\mathcal{T}_\mathrm{d} \subset \mathcal{T} \subset \mathcal{C}_0$ of the feasible state space. Asymptotic stability applies if the definitions 3.3.1 and 3.3.2 hold for the GT.

Definition 3.3.1 (Local stability [Raw+22, def. B.4])**:** The set $\mathcal{T}_\mathrm{d}$ is locally stable for the given system on the state space $\mathcal{C}$ if $\left\|\underline{\xi}_0\right\|_{\mathcal{T}_\mathrm{d}} < \delta$ is bounded by $\delta \in \mathbb{R}^+$ for each $\epsilon \in \mathbb{R}^+$, so $\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}} < \epsilon$ at all time steps $j \in \mathbb{N}_0$.

Definition 3.3.2 (Attraction [Raw+22, def. B.5])**:** The set $\mathcal{T}_\mathrm{d}$ is attractive for the given system on the state space $\mathcal{C}$ if $\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}} \to 0$ with $j \to \infty$ for all $\underline{\xi}_0 \in \mathcal{C}$.

The local stability property requires each closed-loop trajectory originating from a bounded initial state to stay in a region close to the equilibrium. In combination with the attraction property, the closed-loop trajectory has to converge toward the equilibrium. The MPC achieves asymptotic stability in the nominal case if the OCP's minimum cost

$$V\left(\underline{\xi}_0\right) \coloneqq F\Big(\underline{\xi}^*\left(T, \underline{\xi}_0\right)\Big) + \int_0^T l\Big(\underline{\xi}^*\left(\tau, \underline{\xi}_0\right), \underline{u}^*\left(\tau, \underline{\xi}_0\right)\Big)\, \mathrm{d}\tau \tag{3.3.3}$$

resembles a Lyapunov function.

Definition 3.3.3 (Lyapunov function [Raw+22, def. 2.12])**:** The function $V : \mathcal{C} \mapsto \mathbb{R}^+$ is a Lyapunov function for the set $\mathcal{T}_\mathrm{d}$ if it is bounded by continuous, strictly increasing, and unbounded functions $\alpha_n : \mathbb{R}^+ \mapsto \mathbb{R}^+$, $n = 1, 2, 3$ with $\alpha_n(0) = 0$

$$\alpha_1\Big(\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}}\Big) \geq V\Big(\underline{\xi}_c(j)\Big) \geq \alpha_2\Big(\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}}\Big). \tag{3.3.4}$$

The positive definite and continuous function $\beta_1 : \mathbb{R}^+ \mapsto \mathbb{R}^+$ bounds the descent

$$V\Big(\underline{\xi}_c(j+1)\Big) - V\Big(\underline{\xi}_c(j)\Big) \leq -\beta_1\Big(\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}}\Big). \tag{3.3.5}$$

The scaled running cost with $\lambda_0 \in (0, 1]$ resembles a suitable upper bound [GP11, th. 4.11]

$$\beta_1\Big(\left\|\underline{\xi}_c(j)\right\|_{\mathcal{T}_\mathrm{d}}\Big) \coloneqq \lambda_0 l\Big(\underline{\xi}_c(j), \underline{u}_c(j)\Big). \tag{3.3.6}$$

3.3.2. Stabilizing Conditions

The conditions on the minimum cost translate to conditions on the OCP elements, where $\underline{\xi}' \in \mathcal{C}_0$ denotes the state for a feasible input trajectory $\underline{u}(t) \in \mathcal{U}_0$ at the end of the interval $t \in [0, \Delta t]$:

$$\underline{\xi}' \coloneqq \underline{\xi}_0 + \int_0^{\Delta t} \underline{f}_\xi\Big(\underline{\xi}(\tau), \underline{u}(\tau)\Big)\, \mathrm{d}\tau. \tag{3.3.7}$$

Definition 3.3.4 (Stability conditions [Raw+22, ass. 2.14])**:** A function $\underline{u}(t) \in \mathcal{U}_0$ exists on the interval $t \in [0, \Delta t]$ and for each $\underline{\xi}_0 \in \mathcal{T}$, so $\underline{\xi}' \in \mathcal{T}$, and the terminal cost decrease with

$$F\left(\underline{\xi}'\right) - F\left(\underline{\xi}_0\right) \leq -l\big(\underline{\xi}_0, \underline{u}(0)\big). \tag{3.3.8}$$

Also, the running cost is bounded from below for all $\underline{\xi}_0 \in \mathcal{C}_0$ yielding a feasible trajectory to equation (1.2.2):

$$l\big(\underline{\xi}_0, \underline{u}\,(0)\big) \geq \alpha_2\left(\left\|\underline{\xi}_0\right\|_{\mathcal{T}_\mathrm{d}}\right). \tag{3.3.9}$$

The terminal cost is bounded from above for all $\underline{\xi}_0 \in \mathcal{T}$:

$$F\big(\underline{\xi}_0\big) \leq \alpha_3\left(\left\|\underline{\xi}_0\right\|_{\mathcal{T}_\mathrm{d}}\right). \tag{3.3.10}$$

The equations (3.3.4) and (3.3.9) establish the bound $V\left(\underline{\xi}_c\,(j)\right) \geq l\big(\underline{\xi}_c\,(j)\,, \underline{u}_c\,(j)\big)$. The running cost is usually suitable if it is a positive definite quadratic function of the state and input [Raw+22, p. 115]. The equation (3.3.10) ensures $F\left(\underline{\xi}_c\,(j)\right) \geq V\left(\underline{\xi}_c\,(j)\right)$ if $\underline{\xi}_c\,(j) \in \mathcal{T}$. If equation (3.3.8) applies, the required descent in equation (3.3.5) is achieved. Therefore, the terminal cost resembles a Lyapunov function for a distinct locally stabilizing control law [GP11, sec. 5.3].

3.4. Vehicle Model

The vehicle model shall achieve sufficient accuracy for normal highway driving situations. Kinematic models rely on geometric constraints and assume zero tire sideslip, which is sufficient at velocities below $18\,\mathrm{km\,h^{-1}}$ [Raj06, p. 21]. Polack et al. [Pol+17] indicate the applicability at higher velocities if the lateral acceleration is limited to $4.9\,\mathrm{m\,s^{-2}}$ on dry roads. However, it is not clear how far the results generalize to different models. Dynamic models consider the forces on the vehicle and are therefore more accurate at higher velocities. The linearized single-track model assumes low sideslip and steering angles, and it considers the over- or understeering behavior sufficiently accurate up to $4\,\mathrm{m\,s^{-2}}$ lateral acceleration [Mey15, p. 202]. Higher accelerations may require a nonlinear model. Thus, sufficient accuracy is expected from the linearized single-track model described in section 3.4.2.
This work uses differential flatness [Fli+92; Fli+95] to describe the vehicle's motion with differentiable functions, avoiding integration errors. The concept of differential flatness is introduced, followed by the linear single-track model and its flat output inversion. The time argument is omitted for brevity.

3.4.1. Differential Flatness

The differential flatness property allows to derive the state and input of a system

$$\dot{\underline{\xi}} = \underline{f}_{\,\xi}\left(\underline{\xi}, \underline{u}\right), \quad \underline{z} = \underline{f}_{\,z}\left(\underline{\xi}\right), \tag{3.4.1}$$

from finitely many derivatives of the flat output and vice versa. The function $\underline{f}_{\,z} : \mathcal{C} \mapsto \mathcal{Z}$ maps the state to the output trajectory $\underline{z} : \mathbb{R} \mapsto \mathcal{Z}$ in the output space $\mathcal{Z}$ with $\dim \mathcal{Z} = \breve{z}$.

The nonlinear system (3.4.1) is called differentially flat if a diffeomorphism $\underline{\Psi} : \mathcal{C} \mapsto \mathcal{L}$ exists with the transformed state trajectory $\underline{\mathbf{z}} : \mathbb{R} \mapsto \mathcal{L}$:

$$\underline{\mathbf{z}} = \underline{\Psi}\left(\underline{\xi}\right) \Leftrightarrow \underline{\xi} = \underline{\Psi}^{-1}\left(\underline{\mathbf{z}}\right), \tag{3.4.2}$$

$$\underline{\mathbf{z}} = \begin{bmatrix} z_0 & \dots & z_0^{(\delta_0-1)} & z_1 & \dots & z_1^{(\delta_1-1)} & \dots & z_{\check{z}-1} & \dots & z_{\check{z}-1}^{(\delta_{\check{z}-1}-1)} \end{bmatrix}^\intercal. \tag{3.4.3}$$

The controllability indices $\delta_\iota \in \mathbb{N}_0$ denote the minimum time derivative of the $\iota = 0, 1, \dots, \check{z}-1$ output elements z_ι, so an output directly depends on at least one element of the system input. The nonlinear system's output is described by the transformed linear and time-invariant ordinary differential equation with the transformed input trajectory $\underline{\mathbf{v}} : \mathbb{R} \mapsto \mathbb{R}^{\check{z}}$, transformed state trajectory $\underline{\mathbf{z}}$, and transformed state space $\mathcal{L}$:

$$\dot{\underline{\mathbf{z}}} = \underline{A}_{\mathbf{z}}\underline{\mathbf{z}} + \underline{B}_{\mathbf{v}}\underline{\mathbf{v}}. \tag{3.4.4}$$

The linear system is characterized by the matrices $\underline{A}_{\mathbf{z}} \in \mathbb{R}^{\check{\xi}\times\check{\xi}}$ and $\underline{B}_{\mathbf{v}} \in \mathbb{R}^{\check{\xi}\times\check{z}}$. The output $\underline{z}$ is called flat output for the differentially flat system. The functions $\underline{\Phi}_\xi : \mathcal{Z} \times \mathcal{Z} \times \dots \times \mathcal{Z} \mapsto \mathcal{C}$ and $\underline{\Phi}_u : \mathcal{Z} \times \mathcal{Z} \times \dots \times \mathcal{Z} \mapsto \mathcal{U}$ map from finitely many $\omega_z \in \mathbb{N}_0$ output time derivatives to the nonlinear system's state and input:

$$\underline{\xi} = \underline{\Phi}_\xi\Big(\underline{z}, \dot{\underline{z}}, \dots, \underline{z}^{(\omega_z)}\Big), \qquad \underline{u} = \underline{\Phi}_u\Big(\underline{z}, \dot{\underline{z}}, \dots, \underline{z}^{(\omega_z)}\Big). \tag{3.4.5}$$

In addition, a function $\underline{\Phi}_z : \mathcal{C} \times \mathcal{U} \times \mathcal{U} \times \dots \times \mathcal{U} \mapsto \mathcal{Z}$ exists that maps the nonlinear system state and finitely many $\omega_u \in \mathbb{N}_0$ input time derivatives to the flat output:

$$\underline{z} = \underline{\Phi}_z\Big(\underline{\xi}, \underline{u}, \dot{\underline{u}}, \dots, \underline{u}^{(\omega_u)}\Big). \tag{3.4.6}$$

3.4.2. Vehicle Motion Model

This section describes the linearized single-track model according to [Mey15, ch. 11-12] assuming a constant velocity and steering angle. The model captures the velocity-dependent turn rate neglected by kinematic models. The single-track model results from equations describing the equilibria of forces in the x- and y-direction and the moment equilibrium:

$$ {}_{\mathrm{c}}F_y \sin\beta_{\mathrm{v}} - {}_{\mathrm{c}}F_x \cos\beta_{\mathrm{v}} + {}_{\mathrm{r}}F_x + {}_{\mathrm{f}}F_x \cos\delta_{\mathrm{v}} - {}_{\mathrm{f}}F_y \sin\delta_{\mathrm{v}} = 0, \tag{3.4.7}$$

$$ {}_{\mathrm{c}}F_y \cos\beta_{\mathrm{v}} + {}_{\mathrm{c}}F_x \sin\beta_{\mathrm{v}} - {}_{\mathrm{r}}F_y - {}_{\mathrm{f}}F_x \sin\delta_{\mathrm{v}} - {}_{\mathrm{f}}F_y \cos\delta_{\mathrm{v}} = 0, \tag{3.4.8}$$

$$ J_{\mathrm{v}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\ddot{\psi} - \left({}_{\mathrm{f}}F_y \cos\delta_{\mathrm{v}} + {}_{\mathrm{f}}F_x \sin\delta_{\mathrm{v}}\right) l_{\mathrm{f}} + {}_{\mathrm{r}}F_y l_{\mathrm{r}} = 0. \tag{3.4.9}$$

Figure 3.3 shows the free-body model and essential quantities of the single-track model. The functions ${}_{\square}F_{\triangledown} : \mathbb{R} \mapsto \mathbb{R}$ describe the forces acting on the front and rear tires and the COG $\square = \mathrm{f}, \mathrm{r}, \mathrm{c}$ in $\triangledown = x, y$ direction. The front tire rotates in the plane by the steering angle $\delta_{\mathrm{v}} : \mathbb{R} \mapsto \mathbb{R}$. The vehicle velocity $v_{\mathrm{v}} : \mathbb{R} \mapsto \mathbb{R}^+$ deviates by the sideslip angle $\beta_{\mathrm{v}} : \mathbb{R} \mapsto \mathbb{R}$ from the vehicle's longitudinal axis. The vehicle's inertia $J_{\mathrm{v}} \in \mathbb{R}^+$ counteracts the moments at the front and rear tires $\square = \mathrm{f}, \mathrm{r}$, placed at distances $l_\square \in \mathbb{R}^+$ from the COG. The force along the COG's direction of motion ${}_{\mathrm{c}}F_x = m_{\mathrm{v}} a_x$ depends on the vehicle's longitudinal acceleration $a_x : \mathbb{R} \mapsto \mathbb{R}$ and its mass $m_{\mathrm{v}} \in \mathbb{R}^+$. The force in

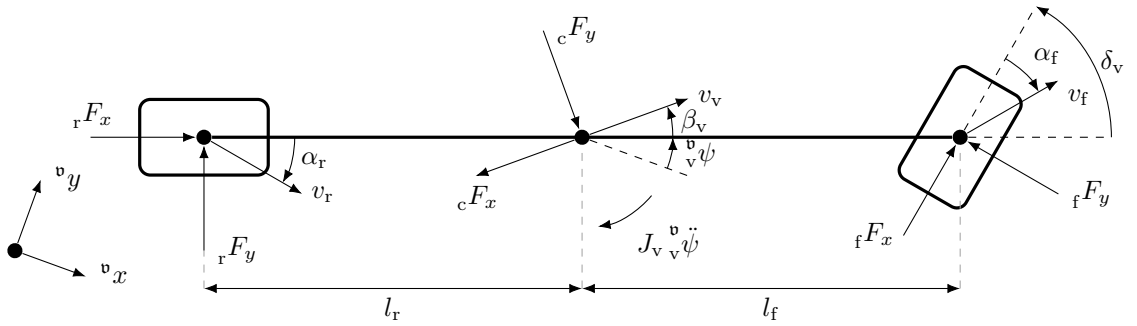


Figure 3.3.: Free-body model of a single-track vehicle model. At $t = 0\,\mathrm{s}$, the COG is located in the origin of the current vehicle frame, and the velocity is aligned with the frame's abscissa. With growing time $t > 0\,\mathrm{s}$, the COG and the velocity vector move along the vehicle's future trajectory.

the y-direction yields ${}_{\mathrm{c}}F_y = m_{\mathrm{v}} v_{\mathrm{v}}^2 \kappa_{\mathrm{v}}$. The curvature $\kappa_{\mathrm{v}} : \mathbb{R} \mapsto \mathbb{R}$ is proportional to the course rate $\kappa_{\mathrm{v}} = ({}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} + \dot{\beta}_{\mathrm{v}})/v_{\mathrm{v}}$.
In contrast to the kinematic model, the velocities $v_\square : \mathbb{R} \mapsto \mathbb{R}^+$ of the dynamic model's front and rear tires $\square = \mathrm{f}, \mathrm{r}$ are not aligned with their orientation. The tire slip angles $\alpha_\square : \mathbb{R} \mapsto \mathbb{R}$ describe the orientation deviation. The lateral tire force almost grows linearly with the slip angle ${}_{\square}F_y \approx c_\square \alpha_\square$, considering the cornering stiffness $c_\square \in \mathbb{R}^+$ for tire slip angles $|\alpha_\square| < 4°$. The tires' and the COG's velocity components tangential and normal to the vehicle's longitudinal axis coincide. The tangential components are given by

$$v_{\mathrm{v}} \cos\beta_{\mathrm{v}} = v_{\mathrm{f}} \cos\left(\delta_{\mathrm{v}} - \alpha_{\mathrm{f}}\right), \qquad v_{\mathrm{v}} \cos\beta_{\mathrm{v}} = v_{\mathrm{r}} \cos\alpha_{\mathrm{r}}. \tag{3.4.10}$$

In the normal direction, the rotational velocity is considered in addition to the translational one:

$$v_{\mathrm{v}} \sin\beta_{\mathrm{v}} + l_{\mathrm{f}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} = v_{\mathrm{f}} \sin\left(\delta_{\mathrm{v}} - \alpha_{\mathrm{f}}\right), \qquad -v_{\mathrm{v}} \sin\beta_{\mathrm{v}} + l_{\mathrm{r}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} = v_{\mathrm{r}} \sin\alpha_{\mathrm{r}}. \tag{3.4.11}$$

The quotients of the directional velocity components at each tire yield

$$\tan(\delta_{\mathrm{v}} - \alpha_{\mathrm{f}}) = \frac{\tan\delta_{\mathrm{v}} - \tan\alpha_{\mathrm{f}}}{1 + \tan\delta_{\mathrm{v}} \tan\alpha_{\mathrm{f}}} = \frac{l_{\mathrm{f}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} + v_{\mathrm{v}} \sin\beta_{\mathrm{v}}}{v_{\mathrm{v}} \cos\beta_{\mathrm{v}}}, \tag{3.4.12}$$

$$\tan\alpha_{\mathrm{r}} = \frac{l_{\mathrm{r}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} - v_{\mathrm{v}} \sin\beta_{\mathrm{v}}}{v_{\mathrm{v}} \cos\beta_{\mathrm{v}}}. \tag{3.4.13}$$

Assuming $\tan\alpha_\square \approx \alpha_\square$, $\tan\delta_{\mathrm{v}} \approx \delta_{\mathrm{v}}$, $\tan(\delta_{\mathrm{v}}\alpha_\square) \approx 0$, $\square = \mathrm{f}, \mathrm{r}$, $\cos\beta_{\mathrm{v}} \approx 1$, and $\sin\beta_{\mathrm{v}} \approx \beta_{\mathrm{v}}$ yields

$$\delta_{\mathrm{v}} - \alpha_{\mathrm{f}} = \beta_{\mathrm{v}} + \frac{l_{\mathrm{f}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_{\mathrm{v}}}, \qquad \alpha_{\mathrm{r}} = \frac{l_{\mathrm{r}}\,{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_{\mathrm{v}}} - \beta_{\mathrm{v}}. \tag{3.4.14}$$

The linearized equations are substituted into linearizations of equations (3.4.8) and (3.4.9), assuming ${}_{\mathrm{f}}F_x \sin\delta_{\mathrm{v}} \approx 0$. Stationary circular driving with $\dot{v}_{\mathrm{v}} = 0$, $\dot{\beta}_{\mathrm{v}} = 0$, and ${}^{\mathfrak{v}}_{\mathrm{v}}\ddot{\psi} = 0$

results in a linear differential equation of order one:

$$m_\mathrm{v} v_\mathrm{v} \, {}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} - c_\mathrm{r}\left(\frac{{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_\mathrm{v}} l_\mathrm{r} - \beta_\mathrm{v}\right) - c_\mathrm{f}\left(\delta_\mathrm{v} - \beta_\mathrm{v} - l_\mathrm{f}\frac{{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_\mathrm{v}}\right) = 0, \tag{3.4.15}$$

$$c_\mathrm{r} l_\mathrm{r}\left(\frac{{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_\mathrm{v}} l_\mathrm{r} - \beta_\mathrm{v}\right) - c_\mathrm{f} l_\mathrm{f}\left(\delta_\mathrm{v} - \beta_\mathrm{v} - l_\mathrm{f}\frac{{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi}}{v_\mathrm{v}}\right) = 0. \tag{3.4.16}$$

The equations are solved for the heading rate with the characteristic velocity $v_\mathrm{ch} \in \mathbb{R}^+$ and the wheelbase $l_\mathrm{v} = l_\mathrm{f} + l_\mathrm{r}$:

$${}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} = \frac{{v_\mathrm{ch}}^2}{{v_\mathrm{ch}}^2 + v_\mathrm{v}^2}\frac{v_\mathrm{v}}{l_\mathrm{v}}\delta_\mathrm{v}, \tag{3.4.17}$$

$$v_\mathrm{ch} = \sqrt{\frac{c_\mathrm{f} c_\mathrm{r} l_\mathrm{v}^2}{m_\mathrm{v}(c_\mathrm{r} l_\mathrm{r} - c_\mathrm{f} l_\mathrm{f})}}. \tag{3.4.18}$$

Steady-State Yaw Dynamics in the Frénet Frame

The vehicle shall be controlled relative to the road course, requiring a differential equation with states in the Frénet frame. Therefore, the equation (3.1.3) is differentiated with respect to time using ${}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{\mathbf{n}}}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right) = -\kappa_\mathrm{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right) \, {}^{\mathfrak{v}}_{\mathrm{v}}\dot{p}_x \, {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)$:

$${}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_y = {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{\mathbf{n}}}^\intercal\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\right) + {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^\intercal\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}} - {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{p}}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\right) = v_\mathrm{v}\sin{}^{\mathfrak{p}}_{\mathrm{v}}\psi. \tag{3.4.19}$$

Since the reference path's normal and tangential are perpendicular, their scalar product becomes zero. The condition always holds if $\frac{\mathrm{d}}{\mathrm{d}t}\left[{}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^\intercal\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\right)\right] = 0$. The equation is solved for the velocity in the x-direction:

$${}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x = \frac{v_\mathrm{v}\cos{}^{\mathfrak{p}}_{\mathrm{v}}\psi}{1 - \kappa_\mathrm{p}({}^{\mathfrak{p}}_{\mathrm{v}}p_x)\,{}^{\mathfrak{p}}_{\mathrm{v}}p_y}. \tag{3.4.20}$$

The derivation of velocities in the Frénet frame is described in more detail in section A.1. The first time derivative of equation (3.1.4) becomes ${}^{\mathfrak{p}}_{\mathrm{v}}\dot{\psi} = {}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} - \kappa_\mathrm{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right) \, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x$, where the reference path's yaw rate is proportional to its curvature $\kappa_\mathrm{p} : \mathbb{R} \mapsto \mathbb{R}$. The substitution of equation (3.4.17) yields

$${}^{\mathfrak{p}}_{\mathrm{v}}\dot{\psi} = \frac{{v_\mathrm{ch}}^2}{{v_\mathrm{ch}}^2 + v_\mathrm{v}^2}\frac{v_\mathrm{v}}{l_\mathrm{v}}\delta_\mathrm{v} - \kappa_\mathrm{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right) \, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x. \tag{3.4.21}$$

Thus, this work follows a similar approach to [Lie22] and describes the vehicle motion using the steady-state yaw rate dynamic model. The vehicle's position, the yaw angle, and the velocity represent the state values $\underline{\xi} = \begin{bmatrix}\xi_x & \xi_y & \xi_\psi & \xi_v\end{bmatrix}^\intercal = \begin{bmatrix}{}^{\mathfrak{p}}_{\mathrm{v}}p_x & {}^{\mathfrak{p}}_{\mathrm{v}}p_y & {}^{\mathfrak{p}}_{\mathrm{v}}\psi & v_\mathrm{v}\end{bmatrix}^\intercal$. The longitudinal acceleration and steering angle are chosen as the inputs $\underline{u} = \begin{bmatrix}u_a & u_{\delta_\mathrm{v}}\end{bmatrix}^\intercal = \begin{bmatrix}a_x & \delta_\mathrm{v}\end{bmatrix}^\intercal$ to the nonlinear differential equation:

$$\begin{aligned}
\dot{\xi}_x &= \frac{\xi_v\cos\xi_\psi}{1 - \kappa_\mathrm{p}({}^{\mathfrak{p}}_{\mathrm{v}}p_x)\xi_y}, & \dot{\xi}_y &= \xi_v\sin\xi_\psi, \\
\dot{\xi}_\psi &= \frac{{v_\mathrm{ch}}^2}{{v_\mathrm{ch}}^2 + \xi_v^2}\frac{\xi_v}{l_\mathrm{v}}u_{\delta_\mathrm{v}} - \kappa_\mathrm{p}\left({}^{\mathfrak{p}}_{\mathrm{v}}p_x\right)\dot{\xi}_x, & \dot{\xi}_v &= u_a.
\end{aligned} \tag{3.4.22}$$

Circle of Forces

The introduced model neglects the boundness of the absolute tire forces. The absolute force acting on the COG cannot exceed the adhesive force between the tires and the road [Mey15, ch. 11.5]:

$$\sqrt{{}_{\mathrm{c}}F_x{}^2 + {}_{\mathrm{c}}F_y{}^2} \leq \mu_{\mathrm{ad}} m_{\mathrm{v}} g. \tag{3.4.23}$$

The adhesive force depends on the friction coefficient $\mu_{\mathrm{ad}} \in \mathbb{R}^+$ and the vertical force due to the gravitational acceleration g. Thus, the longitudinal and lateral forces are interdependent in the limit of adhesion. The limit is described geometrically by Kamm's circle. The friction coefficient depends mainly on the road surface, which is assumed to be dry, resulting in $\mu_{\mathrm{ad}} = 1$.

3.4.3. Flat Output

The position resembles the output $\underline{z} = \begin{bmatrix} z_x & z_y \end{bmatrix}^\intercal = {}^{\mathrm{p}}_{\mathrm{v}}\underline{p}$ of the nonlinear vehicle model. The first time derivatives are given in equation (3.4.22) with $\dot{z}_x = \dot{\xi}_x$ and $\dot{z}_y = \dot{\xi}_y$. The second time derivatives yield

$$\ddot{z}_x = \dot{z}_x \left(\frac{u_a}{\xi_v} + \frac{\kappa_{\mathrm{p}}'(z_x)\dot{z}_x z_y + \kappa_{\mathrm{p}} \dot{z}_y}{1 - \kappa_{\mathrm{p}}(z_x) z_y} \right) - \dot{z}_y \frac{\dot{\xi}_\psi}{1 - \kappa_{\mathrm{p}}(z_x) z_y}, \tag{3.4.24}$$

$$\ddot{z}_y = u_a \frac{\dot{z}_y}{\xi_v} + \dot{z}_x \dot{\xi}_\psi \Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big), \tag{3.4.25}$$

and depend via $\dot{\xi}_\psi$ on $u_{\delta_{\mathrm{v}}}$ and u_a, resulting in a relative degree of two. Thus, the mapping from the state and input of the nonlinear model to the flat output trajectory is established. The inverse mapping to the nonlinear system state trajectories is given by

$$\begin{aligned} \xi_x &= z_x, & \xi_v &= \sqrt{\dot{z}_x^2 \Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big)^2 + \dot{z}_y^2}, \\ \xi_y &= z_y, & \xi_\psi &= \arctan \frac{\dot{z}_y}{\dot{z}_x \Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big)}, \\ \dot{\xi}_\psi &= \frac{\Big(\ddot{z}_y \dot{z}_x - \ddot{z}_x \dot{z}_y\Big)\big(1 - \kappa_{\mathrm{p}}(z_x) z_y\big) + \dot{z}_x \dot{z}_y \Big(\kappa_{\mathrm{p}}'(z_x) z_y \dot{z}_x + \kappa_{\mathrm{p}}(z_x) \dot{z}_y\Big)}{\xi_v^2}. \end{aligned} \tag{3.4.26}$$

Also, the inputs are derived:

$$\begin{aligned} u_a &= \frac{\ddot{z}_x \dot{z}_x \Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big)^2 + \ddot{z}_y \dot{z}_y - \dot{z}_x^2 \Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big)\Big(\kappa_{\mathrm{p}}'(z_x) z_y \dot{z}_x + \kappa_{\mathrm{p}}(z_x) \dot{z}_y\Big)}{\xi_v}, \\ u_{\delta_{\mathrm{v}}} &= \frac{l_{\mathrm{v}} \Big(\dot{\xi}_\psi + \kappa_{\mathrm{p}}(z_x) \dot{z}_x\Big)\big(\xi_v^2 + {v_{\mathrm{ch}}}^2\big)}{{v_{\mathrm{ch}}}^2 \xi_v}. \end{aligned} \tag{3.4.27}$$

A dynamic extension increases the transformed system's order, such that the transformed state vector yields $\underline{\mathbf{z}} = \begin{bmatrix} \underline{\mathbf{z}}_x^\intercal & \underline{\mathbf{z}}_y^\intercal \end{bmatrix}^\intercal$. The system state $\underline{\mathbf{z}}_\triangledown : \mathbb{R} \mapsto \mathbb{R}^3$ in the $\triangledown = x, y$ is composed of the directional system output derivatives $\underline{\mathbf{z}}_\triangledown = \begin{bmatrix} z_\triangledown & \dot{z}_\triangledown & \ddot{z}_\triangledown \end{bmatrix}^\intercal$. Therefore, the transformed input becomes $\underline{\mathbf{v}} = \begin{bmatrix} \mathbf{v}_x & \mathbf{v}_y \end{bmatrix}^\intercal = \begin{bmatrix} \dddot{z}_x & \dddot{z}_y \end{bmatrix}^\intercal$. The directional transformed input $\mathbf{v}_\triangledown : \mathbb{R} \mapsto \mathbb{R}$ equals the jerk.

The inverse mapping enables a problem formulation equivalent to problem 1.2.1 using the transformed states, inputs, and a linear differential equation. The nonlinear system inputs can be reconstructed and used as feedforward control for the feedback module. Nonlinear state and input constraints are applied to functions of the transformed states and inputs. The lateral acceleration $a_y : \mathbb{R} \mapsto \mathbb{R}$ is limited to $|a_y| \leq \bar{a}_y = 4\,\mathrm{m\,s^{-2}}$, rendering the single-track model sufficiently accurate. In addition, a small steering angle is assumed when linearizing the geometric relations in equation (3.4.14). The steering angle at $\bar{a}_y$ obtained with ${}^{\mathfrak{v}}_{\mathrm{v}}\dot{\psi} = a_y/\xi_v$ is shown in figure 3.4 in dependence on the vehicle velocity for different characteristic velocities and wheelbase lengths. According to [Ise22, fig. 7.12], the characteristic velocity remains in the limits $v_{\mathrm{ch}} \in [68\,\mathrm{km\,h^{-1}}, 112\,\mathrm{km\,h^{-1}}]$. The wheelbase is usually limited to $l_{\mathrm{v}} \in [2.5\,\mathrm{m}, 3\,\mathrm{m}]$. The steering angle increases to maintain the same lateral acceleration with a longer wheelbase and lower characteristic velocity. Above $60\,\mathrm{km\,h^{-1}}$, the steering angle stays below 4.5°, regardless of vehicle parameters. Because this work aims at high operational velocities under normal driving conditions, steering limitations are unnecessary. However, lower speeds may require a steering angle constraint to maintain model accuracy.

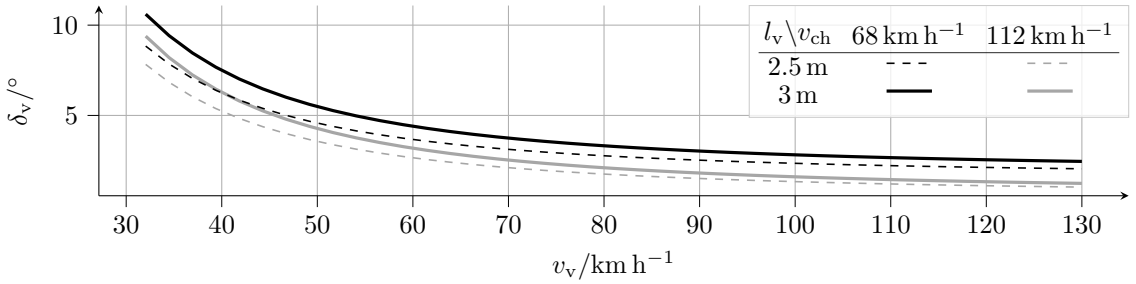


Figure 3.4.: Visualization of the steering angle required to obtain the maximum lateral acceleration of $4\,\mathrm{m\,s^{-2}}$.

3.5. Background on Splines in Basis Form

Polynomial functions are used frequently to parameterize the input and state trajectory of differentially flat systems. The degrees of freedom can be increased by the sequential connection of multiple polynomials at the $\breve{\mathbf{k}} \in \mathbb{N}_2$ breakpoints $\underline{\mathbf{k}} \in \mathbb{R}^{\breve{\mathbf{k}}}$ to a spline $s_{\rho,\underline{w}} : \mathbb{R} \times \mathbb{R}^{\breve{\mathbf{c}}} \times \mathbb{R}^{\breve{\mathbf{k}}} \mapsto \mathbb{R}$. The spline's order of continuity at the inner breakpoints is limited to $\underline{w} \in \mathbb{N}_0^{\breve{\mathbf{k}}-2}$. The $\breve{\mathbf{c}} \in \mathbb{N}_0$ coefficients $\underline{\mathbf{c}} \in \mathbb{R}^{\breve{\mathbf{c}}}$ include the polynomial parameters either explicitly or implicitly. The shorthand $s_{\square}(t) \coloneqq s_{\rho_{\square},\underline{w}_{\square}}(t, \underline{\mathbf{c}}_{\square}, \underline{\mathbf{k}}_{\square})$ is introduced, where a common subscript $\square$ identifies the spline order, continuity, coefficients, and breakpoints. Splines in basis form are equivalent to splines but use a truncated power basis. The basis is constructed recursively from truncated powers, called basis splines (B-splines). The B-splines $\underline{b}_{\rho,\underline{k}}(t, \underline{k})$ with $j = 0, 1, \ldots, \breve{\mathbf{c}} - 1$ elements $b_{j,\rho,\underline{k}} : \mathbb{R} \mapsto [0, 1]$ are defined on $\breve{k} \in \mathbb{N}_2$ knots $\underline{k} \in \mathbb{R}^{\breve{k}}$ [Boo72]. The B-splines form a basis for the space of piecewise polynomials $\mathcal{S}_{\underline{\mathbf{k}},\underline{w},\rho}$ if the knots are repetitions of the $l = 0, 1, \ldots, \breve{\mathbf{k}} - 2$ breakpoints $\mathbf{k}_l < \mathbf{k}_{l+1}$ [CS66, th. 4]:

$$\underline{k} = \Big[\underbrace{\mathbf{k}_0 \quad \mathbf{k}_0 \quad \ldots \quad \mathbf{k}_0}_{\rho} \quad \underbrace{\mathbf{k}_1 \quad \mathbf{k}_1 \quad \ldots \quad \mathbf{k}_1}_{\rho - w_0} \quad \ldots \quad \mathbf{k}_{\breve{\mathbf{k}}-1} \quad \mathbf{k}_{\breve{\mathbf{k}}-1}\Big]^{\intercal}. \tag{3.5.1}$$

Each of the $\iota = 1, \ldots, \breve{\mathbf{k}}-2$ inner breakpoints appears $\rho - w_{\iota-1} \geq 0$ times. This work applies clamped splines, so the outer breakpoints are repeated ρ times. The basis dimensionality $\breve{\mathbf{c}} = \breve{k} - \rho = \dim(\mathcal{S}_{\underline{\mathbf{k}},\underline{w},\rho})$ depends on the number of breakpoints and the spline order.
The B-spline's order grows with the recursion depth. A shorthand $\underline{b}_{\square}(t) \coloneqq \underline{b}_{\rho_{\square},\underline{k}_{\square}}(t, \underline{k}_{\square})$ is introduced for B-splines of order $\rho_{\square}$ and with knots $\underline{k}_{\square}$. Starting at the order $\rho = 1$, the B-splines in the $n = 0, 1, \ldots, \breve{k} - 2$ knot intervals are given by piecewise constant functions:

$$b_{n,1,\underline{k}}(t) = \begin{cases} 1 & \text{if } k_n \leq t < k_{n+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.5.2}$$

B-splines of higher order $\rho > 1$ are constructed from a combination of lower-order B-splines with $l = 0, 1, \ldots, \breve{k} - \rho - 1$ [Boo01, B-spline prop. (i)]:

$$b_{l,\rho,\underline{k}}(t) = \frac{t - k_l}{k_{l+\rho-1} - k_l} b_{l,\rho-1,\underline{k}}(t) + \frac{k_{l+\rho} - t}{k_{l+\rho} - k_{l+1}} b_{l+1,\rho-1,\underline{k}}(t). \tag{3.5.3}$$

Due to the recursive definition, each B-spline overarches $\rho+1$ breakpoints. Thus, B-splines are nonzero and positive on the interval $t \in [k_l, k_{l+\rho}]$. Also, each B-spline is a partition of unity [Boo01, B-spline prop. (iv)]:

$$\sum_{j=0}^{\breve{\mathbf{c}}-1} b_j(t) = 1, \qquad t \in [k_0, k_{\breve{k}-1}]. \tag{3.5.4}$$

The basis form of a piecewise polynomial function $s(t) \in \mathcal{S}_{\underline{\mathbf{k}},\underline{w},\rho}$ is a linear combination of B-splines and the coefficients [Boo01, def. (51)]:

$$s(t) = \sum_{n=0}^{\breve{\mathbf{c}}-1} \mathbf{c}_n b_n(t) = \sum_{n=l-\rho+1}^{l} \mathbf{c}_n b_n(t), \quad t \in [k_l, k_{l+1}], \quad l = \rho - 1, \rho, \ldots, \breve{k} - \rho - 1. \tag{3.5.5}$$

Equation (3.5.5) states that only ρ coefficients and B-splines are required to define a spline segment between two breakpoints. With an increasing order of continuity, the knots are composed of fewer breakpoints. In the extreme case of maximum continuity, the number of segments and knot intervals coincide. Otherwise, the number of segments is smaller, so coefficients are shared among neighboring segments. As a consequence of equation (3.5.4), the spline function is a convex combination of its coefficients [Boo01, B-spline prop. (ix)] and bounded on the $l = \rho - 1, \rho, \ldots, \breve{k} - \rho - 1$ knot intervals [Boo01, cor. 2] to

$$\min\{\mathbf{c}_{l-\rho+1}, \mathbf{c}_{l-\rho+2}, \ldots, \mathbf{c}_l\} \leq s(t) \leq \max\{\mathbf{c}_{l-\rho+1}, \mathbf{c}_{l-\rho+2}, \ldots, \mathbf{c}_l\}, \quad t \in [k_l, k_{l+1}]. \tag{3.5.6}$$

The coefficients are represented intuitively in figure 3.5 using the control points $(\tilde{k}_n, \mathbf{c}_n)$ at the knot averages $\tilde{\underline{k}} \in \mathbb{R}^{\breve{\mathbf{c}}}$:

$$\tilde{k}_j \coloneqq \frac{1}{\rho - 1} \sum_{n=j+1}^{j+\rho-1} k_n, \qquad j = 0, 1, \ldots, \breve{\mathbf{c}} - 1. \tag{3.5.7}$$

The control points form convex hull around each spline segment.

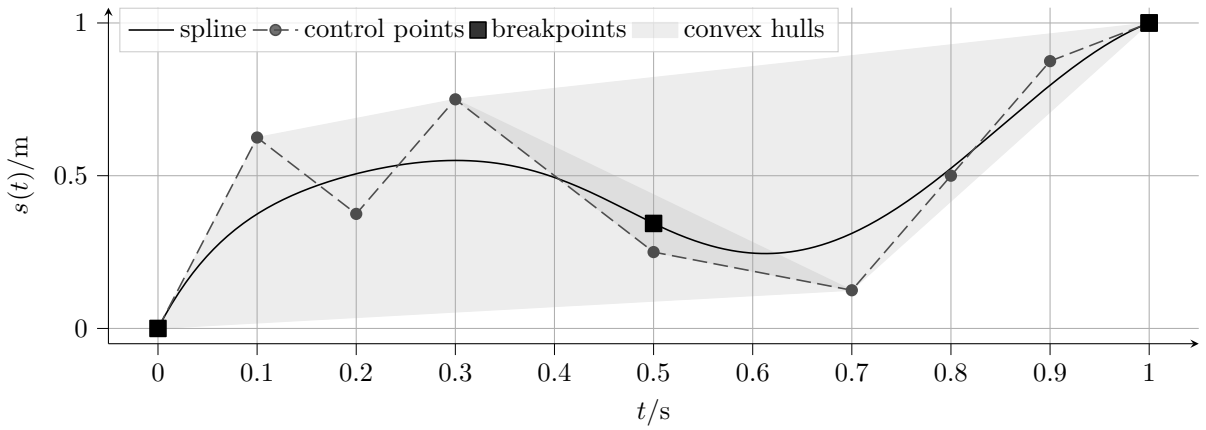


Figure 3.5.: Visualization of a spline in basis form of order $\rho = 6$ with continuity $w_0 = 3$.

The goal is to formulate semi-infinite inequality constraint functions in problem 1.2.1 with splines in basis form and exploiting equation (3.5.6) to guaranteed feasibility. The constraints are functions of the EV trajectory involving basic algebraic operations. Due to the equivalence between polynomials and splines in basis form, the sum, product, derivative, and integral of splines are still splines. Thus, splines describe polynomial constraints of the transformed state trajectory exactly. The operations and their implementation with splines in basis form are introduced in the following.

Differentiation

The spline derivative $\dot{s}(t) \coloneqq \frac{\mathrm{d}}{\mathrm{d}t} s(t) \in \mathcal{S}_{\underline{\mathbf{k}},\underline{w}-1,\rho-1}$ has a reduced order and continuity compared to the initial spline. The derivative's coefficients are determined by divided differences [Boo01, B-spline prop. (viii)]:

$$\frac{\mathrm{d}}{\mathrm{d}t} s(t) = \frac{\mathrm{d}}{\mathrm{d}t} \sum_{n=0}^{\check{\mathbf{c}}-1} \mathbf{c}_n b_n(t) = (\rho - 1) \sum_{n=1}^{\check{\mathbf{c}}-1} \frac{\mathbf{c}_n - \mathbf{c}_{n-1}}{k_{n+\rho-1} - k_n} b_{n,\rho-1,\underline{k}}(t) \tag{3.5.8}$$

The coefficient transformation $\underline{T}^{\underline{b}\square}_{\underline{b}_\triangle} \in \mathbb{R}^{\check{\mathbf{c}}_\triangle} \times \mathbb{R}^{\check{\mathbf{c}}_\square}$ from spline $s_\square(t)$ to spline $s_\triangle(t)$ enables a compact formulation. $\triangle$ is another placeholder variable indicating the use of the same or a different subscript. In the derivative case, the matrix $\underline{T}^{\underline{b}}_{\underline{\dot{b}}} \in \mathbb{R}^{\check{\mathbf{c}}-1} \times \mathbb{R}^{\check{\mathbf{c}}}$ is constructed based on equation (3.5.8). The derivative spline is calculated with $\dot{s}(t) = \left(\underline{T}^{\underline{b}}_{\underline{\dot{b}}} \underline{\mathbf{c}}\right)^\mathsf{T} \underline{b}_{\rho-1,\underline{k}}(t)$ and

$$\underline{T}^{\underline{b}}_{\underline{\dot{b}}} \coloneqq (\rho - 1) \begin{pmatrix} \frac{1}{k_1 - k_\rho} & \frac{1}{k_\rho - k_1} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{k_2 - k_{\rho+1}} & \frac{1}{k_{\rho+1} - k_2} & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{k_{\check{k}-1} - k_{\check{\mathbf{c}}}} & \frac{1}{k_{\check{\mathbf{c}}} - k_{\check{k}-1}} \end{pmatrix}. \tag{3.5.9}$$

Integration

An equivalence transformation of equation (3.5.8) defines the spline integral $s_\mathrm{I}(t) \coloneqq \int s(t)\, \mathrm{d}t \in \mathcal{S}_{\underline{\mathbf{k}},\underline{w}+1,\rho+1}$ [Boo01, p. 127]. $\underline{\mathbf{c}}_\mathrm{I} \in \mathbb{R}^{\check{\mathbf{c}}+1}$ denotes the coefficients of the spline

integral:

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{n=0}^{\breve{\mathbf{c}}}\mathbf{c}_{n,\mathrm{I}}b_{n,\rho+1,\underline{k}}(t)=\sum_{n=0}^{\breve{\mathbf{c}}-1}\rho\frac{\mathbf{c}_{n+1,\mathrm{I}}-\mathbf{c}_{n,\mathrm{I}}}{k_{n+\rho}-k_n}b_n(t)=\sum_{n=0}^{\breve{\mathbf{c}}-1}\mathbf{c}_n b_n(t). \tag{3.5.10}$$

A comparison of coefficients yields the equation (3.5.11) with $n=0,1,\ldots,\breve{\mathbf{c}}-1$. The coefficient $\mathbf{c}_{0,\mathrm{I}}$ resembles the initial value of the integral. It is defined to zero for all subsequent spline integrals.

$$\mathbf{c}_n=\rho\frac{\mathbf{c}_{n+1,\mathrm{I}}-\mathbf{c}_{n,\mathrm{I}}}{k_{n+\rho}-k_n}\Leftrightarrow\mathbf{c}_{n+1,\mathrm{I}}=\mathbf{c}_{n,\mathrm{I}}+\frac{1}{\rho}\left(k_{n+\rho}-k_n\right)\mathbf{c}_n \tag{3.5.11}$$

In line with the differentiation, the spline integral reads $s_{\mathrm{I}}(t)=\left(\underline{T}^{\underline{b}}_{\underline{b}_{\mathrm{I}}}\underline{\mathbf{c}}\right)^{\intercal}\underline{b}_{\rho+1,\underline{k}}(t)+s_{\mathrm{I}}(0)$. The corresponding transformation $\underline{T}^{\underline{b}}_{\underline{b}_{\mathrm{I}}}\in\mathbb{R}^{\breve{\mathbf{c}}+1}\times\mathbb{R}^{\breve{\mathbf{c}}}$ is constructed from equation (3.5.10):

$$\underline{T}^{\underline{b}}_{\underline{b}_{\mathrm{I}}}\coloneqq\frac{1}{\rho}\begin{pmatrix}0&0&0&\cdots&0\\k_\rho-k_0&0&0&\cdots&0\\k_\rho-k_0&k_{\rho+1}-k_1&0&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\k_\rho-k_0&k_{\rho+1}-k_1&k_{\rho+2}-k_2&\cdots&k_{\breve{k}-1}-k_{\breve{\mathbf{c}}-1}\end{pmatrix}. \tag{3.5.12}$$

Sum and Product

Also, the sum and product of two splines yield a spline. Loock, Pipeleers, and Swevers [Loo+15a] provide the required steps, which are presented subsequently. If the operations are applied to two splines $s_\square(t)$ and $s_\triangle(t)$, the corresponding knots $\underline{k}_\square$ and $\underline{k}_\triangle$ must be combined to the combined knots $\underline{k}_{\square\triangle}$. The combination is accomplished by a sorted and strictly increasing union of the respective breakpoints $\underline{\mathbf{k}}_{\square\triangle}=\underline{\mathbf{k}}_\square\underset{<}{\cup}\underline{\mathbf{k}}_\triangle$. The $\iota=1,2,\ldots,\breve{\mathbf{k}}_{\square\triangle}-2$ combined breakpoint continuities are determined by comparison with the $j=1,2\ldots,\breve{\mathbf{k}}_\square-2$ and $n=1,2,\ldots,\breve{\mathbf{k}}_\triangle-2$ operand splines' breakpoints:

$$w_{\iota-1,\square\triangle}=\begin{cases}\min\{w_{j-1,\square},w_{n-1,\triangle}\}&\text{if }\mathbf{k}_{\iota,\square\triangle}=\mathbf{k}_{j,\square}=\mathbf{k}_{n,\triangle},\\w_{j-1,\square}&\text{if }\mathbf{k}_{\iota,\square\triangle}=\mathbf{k}_{j,\square},\\w_{n-1,\triangle}&\text{if }\mathbf{k}_{\iota,\square\triangle}=\mathbf{k}_{n,\triangle}.\end{cases} \tag{3.5.13}$$

The basis of the sum $s_\boxplus(t)=s_\square(t)+s_\triangle(t)$ is defined by the combined breakpoints $\underline{\mathbf{k}}_\boxplus\coloneqq\underline{\mathbf{k}}_{\square\triangle}$ and combined continuities $\underline{w}_\boxplus\coloneqq\underline{w}_{\square\triangle}$. The sum spline's order is determined by $\rho_\boxplus\coloneqq\max\{\rho_\square,\rho_\triangle\}$. The placeholder $\boxplus$ indicates the subscript of the sum spline. The coefficients of the operands must be transformed to their combined basis. The transformations are derived by interpolating the sum spline along a monotonously increasing sequence $\underline{\tau}_\boxplus\in\mathbb{R}^{\breve{\mathbf{c}}_\boxplus}$:

$$s_\boxplus(\tau_{j,\boxplus})=s_\square(\tau_{j,\boxplus})+s_\triangle(\tau_{j,\boxplus}),\quad j=0,1,\ldots,\breve{\mathbf{c}}_\boxplus-1 \tag{3.5.14}$$

$$\overset{\text{eq. (3.5.5)}}{\Rightarrow}{}^{\boxplus}\underline{B}^{\intercal}_{\boxplus}\underline{\mathbf{c}}_\boxplus={}^{\boxplus}\underline{B}^{\intercal}_{\square}\underline{\mathbf{c}}_\square+{}^{\boxplus}\underline{B}^{\intercal}_{\triangle}\underline{\mathbf{c}}_\triangle, \tag{3.5.15}$$

$$\Leftrightarrow\underline{\mathbf{c}}_\boxplus=\underbrace{\left({}^{\boxplus}\underline{B}^{-1}_{\boxplus}{}^{\boxplus}\underline{B}_\square\right)^{\intercal}}_{=:\underline{T}^{\underline{b}_\square}_{\underline{b}_\boxplus}}\underline{\mathbf{c}}_\square+\underbrace{\left({}^{\boxplus}\underline{B}^{-1}_{\boxplus}{}^{\boxplus}\underline{B}_\triangle\right)^{\intercal}}_{=:\underline{T}^{\underline{b}_\triangle}_{\underline{b}_\boxplus}}\underline{\mathbf{c}}_\triangle. \tag{3.5.16}$$

The matrix ${}^{\boxplus}\underline{B}_{\diamond} \coloneqq [\underline{b}_{\diamond}(\tau_{j,\boxplus})]_{j=0,1,\ldots,\check{\mathbf{c}}_{\boxplus}-1} \in \mathbb{R}^{\check{\mathbf{c}}_{\diamond}\times\check{\mathbf{c}}_{\boxplus}}$ includes the evaluations of all B-splines with subscripts $\diamond = \square, \triangle, \boxplus$.
The basis of the product $s_{\boxtimes}(t) = s_{\square}(t)\cdot s_{\triangle}(t)$ is defined by the same breakpoints $\underline{\mathbf{k}}_{\boxtimes} \coloneqq \underline{\mathbf{k}}_{\square\triangle}$ and continuities $\underline{w}_{\boxtimes} \coloneqq \underline{w}_{\square\triangle}$ as the spline sum. In contrast, the order yields $\rho_{\boxtimes} \coloneqq \rho_{\square} + \rho_{\triangle} - 1$. The placeholder $\boxtimes$ indicates the subscript of the product spline. The coefficients of $s_{\boxtimes}(t)$ are determined analogously to the sum coefficients using an interpolation at monotonously the increasing points $\underline{\tau}_{\boxtimes} \in \mathbb{R}^{\check{\mathbf{c}}_{\boxtimes}}$. The symbol $\circ$ denotes the Hadamard product and $\odot$ the Khatri-Rao product:

$$s_{\boxtimes}(\tau_{j,\boxtimes}) = s_{\square}(\tau_{j,\boxtimes}) \cdot s_{\triangle}(\tau_{j,\boxtimes}), \quad j = 0, 1, \ldots, \check{\mathbf{c}}_{\boxtimes} - 1, \tag{3.5.17}$$

$$\overset{\text{eq. (3.5.5)}}{\Rightarrow} {}^{\boxtimes}\underline{B}_{\boxtimes}^{\intercal}\underline{\mathbf{c}}_{\boxtimes} = {}^{\boxtimes}\underline{B}_{\square}^{\intercal}\underline{\mathbf{c}}_{\square} \circ {}^{\boxtimes}\underline{B}_{\triangle}^{\intercal}\underline{\mathbf{c}}_{\triangle}, \tag{3.5.18}$$

$$= \left({}^{\boxtimes}\underline{B}_{\square} \odot {}^{\boxtimes}\underline{B}_{\triangle}\right)^{\intercal} (\underline{\mathbf{c}}_{\square} \otimes \underline{\mathbf{c}}_{\triangle}), \tag{3.5.19}$$

$$\Leftrightarrow \underline{\mathbf{c}}_{\boxtimes} = \underbrace{{}^{\boxtimes}\underline{B}_{\boxtimes}^{-\intercal}\left({}^{\boxtimes}\underline{B}_{\square} \odot {}^{\boxtimes}\underline{B}_{\triangle}\right)^{\intercal}}_{=:\underline{T}_{\underline{b}_{\boxtimes}}^{\underline{b}_{\square}\odot\underline{b}_{\triangle}}} (\underline{\mathbf{c}}_{\square} \otimes \underline{\mathbf{c}}_{\triangle}). \tag{3.5.20}$$

The matrix ${}^{\boxtimes}\underline{B}_{\diamond} \coloneqq [\underline{b}_{\diamond}(\tau_{j,\boxtimes})]_{j=0,1,\ldots,\check{\mathbf{c}}_{\boxtimes}-1} \in \mathbb{R}^{\check{\mathbf{c}}_{\diamond}\times\check{\mathbf{c}}_{\boxtimes}}$ includes the evaluations of all B-splines with subscripts $\diamond = \square, \triangle, \boxtimes$. In the product case only a single transformation $\underline{T}_{\underline{b}_{\boxtimes}}^{\underline{b}_{\square}\odot\underline{b}_{\triangle}} \in \mathbb{R}^{\check{\mathbf{c}}_{\boxtimes}\times\check{\mathbf{c}}_{\square}\check{\mathbf{c}}_{\triangle}}$ is derived to map the Kronecker product of the operators' coefficients to the coefficients of the product spline. The equality between equation (3.5.18) and equation (3.5.19) follows from the Kronecker product rules [Hen+83].

4

Development of the Spline-Based Semi-Infinite Program

The provided background and problem 1.2.1 establish the components of a SIP that formalizes the optimal trajectory for automated highway driving. In the beginning, the development criteria that provide a guideline for the subsequent design decisions are introduced. Considering the criteria, a variational formalization of the optimal trajectory is proposed, leveraging the flat output of the model described in section 3.4. In addition, the GT, the terminal set, the cost functional, and the incorporation of other vehicles are concretized. A spline trajectory parameterization is proposed, rendering the problem finite-dimensional in the decision variables but not in the constraints. The spline parameterization allows for finding an approximation of the SIP solution with a finite number of constraints. Because each spline segment is contained in a convex null, formed by the coefficients, the approximate solution is feasible for the semi-infinite constraints.

4.1. Development Criteria

The trajectories of the planning algorithm shall result in a desirable behavior of the EV for the vehicle passengers. Therefore, criteria and guidelines for development and evaluation are derived. The criteria are organized into automated driving objectives, system stability, complexity, and performance.

4.1.1. Automated Driving Objectives

An MPC in application to automated driving has to consider multiple contradicting objectives. The objectives are either implemented in the constraints or considered in the cost. An overview of relevant objectives and metrics for highway driving is provided below.

Progress

Vehicle passengers desire to reach their destination as fast as possible. Therefore, considering the travel time, a route planner usually finds the optimal path through the road network [Bas+16]. The subsequent behavior and motion planning modules should maximize the progress toward the GTs along the route to minimize the time to the destination.

Different progress metrics exist in the literature, categorized into time- and distance-based metrics.
Time-based metrics, which are used in [Wer+12; Rat+15], consider the time into a terminal set. The terminal set is spanned laterally by a region about a lane center. In the longitudinal direction, the terminal set contains the target velocity or a save trajectory behind the leading vehicle.
Distance-based metrics measure a distance-related quantity toward an LT. The squared deviation from a reference velocity and the target lane center is used in [Zie+14a]. Fassbender et al. [Fas+17] consider the distance toward a position behind a leading vehicle. The deviation from a goal position and heading angle is used by [Göt+17].

Passenger Comfort and Safety

When approaching an LT, the vehicle passengers desire a comfortable driving experience. However, no unique definition of driving comfort exists. It can generally be understood as the absence of uneasiness and anxiety [Bel18]. Simultaneously, a feeling of well-being is present in connection with the vehicle and the execution of a maneuver. Several factors that favor passenger comfort in maneuver execution are mentioned in [Elb+15] and discussed subsequently.
The vehicle's acceleration, braking, and steering controls induce horizontal forces onto the passengers, which have to withstand the forces to maintain posture. Winkel et al. [Win+23] investigate the relation between discomfort and vehicle motion. The study observes force perception as directional, amplitude, and frequency-dependent. Usually, the driving comfort decreases with an increasing force amplitude. Passengers are more sensitive to lateral forces than longitudinal forces. Motion planning algorithms consider forces frequently via a trajectory's acceleration or jerk [Nau+20].
Another factor contributing to discomfort is motion sickness. Motion sickness results from a discrepancy in perceived force and visually observed environment motions. Two models are applied in the motion planning literature to reduce motion sickness. In [Hti+20], the motion sickness dose value [Int97] is subject to minimization. Steinke and Konigorski [SK22] propose a less complex OCP derived from the linearization of the subjective vertical conflict model [Wad+10]. The time integral over the squared jerk is shown to approximate the effect of considering the subjective vertical conflict model in the cost function.
The vehicle passengers must be willing to trust the vehicle automation. Therefore, the automation has to convey the impression of safe operation to the passengers. Maintaining a sufficient distance from other vehicles while performing maneuvers consistently and continuously can contribute to the impression. A time headway of at least 2 s should be maintained, as the feeling of discomfort grows significantly with a decreasing distance below this threshold [Lew+10]. Announcing the next maneuver via vestibular feedback can also improve the impression of safety [Lan18].
Simultaneously, vehicle automations should behave like a defensive human driver [Bas+17]. Human-like trajectories are characterized by a high degree of continuity and alignment with the road [Elb+15]. However, formalizing all effects via rules is not practical. Instead, inverse reinforcement learning can determine a cost function from observed human-driven vehicles [Nau+20].

Traffic Rules

Applying an automated vehicle on public roads requires the algorithmic consideration of traffic rules. This work considers the German traffic rules [Deu13b, StVO][1] that apply to highway-like environments and fall under the behavior and motion planners' responsibility. The passenger preferences for progress and comfort are subordinated to the traffic rules, which introduces a prioritization of objectives. The following rules are considered in the development:

1. Vehicles have to use the right-most lane if possible [Deu13b, §7 I StVO].

2. Other vehicles have to be overtaken on the left side [Deu13b, §5 I StVO].

3. Vehicles have to keep a sufficient distance to the front vehicle to avoid a collision in case of sudden breaking [Deu13b, §4 I StVO].

4. The velocity limit has to be obeyed.

Although the German traffic rules do not specify a minimum distance, fees are charged below a time headway of 0.9 s to the leading vehicle [Deu13a, BABRiGeschwV][2]. In this work, the EV velocity is limited to the recommended velocity of $130\,\mathrm{km\,h^{-1}}$ [Deu78, §1 III BKatV][3]. Although no lower velocity limit exists, vehicles driving on highways have to be capable of driving at least $60\,\mathrm{km\,h^{-1}}$ [Deu13b, §18 I StVO].

4.1.2. System Stability

The EV shall be stabilized in the GT that is provided by the route planning module. The stability requirement imposes conditions on the design choices of the terminal set, the cost, the vehicle, and environment model.

Environment and Vehicle Model

Future motions must be planned so that the vehicle can follow closely, which demands the consideration of a sufficiently accurate motion model in the OCP constraints (1.2.2). The model choice depends on the ODD. This work focuses on normal highway driving situations, where the tires adhere to the road. In contrast, the OCP design may not consider exceptional situations, like evasive maneuvers and traffic jams. The vehicle model, the traffic rules, and other vehicles restrict the EV to the feasible state space $\mathcal{C}_0$. A vehicle can only apply a limited acceleration and steering angle, which is considered by $\mathcal{U}_0$. In contrast to urban environments, highways are characterized by parallel and wide lanes of low curvature, large longitudinal inter-vehicle distances, and high velocities. Thus, the accuracy requirements for the environment model are considered lower on highways.

[1] Straßenverkehrs-Ordnung (StVO) [2] Bundesautobahn-Richtgeschwindigkeits-Verordnung (BABRiGeschwV) [3] Bußgeldkatalog-Verordnung (BKatV)

Cost and Terminal Set

In the automated driving context, the stability conditions introduced in section 3.3.1 demand continuous progress of the EV toward the GT. The minimum cost (3.3.3), which has to grow monotonously and bounded with a growing distance from the target, measures the progress. Also, the minimum cost has to descent over time until reaching the GT. Definition 3.3.4 states the requirements on the OCP components. The running cost is a positive definite, resembling a contiuous lower bound on the minimum cost. The terminal cost provides a contiuous upper bound for the minimum cost in the terminal set. Simultaneously, it is a Lyapunov function for a stabilizing control law for the GT.
The terminal set extends the GT, which is important for automated driving since other vehicles might prevent reaching the GT on a tractable control horizon. On multi-lane highways, the terminal set is composed of distinct regions the EV can safely follow for an extended duration. Such regions may encompass the neighboring lane center paths and staying at a safe distance behind a leading vehicle. If the GT is not reachable in the control horizon, the EV transitions between the regions to progress toward the GT. Such a design is advantageous if a queue of slower-driving vehicles occupies the target lane. The EV has to switch to the left lane for an overtaking maneuver until the target lane becomes free. Then, it can change back to the target lane. The distinct regions are called LTs and render the terminal set a union $\mathcal{T} = \bigcup_{n \in \mathcal{I}_\mathrm{T}} \mathcal{T}_n$ of finitely many disconnected subsets $\mathcal{T}_n \subset \mathcal{C}_0$ with $\mathcal{I}_\mathrm{T} \subset \mathbb{N}$.
However, the disconnected terminal set renders the terminal cost discontinuous, so it cannot provide a continuous upper bound for the minimum cost. Furthermore, the terminal cost descent property (3.3.8) is insufficient for transitions between the LTs since the EV has to leave the terminal set, which violates the assumptions in definition 3.3.4. Thus, stabilizing conditions on the terminal cost are defined in the following, considering the discontinuity.
Assume the EV is in a state $\underline{\xi}_c(j)$, so it reaches the current LT $\underline{\xi}_c(j+1) \in \mathcal{T}_n$ with $n \in \mathcal{I}_\mathrm{T}$ at the next time step. In addition, no progress toward $\mathcal{T}_\mathrm{d}$ is possible by staying in the current LT, and the EV has to approach a new LT $\mathcal{T}_l$, $l \in \mathcal{I}_\mathrm{T} \setminus \{n\}$. The next open-loop state trajectory $\underline{\xi}^*_{j+1} : [0, T] \mapsto \mathcal{C}_0$ with $\underline{\xi}^*_{j+1}(t) \coloneqq \underline{\xi}^*\Big(t, \underline{\xi}_c(j+1)\Big)$ has to fulfill $\underline{\xi}^*_{j+1}(T) \in \mathcal{T}_l$. The substitution of the minimum cost (3.3.3) in the minimum cost descent (3.3.5), the next open-loop state trajectory and the corresponding input $\underline{u}^*_{c,j+1} : [0, T] \mapsto \mathcal{U}_0$ with $\underline{u}^*_{c,j+1}(t) \coloneqq \underline{u}^*\Big(t, \underline{\xi}_c(j+1)\Big)$ yield the condition

$$\begin{aligned} & F\Big(\underline{\xi}^*_{j+1}(T)\Big) + \int_0^T l\Big(\underline{\xi}^*_{j+1}(t), \underline{u}^*_{c,j+1}(t)\Big)\,\mathrm{d}t + \\ & \quad - F\Big(\underline{\xi}^*_j(T)\Big) - \int_0^T l\Big(\underline{\xi}^*_j(t), \underline{u}^*_{c,j}(t)\Big)\,\mathrm{d}t \leq -\lambda_0 l\Big(\underline{\xi}_c(j), \underline{u}_c(j)\Big), \end{aligned} \tag{4.1.1}$$

$$\begin{aligned} \Leftrightarrow\ & F\Big(\underline{\xi}^*_{j+1}(T)\Big) - F\Big(\underline{\xi}^*_j(T)\Big) \leq - \int_0^T l\Big(\underline{\xi}^*_{j+1}(t), \underline{u}^*_{c,j+1}(t)\Big)\,\mathrm{d}t + \\ & - \lambda_0 l\Big(\underline{\xi}_c(j), \underline{u}_c(j)\Big) + \int_0^{\Delta t} l\Big(\underline{\xi}^*_j(t), \underline{u}^*_{c,j}(t)\Big)\,\mathrm{d}t + \underbrace{\int_{\Delta t}^T l\Big(\underline{\xi}^*_j(t), \underline{u}^*_{c,j}(t)\Big)\,\mathrm{d}t}_{=0}, \end{aligned} \tag{4.1.2}$$

$$\overset{\lambda_0 \to 0}{\Rightarrow} F\Big(\underline{\xi}^*_{j+1}(T)\Big) - F\Big(\underline{\xi}^*_j(T)\Big) < - \int_0^T l\Big(\underline{\xi}^*_{j+1}(t), \underline{u}^*_{c,j+1}(t)\Big)\,\mathrm{d}t. \tag{4.1.3}$$

The running cost for trajectories contained in the terminal set at $t > \Delta t$ is assumed to be zero $l\left(\underline{\xi}_j^*(t), \underline{u}_{c,j}^*(t)\right) = 0$. Equation (4.1.3) is a sufficient condition because it neglects the last term in the right-hand-side of equation (4.1.2). The condition demands that the terminal cost descent bounds the transition cost between the LTs from above. The condition is plausible because a new LT is only approached if it yields a lower overall cost than the terminal cost along the current LT. The terminal cost upper bound (3.3.10) applies if an open-loop trajectory and a factor $\lambda_1 \in \mathbb{R}^+$ exist on a possibly infinite control horizon $T \to \infty$ such that $\underline{\xi}^*\left(T, \underline{\xi}_c(j)\right) \in \mathcal{T}_\mathrm{d}$ and

$$F\left(\underline{\xi}_c(j)\right) \leq \lambda_1 \int_0^T l\left(\underline{\xi}^*\left(t, \underline{\xi}_c(j)\right), \underline{u}^*\left(t, \underline{\xi}_c(j)\right)\right) \mathrm{d}t. \tag{4.1.4}$$

In the presence of other vehicles, it is difficult to design terminal cost fulfilling equations (4.1.3) and (4.1.4) due to the vehicles' uncertain future motion. This work aims to design the terminal cost as a heuristic. Assumptions are imposed on the other vehicles' behavior, so the EV is guided towards the GT via the LTs. The terminal cost prioritizes the LTs selection. A lower terminal cost promises a higher control performance in the convergence to the GT. Even if the terminal cost does not achieve the desired cost reduction, the terminal constraints and positive definite running cost ensure convergence toward the terminal set.

4.1.3. Complexity and Performance

The time until the OCP's solution is available depends on the problem's complexity and the solution algorithm design. The algorithm's running time is essential because the deviations from the nominal state cannot be considered during the computation of the optimal trajectory. Thus, the behavior and motion planning modules usually have to provide their results in a fixed time interval. This real-time constraint is specific to the modules' tasks and the hardware, so a generally applicable time interval cannot be given. Still, 100 ms is commonly used as upper limit for the motion planning module's running time in experimental automated vehicles [Kuw+09; Li+16]. Ziegler et al. [Zie+14a] report an even longer time of 500 ms.
This work does not aim to fulfill a specific real-time constraint but focuses its design on obtaining an OCP solution efficiently. The goal is finding a reasonable compromise between the OCP's complexity and the resulting control performance, while the nominal stability shall be mostly invariant to the problem complexity. In turn, models of the vehicle motion, the environment, and the objectives should only be as complex as necessary. Since the lanes are assumed parallel and of low curvature, the environment in the Frénet frame yields a simple representation. The solution method for the OCP should result in a problem with a few decision variables and constraints. Simultaneously, the problem sparsity shall be used for an efficient solution [Die+06]. Generally, the iterations required until convergence are reduced with an initial guess close to the optimal solution. Limiting the maximum allowed iterations also limits the optimization algorithm's running time. However, a tight iteration limit might prevent the algorithm from finding a feasible solution. Graph search algorithms use the DP principle [Bel54] to reduce the search complexity. Since they rely on a finite number of candidate trajectories, the candidates should be distinct to cover a

large portion of the state space and thus increase the chance of finding a feasible solution [GK11].

4.2. Variational Problem

The criteria proposed in the previous section are used to formulate the transformed OCP in problem 4.2.1, based on problem 1.2.1.

Problem 4.2.1 (Transformed optimal control problem)**:** Reformulation of problem 1.2.1 which seeks the transformed states and inputs as solutions to

$$\min_{\substack{\underline{\mathbf{z}}(t),\, T_x,\\ \underline{\mathbf{v}}(t),\, T_y,\\ m \,\in\, \mathcal{I}_\mathrm{T}}} \tilde{F}\Big(\underline{\mathbf{z}}_x(H), \underline{\mathbf{z}}_y(H)\Big) + \sum_{\triangledown=x,y} \int_0^{T_\triangledown} \tilde{l}_\triangledown\Big(\underline{\mathbf{z}}_\triangledown(t), \mathbf{v}_\triangledown(t)\Big)\, \mathrm{d}t \tag{4.2.1}$$

subject to

$$\begin{aligned}
\underline{h}\Big(\underline{\mathbf{z}}_x(t), \underline{\mathbf{z}}_y(t)\Big) &\geq \underline{0}, && t \in [0, H], && \\
\underline{\mathbf{z}}_\triangledown(0) &= \underline{\mathbf{z}}_{\triangledown,0}, && && \triangledown = x, y, \\
\underline{\mathbf{z}}_\triangledown(t) &\in \tilde{\mathcal{T}}_{\triangledown,m}, && t \in [T_\triangledown, H], && \triangledown = x, y, \\
\dot{\underline{\mathbf{z}}}(t) &= \underline{A}_\mathbf{z}\underline{\mathbf{z}}(t) + \underline{B}_\mathbf{v}\underline{\mathbf{v}}(t), && t \in [0, H]. &&
\end{aligned} \tag{4.2.2}$$

The inequality constraints depend on the transformed state vector and are described by the transformed inequality constraint function $\underline{h} : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^{\check{h}}$. Similarly, transformed terminal cost $\tilde{F} : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^+$ and transformed running cost $\tilde{l}_\triangledown : \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}^+$ in $\triangledown = x, y$ direction are introduced. Further constraints fix the trajectory's initial state to the initial values $\underline{\mathbf{z}}_{\triangledown,0} \in \mathbb{R}^3$ in $\triangledown = x, y$ direction and the trajectory's end to the transformed terminal set $\tilde{\mathcal{T}}$. In contrast to the previous problem formulation, the terminal set is explicitly considered by disconnected subsets $\tilde{\mathcal{T}}_{x,m} \times \tilde{\mathcal{T}}_{y,m} \subset \tilde{\mathcal{T}}$. They represent the LTs. The current LT is subject to optimization and introduces an integer variable $m \in \mathcal{I}_\mathrm{T} = \{-2, -1, \ldots, \eta_\mathrm{ov}\}$. In contrast to problem 1.2.1, a fixed prediction horizon $H \in \mathbb{R}^+$ is used. The terminal set must be reached and retained at the control horizons $T_\triangledown \in [0, H)$ in $\triangledown = x, y$ direction. Consequently, the trajectory can reach the terminal set at different times in each direction. The terminal constraint and the running cost choices are explained below. In addition, the inequality constraints enabling clearance from other vehicles are specified.

4.2.1. Global Target

The MPC has to ensure progress along the route toward the destination and a comfortable ride for the passengers, as pointed out in section 4.1.1. The GT is assumed to be time-invariant and given along the right-most lane center. Driving along the target lane center with maximum velocity $\hat{\xi}_v \in \mathbb{R}^+$ maximizes progress and passenger comfort simultaneously on a straight road if vehicle vibrations are neglected. Still, choosing a lower target velocity $\tilde{v} \in \left[\check{\xi}_v, \hat{\xi}_v\right]$ between the upper and lower velocity bounds $\check{\xi}_v \in \left[0, \hat{\xi}_v\right]$ might be necessary for safety reasons and comfort on curved roads. The GT in the x-direction is described by the set $\tilde{\mathcal{T}}_{x,\mathrm{d}} = \{\underline{\mathbf{z}}_x(t) | \dot{z}_x(t) = \tilde{v} \wedge \ddot{z}_x(t) = 0\}$. The right-most lane center resembles the

target lane center $\tilde{\mathcal{T}}_{y,\mathrm{d}} = \left\{ \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal \right\}$. The choice is an incentive for the EV to return to the right-most lane if possible, which is motivated by the first traffic rule stated in section 4.1.1.

4.2.2. Terminal Set

The terminal set extends the GT to a region of the state space where a stabilizing control law exists to keep the EV in the terminal set and stabilize the vehicle in the GT asymptotically. The control horizon required to plan a trajectory into the GT might become intractable in the presence of other vehicles. Thus, the EV may stay in the terminal set for an extended time duration. In addition, the terminal set design should consider the same comfort and progress objectives as the GT. The lane-structured highway environment motivates a disconnected terminal set.
If the target lane center is occupied, the EV may track the centers of the possibly unoccupied other lanes. Simultaneously, the EV can track the target velocity if no other vehicle is in front. Thus, each LT is composed of two sets determined by

$$\tilde{\mathcal{T}}_{x,m} = \tilde{\mathcal{T}}_{x,\mathrm{d}}, \qquad m = -2, -1, 0, \tag{4.2.3}$$

$$\tilde{\mathcal{T}}_{y,m} = \left\{ \begin{bmatrix} d_\square & 0 & 0 \end{bmatrix}^\intercal \right\}, \qquad (m, \square) = (-2, \mathrm{lc}), (-1, \mathrm{mc}), (0, \mathrm{rc}), \tag{4.2.4}$$

assuming the absence of other vehicles. If other vehicles are present, they might require the EV to follow at a desired time headway $t_{\mathrm{hw}} \in \mathbb{R}^+$. Each other vehicle $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ provides a target trajectory $z_{\tilde{x},m} : \mathbb{R} \mapsto \mathbb{R}$ determined by $z_{\tilde{x},m}(t) = z_{x,m}(t) - t_{\mathrm{hw}} \dot{z}_{x,m}(t)$. The EV shall track the trajectory in combination with the vehicles' associated lane center $d_m(t) : \mathbb{R} \mapsto \mathbb{R}$. Thus, the EV still maximizes the progress toward the destination while keeping a safe distance to the vehicle in front. The LTs considering the other vehicles as leading vehicles are defined by

$$\tilde{\mathcal{T}}_{x,m} = \left\{ \begin{bmatrix} z_{\tilde{x},m}(t) & \dot{z}_{\tilde{x},m}(t) & \ddot{z}_{\tilde{x},m}(t) \end{bmatrix}^\intercal \mid t \in [0, H] \right\}, \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}, \tag{4.2.5}$$

$$\tilde{\mathcal{T}}_{y,m} = \left\{ \begin{bmatrix} d_m(t) & 0 & 0 \end{bmatrix}^\intercal \mid t \in [0, H] \right\}, \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}. \tag{4.2.6}$$

The union of all LTs is the transformed terminal set $\tilde{\mathcal{T}} = \bigcup_{m=-2,-1,\ldots,\eta_{\mathrm{ov}}} \left(\tilde{\mathcal{T}}_{x,m} \times \tilde{\mathcal{T}}_{y,m} \right)$. New vehicles can appear at the control horizons as the EV follows the terminal set. Thus, a receding horizon is used to guarantee the feasibility of the planned trajectory on a fixed prediction horizon. The distinct control horizons allow the MPC to find the optimal time instances for reaching the terminal set. The terminal control law defines a sequence of LTs toward the GT, assuming the existence of feasible trajectories to implement the LT transitions. Such trajectories usually exist in the absence of other vehicles or if other vehicles behave defensively and give way to the EV. The LT sequences are defined a-priori in dependence on the current vehicle state, the terminal state, and the other vehicles' states. The terminal cost conditions (4.1.3) and (4.1.4) are fulfilled if the terminal cost bound the minimum cost for each LT transition from above. If the assumptions do not apply, the EV cannot make progress toward the GT. However, safety is not affected since all LTs include save and desirable states. Detailed descriptions of the sequences, assumptions, and the terminal cost are given in section 8.1.

4.2.3. Running Cost

The running cost shall be minimum for trajectories implementing a desirable compromise between progress and comfort. In addition, a continuous lower bound (3.3.9) must exist. Simultaneously, the running cost should include a low number of features to limit the OCP's complexity.
In this work, the transformed running cost is chosen from the class of linear quadratic minimum-time problems and considers the transformed states and inputs in the $\triangledown = x, y$ direction. Such problems can yield an analytic solution [VL91] that favors a high control performance with a few optimization variables in special cases. The comfort aspect is considered by the squared jerk. The time required to reach the terminal set is included to consider the desire for progress:

$$\tilde{l}_{\triangledown}\Big(\underline{\mathbf{z}}_{\triangledown}(t), \mathbf{v}_{\triangledown}(t)\Big) = w_{T,\triangledown} + w_{\mathrm{j},\triangledown}\dddot{z}^2_{\triangledown}(t), \qquad \triangledown = x, y. \tag{4.2.7}$$

The weights $w_{\square,\triangledown} \in \mathbb{R}^+$ on the cost features time and jerk $\square \in \{T, \mathrm{j}\}$ in $\triangledown = x, y$ direction allow the implementation of preferences. The weighted time to the terminal set provides a continuous lower bound $w_{T,\triangledown} \leq \tilde{l}_{\triangledown}\Big(\underline{\mathbf{z}}_{\triangledown}(t), \mathbf{v}_{\triangledown}(t)\Big)$ in the sense of equation (3.3.9). Assuming the existence of a feasible LT sequence to the GT, the minimum time transitions establish a lower bound for the minimum cost to the GT, so the right-hand side of equation (3.3.4) is fulfilled.

4.2.4. Other Vehicles

The trajectory planner has to take other vehicles into account. The CARLA simulation provides the rectangular vehicle shapes, which are transformed into the Frénet frame. The lanes' radii are large in comparison to the vehicles, so their rectangular shapes are preserved approximately during the transformation. In highway scenarios, a limited heading angle deviation from the reference path can be expected. Vehicles in the HighD dataset [Kra+18] do not exceed a heading angle limit of $\psi_{\mathrm{ov}} = 7°$ relative to the lane center at velocities beyond $60\,\mathrm{km\,h^{-1}}$, shown in section A.5. Similar to [Ada+22], the limited heading angle motivates to enclose the vehicle shapes with axis-aligned ellipses. The ellipses are defined by their semi-axis lengths for the EV $\Delta_{\triangledown} \in \mathbb{R}^+$ and the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles $\Delta_{\triangledown,m} \in \mathbb{R}^+$ in $\triangledown = x, y$ direction. The latter ones are represented by the inner-most ellipse in figure 4.1. The Minkowski sum of the EV ellipse with the other vehicles' ellipses enables an efficient collision check [ZS10]. The Minkowski sum reduces to summing the ellipses' semi-axes for axis-aligned ellipses [YC15]. The result is represented by the dashed ellipse in figure 4.1.
The motion of the other vehicles' shapes must be predicted to avoid collisions. Recently, many contributions with data-driven models [Hua+22] have been made to the field of trajectory prediction for automated driving. In highway driving, phases of constant velocity are observed frequently [AL17]. Thus, a constant velocity prediction is assumed sufficient for the scope of this work. However, deviations from the prediction are likely, possibly rendering a previously feasible trajectory infeasible. Following the idea of Schwarting et al. [Sch+18], the vehicle shapes are inflated with a growing uncertainty ellipse around the nominal trajectory prediction. While the approach does not guarantee a collision-free trajectory, it increases the chance of the recursive feasibility property to hold. In

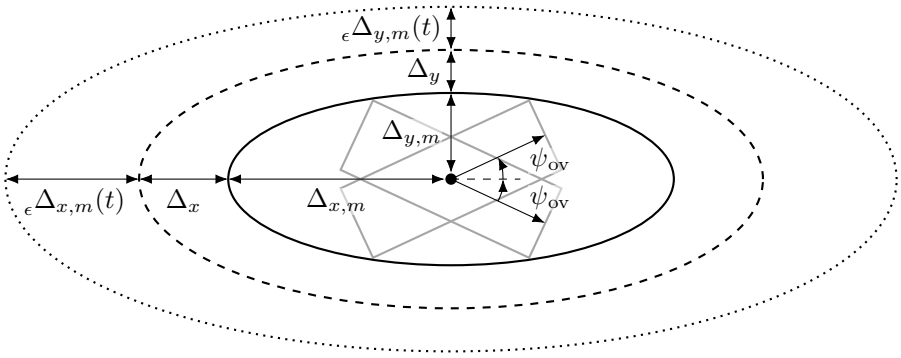


Figure 4.1.: Visualization of the ellipse shape design assuming a limited heading angle. The other vehicles' ellipses are inflated considering the EV shape and prediction uncertainty.

addition, the shape inflation induces a safety distance, considering the passengers' desire for safety. Therefore, the inflations $_{\epsilon}\Delta_{\triangledown,m} : \mathbb{R} \mapsto \mathbb{R}^+$ are introduced for all other vehicles $m = 1, 2, \ldots, \eta_{\text{ov}}$, represented by the dotted ellipse in figure 4.1. The inequality constraints preventing overlap between the EV's and the other vehicles' ellipses are described by

$$\frac{\Big(z_{x,m}(t) - z_x(t)\Big)^2}{\tilde{\Delta}_{x,m}(t)} + \frac{\Big(z_{y,m}(t) - z_x(t)\Big)^2}{\tilde{\Delta}_{y,m}(t)} \geq 1, \qquad m = 1, 2, \ldots, \eta_{\text{ov}}, \tag{4.2.8}$$

with the squared semi-axis lengths $\tilde{\Delta}_{\triangledown,m}(t) = \Big(\Delta_{\triangledown} + \Delta_{\triangledown,m} + {}_{\epsilon}\Delta_{\triangledown,m}(t)\Big)^2$ for $\triangledown = x, y$.

4.3. Polynomial Spline Trajectory Parameterization

A finite-dimensional problem must be inferred from problem 4.2.1 to find a solution numerically. The solution to problem 4.3.1, obtained analytically [Tak+89], provides a trajectory parameterization enabling a high control performance with few variables.

Problem 4.3.1 (Simplified optimal control problem)**:** In the absence of inequality constraints, considering a given LT and fixed control horizons, problem 4.2.1 becomes two independent problems in the $\triangledown = x, y$ direction. The problems are described by

$$\min_{\mathbf{v}_{\triangledown}(t), \underline{\mathbf{z}}_{\triangledown}(t)} \int_0^{T_{\triangledown}} w_{T,\triangledown} + w_{\text{j},\triangledown} {\mathbf{v}_{\triangledown}}^2(t) \, \mathrm{d}t \tag{4.3.1}$$

subject to

$$\begin{aligned} \dot{\underline{\mathbf{z}}}_{\triangledown}(t) &= \underline{A}_{\mathbf{z}} \underline{\mathbf{z}}_{\triangledown}(t) + \underline{B}_{\mathbf{v}} \mathbf{v}_{\triangledown}(t), \quad t \in [0, T_{\triangledown}], \\ \underline{\mathbf{z}}_{\triangledown}(0) &= \underline{\mathbf{z}}_{\triangledown,0}, \\ \underline{\mathbf{z}}_{\triangledown}(T_{\triangledown}) &= \underline{\mathbf{z}}_{\triangledown,\text{T}}. \end{aligned} \tag{4.3.2}$$

The solution is obtained via the Lagrange functions $L_{\triangledown} : \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}$ with Lagrange multipliers $\underline{\Lambda}_{\triangledown} : \mathbb{R} \mapsto \mathbb{R}^3$ in the $\triangledown = x, y$ direction:

$$L_{\triangledown}\Big(\mathbf{v}_{\triangledown}(t), \underline{\mathbf{z}}_{\triangledown}(t), \dot{\underline{\mathbf{z}}}_{\triangledown}(t)\Big) = w_{T,\triangledown} + w_{\text{j},\triangledown} \mathbf{v}_{\triangledown}^2(t) + \underline{\Lambda}_{\triangledown}^{\intercal}(t)\Big(\dot{\underline{\mathbf{z}}}_{\triangledown}(t) - \underline{A}_{\mathbf{z}} \underline{\mathbf{z}}_{\triangledown}(t) - \underline{B}_{\mathbf{v}} \mathbf{v}_{\triangledown}(t)\Big) \tag{4.3.3}$$

$$\begin{aligned} &= w_{T,\triangledown} + w_{\text{j},\triangledown} \mathbf{v}_{\triangledown}^2(t) + \Lambda_{0,\triangledown}(t)\Big(\dot{\mathbf{z}}_{0,\triangledown}(t) - \mathbf{z}_{1,\triangledown}(t)\Big) + \\ &\quad + \Lambda_{1,\triangledown}(t)\Big(\dot{\mathbf{z}}_{1,\triangledown}(t) - \mathbf{z}_{2,\triangledown}(t)\Big) + \Lambda_{2,\triangledown}(t)\Big(\dot{\mathbf{z}}_{2,\triangledown}(t) - \mathbf{v}_{\triangledown}(t)\Big). \end{aligned} \tag{4.3.4}$$

Firstly, the necessary condition for the optimal input is applied [Föl94, eq. (2.53)]. The arguments of the Lagrangian functions are omitted for brevity:

$$\frac{\partial L_\triangledown}{\partial \mathbf{v}_\triangledown(t)} = 2w_{\mathrm{j},\triangledown}\mathbf{v}_\triangledown(t) - \Lambda_{2,\triangledown}(t) = 0, \qquad \triangledown = x, y. \tag{4.3.5}$$

Secondly, the Euler-Lagrange equations yield conditions for the Lagrange multipliers [Föl94, eq. (2.52)]:

$$\begin{aligned} &\frac{\partial L_\triangledown}{\partial \underline{\mathbf{z}}_\triangledown(t)} - \frac{\mathrm{d}}{\mathrm{d}t}\frac{\partial L_\triangledown}{\partial \dot{\underline{\mathbf{z}}}_\triangledown(t)} = -\underline{A}_{\mathbf{z}}{}^{\mathsf{T}}\underline{\Lambda}_\triangledown(t) - \dot{\underline{\Lambda}}_\triangledown(t) = \underline{0}, && \triangledown = x, y, \\ &= \begin{bmatrix} -\dot{\Lambda}_{0,\triangledown}(t) & -\Lambda_{0,\triangledown}(t) - \dot{\Lambda}_{1,\triangledown}(t) & -\Lambda_{1,\triangledown}(t) - \dot{\Lambda}_{2,\triangledown}(t) \end{bmatrix}^{\mathsf{T}}, && \triangledown = x, y. \end{aligned} \tag{4.3.6}$$

Equation (4.3.6) requires $\dot{\Lambda}_{0,\triangledown}(t) = 0$. Thus, the first Lagrange multiplier is constant and defined to $\Lambda_{0,\triangledown}(t) = 2w_{\mathrm{j},\triangledown}q_{0,\triangledown}$ with the parameters $\underline{q}_\triangledown \in \mathbb{R}^6$ in $\triangledown = x, y$ direction. The equations (4.3.5) and (4.3.6) yield the optimal solution to problem 4.3.1 as polynomials of degree five:

$$\begin{bmatrix} \underline{\mathbf{z}}_\triangledown(t) \\ \underline{\Lambda}_\triangledown(t) \end{bmatrix} = \begin{bmatrix} \frac{1}{120}t^5 & \frac{1}{24}t^4 & \frac{1}{6}t^3 & \frac{1}{2}t^2 & t & 1 \\ \frac{1}{24}t^4 & \frac{1}{6}t^3 & \frac{1}{2}t^2 & t & 1 & 0 \\ \frac{1}{6}t^3 & \frac{1}{2}t^2 & t & 1 & 0 & 0 \\ 2w_{\mathrm{j},\triangledown} & 0 & 0 & 0 & 0 & 0 \\ -2w_{\mathrm{j},\triangledown}t & -2w_{\mathrm{j},\triangledown} & 0 & 0 & 0 & 0 \\ w_{\mathrm{j},\triangledown}t^2 & 2w_{\mathrm{j},\triangledown}t & 2w_{\mathrm{j},\triangledown} & 0 & 0 & 0 \end{bmatrix} \underline{q}_\triangledown, \qquad \triangledown = x, y. \tag{4.3.7}$$

The polynomial trajectories are optimal for problem 4.2.1 if the neglected constraints are inactive. Otherwise, the trajectories are suboptimal and the degrees of freedom must be increased to improve the solution accuracy. Additional degrees of freedom can be introduced by connectiong multiple polynomials to splines. A spline of order six is still optimal for problem 4.3.1 since it extends the function class of fifth-degree polynomials. Furthermore, by applying Potryagin's maximum principle [Rat16, sec. 4.5.2], the class of spline functions is shown to be optimal for a state-bounded version of problem 4.3.1.

Problem 4.3.2 (Semi-infinite program)**:** The spline parameterization $z_\triangledown(t) \coloneqq s_\triangledown(t)$ of the trajectories in $\triangledown = x, y$ direction introduces $\underline{\mathbf{c}}_\triangledown$, $\underline{\mathbf{k}}_\triangledown$, and $\check{\mathbf{k}}_\triangledown \in \mathcal{I}_{\check{\mathbf{k}},\triangledown}$ for finite sets $\mathcal{I}_{\check{\mathbf{k}},\triangledown} \subset \mathbb{N}_2$ as decision variables. The constraints for collision avoidance and the bounds on the state and input trajectories apply to the whole prediction horizon. Thus, the spline parameterization renders problem 4.2.1 a SIP described by

$$\min_{\substack{T_\triangledown, \underline{\mathbf{c}}_\triangledown, \underline{\mathbf{k}}_\triangledown, \\ \check{\mathbf{k}}_\triangledown \in \mathcal{I}_{\check{\mathbf{k}},\triangledown}, \\ \triangledown = x, y, \\ m \in \mathcal{I}_{\mathrm{T}}}} \tilde{F}\Big(\underline{s}_x(H), \underline{s}_y(H)\Big) + \sum_{\triangledown = x,y} \int_0^{T_\triangledown} \tilde{l}_\triangledown\Big(\underline{s}_\triangledown(t), \dddot{\underline{s}}_\triangledown(t)\Big)\,\mathrm{d}t \tag{4.3.8}$$

$$\begin{aligned} &\text{subject to} \\ &\underline{h}\Big(\underline{s}_x(t), \underline{s}_y(t)\Big) \geq \underline{0}, && t \in [0, H], \\ &\underline{s}_\triangledown(0) = \underline{\mathbf{z}}_{\triangledown,0}, && \triangledown = x, y, \\ &\underline{s}_\triangledown(t) \in \tilde{\mathcal{T}}_{\triangledown,m}, && t \in [T_\triangledown, H], \quad \triangledown = x, y. \end{aligned} \tag{4.3.9}$$

The functions $\underline{s}_\nabla : \mathbb{R} \mapsto \mathbb{R}^3$ also include the first and second time derivatives in $\nabla = x, y$ direction $\underline{s}_\nabla(t) \coloneqq \left[s_\nabla^{(j)}(t)\right]_{j=0,1,2}^\mathsf{T}$. The infinite constraint set must be reduced to a finite one to solve problem 4.3.2 numerically. The works presented in section 2.1 discretize the inequality constraints over time. The discretization method renders the constraints in problem 4.3.2 finite-dimensional, but it is likely to yield an infeasible approximate solution to problem 4.3.2. A solution of sufficient accuracy would require an intractable number of constraints. Systematic approaches to the solution of SIPs include discretization, reduction-based, semi-continuous, and hybrid methods. The methods are briefly reviewed in section 2.2. Solution algorithms employing a semi-continuous method are proposed frequently in works from robotic and automated driving fields. In contrast to the discretization methods, they yield a feasible solution to the SIP. Simultaneously, semi-continuous methods avoid the computational complexity and possible convergence issues of the reduction-based and hybrid methods.
Therefore, this work uses a semi-continuous method based on the splines' convex hull. The inequality constraints in equation (4.3.9) demand the positivity of the constraint functions. Positive spline coefficients certify feasibility if the constraints are replaced by spline functions that are equivalent to or lower bounding the constraint functions.

5

Polynomial Constraint Formulation

This chapter specifies the semi-infinite inequality constraints in problem 4.3.2 while including the vehicle model from section 3.4.2 and the collision constraints (4.2.8) as polynomial functions of the spline-parameterized transformed state trajectory. However, the mapping from the flat output to the vehicle state in the equation (3.4.26) introduces nonlinear functions. Although the constraints are approximated, feasibility is preserved at the cost of control performance. The time argument is omitted in the following to simplify the notation. Table A.5 quantifies the bounds on the state and input trajectories and the reference path, which are introduced in this chapter. Parts of this chapter have been published in [Dor+23].

5.1. Velocity Constraints

The traffic rules motivate an upper bound on the EV's velocity to not exceed the allowed speed limit. The lower velocity bound increases the feasible space for the applied constraint approximations. The velocity ξ_v from equation (3.4.26) is squared to obtain the polynomial constraints

$$\check{\xi}_v^2 \leq \dot{z}_x^2\Big(1 - \kappa_\mathrm{p}(z_x) z_y\Big)^2 + \dot{z}_y^2 \leq \hat{\xi}_v^2. \tag{5.1.1}$$

The right inequality is transformed to demand positivity. In combination with the substitution $\dot{z}_y^2 \coloneqq \dot{z}_x^2\Big(1 - \kappa_\mathrm{p}(z_x) z_y\Big)^2 \tan^2 \xi_\psi$, resulting from the division of equation (3.4.19) by equation (3.4.20), the transformation results in equation (5.1.2). Still, the constraint depends on $\kappa_\mathrm{p}(z_x)$, which is not yet parameterized. A polynomial curvature approximation would require frequent parameter updates to maintain accuracy. However, parameter estimations render the curvature inconsistent and invalidate recursive feasibility. Instead, a constant symmetric upper bound $\bar{\kappa}_\mathrm{p} \in \mathbb{R}^+$ is assumed. Additional bounds are introduced for the heading angle $\bar{\xi}_\psi \in \mathbb{R}^+$ and the position $\bar{z}_y \in \mathbb{R}^+$. Substituting the trajectories with their respective bounds yields equation (5.1.3), which is a lower bound to the left-hand side of equation (5.1.2). Finally, the inequality is solved for the upper bound $\hat{\dot{z}}_x \in \mathbb{R}^+$ in

equation (5.1.4).

$$\hat{\xi}_v^2 - \dot{z}_x^2\Big(1-\kappa_\mathrm{p}(z_x)z_y\Big)^2\left(1+\tan^2\xi_\psi\right) \geq 0 \tag{5.1.2}$$

$$\Rightarrow \hat{\xi}_v^2 - \dot{z}_x^2(1+\bar{\kappa}_\mathrm{p}\bar{z}_y)^2\left(1+\tan^2\bar{\xi}_\psi\right) \geq 0 \tag{5.1.3}$$

$$\Leftrightarrow \dot{z}_x \leq \hat{\dot{z}}_x = \frac{\hat{\xi}_v}{(1+\bar{\kappa}_\mathrm{p}\bar{z}_y)\sqrt{1+\tan^2\bar{\xi}_\psi}} \tag{5.1.4}$$

The lower bound $\check{\dot{z}}_x \in \mathbb{R}^+$ is derived similarly. The left inequality in equation (5.1.1) is transformed into equation (5.1.5). The functions are substituted with their respective bounds to obtain equation (5.1.6). In contrast to the upper bound constraint, the substitution $\dot{z}_y \coloneqq 0$ is applied. Then, equation (5.1.6) is solved for $\check{\dot{z}}_x$ to get equation (5.1.7).

$$\dot{z}_x^2\Big(1-\kappa_\mathrm{p}(z_x)z_y\Big)^2 + \dot{z}_y^2 - \check{\xi}_v^2 \geq 0 \tag{5.1.5}$$

$$\Rightarrow \dot{z}_x^2(1-\bar{\kappa}_\mathrm{p}\bar{z}_y)^2 - \check{\xi}_v^2 \geq 0 \tag{5.1.6}$$

$$\Leftrightarrow \dot{z}_x \geq \check{\dot{z}}_x = \frac{\check{\xi}_v}{1-\bar{\kappa}_\mathrm{p}\bar{z}_y} \tag{5.1.7}$$

5.2. Lateral Acceleration Constraints

The lateral acceleration is derived from the yaw rate in equation (3.4.26). After adding the yaw rate ${}^{\mathfrak{v}}_{\mathrm{p}}\dot{\psi} = \kappa_\mathrm{p}(z_x)\dot{z}_x$ along the reference path and the multiplication with the vehicle velocity, the equation yields the lateral acceleration $a_y = \xi_v\Big(\dot{\xi}_\psi + \kappa_\mathrm{p}(z_x)\dot{z}_x\Big)$. Next, the lateral acceleration is multiplied by ξ_v, resulting in the inequalities

$$-\bar{a}_y\xi_v \leq \xi_v^2\Big(\dot{\xi}_\psi + \kappa_\mathrm{p}(z_x)\dot{z}_x\Big) \leq \bar{a}_y\xi_v. \tag{5.2.1}$$

The right inequality in equation (5.2.1) is transformed into a positivity constraint, where $\dot{\xi}_\psi$ is substituted from equation (3.4.26). The expression is further simplified with the function $\tilde{a}_y : \mathbb{R} \mapsto \mathbb{R}$, which includes the curvature rate $\kappa_\mathrm{p}' : \mathbb{R} \mapsto \mathbb{R}$:

$$\tilde{a}_y = \kappa_\mathrm{p}(z_x)\xi_v^2 + \dot{z}_y\Big(\kappa_\mathrm{p}'(z_x)\dot{z}_x z_y + \kappa_\mathrm{p}(z_x)\dot{z}_y\Big), \tag{5.2.2}$$

$$\overset{\text{eq. (3.4.26)}}{\Rightarrow} \xi_v^2\dot{\xi}_\psi \coloneqq (\ddot{z}_y\dot{z}_x - \ddot{z}_x\dot{z}_y)\left(1-\kappa_\mathrm{p}(z_x)z_y\right) + \dot{z}_x\left(\tilde{a}_y - \kappa_\mathrm{p}(z_x)\xi_v^2\right). \tag{5.2.3}$$

The positivity constraint finally yields equation (5.2.4). The left-hand side is lower bounded by substituting a lower bound $\xi_v \coloneqq \dot{z}_x\Big(1-\kappa_\mathrm{p}(z_x)z_y\Big)$ on the velocity to get equation (5.2.5).

$$\bar{a}_y\xi_v - \tilde{a}_y\dot{z}_x - (\ddot{z}_y\dot{z}_x - \ddot{z}_x\dot{z}_y)\left(1-\kappa_\mathrm{p}(z_x)z_y\right) \geq 0 \tag{5.2.4}$$

$$\Rightarrow \Big[(\bar{a}_y - \ddot{z}_y)\left(1-\kappa_\mathrm{p}(z_x)z_y\right) - \tilde{a}_y\Big]\dot{z}_x + \Big(1-\kappa_\mathrm{p}(z_x)z_y\Big)\ddot{z}_x\dot{z}_y \geq 0 \tag{5.2.5}$$

Similar to the velocity constraints, bounds on the involved trajectories are substituted to lower bound the left-hand side of the equation (5.2.5). Aiming at a reasonable compromise between performance and complexity, the functions contributing to a high constraint sensitivity are retained, including $\ddot{z}_x$, $\dot{z}_x$, and $\ddot{z}_y$. The function $\tilde{a}_y$ is substituted with

its bound $\bar{\tilde{a}}_y \in \mathbb{R}^+$, considering the curvature rate bound $\bar{\kappa}'_\mathrm{p} \in \mathbb{R}^+$: $\tilde{a}_y \coloneqq \bar{\tilde{a}}_y = \bar{\kappa}_\mathrm{p}\hat{\xi}_v^2 + \bar{\dot{z}}_y\left(\bar{\kappa}'_\mathrm{p}\hat{\dot{z}}_x\bar{z}_y + \bar{\kappa}_\mathrm{p}\bar{\dot{z}}_y\right)$. Two cases are considered due to a sign switch of $\ddot{z}_x$:

$$\Big(\left(\bar{a}_y - \ddot{\bar{z}}_y\right)\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_y\Big)\dot{z}_x - \left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right)\ddot{z}_x\bar{\dot{z}}_y \geq 0, \qquad \text{if } \ddot{z}_x \geq 0, \tag{5.2.6}$$

$$\Big(\left(\bar{a}_y - \ddot{\bar{z}}_y\right)\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_y\Big)\dot{z}_x + \left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right)\ddot{z}_x\bar{\dot{z}}_y \geq 0, \qquad \text{if } \ddot{z}_x < 0. \tag{5.2.7}$$

The left inequality in equation (5.2.1) yields the lower bound constraint in equation (5.2.8) after the transformation to a positivity constraint and the substitution of equation (5.2.3). The substitution $\xi_v \coloneqq \dot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)$ results in equation (5.2.9).

$$\bar{a}_y\xi_v + \tilde{a}_y\dot{z}_x + \left(\ddot{z}_y\dot{z}_x - \ddot{z}_x\dot{z}_y\right)\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big) \geq 0 \tag{5.2.8}$$

$$\Rightarrow \Big[\left(\bar{a}_y + \ddot{\bar{z}}_y\right)\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big) + \tilde{a}_y\Big]\dot{z}_x + \Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)\ddot{z}_x\dot{z}_y \geq 0 \tag{5.2.9}$$

Like the upper bound constraint, the left-hand side of equation (5.2.9) is lower bounded by substituting bounds on the involved functions and preserving $\ddot{z}_x$, $\dot{z}_x$, and $\ddot{z}_y$. Two cases are considered for the lower bound:

$$\Big(\left(\bar{a}_y + \ddot{\bar{z}}_y\right)\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_y\Big)\dot{z}_x - \left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right)\ddot{z}_x\bar{\dot{z}}_y \geq 0, \qquad \text{if } \ddot{z}_x \geq 0, \tag{5.2.10}$$

$$\Big(\left(\bar{a}_y + \ddot{\bar{z}}_y\right)\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_y\Big)\dot{z}_x + \left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right)\ddot{z}_x\bar{\dot{z}}_y \geq 0, \qquad \text{if } \ddot{z}_x < 0. \tag{5.2.11}$$

5.3. Longitudinal Acceleration Constraints

The absolute acceleration at the tires cannot exceed the friction force in equation (3.4.23), which bounds the acceleration input in dependence on the maximum lateral acceleration in equation (5.3.1). The breaks are assumed to utilize the full acceleration potential, resulting in the lower bound $-\check{u}_a \in \mathbb{R}^+$ in equation (5.3.2).

$$|u_a| \leq \sqrt{(\mu_\mathrm{ad}g)^2 - \bar{a}_y{}^2} \tag{5.3.1}$$

$$\Rightarrow \check{u}_a = -\sqrt{(\mu_\mathrm{ad}g)^2 - \bar{a}_y{}^2} \tag{5.3.2}$$

The available engine torque limits the forward acceleration capability, inducing a distinct upper bound $\hat{u}_a \in \mathbb{R}^+$. The acceleration input to the vehicle from equation (3.4.27) is bounded by

$$\check{u}_a\xi_v \leq \Big[\ddot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)^2 - \tilde{a}_x\Big]\dot{z}_x + \ddot{z}_y\dot{z}_y \leq \hat{u}_a\xi_v. \tag{5.3.3}$$

The function $\tilde{a}_x : \mathbb{R} \mapsto \mathbb{R}$ simplifies the expression:

$$\tilde{a}_x = \dot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)\Big(\kappa'_\mathrm{p}(z_x)\dot{z}_x z_y + \kappa_\mathrm{p}(z_x)\dot{z}_y\Big). \tag{5.3.4}$$

The right inequality in equation (5.3.3) is transformed into a positivity constraint. The velocity ratio $r_v : \mathbb{R} \mapsto \mathbb{R}$

$$r_v = \frac{\dot{z}_y}{\dot{z}_x} = \Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)\tan\xi_\psi \tag{5.3.5}$$

enables the substitution $\dot{z}_y \coloneqq \dot{z}_x r_v$. In addition, the substitution $\xi_v \coloneqq \check{\xi}_v = \dot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)$, resulting from equation (5.1.5), is applied to equation (5.3.3). All terms are divided by $\dot{z}_x$ to get

$$\hat{u}_a\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big) + \tilde{a}_x - \ddot{z}_y r_v - \ddot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)^2 \geq 0. \tag{5.3.6}$$

The final constraint expression is obtained similar to section 5.2 by underestimating the left-hand side of equation (5.3.6) with bounds on the involved functions, so $\ddot{z}_x$ and $\ddot{z}_y$ remain. The substitution $\tilde{a}_x \coloneqq -\bar{\tilde{a}}_x = -\hat{\dot{z}}_x\left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right)\left(\bar{\kappa}'_\mathrm{p}\hat{\dot{z}}_x\bar{z}_y + \bar{\kappa}_\mathrm{p}\hat{\ddot{z}}_y\right)$ uses the bound $\bar{\tilde{a}}_x \in \mathbb{R}^+$. Since $\ddot{z}_y$ can switch sign, two cases are considered. The substitution $r_v \coloneqq \bar{r}_v = (1 + \bar{\kappa}_\mathrm{p}\bar{z}_y)\tan\bar{\xi}_\psi$ yields equation (5.3.7) with the bound $\bar{r}_v \in \mathbb{R}^+$. The second case, considered in equation (5.3.8), uses $r_v \coloneqq -\bar{r}_v$.

$$\hat{u}_a\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_x - \ddot{z}_y\bar{r}_v - \ddot{z}_x(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y)^2 \geq 0, \qquad \text{if } \ddot{z}_y \geq 0, \tag{5.3.7}$$

$$\hat{u}_a\left(1 - \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_x + \ddot{z}_y\bar{r}_v - \ddot{z}_x(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y)^2 \geq 0, \qquad \text{if } \ddot{z}_y < 0. \tag{5.3.8}$$

The left inequality in equation (5.3.3) is also reformulated to a positivity constraint and undergoes the same substitutions of $\dot{z}_y$ and ξ_v as the upper bound. After eliminating $\dot{z}_x$, the lower bound constraint becomes

$$-\check{u}_a\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big) - \tilde{a}_x + \ddot{z}_y r_v + \ddot{z}_x\Big(1 - \kappa_\mathrm{p}(z_x)z_y\Big)^2 \geq 0. \tag{5.3.9}$$

In contrast to the upper bound, $\tilde{a}_x \coloneqq \bar{\tilde{a}}_x$ applies. Equations (5.3.10) and (5.3.11) result from the substitution of the respective velocity ratio bounds $r_v \coloneqq -\bar{r}_v$ and $r_v \coloneqq \bar{r}_v$:

$$-\check{u}_a\left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_x - \ddot{z}_y\bar{r}_v + \ddot{z}_x(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y)^2 \geq 0, \qquad \text{if } \ddot{z}_y \geq 0, \tag{5.3.10}$$

$$-\check{u}_a\left(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y\right) - \bar{\tilde{a}}_x + \ddot{z}_y\bar{r}_v + \ddot{z}_x(1 + \bar{\kappa}_\mathrm{p}\bar{z}_y)^2 \geq 0, \qquad \text{if } \ddot{z}_y < 0. \tag{5.3.11}$$

The feasible accelerations are depicted in figure 5.1 for different velocities. The constraints on u_a are linear. The constraints on a_y are bilinear due to a multiplication of $\dot{z}_x$ with $\ddot{z}_y$. Consequently, the feasible state space grows with the velocity.

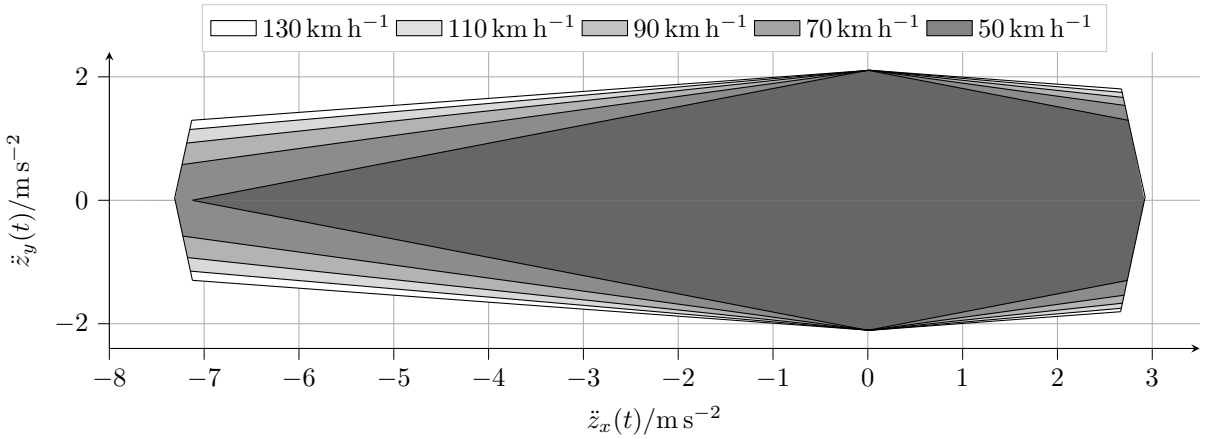


Figure 5.1.: The areas mark the connected feasible acceleration sets at different velocities $\dot{z}_x$.

5.4. Other Vehicle Constraints

The vehicle shapes are described by axis-aligned ellipses assuming a limited heading angle deviation from the reference path, as described in section 4.2.4. The other vehicles' shapes are inflated with the EV shape and a growing prediction uncertainty. The collision constraints require the origin to stay outside the other vehicles' ellipses, which are translated by the EV position. An equivalence transformation obtains polynomial constraints for the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles:

$$\frac{(z_x - z_{x,m})^2}{\tilde{\Delta}_{x,m}} + \frac{(z_y - z_{y,m})^2}{\tilde{\Delta}_{y,m}} - 1 \geq 0, \tag{5.4.1}$$

$$\Leftrightarrow (z_x - z_{x,m})^2 \, \tilde{\Delta}_{y,m} + (z_y - z_{y,m})^2 \, \tilde{\Delta}_{x,m} - \tilde{\Delta}_{x,m}\tilde{\Delta}_{y,m} \geq 0. \tag{5.4.2}$$

A linear increase in the ellipses semi-axes lengths is assumed to parameterize the uncertainty inflations ${}_{\epsilon}\Delta_{\triangledown,m}$ with $\triangledown = x, y$:

$${}_{\epsilon}\Delta_{x,m} = \hat{\dot{z}}_x \overset{\circ}{t}_{\mathrm{hw}} \frac{t}{H}, \tag{5.4.3}$$

$${}_{\epsilon}\Delta_{y,m} = \overset{\circ}{\Delta}_y \frac{t}{H}. \tag{5.4.4}$$

The approach is similar to the linear growing position covariance around the predicted trajectories proposed by [Sch+18]. In the x-direction, the maximum inflation $\overset{\circ}{t}_{\mathrm{hw}} \in [0, t_{\mathrm{hw}}\check{\dot{z}}_x/\hat{\dot{z}}_x)$ at the prediction horizon end still allows to follow a leading vehicle at the minimum feasible velocity. The maximum lateral inflation $\overset{\circ}{\Delta}_y \in \mathbb{R}^+$ is limited so that the EV can pass another vehicle on a neighboring lane. The parameters involved in the collision constraints are given in table A.8.
As the planned trajectory reaches a lane center at $t \in [T_y, H]$, it has to stay in the lane center until the prediction horizon's end. Thus, it is sufficient to check for a collision with the next vehicle in front in equation (5.4.5) and to the rear in equation (5.4.6) on the same lane in the x-direction. The constraints demanding clearance of the elliptical shapes apply to the time interval $t \in [0, \max\{T_x, T_y\}]$.

$$z_{x,m} - z_x - \sqrt{\tilde{\Delta}_{x,m}} \geq 0, \qquad \text{next front vehicle } m = 1, 2, \ldots, \eta_{\mathrm{ov}} \tag{5.4.5}$$

$$z_x - z_{x,m} - \sqrt{\tilde{\Delta}_{x,m}} \geq 0, \qquad \text{next rear vehicle } m = 1, 2, \ldots, \eta_{\mathrm{ov}} \tag{5.4.6}$$

5.5. Heading Angle Constraints

The collision constraints assume a limited heading angle deviation from the reference path. While the heading of the other vehicles cannot be controlled, the EV's heading angle ξ_ψ from equation (3.4.26) shall not exceed the symmetric bound $\bar{\xi}_\psi \in \mathbb{R}^+$:

$$-\bar{\xi}_\psi \leq \arctan \frac{\dot{z}_y}{\dot{z}_x\Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big)} \leq \bar{\xi}_\psi. \tag{5.5.1}$$

The upper bound constraint (5.5.2) is derived from the right inequality in equation (5.5.1). The functions $\kappa_{\mathrm{p}}(z_x) \coloneqq \bar{\kappa}_{\mathrm{p}}$ and $\dot{z}_y \coloneqq \bar{z}_y$ are substituted to lower bound the left-hand side

of the equation (5.5.2), yielding the constraint expression (5.5.3). Thus, the heading angle constraint is a linear function of $\dot{z}_x$ and $\dot{z}_y$.

$$\Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big) \dot{z}_x \tan \bar{\xi}_\psi - \dot{z}_y \geq 0 \tag{5.5.2}$$

$$\Rightarrow (1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y)\, \dot{z}_x \tan \bar{\xi}_\psi - \dot{z}_y \geq 0 \tag{5.5.3}$$

The lower bound constraint is derived from the left inequality in equation (5.5.1). The substitutions applied to the upper bound constraint are applied to the lower bound equation (5.5.4) to obtain equation (5.5.5).

$$\Big(1 - \kappa_{\mathrm{p}}(z_x) z_y\Big) \dot{z}_x \tan \bar{\xi}_\psi + \dot{z}_y \geq 0 \tag{5.5.4}$$

$$\Rightarrow (1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y)\, \dot{z}_x \tan \bar{\xi}_\psi + \dot{z}_y \geq 0 \tag{5.5.5}$$

6

Local Optimal Trajectory Planning

The spline-based trajectory parameterization from section 4.3 is used to parameterize the constraints in chapter 5. Consequently, the problem 4.3.1 is reformulated to a sparse NLP, which is solved at the core of the LP. The solution achieves a high control performance and resembles an inner approximation to problem 4.3.2 for a given LT. The convergence to an LT is guaranteed by a shrinking horizon approach. It ensures recursive feasibility under nominal conditions. The stability, complexity, and control performance are demonstrated in two scenarios. Parts of this chapter have been published in [Dor+23].

6.1. Requirements

In the hierarchy of automated driving in figure 1.2, the motion planning module finds a trajectory toward an LT defined by the preceding behavior module. In this chapter, the behavior module is assumed to exist. It provides one of the $m \in \mathcal{I}_\mathrm{T}$ LTs $\tilde{\mathcal{T}}_{x,m} \times \tilde{\mathcal{T}}_{y,m} \subset \tilde{\mathcal{T}}$ to the LP. Furthermore, it provides an initial guess in the form of a suboptimal spline trajectory. The trajectory terminates in the LT and may be infeasible for the given constraints. The goal is to develop and implement a trajectory planning algorithm, which returns a locally optimal trajectory. The planner shall achieve a high control performance during convergence into the selected LT at a low computational complexity.

6.1.1. Stability Requirements

The terminal cost in the equation (4.3.8) estimates the cost-to-go to the GT. The terminal cost is assumed sensitive to $m \in \mathcal{I}_\mathrm{T}$ but not to the spline coefficients and breakpoints. Thus, the NLP does not have to consider the terminal cost in its cost function.
Even if the stability requirements in definition 3.3.4 and section 4.1.2 are fulfilled, stability might still not hold when using a spline-based trajectory parameterization. If the parameterization is not considered correctly, the EV possibly visits states where no feasible solution exists and recursive feasibility does not hold.

Definition 6.1.1 (Recursive feasibility [GP11, p. 49])**:** If at a time step $j \in \mathbb{N}_0$ a state $\underline{\xi}_c(j) \in \mathcal{C}_0$ is feasible for problem 1.2.1, the next closed-loop state $\underline{\xi}_c(j+1) \in \mathcal{C}_0$ remains feasible.

Consequently, a solution to problem 1.2.1 always exists if the initial state yields a feasible solution. Thus, recursive feasibility is a necessary condition for asymptotic stability. Recursive feasibility applies if the segments $t \in [\Delta t, H + \Delta t]$ of the current open-loop trajectories $\underline{\xi}_j^*(t) \in \mathcal{C}_0$ and $\underline{u}_{c,j}^*(t) \in \mathcal{U}_0$ at time step $j \in \mathbb{N}_0$ are still feasible solutions at the next state $\underline{\xi}_c(j+1)$. The terminal part $t \in [H, H + \Delta t]$ is an extension along the terminal set, so $\underline{\xi}_j^*(t) \in \mathcal{T}$. In this case, the optimal solution is not necessary and initial feasibility suffices for asymptotic stability [Sco+99, th. 1]. The desired property may not be achieved in all situations if a uniform spline trajectory parameterization is applied. Instead, a nonuniform spline parameterization is proposed to refine the breakpoints, so the set of feasible splines from the previous time step is retained [Boo01, eq. XI(16)].

6.1.2. Performance and Complexity Requirements

The initial guess and the terminal set determine the homotopy class of the LP's solution trajectory. The LP's task is to improve the initial guess in the given homotopy class. The requirement for a locally optimal solution motivates the application of a numerical optimization algorithm. In contrast to sampling-based algorithms, algorithms based on numerical optimization generally scale better with the degrees of freedom and are not restricted to a discrete set of solutions.

The semi-infinite constraints of problem 4.3.2 are formulated as polynomial functions of the spline parameterized trajectory in chapter 5, aiming at a finite-dimensional optimization problem. Thus, the constraint functions are still splines in basis form. Requiring positive spline coefficients achieves a feasible solution to problem 4.3.2. Ensuring the recursive feasibility property does not allow time scaling, as applied by [Mer+18b]. However, a nonuniform parameterization denies an offline computation of the coefficient transformations. Thus, the transformations are performed efficiently online while solving the NLP.

6.2. Spline-Based Optimal Control Problem

Problem 4.3.2 is turned into a finite-dimensional OCP in the framework of the LP, as described by problem 6.2.1. Therefore, all time-dependent functions are parameterized with spline functions. The inequality constraints introduced in chapter 5 are parameterized by splines as well. Demanding the positivity of their coefficients ensures constraint feasibility at all times.

Problem 6.2.1 (Nonlinear program)**:** For a given transformed LT with index $m \in \mathcal{I}_\mathrm{T}$, the spline parameterization of the constraints from chapter 5 and the involved trajectories

result in

$$\min_{\underline{o},\underline{\mathbf{k}}_x,\underline{\mathbf{k}}_y} \sum_{\nabla=x,y} \int_0^{T_\nabla} w_{T,\nabla} + w_{\mathrm{j},\nabla} s_{\dddot{\nabla}^2}(t, \underline{\mathbf{c}}_{\dddot{\nabla}^2}, \underline{\mathbf{k}}_\nabla)\, \mathrm{d}t, \tag{6.2.1}$$

subject to

State bounds	**Boundary conditions**	**Derivative constr.**
$\check{\dot{z}}_x \geq \underline{\mathbf{c}}_{\dot{x}} \geq \hat{\dot{z}}_x,$	$\mathbf{c}_{0,\nabla^{(\iota)}} = \mathbf{z}_{\iota,\nabla,0},$	$\underline{g}_{\nabla^{(l)}}(\underline{\mathbf{c}}_{\nabla^{(l)}}, \underline{\mathbf{c}}_{\nabla^{(l-1)}}, \underline{\mathbf{k}}_\nabla) = \underline{0},$
$-\bar{\dot{z}}_y \geq \underline{\mathbf{c}}_{\dot{y}} \geq \bar{\dot{z}}_y,$	$\iota = 0,1,2,\ \nabla = x,y,$	$l = 1,2,3,\ \nabla = x,y,$
$\Delta_y + \underline{\mathbf{c}}_y \leq z_{y,\mathrm{l}},$	$\underline{g}_{\nabla,m,\iota}(\underline{\mathbf{c}}_{\nabla^{(\iota)}}, \underline{\mathbf{k}}_\nabla) = \underline{0},$	**Spline constr.** $\underline{g}_{\mathrm{b}}(\underline{o}, \underline{\mathbf{k}}_x, \underline{\mathbf{k}}_y) = \underline{0},$
$-\Delta_y + \underline{\mathbf{c}}_y \geq z_{y,\mathrm{r}},$	$\iota = 0,1,2,\ \nabla = x,y,$	$o_j \geq 0,\ j \in \mathcal{I}_h,$
$\mathbf{k}_{n,xy} - \mathbf{k}_{n-1,xy} \geq \Delta\check{\mathbf{k}},$	$n = 0,1,\ldots,\breve{\mathbf{k}}_{xy} - 1$	**Breakpoint constr.**

(6.2.2)

The optimization variables are composed of all spline coefficients $\underline{o} \in \mathbb{R}^{\check{o}}$ and the breakpoints $\underline{\mathbf{k}}_x$ and $\underline{\mathbf{k}}_y$. Thus, the variables include the coefficients of the EV position trajectory and its time derivatives $z_\nabla^{(j)}(t) \coloneqq s_{\nabla^{(j)}}(t)$ up to the 3rd order $j = 0,1,2,3$ in the $\nabla = x,y$ direction. The variables also include the coefficients related to the inequality constraint splines. The NLP is implemented via the Python [RD09] interface of the open-source automatic differentiation framework CasADi [And+18]. The gradients are supplied to the interior point optimization algorithm Interior Point Optimizer (IPOPT) [WB05], which solves sparse large-scale optimization problems efficiently. IPOPT uses the linear solver Multifrontal Massively Parallel Solver (MUMPS) [Ame+00] to compute the update for the optimization variables. Section A.3 provides the non-default IPOPT parameter choices. The parameters used in the NLP formulation are summarized in table A.6. The constraints of problem 6.2.1 and the corresponding spline functions are detailed below.

Breakpoint Constraints

The EV motion is described in $\nabla = x,y$ direction by the splines $s_\nabla(t)$ with the breakpoints $\underline{\mathbf{k}}_\nabla$. The second-last breakpoints provide the control horizons $T_\nabla \coloneqq \mathbf{k}_{\breve{\mathbf{k}}_\nabla - 2,\nabla}$. Both breakpoint vectors share the same initial time $\mathbf{k}_{0,\nabla} \coloneqq 0$ and prediction horizon $\mathbf{k}_{\breve{\mathbf{k}}_\nabla - 1,\nabla} \coloneqq H$. The implementation of constraints involving the trajectories in both directions requires a combination of the breakpoints to $\underline{\mathbf{k}}_{xy} \coloneqq \underline{\mathbf{k}}_x \bigcup_{<} \underline{\mathbf{k}}_y$ in a sorted, strictly increasing manner. The assumption of a strictly increasing sequence $\underline{\mathbf{k}}_{xy}$ mentioned in section 3.5 holds because of the minimum breakpoint interval $\Delta\check{\mathbf{k}} \in \mathbb{R}^+$. The optimization algorithm cannot change the ordering of the breakpoints and their count, which are determined by the initial guess.

Derivative Constraints

New splines are introduced for $j = 1,2,3$ time derivatives of the position trajectory. The splines' coefficients are subject to optimization and coupled by the transformations $\underline{T}_{\underline{b}_{\nabla(j)}}^{\underline{b}_{\nabla(j-1)}}$, introduced in equation (3.5.9). The transformations are considered in the constraint functions $\underline{g}_{\nabla(j)} : \mathbb{R}^{\check{\mathbf{c}}_{\nabla(j)}} \times \mathbb{R}^{\check{\mathbf{c}}_{\nabla(j-1)}} \times \mathbb{R}^{\check{\mathbf{k}}_\nabla} \mapsto \mathbb{R}^{\check{\mathbf{c}}_{\nabla(j)}}$ in $\nabla = x,y$ direction:

$$\underline{g}_{\nabla(j)}(\underline{\mathbf{c}}_{\nabla(j)}, \underline{\mathbf{c}}_{\nabla(j-1)}, \underline{\mathbf{k}}_\nabla) = \underline{\mathbf{c}}_{\nabla(j)} - \underline{T}_{\underline{b}_{\nabla(j)}}^{\underline{b}_{\nabla(j-1)}} \underline{\mathbf{c}}_{\nabla(j-1)}. \tag{6.2.3}$$

The derivative splines maintain distinct coefficients, but they share the same breakpoints $\underline{\mathbf{k}}_{\nabla^{(j)}} \coloneqq \underline{\mathbf{k}}_{\nabla}$.

State Bounds

The spline coefficients are bounded to impose limits on the EV velocity and position. Bounds are applied to the x- and y-velocity, and the y-position. The latter represents the left $z_{y,\mathrm{l}} \in [-\bar{z}_y, \bar{z}_y]$ and right $z_{y,\mathrm{r}} \in [-\bar{z}_y, z_{y,\mathrm{l}}]$ non-traversable lane boundaries, so the EV stays in the drivable road area.

Boundary Conditions

Additional equality constraints ensure that the planned trajectory starts at the initial state and ends in one of the $m = -2, -1, \dots, \eta_{\mathrm{ov}}$ LTs. The initial condition demands the equality $\mathbf{c}_{0,\nabla^{(\iota)}} = \mathbf{z}_{\iota,\nabla,0}$ for the order $\iota = 0, 1, 2$ time derivatives in $\nabla = x, y$ direction. The equality constraints $\underline{g}_{\nabla,m,\iota} : \mathbb{R}^{\check{\mathbf{c}}_{\nabla^{(\iota)}}} \times \mathbb{R}^{\check{\mathbf{k}}_{\nabla}} \mapsto \mathbb{R}^2$ ensure that the terminal spline segment is retained in the selected LT. The LTs in the x-direction, which follow the $m = 1, 2, \dots, \eta_{\mathrm{ov}}$ other vehicle target trajectories, are described by the splines $s_{\tilde{x},m}(t) = s_{x,m}(t) - t_{\mathrm{hw}} \dot{s}_{x,m}(t)$. $z_{\tilde{x},m}(t) \coloneqq s_{\tilde{x},m}(t)$ denotes the spline parameterization of the target trajectories and $z_{x,m}(t) \coloneqq s_{x,m}(t)$ the parameterized predicted motion in the x-direction. The remaining LTs, $m = -2, -1, 0$, include the target velocity. As an exception, the terminal constraint $\underline{g}_{x,0,0} : \mathbb{R}^{\check{\mathbf{c}}_{\nabla^{(\iota)}}} \times \mathbb{R}^{\check{\mathbf{k}}_{\nabla}} \mapsto \mathbb{R}$ maps to a scalar value.

$$\underline{g}_{x,m,\iota}(\underline{\mathbf{c}}_{x^{(\iota)}}, \underline{\mathbf{k}}_x) = \left[s_{x^{(\iota)}}(t) - s^{(\iota)}_{\tilde{x},m}(t) \right]^{\intercal}_{t=T_x,H}, \qquad m = 1, 2, \dots, \eta_{\mathrm{ov}}, \quad \iota = 0, 1, 2 \tag{6.2.4}$$

$$\underline{g}_{x,m,0}(\underline{\mathbf{c}}_x, \underline{\mathbf{k}}_x) = s_x(H) - s_x(T_x) - (H - T_x)\tilde{v}, \quad m = -2, -1, 0 \tag{6.2.5}$$

$$\underline{g}_{x,m,1}(\underline{\mathbf{c}}_{\dot{x}}, \underline{\mathbf{k}}_x) = \begin{bmatrix} s_{\dot{x}}(T_x) & s_{\dot{x}}(H) \end{bmatrix}^{\intercal} - \tilde{v}, \qquad m = -2, -1, 0 \tag{6.2.6}$$

$$\underline{g}_{x,m,2}(\underline{\mathbf{c}}_{\ddot{x}}, \underline{\mathbf{k}}_x) = \begin{bmatrix} s_{\ddot{x}}(T_x) & s_{\ddot{x}}(H) \end{bmatrix}^{\intercal}, \qquad m = -2, -1, 0 \tag{6.2.7}$$

The LTs in the y-direction either include the lane centers with a lateral offset $d_m(t)$ in equation (6.2.8), associated with the other vehicles $m = 1, 2, \dots, \eta_{\mathrm{ov}}$, or one specific center $\square = \mathrm{lc}, \mathrm{mc}, \mathrm{rc}$ in equation (6.2.9). The equation (6.2.10) ensures the absence of a lateral motion beyond T_y.

$$\underline{g}_{y,m,0}(\underline{\mathbf{c}}_y, \underline{\mathbf{k}}_y) = \left[s_y(t) - d_m(t) \right]^{\intercal}_{t=T_y,H}, \qquad m = 1, 2, \dots, \eta_{\mathrm{ov}} \tag{6.2.8}$$

$$\underline{g}_{y,m,0}(\underline{\mathbf{c}}_y, \underline{\mathbf{k}}_y) = \left[s_y(t) - d_{\square} \right]^{\intercal}_{t=T_y,H}, \qquad (\square, m) = (\mathrm{lc}, -2), (\mathrm{mc}, -1), (\mathrm{rc}, 0) \tag{6.2.9}$$

$$\underline{g}_{y,m,\iota}(\underline{\mathbf{c}}_{y^{(\iota)}}, \underline{\mathbf{k}}_y) = \left[s_{y^{(\iota)}}(t) \right]^{\intercal}_{t=T_y,H}, \qquad m = -2, -1, \dots, \eta_{\mathrm{ov}}, \quad \iota = 1, 2 \tag{6.2.10}$$

Spline Constraints

The coefficient transformations in the more complex polynomial constraints introduced in chapter 5 are not computed explicitly. Because the splines are nonuniform and the breakpoints are subject to optimization, the equation systems determining the transformations of the spline sum (3.5.15) and product (3.5.19) cannot be precomputed but must be solved online. An efficient online solution is achieved by integrating the transformations in the

NLP. Instead of solving the linear equation systems separately in each iteration of the optimization algorithm, the spline interpolations from the equations (3.5.14) and (3.5.17) are introduced as equality constraints:

$$s_{\boxplus}(\tilde{k}_{j,\boxplus}) - s_{\square}(\tilde{k}_{j,\boxplus}) - s_{\triangle}(\tilde{k}_{j,\boxplus}) = 0, \qquad j = 0, 1, \ldots, \breve{\mathbf{k}}_{\boxplus} - 1 \tag{6.2.11}$$

$$s_{\boxtimes}(\tilde{k}_{j,\boxtimes}) - s_{\square}(\tilde{k}_{j,\boxtimes}) \cdot s_{\triangle}(\tilde{k}_{j,\boxtimes}) = 0, \qquad j = 0, 1, \ldots, \breve{\mathbf{k}}_{\boxtimes} - 1. \tag{6.2.12}$$

Each sum or product of two splines $s_{\square}(t)$ and $s_{\triangle}(t)$ is interpolated at the knot averages $\underline{\tilde{k}}_{\diamond}$ of the new interpolating splines $s_{\diamond}(t)$ with $\diamond = \boxplus, \boxtimes$. The latter splines' coefficients are included in the optimization variables o_j with the indices $j \in \tilde{\mathcal{I}}_{\diamond} \subset \mathbb{N}_0$. Simultaneously, the equations' (6.2.11) and (6.2.12) left-hand sides are included in $\underline{g}_{\mathrm{b}} : \mathbb{R}^{\breve{o}} \times \mathbb{R}^{\breve{\mathbf{k}}_x} \times \mathbb{R}^{\breve{\mathbf{k}}_y} \mapsto \mathbb{R}^{\breve{g}_{\mathrm{b}}}$. The order $\rho_{\diamond}$, breakpoints $\underline{\mathbf{k}}_{\diamond}$, and continuities $\underline{w}_{\diamond}$ of the new splines are determined according to the rules for the spline sum and product introduced in section 3.5. The proposed implicit coefficient transformation increases the number of constraints and optimization variables compared to an explicit calculation. However, a sparse problem formulation is obtained.

Aiming at a suitable computational complexity, the number of interpolation sites should be low, while their locations are easily determined. Simultaneously, the interpolation error resulting from numerical errors must be sufficiently small. The knot averages (3.5.7) are a suitable choice in practice [Boo01, p. 192]. The interpolation error tends to increase with the spline order, which should be limited for that reason. Also, the minimum time interval $\Delta\breve{\mathbf{k}}$ cannot be chosen arbitrarily small. Otherwise, the NLP becomes degenerate due to the mutual proximity of interpolation points. The splines that contribute to the constraint implementation are introduced below.

Other vehicles are considered in the equation (5.4.2). At first, the position differences between the EV's trajectory and the other vehicles' predictions are described by the splines $s_{\Delta\nabla,m}(t)$, which are squared subsequently to $s_{\Delta\nabla^2,m}(t)$:

$$s_{\Delta\nabla,m}(t) = s_{\nabla}(t) - s_{\nabla,m}(t), \qquad \nabla = x, y, \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}, \tag{6.2.13}$$

$$s_{\Delta\nabla^2,m}(t) = s^2_{\Delta\nabla,m}(t), \qquad \nabla = x, y, \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}. \tag{6.2.14}$$

$s_{\tilde{\nabla},m}(t)$ describe the products of the ellipses' squared semi-axis lengths $\tilde{\Delta}_{\nabla,m}(t) \coloneqq s_{\tilde{\Delta},\nabla,m}(t)$ in $\nabla = x, y$ direction with the squared position differences:

$$s_{\tilde{x},m}(t) = s_{\Delta x^2,m}(t) s_{\tilde{\Delta},y,m}(t), \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}, \tag{6.2.15}$$

$$s_{\tilde{y},m}(t) = s_{\Delta y^2,m}(t) s_{\tilde{\Delta},x,m}(t), \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}. \tag{6.2.16}$$

The ellipse extends are splines of order $\rho_{\tilde{\Delta},\nabla,m} = 3$ with breakpoints $\underline{\mathbf{k}}_{\tilde{\Delta},\nabla,m} = \underline{\mathbf{k}}_{0H} = \begin{bmatrix} 0 & H \end{bmatrix}^{\mathsf{T}}$. The sum of equations (6.2.15) and (6.2.16) yields $s_{xy,m}(t)$, which is added to the product of the negative semi-axis lengths $\tilde{\Delta}_{x,m}(t)\tilde{\Delta}_{y,m}(t) \coloneqq s_{\tilde{\Delta},xy,m}(t)$ of order $\rho_{\tilde{\Delta},xy,m} = 5$ to get the final collision constraints $s_{\mathrm{ov},m}(t)$:

$$s_{xy,m}(t) = s_{\tilde{x},m}(t) + s_{\tilde{y},m}(t), \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}, \tag{6.2.17}$$

$$s_{\mathrm{ov},m}(t) = s_{xy,m}(t) - s_{\tilde{\Delta},xy,m}(t), \qquad m = 1, 2, \ldots, \eta_{\mathrm{ov}}. \tag{6.2.18}$$

The inequality constraints demand positivity of the $n = 0, 1, \ldots, \breve{\underline{\mathbf{c}}}_{\mathrm{ov},m} - \rho_{\mathrm{ov},m} - 1$ coefficients $\mathbf{c}_{n,\mathrm{ov},m}$, which are equal to o_j with $j \in \tilde{\mathcal{I}}_{\mathrm{ov},m} \subset \mathcal{I}_h$. Therefore, each other vehicle $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ introduces six sets of equality constraints and six sets of spline coefficients to the optimization variables. The last $\rho_{\mathrm{ov},m}$ coefficients are unbounded because they enclose the terminal segment, where the simplified collision constraints (5.4.5) and (5.4.6) apply. The collision constraints along the terminal segment only consider the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' extend in the x-direction, which are described by the splines $\sqrt{\tilde{\Delta}_{x,m}(t)} \coloneqq s_{\tilde{\epsilon},x,m}(t) \coloneqq s_{2,\underline{0}}(t, \underline{\mathbf{c}}_{\tilde{\epsilon},x,m}, \underline{\mathbf{k}}_{0H})$. The constraint considering the next rear vehicle entering the target lane is described by $s_{\mathrm{ov,r}}(t)$. Similarly, $s_{\mathrm{ov,f}}(t)$ resembles the constraint corresponding to the next front vehicle:

$$s_{\mathrm{ov,r}}(t) = s_{\Delta x,m}(t) - s_{\tilde{\epsilon},x,m}(t), \qquad \text{next front vehicle } m = 1, 2, \ldots, \eta_{\mathrm{ov}}, \tag{6.2.19}$$

$$s_{\mathrm{ov,f}}(t) = -s_{\Delta x,m}(t) - s_{\tilde{\epsilon},x,m}(t), \qquad \text{next rear vehicle } m = 1, 2, \ldots, \eta_{\mathrm{ov}}. \tag{6.2.20}$$

Only the $l = \breve{\mathbf{c}}_{\mathrm{ov},\square} - 6, \breve{\mathbf{c}}_{\mathrm{ov},\square} - 5, \ldots, \breve{\mathbf{c}}_{\mathrm{ov},\square} - 1$ coefficients $\mathbf{c}_{l,\mathrm{ov},\square}$, that enclose the terminal segments, are lower bounded variables o_j with $j \in \tilde{\mathcal{I}}_{\mathrm{ov},\square} \subset \mathcal{I}_h$ and $\square = \mathrm{r}, \mathrm{f}$.

Heading angle bounds are implemented using the splines $s_{\hat{\psi}}(t)$ and $s_{\breve{\psi}}(t)$, which represent the upper (5.5.3) and the lower bound (5.5.5) constraints:

$$s_{\hat{\psi}}(t) = \tan \bar{\xi}_{\psi} \Big(1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y\Big) s_{\dot{x}}(t) + s_{\dot{y}}(t), \tag{6.2.21}$$

$$s_{\breve{\psi}}(t) = \tan \bar{\xi}_{\psi} \Big(1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y\Big) s_{\dot{x}}(t) - s_{\dot{y}}(t). \tag{6.2.22}$$

The splines' coefficients coincide with the variables o_j with $j \in \tilde{\mathcal{I}}_{\square} \subset \mathcal{I}_h$ and $\square = \breve{\psi}, \hat{\psi}$.

Lateral acceleration constraint implementation of equations (5.2.6) and (5.2.7), and equations (5.2.10) and (5.2.11) involve a product and a sum. The product yields two intermediate splines distinguishing between the cases $s_{\ddot{y}}(t) \geq 0$ with $s_{\hat{a},\tilde{y}}(t)$ and $s_{\ddot{y}}(t) < 0$ with $s_{\breve{a},\tilde{y}}(t)$:

$$s_{\hat{a},\tilde{y}}(t) = \Big[\Big(\bar{a}_y - s_{\ddot{y}}(t)\Big)(1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y) - \bar{\bar{a}}_y\Big] s_{\dot{x}}(t), \tag{6.2.23}$$

$$s_{\breve{a},\tilde{y}}(t) = \Big[\Big(\bar{a}_y + s_{\ddot{y}}(t)\Big)(1 - \bar{\kappa}_{\mathrm{p}} \bar{z}_y) - \bar{\bar{a}}_y\Big] s_{\dot{x}}(t). \tag{6.2.24}$$

Subsequently, the constraint splines are calculated with four sets of equality constraints. The case $s_{\ddot{x}}(t) \geq 0$ is implemented for the upper and lower bound constraints $\square = \hat{a}, \breve{a}$ with $s_{\square,y,\mathrm{p}}(t)$. Similarly, $s_{\square,y,\mathrm{n}}(t)$ consider the cases $s_{\ddot{x}}(t) < 0$ and $\square = \hat{a}, \breve{a}$:

$$s_{\square,y,\mathrm{p}}(t) = s_{\square,\tilde{y}}(t) - (1 + \bar{\kappa}_{\mathrm{p}} \bar{z}_y) \bar{z}_{\dot{y}} s_{\ddot{x}}(t), \qquad \square = \hat{a}, \breve{a}, \tag{6.2.25}$$

$$s_{\square,y,\mathrm{n}}(t) = s_{\square,\tilde{y}}(t) + (1 + \bar{\kappa}_{\mathrm{p}} \bar{z}_y) \bar{z}_{\dot{y}} s_{\ddot{x}}(t), \qquad \square = \hat{a}, \breve{a}. \tag{6.2.26}$$

The constraint splines' coefficients correspond to the $j \in \tilde{\mathcal{I}}_{\square,y,\triangle} \subset \mathcal{I}_h$ variables o_j, with $\square = \hat{a}, \breve{a}$ and $\triangle = \mathrm{p}, \mathrm{n}$.

Longitudinal acceleration constraints must also distinguish four cases, but they do not require intermediate splines. The splines $s_{\hat{a},x,\mathrm{p}}(t)$ and $s_{\hat{a},x,\mathrm{n}}(t)$ consider the cases $s_{\ddot{x}}(t) \geq 0$ and $s_{\ddot{x}}(t) < 0$, respectively, for the acceleration upper bound constraints in equations (5.3.7) and (5.3.8). The same applies to the lower bound constraints (5.3.10) and (5.3.11), which are parameterized by the splines $s_{\check{a},x,\mathrm{p}}(t)$ and $s_{\check{a},x,\mathrm{n}}(t)$:

$$s_{\hat{a},x,\mathrm{p}}(t) = \hat{u}_a\,(1 - \bar{\kappa}_{\mathrm{p}}\bar{z}_y) - \bar{\bar{a}}_x - s_{\ddot{y}}\bar{r}_v - s_{\ddot{x}}(1 + \bar{\kappa}_{\mathrm{p}}\bar{z}_y)^2, \tag{6.2.27}$$

$$s_{\hat{a},x,\mathrm{n}}(t) = \hat{u}_a\,(1 - \bar{\kappa}_{\mathrm{p}}\bar{z}_y) - \bar{\bar{a}}_x - s_{\ddot{y}}\bar{r}_v + s_{\ddot{x}}(1 + \bar{\kappa}_{\mathrm{p}}\bar{z}_y)^2, \tag{6.2.28}$$

$$s_{\check{a},x,\mathrm{p}}(t) = \check{u}_a\,(1 - \bar{\kappa}_{\mathrm{p}}\bar{z}_y) - \bar{\bar{a}}_x + s_{\ddot{y}}\bar{r}_v - s_{\ddot{x}}(1 + \bar{\kappa}_{\mathrm{p}}\bar{z}_y)^2, \tag{6.2.29}$$

$$s_{\check{a},x,\mathrm{n}}(t) = \check{u}_a\,(1 - \bar{\kappa}_{\mathrm{p}}\bar{z}_y) - \bar{\bar{a}}_x + s_{\ddot{y}}\bar{r}_v + s_{\ddot{x}}(1 + \bar{\kappa}_{\mathrm{p}}\bar{z}_y)^2. \tag{6.2.30}$$

The coefficients of all transformed splines are part of the optimization variables o_j with $j \in \tilde{\mathcal{I}}_{\square,x,\triangle} \subset \mathcal{I}_h$, with $\square = \hat{a}, \check{a}$ and $\triangle = \mathrm{p}, \mathrm{n}$.

Cost function involves the squared jerk $s_{\dddot{\nabla}2}(t) = s^2_{\ddot{\nabla}}(t)$ in $\nabla = x, y$ direction. Thus, two additional sets of equality constraints are dedicated to determining the squared jerk coefficients.

It is sufficient to provide an initial guess for the splines' $s_x(t)$ and $s_y(t)$ coefficients and breakpoints despite the introduction of additional spline coefficients in the optimization variables. The coefficient transformations involved in the spline sums, products, and derivatives, that are implemented via equality constraints in the NLP, are performed explicitly with the position splines' initial coefficients before starting the optimization. Thus, coefficients for all spline functions fulfilling the previously introduced equality constraints are provided to IPOPT.

6.3. Breakpoint Adaption

The nonuniform spline trajectory parameterization enables the recursive feasibility property required in section 6.1.1 if the nominal case applies. The nominal case refers to scenes where the vehicle motion and the environment evolve as expected by the respective model assumptions. Although the nominal case does not hold in practice, the recursive feasibility prevents problem 6.2.1 to not yield a solution due to the splines' limited degrees of freedom or a too short control horizon. An empty feasible set occurs most likely without recursive feasibility if the feasible set at the previous time step is small already. A trajectory near the acceleration bounds, barely avoiding an obstacle, is an exemplary situation where a small feasible set occurs. Also, a constraint that becomes active at the end of a short control horizon might require a solution that violates the input limitations.
Terminal equality constraints are one option for avoiding the issue of a short control horizon [GP11, p. 88]. However, the constraints require a sufficiently long control horizon, so the trajectory ends in the terminal set. A termination in the GT likely requires an intractable control horizon, but a trajectory terminating in one of the LTs is assumed feasible. Still, the splines' breakpoints must comply with additional conditions to achieve recursive feasibility. Consider ${}_j s_\nabla : \mathbb{R} \mapsto \mathbb{R}$, the feasible spline trajectory ${}_j s_\nabla(t) \in {}_j\mathcal{S}_\nabla$,

and the spline space ${}_j\mathcal{S}_\triangledown$ in $\triangledown = x, y$ direction at one of the $j \in 0, 1, \ldots, \breve{t} - 1$ time steps t_j. The space ${}_j\mathcal{S}_\triangledown \coloneqq \mathcal{S}_{\underline{\mathbf{k}}_\triangledown, \underline{w}_\triangledown, \rho_\triangledown}$ includes all splines at t_j with breakpoints $\underline{\mathbf{k}}_\triangledown \coloneqq {}_j\underline{\mathbf{k}}_\triangledown$, continuity $\underline{w}_\triangledown \coloneqq {}_j\underline{w}_\triangledown$, and of order $\rho_\triangledown$. If the existence of ${}_{j+1}s_\triangledown(t) \in {}_{j+1}\mathcal{S}_\triangledown$ at the next step is guaranteed, the recursive feasibility from definition 6.1.1 applies. The condition is fulfilled if ${}_j\mathcal{S}_\triangledown \subseteq {}_{j+1}\mathcal{S}_\triangledown$ as a consequence of a knot refinement [Boo01, eq. XI(16)]. Thus, the feasible spline trajectory from t_j would be among the solutions at t_{j+1}. Concretely, the breakpoints

$$ {}_{j+1}\underline{\mathbf{k}}_\triangledown = \begin{bmatrix} {}_j\mathbf{k}_{0,\triangledown} & {}_j\mathbf{k}_{0,\triangledown} + \Delta t & {}_j\mathbf{k}_{1,\triangledown} & \ldots & T_\triangledown & H + \Delta t \end{bmatrix}^\intercal \tag{6.3.1} $$

are obtained by inserting a breakpoint ${}_j\mathbf{k}_{0,\triangledown} + \Delta t$ in ${}_j\underline{\mathbf{k}}_\triangledown$. Until the next time step, the EV travels along the initial segment ${}_js_\triangledown(t) = {}_{j+1}s_\triangledown(t)$ with $t \in [{}_j\mathbf{k}_{0,\triangledown}, {}_j\mathbf{k}_{0,\triangledown} + \Delta t]$. All subsequent segments are subject to optimization. Still, the breakpoints and coefficients from the previous step can be recovered. Since the OCP relies only on the current vehicle state and does not require knowledge of the previous states, ${}_{j+1}\mathbf{k}_{0,\triangledown}$ can be dropped from the breakpoints ${}_{j+1}\underline{\mathbf{k}}_\triangledown$. The planning algorithm operates in a shrinking horizon manner toward the control horizons. The initial segment is shortened by Δt in each time step. If the nominal case applies, a feasible initial guess is always derived from the previous solution. The feasible solution is sufficient for stability and can provide a backup solution if IPOPT does not converge. The proposed knot refinement does not allow a uniform spline parameterization, which is applied frequently in the literature. Another consequence of the knot refinement is an increasingly tight convex hull [Boo01, props. XI(12)] and a relaxation of the feasible set.

As the initial segment shrinks, the length of the terminal segment is increased by Δt to maintain a constant prediction horizon. In general, an increase in the distance between breakpoints means an increase in the distance between spline coefficients and the spline function. A growing convex hull for the constraint functions is avoided by designing the terminal set, so the constraint splines and their coefficients always coincide along the terminal segment. Therefore, the polynomials along the terminal segment reduce to linear functions. Since the velocity is always constant in the terminal set, the condition is fulfilled for all constraints introduced in section 5.2 except for the other vehicle constraints (5.5.5), which are deactivated along the terminal segment.

In close vicinity to the terminal set, the control horizons become small. However, the sensitivity of the result trajectory to disturbances becomes larger with a shrinking control horizon. The increased sensitivity can result in an undesired oscillation around the LT. A dual-mode control strategy is applied in the sense of [MM93], assuming the existence of a robust trajectory tracking control law for the $\triangledown = x, y$ direction. Instead of the LP, the tracking controller would stabilize the EV if the distance to one of the $m = -2, -1, \ldots, \eta_{\text{ov}}$ LTs $\|\underline{\mathbf{z}}_\triangledown\|_{\tilde{\mathcal{T}}_{\triangledown,m}} \leq \epsilon_\triangledown$ surpasses a threshold $\epsilon_\triangledown \in \mathbb{R}^+$. Below the threshold, the EV state is projected onto the current LT to provide a reference trajectory along the terminal set for the tracking controller. As the controller is not applied in the current implementation, the EV state is instead placed in the LT to exactly follow the reference trajectory.

The NLP requires a minimum distance $\Delta\breve{\mathbf{k}}$ between breakpoints. A further shortening of the initial spline may violate the minimum interval when the breakpoints are close together. Consequently, breakpoints are removed in the shrinking process to maintain feasibility. The process is depicted in figure 6.1, starting with the exemplary splines ${}_0s_\triangledown(t)$

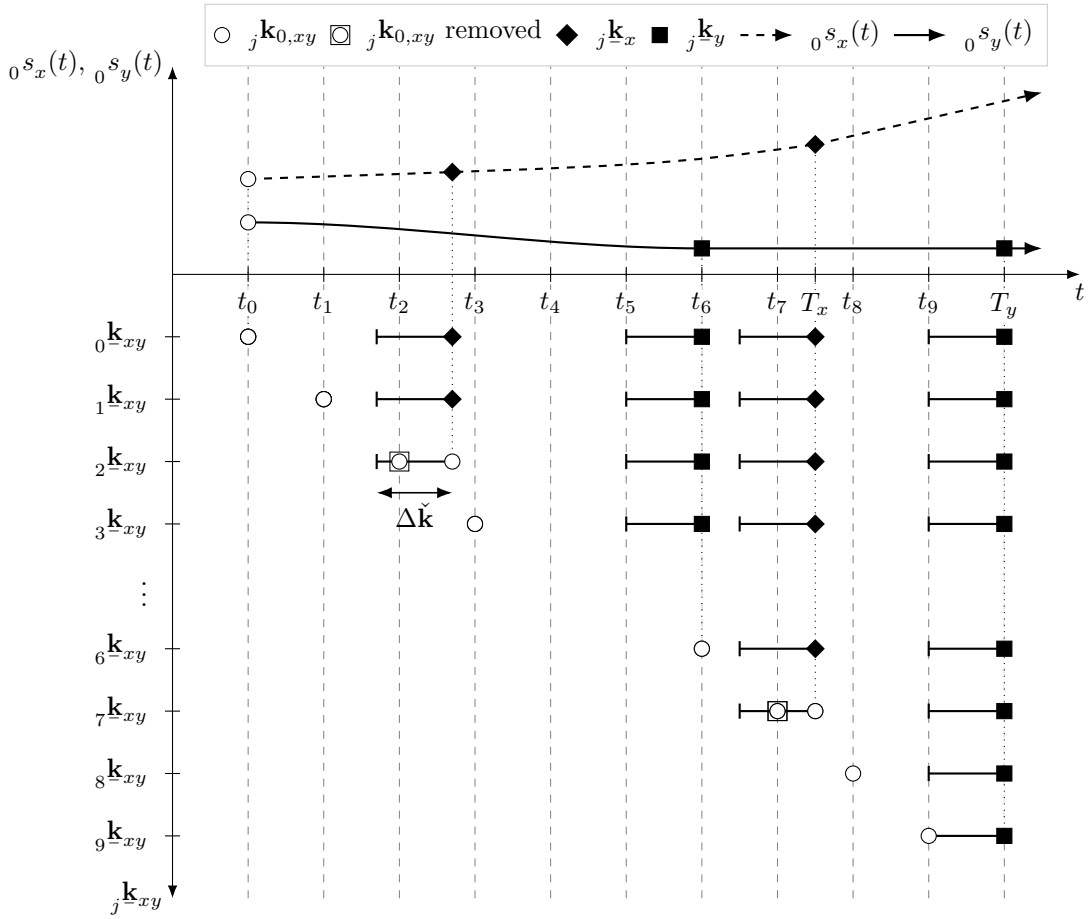


Figure 6.1.: Process of breakpoint removal on the shrinking horizon toward the LT to retain recursive feasibility despite a lower bound on the breakpoint interval. The top part shows splines representing trajectories in the x- and y-direction at time step t_0. The bottom part shows the development of the splines' breakpoints over the time steps until reaching the control horizon T_y.

in $\triangledown = x, y$ direction. The breakpoints are adapted over the $j = 0, 1, \ldots, \breve{t} - 1$ time steps t_j. The splines ${}_js_{\triangledown}(t)$ are not depicted explicitly since they coincide with ${}_0s_{\triangledown}(t)$ on the shrinking control horizons. The splines are fixed to the terminal set at the breakpoints ${}_0\mathbf{k}_{2,\triangledown} = T_{\triangledown}$ and are extended along their terminal segment until the prediction horizon. The breakpoints are combined to ${}_j\underline{\mathbf{k}}_{xy} = {}_j\underline{\mathbf{k}}_x \underset{<}{\cup} {}_j\underline{\mathbf{k}}_y$.

Between each time step, the initial segment of both splines shortens by Δt. After two steps, the minimum interval constraint is violated at t_2. Truncating the initial segment off ${}_2s_x(t)$ retains the recursive feasibility property. Therefore, a breakpoint is inserted along ${}_2s_y(t)$, which becomes the new initial breakpoint ${}_2\mathbf{k}_{0,xy}$. The former initial segment is not subject to optimization and is removed at t_3. In the subsequent steps, the trajectories shrink until the next breakpoint in the y-direction is replaced by the common initial breakpoint at t_6. At t_7, the terminal breakpoint in the x-direction is replaced with a common initial breakpoint. After $j \geq 8$, ${}_js_x(t)$ is only composed of a single polynomial. The same applies to ${}_js_y(t)$ after $j \geq 9$.

6.4. Evaluation

The local planner fulfills several desirable properties, including recursive feasibility and continuous-time feasibility. However, the recursive feasibility property is only guaranteed to hold in the nominal case. The properties are demonstrated in two highway scenarios at the starting point $s2$, shown in figure 3.2. The first one is an overtaking scenario to the left lane, where nominal conditions almost apply. Introducing additional breakpoints increases the trajectory's degrees of freedom to show the possible benefit in control performance and the increase in computational complexity. In the second scenario, the EV starts in the same initial scene and follows a leading vehicle in the right-most lane. The leading vehicle violates the constant velocity assumption and accelerates to a higher speed. Nevertheless, convergence to the LT can be achieved. Both scenarios have a duration of 8 s at a time increment of $\Delta t = 0.1\,\mathrm{s}$. A prediction horizon of $H = 10\,\mathrm{s}$ is used.

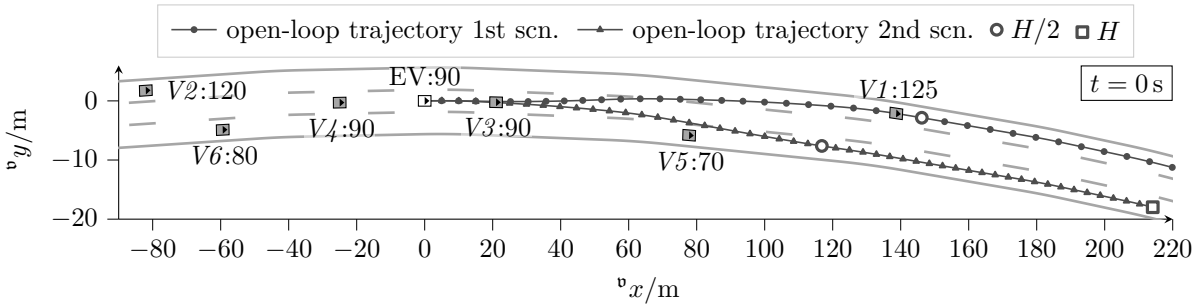


Figure 6.2.: Initial scene with the planned trajectories for the scenario left overtaking (1st scenario) and the scenario right vehicle following (2nd scenario). The EV is indicated by EV: $\dot{z}_x/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles by $Vm : \dot{z}_{x,m}/\mathrm{km\,h^{-1}}$.

The initial scene in both scenarios is depicted in figure 6.2. The EV is located in the middle lane and surrounded by six other vehicles. The vehicles *V1* and *V2* are in the left lane, targeting $122\,\mathrm{km\,h^{-1}}$. The EV is surrounded by *V3* and *V4* on the middle lane, which keep their current velocity. The target velocity of *V5* and *V6* is $80\,\mathrm{km\,h^{-1}}$.
The cost function's decrease over the simulation time is used as a metric to analyze the EV's convergence toward the LT. The decrease is quantified by the function $\tilde{\lambda} : \mathbb{R} \mapsto \mathbb{R}$, which is based on the cost descent requirement (3.3.5), using the upper bound (3.3.6). In contrast, equality is assumed in equation (6.4.1), rendering $\tilde{\lambda}(t_j)$ the current cost descent factor over the $j = 0, 1, \ldots, \breve{t} - 1$ steps:

$$\tilde{\lambda}(t_j) \coloneqq \frac{V\Big(\underline{\xi}_c(t_j, \underline{\xi}_0)\Big) - V\Big(\underline{\xi}_c(t_{j+1}, \underline{\xi}_0)\Big)}{l\Big(\underline{\xi}_c(t_j, \underline{\xi}_0), \underline{u}_c(t_j, \underline{\xi}_0)\Big)}. \tag{6.4.1}$$

The factor provides insights into the convergence properties toward the LT for a given scenario. Four cases are distinguished for the current cost descent factor:

1. $\tilde{\lambda}(t_j) \leq 0$: According to definition 3.3.3, the planning algorithm's cost is not a Lyapunov function for the LT on the visited states $\underline{\xi}_c(t_j, \underline{\xi}_0)$. The previous time step's solution becomes infeasible, and the EV might not progress toward the LT.

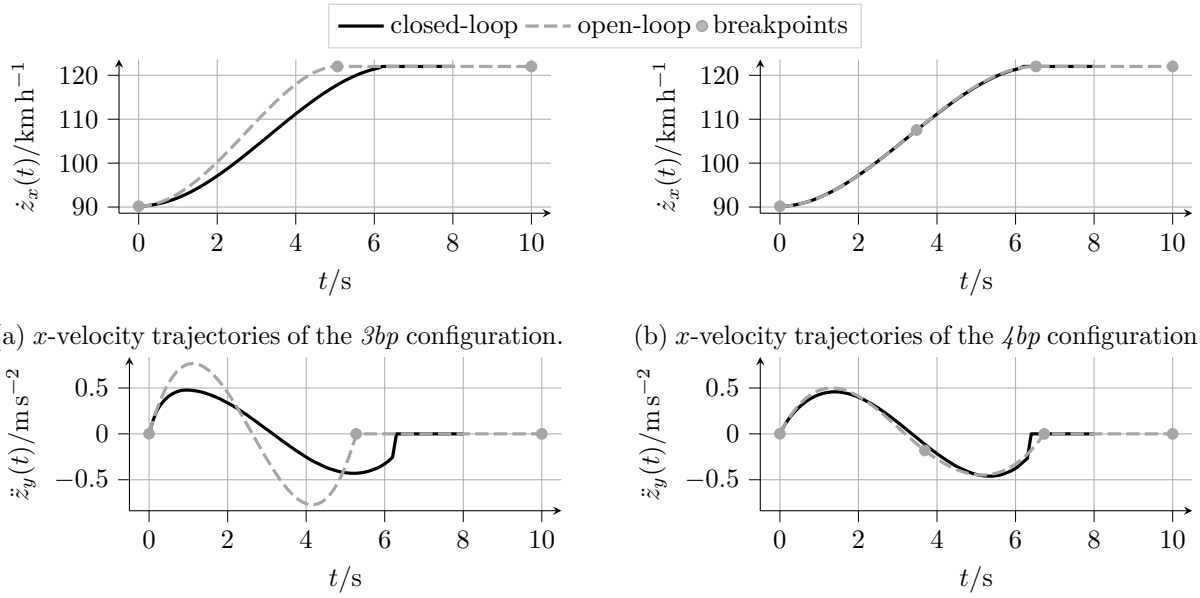
2. $\tilde{\lambda}(t_j) \in (0, 1)$: The cost descent is lower than expected by the principle of optimality. The previous time step's solution becomes infeasible, but the optimization algorithm still finds a solution trajectory to progress toward the LT.

3. $\tilde{\lambda}(t_j) = 1$: The cost value decreases as expected by the principle of optimality. The previous time step's solution is still feasible, resulting in time consistent open-loop trajectories.

4. $\tilde{\lambda}(t_j) > 1$: The cost value decreases faster than expected by the principle of optimality. The previous time step's solution is still feasible, and the feasible state space is relaxed so that the optimization algorithm can improve on the initial guess.

Scenario 1: Left Overtaking

In the first scenario, the EV changes to the left lane and accelerates to $122\,\mathrm{km\,h^{-1}}$. *V3* continues at $90\,\mathrm{km\,h^{-1}}$ and activates the respective collision constraints. Two trajectory parameterizations are compared. The first one uses three breakpoints for each spline trajectory, referred to as the *3bp* configuration. The second configuration, called *4bp*, uses four breakpoints. The initial guess is provided by a squared jerk optimal polynomial interpolation between the initial EV state and the LT. In the x-direction, a polynomial of degree four provides a trajectory toward the target velocity. A spline with three breakpoints is formed by attaching a second polynomial, following the target velocity at the control horizon end, until the prediction horizon is reached. The initial spline in the y-direction is formed similarly. The first polynomial segment of degree five terminates in the target lane center and is extended along the same lane center. The initial control horizon in the x-direction amounts to 8 s and in the y-direction to 9 s since a long control horizon increases the chance of a dynamically feasible initial guess. Still, the resulting spline might result in a collision. The initial guess for the *4bp* configuration is obtained by inserting an additional breakpoint at $H/2$.
The initial guess for both breakpoint configurations violates the collision constraints from *V3*. The constraint violation amounts to $\min\{\mathbf{c}_{j,\mathrm{ov},3} | j = 0, 1, \ldots, \check{\mathbf{c}}_{\mathrm{ov},3} - 1\} = -3.77$ with the *3bp* configuration. An additional breakpoint reduces the distance of the constraint spline from its convex hull, resulting in a lower constraint violation of $\min\{\mathbf{c}_{j,\mathrm{ov},3} | j = 0, 1, \ldots, \check{\mathbf{c}}_{\mathrm{ov},3} - 1\} = -2.38$ when using *4bp*. The *4bp* configuration results in lower open-loop cost than *3bp*, as reported in table 6.1. Open-loop cost refers to the cost of the LP's result trajectory as measured by the cost function (4.3.8). The additional degrees of freedom allow longer control horizons in the x- and y-direction, which can be observed by comparing the open-loop x-velocity and y-acceleration trajectories in figure 6.3. Consequently, the required acceleration and jerk values are reduced, achieving lower overall cost. Simultaneously, an additional breakpoint increases the OCP complexity. Table 6.1 reports the number of constraints, optimization variables, and required iterations until IPOPT converges for both breakpoint configurations at $t = 0\,\mathrm{s}$. The additional breakpoint increases the number of constraints and variables. The *3bp* configuration requires a higher number of iterations until convergence since the feasible set is reduced to the boundary of the unoccupied space and the feasible x-acceleration, leading to numerical problems during the computation of the next iterates with IPOPT due to the strict relative interior



(a) x-velocity trajectories of the *3bp* configuration.

(b) x-velocity trajectories of the *4bp* configuration.

(c) y-acceleration trajectories of the *3bp* configuration.

(d) y-acceleration trajectories of *4bp* configuration.

Figure 6.3.: Initial open-loop and closed-loop trajectories for the left overtaking scenario.

assumption [WB05, sec. 3.5]. Still, the computational cost per iteration is higher in the *4bp* case.

Table 6.1.: Performance and complexity metrics evaluated on the initial scene of the overtaking scenario and closed-loop cost.

config.	performance metrics		complexity metrics		
	cost closed-loop	cost open-loop	constraints	variables	iterations
3bp	17.48	20.11	1483	1466	29
4bp	17.12	17.36	2205	2196	27

In the following 6.3 s, the EV converges toward its target lane center and target velocity using the *3bp* configuration, as shown in figure 6.3. The open-loop cost decreases monotonously in each time step, inducing a positive current cost descent factor in figure 6.4. In the first 0.9 s, the current cost descent factor is greater than one. A value beyond one indicates an inconsistency between the planned trajectories, resulting in a deviation between the open-loop and closed-loop trajectories, visible in figures 6.3a and 6.3c. The shrinking horizon approach decreases the distance between the splines and their convex hulls, resulting in a relaxed feasible space. IPOPT uses the additional space to find a lower-cost trajectory. The increase from $\tilde{\lambda}(0\,\mathrm{s})$ to $\tilde{\lambda}(0.1\,\mathrm{s})$ hints at a suboptimal solution due to the issues during the update step computation. After 0.9 s, the front vehicle no longer influences the trajectory, making the subsequent results time consistent. When the EV is near the target velocity and target lane center, it is placed in the LT. Thus, the cost is reduced to zero, resulting in a discontinuous increase in the current cost descent

factor beyond one after 5.9 s.
In contrast, the *4bp* configuration initially yields a lower current cost descent factor in figure 6.4 than the *3bp* configuration, which reduces to one after 0.2 s. Thus, subsequent open-loop trajectories are more consistent compared with the *3bp* results, and the initial open-loop trajectory is closer to the closed-loop trajectory in figures 6.3b and 6.3d. Due to the additional degrees of freedom and the less conservative convex hull, the front vehicle does not influence the open-loop trajectories as much.
The deviations between the closed-loop trajectories are smaller than those between the open-loop trajectories of the *3bp* and *4bp* configurations in figure 6.3. However, the *4bp* configuration reaches the target only 0.1 s later. Also, the difference in closed-loop cost in table 6.1 is smaller than the initial open-loop cost difference. The closed-loop cost refers to the time integral over the transformed running cost (4.2.7) resulting from the closed-loop trajectory until reaching the LT. The *3b* configuration results in a higher jerk in both directions, which is due to the higher-cost open-loop trajectories in the first time steps. As inequality constraints on the spline coefficients are not active in the later time steps, the additional breakpoint is not beneficial, enabling a similar closed-loop performance.

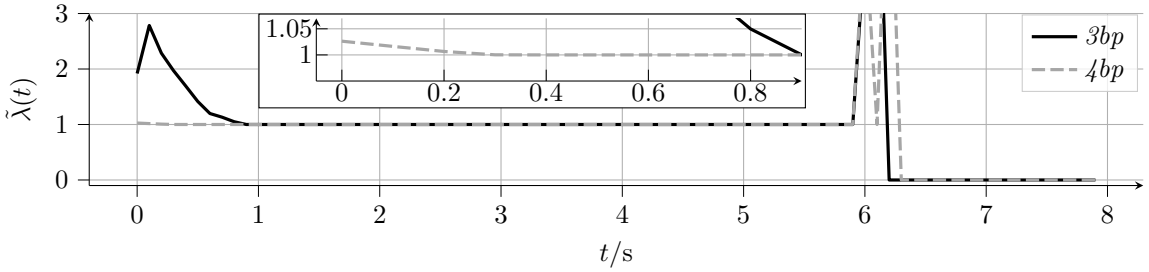


Figure 6.4.: Current cost descent factors due to the *3bp* and *4bp* configurations in the overtaking scenario.

Scenario 2: Right Vehicle Following

In the second scenario, the EV shall follow *V5* on the right lane at a time headway of $t_{\mathrm{hw}} = 2.5\,\mathrm{s}$ using the *3bp* configuration only. In the y-direction, the initial guess is determined in line with the previous scenario but targets the center of the right lane. In the x-direction, a polynomial of degree five interpolates between the EV state and the target trajectory behind the lead vehicle at 8 s. An additional polynomial segment is attached, extending the initial trajectory along the LT until the prediction horizon. In contrast to the first scenario, the initial guess is feasible.
In the following 4.2 s, the EV approaches the target lane center and desired time headway with a positive current cost descent factor, as shown in figure 6.5. However, in figure 6.6a, the leading vehicle violates the constant velocity assumption and accelerates toward a velocity of $80\,\mathrm{km\,h^{-1}}$. Thus, time consistent open-loop trajectories with a current cost descent factor of one are not possible. In addition, the disturbance induced by the leading vehicle acceleration prevent an asymptotic convergence to the LT in the x-direction. Consequently, the current cost descent factor becomes negative between 4.2 s and 4.6 s. Still, the EV closes up to the desired position after 4.6 s, as the leading vehicle settles

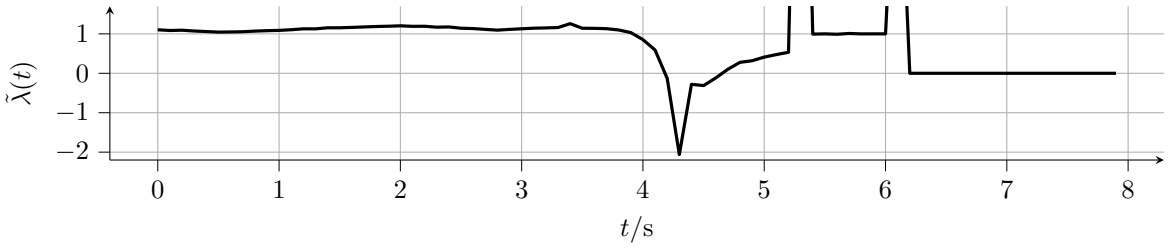
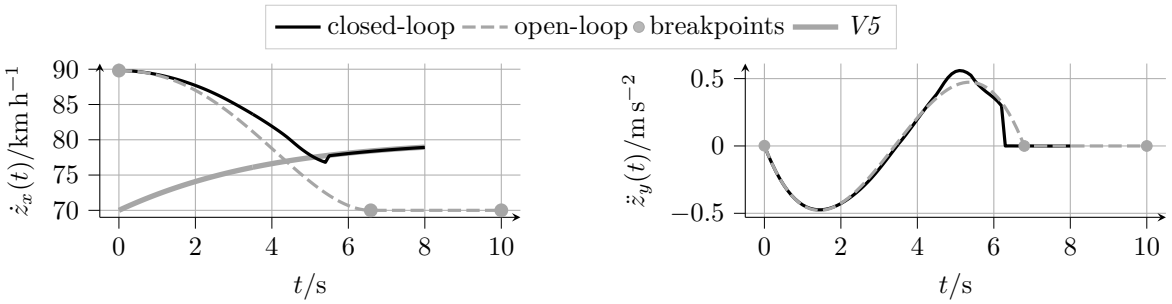


Figure 6.5.: Current cost descent factors due to the *3bp* configuration in the vehicle following scenario.

at its target velocity. At 5.5 s, the desired distance behind the leading vehicle is reached, which results in a discontinuous change of the current cost decent factor. Since the control horizon in x-direction is longer than in the y-direction, the disturbed target trajectory also influences the lateral convergence. The influence is observable in figure 6.6b, resulting in a deviation between the initially planned and the closed-loop trajectory. The target lane center is reached at 6.3 s, after reaching the target trajectory.



(a) x-velocity trajectories of the EV and *V5*.

(b) y-velocity trajectories of the EV.

Figure 6.6.: Initial open-loop and closed-loop trajectories for the vehicle following scenario.

The two scenarios show the applicability of the LP to highway scenarios assuming a given LT. In the first scenario, the recursive feasibility property and the convergence in the vicinity of another vehicle are demonstrated, resulting in a feasible initial guess for the NLP throughout the scenario. Additional breakpoints can improve the control performance if inequality constraints are active. Since the polynomial segments are optimal for the chosen cost functional, the benefit vanishes with the increasing distance from the impeding vehicle. On the other hand, additional breakpoints increase the NLP's complexity and should be applied only if an improvement in closed-loop control performance is expected. The second scenario demonstrates the ability to follow a leading vehicle at a desired time headway. Although the leading vehicle's motion and prediction do not coincide, convergence to the LT is achieved. However, the recursive feasibility property is not guaranteed to hold under disturbances, which might prevent finite-time convergence or even finding a feasible solution in the worst case. Implementing more complex maneuvers toward the LT requires a strategy to switch the LT. A target switch is also required if other vehicles render the current LT infeasible. The LP may not find a feasible solution and relies on an alternative LT with a corresponding initial guess.

7

Global Optimal Trajectory Planning

The LP described in chapter 6 provides the local optimal solution for the given trajectory parameterization. However, whether the local optimal solution is globally optimal depends on the LT. The LP retains the trajectory end in the current LT, which can be restrictive in the case of disturbances. This chapter introduces the GP, which takes over the responsibilities of the behavior module to address the mentioned limitations. Aiming at a hierarchical combination of the LP and the GP, the latter explores the LTs with a sampling-based algorithm while considering the same constraints and running cost as the LP. Parts of this chapter have been published in [DB24].

7.1. Requirements

The GP guides the vehicle toward the GT, which is described by the target velocity and the target lane given by the preceding route planning module. The best spline trajectory among the candidates and the corresponding LT are passed to the next module. In the hierarchical combination, the GP's focus is placed on LT exploration instead of finding a solution trajectory with high control performance.

7.1.1. Stability Requirements

The GP's ability to make far-sighted maneuver decisions is implemented via the terminal cost. Ideally, the terminal cost is the minimum cost-to-go from the prediction horizon to the GT. However, the future motion of other vehicles is uncertain. Therefore, the terminal cost is designed as a heuristic based on assumptions about the future motion of other vehicles while considering the cost design requirements from section 4.2. The asymptotic convergence toward the GT is only assured if the assumptions apply. Still, a sufficient control performance might be achieved if the assumptions are fulfilled in approximation.

Recursive feasibility of the GP is desired but not required in combination with the LP. If the GP cannot achieve a cost descent, the LP can still rely on its previous result trajectory. Thus, inconsistent maneuver selections are avoided due to missing recursive feasibility.

7.1.2. Performance and Complexity Requirements

The exploration of LTs is usually performed via a graph-based decomposition of the feasible state space. The graph consists of nodes and connecting edges to encode a discrete set of candidate trajectories. The structured highway environment motivates a deterministic discretization that depends on the vehicle state and the dynamic terminal set. Aiming at a combination with the LP, the GP does not have to closely approximate the optimal solution. Instead, the nodes' distribution and connectivity must be sufficiently distinct to provide multiple solution candidates.
A shortest-path search evaluates the graph edge candidates for feasibility. The candidate generation and evaluation are performed frequently, requiring an implementation of low complexity. Simultaneously, leveraging the DP principle reduces the number of required trajectory segments. Still, all candidates have to be explored in the worst case.

7.2. Graph Search

In the context of this work, the graph encodes the trajectory candidates as splines in basis form. The nodes represent the splines' breakpoints, while the edges correspond to the spline segments. Each $j = 0, 1, \ldots, \breve{\mathbf{N}} - 1$ node $\underline{\mathbf{N}}_j \in \mathbb{R}^{3\times 4}$, with a maximum number of $\breve{\mathbf{N}} \in \mathbb{N}_0$ nodes, carries information on the time and the state at the corresponding breakpoint. According to equation (3.5.5), only a subset of a candidate spline's coefficients and breakpoints are required to evaluate a spline segment. Thus, a node and an edge contribute to multiple candidate trajectories. The information contained in a node has to be sufficient to define a spline segment with two subsequent nodes. Following the LP implementation, the candidate trajectories are parameterized by clamped splines of order six with a continuity up to the second time derivative. Consequently, each spline segment depends on six coefficients and B-splines. The knot vector comprises the breakpoints with a multiplicity of six for the initial and final breakpoints. The intermediate breakpoints are replicated three times. Each B-spline of order six is defined using seven knots, where successive B-splines share the same six knots. Thus, a spline span involving six B-splines requires information on twelve knots per direction or four breakpoints per direction at maximum. Each node

$$\underline{\mathbf{N}}_j \coloneqq \begin{bmatrix} \underline{\tilde{\mathbf{z}}}_{x,j} & \underline{\tilde{\mathbf{k}}}_{x,j} & \underline{\tilde{\mathbf{z}}}_{y,j} & \underline{\tilde{\mathbf{k}}}_{y,j} \end{bmatrix} \tag{7.2.1}$$

contains half of the required information for a spline segment.
The transformed state in $\triangledown = x, y$ direction is described by $\underline{\mathbf{z}}_{\triangledown}(\tilde{\mathbf{k}}_{1,\triangledown,j}) \coloneqq \underline{\tilde{\mathbf{z}}}_{\triangledown,j} \in \mathbb{R}^3$. However, only including the breakpoint related to the state is not sufficient. Instead, the vectors $\underline{\tilde{\mathbf{k}}}_{\triangledown,j} \in \mathbb{R}^3$ in $\triangledown = x, y$ direction store a part of a candidate spline's breakpoints with $\mu = 0, 1, 2$ elements $\tilde{\mathbf{k}}_{\mu,\triangledown,j} \in [0, H]$. $\tilde{\mathbf{k}}_{1,\triangledown,j}$ is the time at the corresponding state $\underline{\tilde{\mathbf{z}}}_{\triangledown,j}$. In addition, the vector contains the breakpoint corresponding to the jth node's predecessor $\underline{\mathbf{N}}_n$, $\tilde{\mathbf{k}}_{0,\triangledown,j} = \tilde{\mathbf{k}}_{1,\triangledown,n}$, and the successor's $\underline{\mathbf{N}}_l$ breakpoint, $\tilde{\mathbf{k}}_{2,\triangledown,j} = \tilde{\mathbf{k}}_{1,\triangledown,l}$. In turn, the successors and the predecessors maintain the current node's breakpoint $\tilde{\mathbf{k}}_{2,\triangledown,n} = \tilde{\mathbf{k}}_{1,\triangledown,j}$ and $\tilde{\mathbf{k}}_{0,\triangledown,l} = \tilde{\mathbf{k}}_{1,\triangledown,j}$. The assignment of breakpoints to nodes is illustrated in figure 7.1. It shows the $\iota = 0, 1, \ldots, 5$ B-splines $b_{\iota,\triangledown}(t)$ required to evaluate the first segment of a spline $s_{\triangledown}(t)$ in the $\triangledown = x, y$ direction between the breakpoints $\mathbf{k}_{0,\triangledown} = 0\,\mathrm{s}$ and $\mathbf{k}_{1,\triangledown} = 3\,\mathrm{s}$.

The B-splines $b_{4,\triangledown}(t)$ and $b_{5,\triangledown}(t)$ also depend on $\mathbf{k}_{2,\triangledown} = 7\,\mathrm{s}$. The first two breakpoints are stored in a node $\underline{\mathbf{N}}_j$. For edge evaluation, the successor node $\underline{\mathbf{N}}_l$ has to contribute $\mathbf{k}_{2,\triangledown}$.

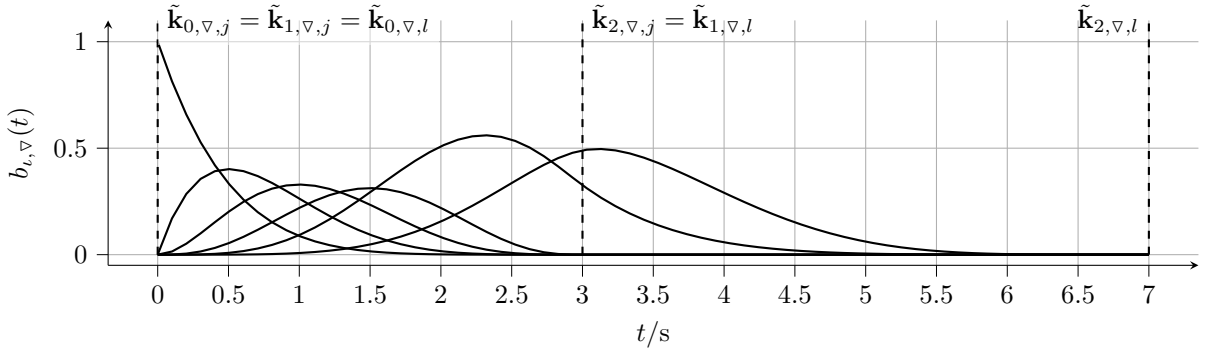


Figure 7.1.: Visualization of B-splines required for the segment between $\mathbf{k}_{0,\triangledown} = 0\,\mathrm{s}$ and $\mathbf{k}_{1,\triangledown} = 3\,\mathrm{s}$, and the assignment of breakpoints to adjacent nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_l$ for the $\triangledown = x, y$ direction.

The sequence of candidate breakpoints must be strictly increasing except at a trajectory's start and end. If $\underline{\mathbf{N}}_j$ is an *initial node*, it contains the initial state. The sequence elements are restricted to $\tilde{\mathbf{k}}_{0,\triangledown,j} = \tilde{\mathbf{k}}_{1,\triangledown,j} = 0$ in the $\triangledown = x, y$ direction. Similarly, if $\underline{\mathbf{N}}_n$ is a *terminal node*, the sequence elements are $\tilde{\mathbf{k}}_{1,\triangledown,n} = \tilde{\mathbf{k}}_{2,\triangledown,n} = H$. A node $\underline{\mathbf{N}}_l$ is a *pre-terminal node* in one or both $\triangledown = x, y$ directions with $\tilde{\mathbf{k}}_{2,\triangledown,l} = H$ and $\tilde{\mathbf{k}}_{1,\triangledown,l} \neq H$. Thus, the successor in the respective direction is a *terminal node*. All *terminal nodes* must connect to the *final node* $\tilde{\underline{\mathbf{k}}}_{x,j} = \tilde{\underline{\mathbf{k}}}_{y,j} = H$. The *final node* does not contribute to a candidate trajectory but represents the possibly distant GT. All other nodes are *intermediate nodes*.

7.2.1. Shortest-Path Search

The candidate trajectories are explored efficiently with a shortest-path search algorithm. Table A.7 provides an overview on the algorithm parameters and the used quantities. The strictly increasing breakpoint sequences render the graph directed and acyclic. The $j = 0, 1, \ldots, \breve{\mathbf{N}} - 1$ cost $V_{j,\mathbf{N}} \in \mathbb{R}^+$ assigned to the nodes $\underline{\mathbf{N}} = \begin{bmatrix}\underline{\mathbf{N}}_0 & \underline{\mathbf{N}}_1 & \ldots & \underline{\mathbf{N}}_{\breve{\mathbf{N}}-1}\end{bmatrix}$ are non-negative. In algorithm 7.2.1, the Dijkstra algorithm [Dij59] is applied for the shortest-path search. This algorithm performs a forward search from the initial nodes with the initial state $(\underline{\mathbf{z}}_{x,0}, \underline{\mathbf{z}}_{y,0})$. The algorithm expands each node in the graph at most once and prioritizes the node with the minimum cost to an initial node for expansion. Node expansion refers to the process of determining a node's successors. In contrast to the A* search, the Dijkstra algorithm does not rely on a cost-to-go heuristic. The heuristic may require substantial computational resources for a sufficiently accurate cost estimate. Among other things, the algorithm 7.2.1 relies on the transformed terminal set $\tilde{\mathcal{T}}$, a set of B-spline evaluations $\mathbb{B}$, and a set of coefficient transformations $\mathbb{T}$. The latter two are computed offline using all possible breakpoint sequences. The nodes' minimum cost predecessor is stored in $\underline{p}_{\mathbf{N}} \in \mathbb{N}^{\breve{\mathbf{N}}}$.
The open set $\mathcal{I}_{\mathbf{N},\mathrm{o}} \subset \mathbb{N}_0$ is created in line 2 and filled with the indices of the *initial nodes*. All nodes are initialized with zeros, and the initial state and the successors' breakpoints are assigned to the *initial nodes*. The open set is maintained until termination and contains the indices of nodes to expand. The algorithm selects the minimum cost node

Algorithm 7.2.1.: Find the shortest path in an implicit graph.

1: **function** SHORESTPATH($\underline{\mathbf{N}}, \underline{V}_{\mathbf{N}}, \tilde{\mathcal{T}}, \mathbb{B}, \mathbb{T}, \underline{p}_{\mathbf{N}}, \underline{\mathbf{z}}_{x,0}, \underline{\mathbf{z}}_{y,0}, \underline{\chi}_x, \underline{\chi}_y$)
2: $\quad\mathcal{I}_{\mathbf{N},\mathrm{o}} \leftarrow$ INITIALNODES($\underline{\mathbf{N}}, \underline{\mathbf{z}}_{x,0}, \underline{\mathbf{z}}_{y,0}$) ▷ initialize open set nodes
3: $\quad$**while** $\mathcal{I}_{\mathbf{N},\mathrm{o}} \neq \emptyset$ **do**
4: $\quad\quad j \leftarrow \arg\min_{n \in \mathcal{I}_{\mathbf{N},\mathrm{o}}} \{V_{n,\mathbf{N}}\}$ ▷ select lowest cost node from open set
5: $\quad\quad$**if** $\tilde{\underline{\mathbf{k}}}_{x,j} == \tilde{\underline{\mathbf{k}}}_{y,j} == H$ **then** ▷ is *final node*
6: $\quad\quad\quad$**return** j ▷ return the final node's index
7: $\quad\quad \mathcal{I}_{\mathbf{N},\mathrm{o}} \leftarrow \mathcal{I}_{\mathbf{N},\mathrm{o}} \setminus \{j\}$ ▷ remove current node from open set
8: $\quad\quad \mathcal{I}_{\mathbf{N},\mathrm{s}} \leftarrow$ GETSUCCESSORS($j, \underline{\mathbf{N}}, \tilde{\mathcal{T}}, \underline{\chi}_x, \underline{\chi}_y$) ▷ potential successors (algorithm 7.2.2)
9: $\quad\quad$**for** $n \in \mathcal{I}_{\mathbf{N},\mathrm{s}}$ **do**
10: $\quad\quad\quad \tilde{\underline{\mathbf{c}}}_x, \tilde{\underline{\mathbf{c}}}_y \leftarrow$ COEFFICIENTS($j, n, \underline{\mathbf{N}}, \mathbb{B}$) ▷ current coefficients (section 7.3.1)
11: $\quad\quad\quad \tilde{h} \leftarrow$ FEASIBLE($j, n, \underline{\mathbf{N}}, \tilde{\underline{\mathbf{c}}}_x, \tilde{\underline{\mathbf{c}}}_y, \mathbb{T}$) ▷ segment feasibility (algorithm 7.3.1)
12: $\quad\quad\quad \tilde{l} \leftarrow$ COST($j, n, \underline{\mathbf{N}}, \tilde{\underline{\mathbf{c}}}_x, \tilde{\underline{\mathbf{c}}}_y, \mathbb{T}$) ▷ segment cost (section 7.3.3)
13: $\quad\quad\quad$**if** $\tilde{h}$ **and** $V_{j,\mathbf{N}} + \tilde{l} < V_{n,\mathbf{N}}$ **then**
14: $\quad\quad\quad\quad V_{n,\mathbf{N}} \leftarrow V_{j,\mathbf{N}} + \tilde{l}$ ▷ update cost to successor
15: $\quad\quad\quad\quad p_{n,\mathbf{N}} \leftarrow j$ ▷ update predecessor
16: $\quad\quad\quad\quad \mathcal{I}_{\mathbf{N},\mathrm{o}} \leftarrow \mathcal{I}_{\mathbf{N},\mathrm{o}} \cup \{n\}$ ▷ add successor to open set

in line 4 for expansion to implement a prioritized breadth-first search considering the DP principle [Sni06]. If the selected node is a *final node*, the algorithm terminates in line 6 and returns the node's index. The node sequence of minimum cost is collected using the stored predecessors $\underline{p}_{\mathbf{N}}$ by backtracking the shortest path from the *final node* to the *initial node*. In line 8, the node expansion process starts by storing the potential successor node indices in $\mathcal{I}_{\mathbf{N},\mathrm{s}} \subset \mathbb{N}_0$. The search algorithm tracks the nodes' placement $\underline{\chi}_\triangledown \in \mathbb{N}_0^{\check{\mathbf{N}}}$ along a breakpoint sequence in $\triangledown = x, y$ direction. An *initial node* $\underline{\mathbf{N}}_j$ starts with $\chi_{j,x} = \chi_{j,y} = 0$ after the node initialization in line 2. The potential successors' coefficients are determined and stored in $\tilde{\underline{\mathbf{c}}}_x \in \mathbb{R}^6$ and $\tilde{\underline{\mathbf{c}}}_y \in \mathbb{R}^6$. Line 11 evaluates the current segment's feasibility according to the inequalities in equation (6.2.2), using the computed coefficients. The variable $\tilde{h} \in \{\mathrm{TRUE}, \mathrm{FALSE}\}$ indicates if the spline segment is feasible. The segment's cost is stored in $\tilde{l} \in \mathbb{R}^+$. The potential successors become actual successors if the corresponding spline segments are feasible and if the new paths to the successor nodes induce a lower cost than the current cost. If the conditions apply, the potential successors become the actual ones, and their cost and predecessor are updated. Finally, the successors are added to the open set for expansion.

The jth node's potential successors are determined by the algorithm 7.2.2. Line 4 retrieves the current breakpoint of the nodes' after the next ones, which are required to classify the expansion direction and to determine the successors' states accordingly. The $\underline{\mathbf{N}}_n$ successor node's breakpoints $\tilde{\underline{\mathbf{k}}}_{\triangledown,n}$ are drawn from the combination of the sets $\mathcal{K}_{j,\triangledown} \subset (0, H]$ in $\triangledown = x, y$ direction in line 6 until all combinations are processed. If the successor node is a *final node*, its index is returned immediately. The successor states are generated based on the current node's state and the successor nodes' breakpoints.

In line with the LP, the GP searches over nonuniform splines. In contrast, the segment

Algorithm 7.2.2.: Determine the potential successor nodes $\mathcal{I}_{\mathbf{N},\mathrm{s}}$ for the current node $\underline{\mathbf{N}}_j$.

function GETSUCCESSORS($j, \underline{\mathbf{N}}, \tilde{\mathcal{T}}, \underline{\chi}_x, \underline{\chi}_y$)
  $\mathcal{I}_{\mathbf{N},\mathrm{s}} \leftarrow \emptyset$
  $n \leftarrow j + 1$
  $\mathcal{K}_{j,x}, \mathcal{K}_{j,y} \leftarrow$ NEXTBREAKPOINTS($\underline{\mathbf{N}}_j, \chi_{j,x}, \chi_{j,y}$) ▷ get breakpoints (section 7.2.2)
  while $\mathcal{K}_{j,x} \times \mathcal{K}_{j,y} \neq \emptyset$ **do**
    $\tilde{\underline{\mathbf{k}}}_{x,n}, \tilde{\underline{\mathbf{k}}}_{y,n} \leftarrow$ DRAWBREAKPOINTS($\mathcal{K}_{j,x} \times \mathcal{K}_{j,y}, \underline{\mathbf{N}}_j$) ▷ draw breakpoints and remove from $\mathcal{K}_{j,x} \times \mathcal{K}_{j,y}$
    if $\tilde{\underline{\mathbf{k}}}_{x,n} == \tilde{\underline{\mathbf{k}}}_{y,n} == H$ **then** ▷ is *final node*
      return $\{n\}$ ▷ return *final node*
    $\mathcal{D}_x \leftarrow$ NEXTSTATESX($\underline{\mathbf{N}}_j, \tilde{\underline{\mathbf{k}}}_{x,n}, \tilde{\mathcal{T}}$) ▷ next states (algorithm 7.2.3)
    $\mathcal{D}_y \leftarrow$ NEXTSTATESY($\underline{\mathbf{N}}_j, \tilde{\underline{\mathbf{k}}}_{y,n}, \tilde{\mathcal{T}}$) ▷ next states (algorithm 7.2.4)
    if $\tilde{\mathbf{k}}_{1,x,j} < \tilde{\mathbf{k}}_{1,y,j}$ **then** ▷ expansion x-direction
      $\tilde{\underline{\mathbf{k}}}_{y,n}, \mathcal{D}_y, \chi_{n,x} \leftarrow \tilde{\underline{\mathbf{k}}}_{y,j}, \{\tilde{\underline{\mathbf{z}}}_{y,j}\}, \chi_{j,x} + 1$
    else if $\tilde{\mathbf{k}}_{1,x,j} > \tilde{\mathbf{k}}_{1,y,j}$ **then** ▷ expansion y-direction
      $\tilde{\underline{\mathbf{k}}}_{x,n}, \mathcal{D}_x, \chi_{n,y} \leftarrow \tilde{\underline{\mathbf{k}}}_{x,j}, \{\tilde{\underline{\mathbf{z}}}_{x,j}\}, \chi_{j,y} + 1$
    while $\mathcal{D}_x \times \mathcal{D}_y \neq \emptyset$ **do**
      $\tilde{\underline{\mathbf{z}}}_{x,n}, \tilde{\underline{\mathbf{z}}}_{y,n} \leftarrow$ DRAWSAMPLE($\mathcal{D}_x \times \mathcal{D}_y$) ▷ draw state pair and remove from $\mathcal{D}_x \times \mathcal{D}_y$
      $\mathcal{I}_{\mathbf{N},\mathrm{s}} \leftarrow \mathcal{I}_{\mathbf{N},\mathrm{s}} \cup \{n\}$ ▷ update successors
      $n \leftarrow n + 1$
  return $\mathcal{I}_{\mathbf{N},\mathrm{s}}$

number of uniform splines would be coupled to the time discretization. A dense discretization induces many short spline segments. However, if one desires long segments, a sparse time discretization is required, which can be insufficient for the task. In this application, spline segments of varying lengths are beneficial for achieving a high control performance with a low number of breakpoints because the polynomials are optimal for the squared jerk cost. Also, the additional breakpoint variations due to the nonuniformity render a sparse state space discretization sufficient to explore the feasible state space.

The constraints that depend on the breakpoints in both directions require a combination of breakpoint sequences during the node expansion. Each node expansion preserves the parameters relevant for a combined spline segment, requiring a node to expand in only the x- or the y-direction, or in both directions simultaneously. Figure 7.2 illustrates the breakpoint combination with a sequence of node expansions for two exemplary breakpoint vectors $\underline{\mathbf{k}}_x$ and $\underline{\mathbf{k}}_y$, which are combined to $\underline{\mathbf{k}}_{xy}$ according to the rules for breakpoint combination in section 3.5. The $j = 0, 1, 2, 3$ nodes $\underline{\mathbf{N}}_j$ describe the corresponding candidate spline. The bottom part of figure 7.2 shows the node connections and the involved breakpoint sequences. The initial node includes the breakpoints $\tilde{\underline{\mathbf{k}}}_{x,0} = \begin{bmatrix} 0 & 0 & \mathbf{k}_{1,x} \end{bmatrix}^\mathsf{T}$ and $\tilde{\underline{\mathbf{k}}}_{y,0} = \begin{bmatrix} 0 & 0 & \mathbf{k}_{1,y} \end{bmatrix}^\mathsf{T}$. The first node expansion from $\underline{\mathbf{N}}_0$ to $\underline{\mathbf{N}}_1$ covers the combined segment $[\mathbf{k}_{0,xy}, \mathbf{k}_{1,xy}]$ because the current breakpoint is the same for both directions $\mathbf{k}_{0,xy} = \tilde{\mathbf{k}}_{1,y,0} = \tilde{\mathbf{k}}_{1,y,0}$. Thus, the node $\underline{\mathbf{N}}_0$ is expanded in both directions, so the next node

includes the breakpoint sequences $\tilde{\underline{\mathbf{k}}}_{x,1} = \begin{bmatrix} 0 & \mathbf{k}_{1,x} & \mathbf{k}_{2,x} \end{bmatrix}^\mathsf{T}$ and $\tilde{\underline{\mathbf{k}}}_{y,1} = \begin{bmatrix} 0 & \mathbf{k}_{1,y} & \mathbf{k}_{2,y} \end{bmatrix}^\mathsf{T}$. The second node expansion from $\underline{\mathbf{N}}_1$ to $\underline{\mathbf{N}}_2$ covers the combined breakpoint segment $[\mathbf{k}_{1,xy}, \mathbf{k}_{2,xy}]$. The second node's breakpoints do not share the same current breakpoint. While the first segment in the x-direction includes the next combined segment $[\mathbf{k}_{1,xy}, \mathbf{k}_{2,xy}] \subset [\mathbf{k}_{0,x}, \mathbf{k}_{1,x}]$, the first segment in the y-direction does not $[\mathbf{k}_{1,xy}, \mathbf{k}_{2,xy}] \not\subset [\mathbf{k}_{0,y}, \mathbf{k}_{1,y}]$. Instead, the next segment in the y-direction $[\mathbf{k}_{1,xy}, \mathbf{k}_{2,xy}] \subset [\mathbf{k}_{1,y}, \mathbf{k}_{2,y}]$ covers the next combined segment. Consequently, the node $\underline{\mathbf{N}}_1$ is only expanded in the y-direction to $\tilde{\underline{\mathbf{k}}}_{y,2} = \begin{bmatrix} \mathbf{k}_{1,y} & \mathbf{k}_{2,y} & \mathbf{k}_{2,y} \end{bmatrix}^\mathsf{T}$. The breakpoints $\tilde{\underline{\mathbf{k}}}_{x,2} = \tilde{\underline{\mathbf{k}}}_{x,1}$ and the corresponding state $\tilde{\underline{\mathbf{z}}}_{x,2} = \tilde{\underline{\mathbf{z}}}_{x,1}$ in the x-direction are preserved. The third node expansion is performed only in the x-direction since the last segment $[\mathbf{k}_{2,xy}, \mathbf{k}_{3,xy}] \subset [\mathbf{k}_{1,y}, \mathbf{k}_{2,y}]$ is already covered in the y-direction. Thus, the breakpoints $\tilde{\underline{\mathbf{k}}}_{y,3} = \tilde{\underline{\mathbf{k}}}_{y,2}$ and states $\tilde{\underline{\mathbf{k}}}_{y,3} = \tilde{\underline{\mathbf{k}}}_{y,2}$ are preserved. Still, $\underline{\mathbf{N}}_2$ is located at $\mathbf{k}_{1,x}$ and requires an expansion to $\mathbf{k}_{2,x}$ with $\tilde{\underline{\mathbf{k}}}_{x,3} = \begin{bmatrix} \mathbf{k}_{1,x} & \mathbf{k}_{2,x} & \mathbf{k}_{2,x} \end{bmatrix}^\mathsf{T}$. The decision for the

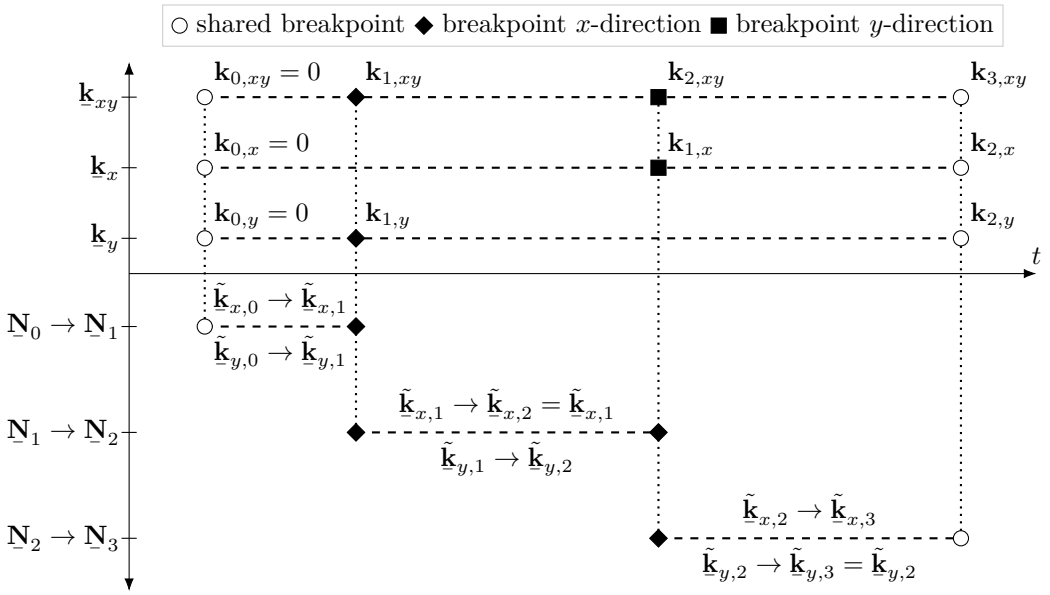


Figure 7.2.: Visualization of directional node expansion and the determination of the successor nodes' breakpoints.

expansion direction is implemented in algorithm 7.2.2 by comparing the jth nodes' current breakpoints in the x- and y-direction. If both current breakpoints are equal, an expansion in both directions is performed with two sets of new successor states $\mathcal{D}_\triangledown \subset \mathbb{R}^3$ in $\triangledown = x, y$ direction. The determination of successor states is described in section 7.2.3. In line 16 of algorithm 7.2.2, the successor states are combined to create the potential successor nodes that are added to $\mathcal{I}_{\mathbf{N},\mathrm{s}}$.

7.2.2. Time Discretization

The breakpoints of candidate trajectories are drawn from a discrete set of times along the planning horizon. The planning horizon is discretized to $\breve{K}_\triangledown \in \mathbb{N}_2$ equidistant intervals in

$\nabla = x, y$ direction, resulting in the discrete sets

$$\mathcal{K}_x = \left\{ \frac{j}{\breve{K}_x} H + \Delta\breve{\mathbf{k}} \mid j = 1, \ldots, \breve{K}_x - 1 \right\} \cup \{0, H\}, \tag{7.2.2}$$

$$\mathcal{K}_y = \left\{ \frac{j}{\breve{K}_y} H \mid j = 0, 1, \ldots, \breve{K}_y \right\}. \tag{7.2.3}$$

Except for the initial time and the prediction horizon, all elements of $\mathcal{K}_x$ are shifted by $\Delta\breve{\mathbf{k}}$ to ensure feasibility for the minimum breakpoint interval from the constraints (6.2.2). The length of spline segments connecting two elements from $\mathcal{K}_\nabla$ must be in the intervals $\mathcal{I}_\chi \subset (0, H)$. Each interval corresponds to a maximum number of breakpoints $\chi \in \mathcal{X} \subset \mathbb{N}_2$ along a candidate spline. The set $\mathcal{X}$ is a design choice, enabling the planning algorithm to explore candidate splines of varying segment lengths. The next breakpoints $\tilde{\mathbf{k}}_{2,\nabla,n} \in \mathcal{K}_{j,\nabla}$ in $\nabla = x, y$ direction of the jth node's successor $\underline{\mathbf{N}}_n$ are drawn from

$$\mathcal{K}_{j,\nabla} = \left\{ (\tilde{\mathbf{k}}_{2,\nabla,j} + \Delta\mathbf{k}) \in \mathcal{K}_\nabla \mid \Delta\mathbf{k} \in \mathcal{I}_\chi,\ \chi_{j,\nabla} < \chi - 3,\ \chi \in \mathcal{X} \right\} \cup \\ \cup \left\{ H \mid \chi - 3 \leq \chi_{j,\nabla} < \chi,\ \chi \in \mathcal{X} \right\}. \tag{7.2.4}$$

If the successor node becomes a *pre-terminal node* in any direction, so $\chi_{j,\nabla} = \chi - 3$ and $\chi_{n,\nabla} = \chi - 2$, the successor's next breakpoint $\tilde{\mathbf{k}}_{2,\nabla,n} \in \mathcal{K}_{j,\nabla} = \{H\}$ must be placed at the prediction horizon. The same applies if the successor node is a *terminal node* with $\chi_{j,\nabla} = \chi - 2$ and $\chi_{n,\nabla} = \chi - 1$ or a *final node* with $\chi_{j,\nabla} = \chi - 1$ and $\chi_{n,\nabla} = \chi$. In the special case of the open set initialization in algorithm 7.2.1 at line 2, $\mathcal{K}_{j,\nabla}$ determines the current node's next breakpoint $\tilde{\mathbf{k}}_{2,\nabla,j} \in \mathcal{K}_{j,\nabla}$ with $\chi_{j,\nabla} = -1$ in $\nabla = x, y$ direction.

7.2.3. State Discretization

The computational complexity of sampling-based planning algorithms is sensitive to the state space dimensionality. Following Ziegler and Stiller [ZS09], the complexity of the graph search algorithm is reduced in this work by restricting states in the y-direction to having zero velocity and acceleration. Also, the acceleration in the x-direction must be zero. In contrast to [ZS09], the x-position is not discretized explicitly. If a node is designated to follow a leading vehicle, the x-position is given by the leading vehicle's predicted trajectory. Otherwise, the x-position is determined to minimize the squared jerk cost for a given velocity. Thus, the terminal position $\mathbf{z}_{0,x}(T_x)$ of problem 4.3.1 is subject to optimization. The solution to the adapted variational problem requires satisfying the transversality condition [Föl94, eq. (2.46)] with the corresponding Lagrange multipliers $\underline{\Lambda}_{T_x} \in \mathbb{R}^2$:

$$\underline{0} = \underline{\Lambda}_x(T_x) - \frac{\mathrm{d}}{\mathrm{d}\underline{\mathbf{z}}_x(T_x)} \begin{bmatrix} \mathbf{z}_{1,\nabla,\mathrm{T}} - \mathbf{z}_{1,x}(T_x) & \mathbf{z}_{2,\nabla,\mathrm{T}} - \mathbf{z}_{2,x}(T_x) \end{bmatrix} \underline{\Lambda}_{T_x}, \tag{7.2.5}$$

$$= \begin{bmatrix} \Lambda_{0,x}(T_x) & \Lambda_{1,x}(T_x) + \Lambda_{0,T_x} & \Lambda_{2,x}(T_x) + \Lambda_{1,T_x} \end{bmatrix}^\intercal. \tag{7.2.6}$$

Since $\underline{\Lambda}_{T_x}$ is free to choose, only the first element resembles a new condition:

$$\Lambda_{0,x}(T_x) = 2w_{\mathrm{j},\nabla} q_{0,\nabla} = 0, \tag{7.2.7}$$

$$\Leftrightarrow q_{0,\nabla} = 0. \tag{7.2.8}$$

In combination with the Euler-Lagrange equation's necessary conditions (4.3.5) and (4.3.6), the first row and column in the matrix of equation (4.3.7) become obsolete. Thus, the solution is a polynomial of degree four $p_4 : \mathbb{R} \times \mathbb{R}^3 \times \mathbb{R} \mapsto \mathbb{R}$ as a function of time, the initial state, and the terminal velocity assuming a zero terminal acceleration.
The set of the successor nodes' state depends on the current node's and the successor nodes' breakpoints. Generally, the states are sampled to comply with the boundary constraints in equation (4.3.9). The initial nodes are placed at the initial state $\underline{\mathbf{z}}_{\nabla,0}$ in $\nabla = x, y$ direction, which is accomplished at the start of the shortest-path search. *Pre-terminal nodes* establish a connection to the terminal set. The subsequent *terminal nodes* retain the terminal segment in the terminal set. *Intermediate nodes* are meant to explore the state space beyond the terminal set.

Algorithm 7.2.3.: Obtain the successor nodes' states in the x-direction.

```
1: function NEXTSTATESX(N_j, k̃_{x,n}, T̃)
2:     if k̃_{1,x,n} ≠ H and k̃_{2,x,n} ≠ H then                ▷ intermediate node
3:         return D_{ẋ,j} ∪ D_{x̃,j}
4:     else if k̃_{2,x,n} == H and k̃_{1,x,n} ≠ H then          ▷ pre-terminal node
5:         return D_{ṽ,j} ∪ D_{x̃,j}
6:     else if k̃_{1,x,n} == H and (z̃_{x,j}, z̃_{y,j}) ∈ T̃ then ▷ terminal node and in terminal set
7:         return { [z̃_{0,j,x} + Δk_j z̃_{1,j,x}   z̃_{1,j,x}   0]^T }
8:     else                                                   ▷ terminal node and outside terminal set
9:         return ∅
```

Algorithm 7.2.3 details the selection of the successors' states in the x-direction. If the successors are *intermediate nodes*, the corresponding candidate trajectories can follow a leading vehicle (7.2.9) or reach states of constant velocity (7.2.10). The velocity set $\mathcal{Z}_{\dot{x}} = \left\{ \check{\dot{z}}_x + \frac{j}{\eta_x}(\tilde{v} - \check{\dot{z}}_x) \mid j = 0, 1, \ldots, \eta_x \right\}$ decomposes the interval between the minimum feasible velocity and the target velocity into $\eta_x \in \mathbb{N}_0$ equidistant intervals. The optimal position to a given velocity is obtained from the fourth-degree polynomial. $\Delta\mathbf{k}_j \coloneqq \tilde{\mathbf{k}}_{2,x,j} - \tilde{\mathbf{k}}_{1,x,j}$ denotes the time interval between the jth current node's and the subsequent nodes' breakpoints.

$$\mathcal{D}_{\tilde{x},j} = \left\{ \begin{bmatrix} z_{\tilde{x},m}(\tilde{\mathbf{k}}_{2,x,j}) & \dot{z}_{\tilde{x},m}(\tilde{\mathbf{k}}_{2,x,j}) & \ddot{z}_{\tilde{x},m}(\tilde{\mathbf{k}}_{2,x,j}) \end{bmatrix}^\intercal \mid m = 1, 2, \ldots, \eta_{\mathrm{ov}} \right\} \tag{7.2.9}$$

$$\mathcal{D}_{\dot{x},j} = \left\{ \begin{bmatrix} p_4(\Delta\mathbf{k}_j, \tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{\dot{\mathbf{z}}}_x) & \tilde{\dot{\mathbf{z}}}_x & 0 \end{bmatrix}^\intercal \mid \tilde{\dot{\mathbf{z}}}_x \in \mathcal{Z}_{\dot{x}} \right\} \tag{7.2.10}$$

If the successors are *pre-terminal nodes* in the x-direction, they must be located in the terminal set. Thus, the nodes are placed at the target velocity or behind a leading vehicle with equation (7.2.9). The state at the target velocity is contained in $\mathcal{D}_{\tilde{v},j}$:

$$\mathcal{D}_{\tilde{v},j} = \left\{ \begin{bmatrix} p_4(\Delta\mathbf{k}_j, \tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{v}) & \tilde{v} & 0 \end{bmatrix}^\intercal \right\}. \tag{7.2.11}$$

If the successors are *terminal nodes* in the x-direction, the trajectory must stay in the current node's LT. Thus, the successor node $\underline{\mathbf{N}}_n$ is placed in front of $\underline{\mathbf{N}}_j$, assuming a constant velocity along the terminal set. Line 9 considers the case where the successor nodes are terminal, but the current one is not in the terminal set. In this case, the

Algorithm 7.2.4.: Obtain the successor nodes' states in the y-direction.

```
1: function NEXTSTATESY(N_j, k̃_{y,n}, T̃)
2:     if k̃_{1,y,n} ≠ H then                                  ▷ not terminal node
3:         return D_{y,j}
4:     else if k̃_{1,y,n} == H and (z̃_{x,j}, z̃_{y,j}) ∈ T̃ then ▷ terminal node and in terminal set
5:         return {z̃_{y,j}}
6:     else                                                  ▷ terminal node and outside terminal set
7:         return ∅
```

corresponding candidate trajectory cannot comply with the boundary conditions, so no node expansion is performed.
The successors' states in the y-direction are obtained considering the same cases as in the x-direction without a distinct *pre-terminal node* case. If the successors are not *terminal nodes*, the trajectories must reach one of the current node's neighboring lane centers (7.2.13). The lane identifiers are obtained from equation (7.2.12), depending on the current node's associated lane.

$$\mathbb{D}_j = \begin{cases} \{\mathrm{lc}, \mathrm{mc}, \mathrm{rc}\}, \text{ if } \tilde{\mathbf{z}}_{0,y,j} \text{ in middle lane} \\ \{\mathrm{lc}, \mathrm{mc}\}, \text{ if } \tilde{\mathbf{z}}_{0,y,j} \text{ in left-most lane} \\ \{\mathrm{mc}, \mathrm{rc}\}, \text{ otherwise} \end{cases} \tag{7.2.12}$$

$$\mathcal{D}_{y,j} = \left\{ \begin{bmatrix} d_\square & 0 & 0 \end{bmatrix}^\mathsf{T} \mid \square \in \mathbb{D}_j \right\} \tag{7.2.13}$$

If the successors are *terminal nodes* and the current node is located in the terminal set, the current y-position is retained to keep the terminal segment in the terminal set. Finally, if a *terminal node* is present but not located in the terminal set, no feasible successor state can be returned.

Consequently, the y-positions and x-velocities are distributed along a regular grid, while the shortest-path search also performs connection attempts to the terminal set. The x-position is chosen optimally and not discretized explicitly, which likely provides feasible polynomial segments and reduces the search complexity. Thus, a set of distinct candidate trajectories of high control performance can be generated, which explores all LTs.

7.3. Edge Evaluation

After obtaining the potential successor nodes, the current node is connected to its successors. The connection process decides whether to establish a connection to a potential successor based on the feasibility and cost of the connecting spline segments. The required operations are detailed in this section.

7.3.1. Segment Coefficients

A prerequisite is the computation of the spline coefficients defining the transformed segments, inferred from the states at the current node $\underline{\mathbf{N}}_j$ and the predecessor's $\underline{\mathbf{N}}_n$ states in

Algorithm 7.3.1.: Test feasibility of a spline segment between $\underline{\mathbf{N}}_j$ and its potential successor $\underline{\mathbf{N}}_n$.

1: **function** FEASIBLE($j, n, \underline{\mathbf{N}}, \underline{\tilde{\mathbf{c}}}_x, \underline{\tilde{\mathbf{c}}}_y, \mathbb{T}$)
2: $\quad\tilde{\mathbb{H}} \leftarrow \mathbb{H}$
3: $\quad$**if** $\tilde{\mathbf{k}}_{1,x,j} == \tilde{\mathbf{k}}_{1,y,n} == H$ **then** ▷ successor is a *terminal node* in both directions
4: $\quad\quad\tilde{\mathbb{H}} \leftarrow \{s_{\mathrm{ov,f}}(t), s_{\mathrm{ov,r}}(t)\}$ ▷ test terminal collision constraints at *terminal node*
5: $\quad$**for** $s_\square(t) \in \tilde{\mathbb{H}}$ **do**
6: $\quad\quad\underline{\tilde{\mathbf{c}}}_\square \leftarrow$ CONSTRAINT$\big(\underline{\mathbf{N}}_j, \underline{\mathbf{N}}_n, \underline{\tilde{\mathbf{c}}}_x, \underline{\tilde{\mathbf{c}}}_y, \mathbb{T}, s_\square(t)\big)$ ▷ constraint coefficients
7: $\quad\quad$**if not** ALL$(\check{s}_\square \leq \underline{\tilde{\mathbf{c}}}_\square \leq \hat{s}_\square)$ **then**
8: $\quad\quad\quad$**return** FALSE ▷ spline segment infeasible
9: $\quad$**return** TRUE ▷ spline segment feasible

$\triangledown = x, y$ direction $\underline{\tilde{\mathbf{z}}}_{\triangledown,l} = {}_{j,n}^{l}\underline{B}_{\triangledown}^{\mathsf{T}}\, {}_{j,n}\underline{\tilde{\mathbf{c}}}_{\triangledown}$. Therefore, the B-splines corresponding to an edge between the $l = j, n$ nodes are evaluated at the nodes' breakpoints:

$$ {}_{j,n}^{l}\underline{B}_{\triangledown} \coloneqq \Big[b_{\iota,\triangledown}(\tilde{\mathbf{k}}_{1,x,l}) \quad b_{\iota,\dot{\triangledown}}(\tilde{\mathbf{k}}_{1,x,l}) \quad b_{\iota,\ddot{\triangledown}}(\tilde{\mathbf{k}}_{1,x,l})\Big]_{\iota\in\, {}_{j,n}\mathcal{I}_b}^{\mathsf{T}} \in \mathbb{R}^{6\times 3}, \quad l = j, n, \quad \triangledown = x, y. \tag{7.3.1} $$

${}_{j,n}\underline{\tilde{\mathbf{c}}}_{\triangledown} = \Big[{}_{n}\underline{\tilde{\mathbf{c}}}_{\triangledown}{}^{\mathsf{T}} \quad {}_{l}\underline{\tilde{\mathbf{c}}}_{\triangledown}{}^{\mathsf{T}}\Big]^{\mathsf{T}}$ comprises the coefficients associated with the individual $j = n, l$ nodes ${}_{l}\underline{\tilde{\mathbf{c}}}_{\triangledown} \in \mathbb{R}^3$ in $\triangledown = x, y$ direction. The set ${}_{j,n}\mathcal{I}_b \subset \mathbb{N}_0$ contains the indices of the nonzero B-splines on the segment under evaluation. The combination of B-spline evaluations at the nodes' breakpoints yields an invertible matrix ${}_{j,n}\underline{B}_{\triangledown} = \Big[{}_{j,n}^{j}\underline{B}_{\triangledown} \quad {}_{j,n}^{n}\underline{B}_{\triangledown}\Big]^{\mathsf{T}}$, enabling the computation of the edges' coefficients ${}_{j,n}\underline{\tilde{\mathbf{c}}}_{\triangledown} = {}_{j,n}\underline{B}_{\triangledown}^{-\mathsf{T}}\Big[\underline{\tilde{\mathbf{z}}}_{x,j}^{\mathsf{T}} \quad \underline{\tilde{\mathbf{z}}}_{x,n}^{\mathsf{T}}\Big]^{\mathsf{T}}$. The matrices ${}_{n,l}\underline{B}_{\triangledown}^{-\mathsf{T}} \in \mathbb{B}$ are computed offline for all possible breakpoint sequences in $\triangledown = x, y$ direction to reduce the computational effort during the graph search.

7.3.2. Segment Feasibility

One requirement for adding a potential successor $\underline{\mathbf{N}}_n$ of the current node $\underline{\mathbf{N}}_j$ to the graph is the feasibility of a candidate spline segment between $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$, subject to the state bounds and spline constraints in equation (6.2.2). Algorithm 7.3.1 evaluates the coefficients $\underline{\tilde{\mathbf{c}}}_\square \in \mathbb{R}^{\rho_\square}$ along the current segment associated with the constraint splines $s_\square(t) \in \tilde{\mathbb{H}}$. If the current spline segment is a terminal segment, it is sufficient to consider the simplified collision constraints. Otherwise, bounds on the velocity, the y-position, the ellipse collision constraints, the heading angle limitations, and the acceleration constraints are evaluated, which are described in chapter 5. The time arguments are omitted for brevity:

$$ \mathbb{H} = \Big\{s_{\dot{\triangledown}}, s_y, s_{\hat{\psi}}, s_{\check{\psi}}, s_{\mathrm{ov},m}, s_{\square,\triangledown,\mathrm{p}}, s_{\square,\triangledown,\mathrm{n}} \mid m = 1, 2, \ldots, \eta_{\mathrm{ov}},\ \square = \hat{a}, \check{a},\ \triangledown = x, y\Big\}. \tag{7.3.2} $$

Line 6 in algorithm 7.3.1 determines the segment coefficients corresponding to a constraint spline, where the required coefficient transformations are performed explicitly. In contrast, the NLP implements the transformations implicitly with $\underline{g}_{\mathrm{b}}(\underline{o}, \underline{\mathbf{k}}_x, \underline{\mathbf{k}}_y) = \underline{0}$. The matrices $\underline{T}_{\underline{b}_\triangle}^{\underline{b}_\square}$, introduced in section 3.5, are used to implement the spline products, sums, and derivatives involved in the constraint formulation. In application to a single spline segment,

only the relevant submatrix ${}_{j,n}\tilde{\underline{T}}^{\underline{b}_\square}_{\underline{b}_\triangle} \in \mathbb{T} \subset \mathbb{R}^{\rho_\triangle \times \rho_\square}$ is computed. The operations are defined analogously to section 3.5:

$${}_{j,n}\tilde{\underline{\mathbf{c}}}_\boxplus = {}_{j,n}\tilde{\underline{T}}^{\underline{b}_\square}_{\underline{b}_\boxplus}\, {}_{j,n}\tilde{\underline{\mathbf{c}}}_\square + {}_{j,n}\tilde{\underline{T}}^{\underline{b}_\triangle}_{\underline{b}_\boxplus}\, {}_{j,n}\tilde{\underline{\mathbf{c}}}_\triangle, \tag{7.3.3}$$

$${}_{j,n}\tilde{\underline{\mathbf{c}}}_\boxtimes = {}_{j,n}\tilde{\underline{T}}^{\underline{b}_\square \odot \underline{b}_\triangle}_{\underline{b}_\boxtimes}({}_{j,n}\tilde{\underline{\mathbf{c}}}_\square \otimes {}_{j,n}\tilde{\underline{\mathbf{c}}}_\triangle), \tag{7.3.4}$$

$${}_{j,n}\tilde{\underline{\mathbf{c}}}_{\dot\square} = {}_{j,n}\tilde{\underline{T}}^{\underline{b}_\square}_{\underline{b}_{\dot\square}}\, {}_{j,n}\tilde{\underline{\mathbf{c}}}_\square. \tag{7.3.5}$$

Like the inverted B-spline evaluations, the coefficient transformations are computed offline and stored in $\mathbb{T}$, considering all possible breakpoint sequences. After obtaining the coefficients, they are checked for compliance with the upper $\hat{s}_\square \in \mathbb{R}$ and lower bounds $\check{s}_\square \in \mathbb{R}$ imposed on the corresponding spline $s_\square(t)$. The bounds $\check{s}_\square = 0$ and $\hat{s}_\square \to \infty$ are associated with $s_\square(t) \in (\tilde{\mathbb{H}} \cup \{s_{\mathrm{ov,f}}(t), s_{\mathrm{ov,r}}(t)\}) \setminus \{s_{\dot x}(t), s_{\dot y}(t), s_y(t)\}$. The vehicle width and the lane bounds are considered in $\check{s}_y = z_{y,\mathrm{r}} + \Delta_y$ and $\hat{s}_y = z_{y,\mathrm{l}} - \Delta_y$. In addition, $\check{s}_{\dot x} = \check{z}_{\dot x}$, $\hat{s}_{\dot x} = \hat{z}_{\dot x}$, $\check{s}_{\dot y} = -\bar{z}_{\dot y}$, and $\hat{s}_{\dot y} = \hat{z}_{\dot y}$ coincide with the respective velocity bounds.

7.3.3. Segment Cost

In addition to a spline segment's feasibility, the transition cost ${}_{j,n}\tilde{l} \in \mathbb{R}^+$ between the nodes $\underline{\mathbf{N}}_j$ and $\underline{\mathbf{N}}_n$ is determined:

$${}_{j,n}\tilde{l} \coloneqq \begin{cases} \tilde{F}(\tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{\underline{\mathbf{z}}}_{y,j}) & \text{if } \tilde{\underline{\mathbf{k}}}_{x,n} = \tilde{\underline{\mathbf{k}}}_{y,n} = H, \\ {}_{j,n}\tilde{l}_x + {}_{j,n}\tilde{l}_y & \text{otherwise.} \end{cases} \tag{7.3.6}$$

If the node $\underline{\mathbf{N}}_n$ is not a *final node*, the cost is determined based on the nodes' states and breakpoints. The cost along a spline segment is the sum of the cost in the $\triangledown = x, y$ direction ${}_{j,n}\tilde{l} \coloneqq {}_{j,n}\tilde{l}_x + {}_{j,n}\tilde{l}_y$ with ${}_{j,n}\tilde{l}_\triangledown \in \mathbb{R}^+$. The directional costs are defined in line with the running cost from equation (6.2.1):

$${}_{j,n}\tilde{l}_\triangledown \coloneqq \begin{cases} 0 & \text{if } \tilde{\mathbf{k}}_{1,\triangledown,n} = H, \\ w_{T,\triangledown}\left(\tilde{\mathbf{k}}_{1,\triangledown,n} - \tilde{\mathbf{k}}_{1,\triangledown,j}\right) + w_{\mathrm{j},\triangledown}\, {}_{j,n}\tilde{\mathbf{c}}_{5,\mathrm{I},\ddot\triangledown^2} & \text{otherwise,} \end{cases} \qquad \triangledown = x, y. \tag{7.3.7}$$

The costs become zero if the successor node is a terminal node to ensure a cost decrease with the shrinking control horizon. Otherwise, the weighted sum of the time along the spline segment and the squared jerk integral is returned. The cost calculation requires the segment coefficients of the squared jerk integral $s_{\mathrm{I},\ddot\triangledown^2}(t) = \int_0^{T_\triangledown} s_{\ddot\triangledown^2}(t)\,\mathrm{d}t$ in the $\triangledown = x, y$ direction. A transformation ${}_{j,n}\tilde{\underline{T}}^{\underline{b}_\triangledown}_{\underline{b}_{\ddot\triangledown}} \in \mathbb{T}$ applied to the position trajectories' spline coefficients yields the third time derivative coefficients in equation (7.3.8). Subsequently, the squared jerk integral coefficients are obtained by equation (7.3.9), with two additional matrices implementing the spline product ${}_{j,n}\tilde{\underline{T}}^{\underline{b}_{\ddot\triangledown} \odot \underline{b}_{\ddot\triangledown}}_{\underline{b}_{\ddot\triangledown^2}} \in \mathbb{T}$ and the time integration ${}_{j,n}\tilde{\underline{T}}^{\underline{b}_{\ddot\triangledown^2}}_{\underline{b}_{\mathrm{I},\ddot\triangledown^2}} \in \mathbb{T}$. The last element of ${}_{j,n}\tilde{\mathbf{c}}_{\mathrm{I},\ddot\triangledown^2} \in \mathbb{R}^6$ yields the squared jerk integral over the segment under evaluation.

$${}_{j,n}\tilde{\underline{\mathbf{c}}}_{\ddot\triangledown} = {}_{j,n}\tilde{\underline{T}}^{\underline{b}_\triangledown}_{\underline{b}_{\ddot\triangledown}}\, {}_{j,n}\tilde{\underline{\mathbf{c}}}_\triangledown \tag{7.3.8}$$

$${}_{j,n}\tilde{\underline{\mathbf{c}}}_{\mathrm{I},\ddot\triangledown^2} = {}_{j,n}\tilde{\underline{T}}^{\underline{b}_{\ddot\triangledown^2}}_{\underline{b}_{\mathrm{I},\ddot\triangledown^2}}\, {}_{j,n}\tilde{\underline{T}}^{\underline{b}_{\ddot\triangledown} \odot \underline{b}_{\ddot\triangledown}}_{\underline{b}_{\ddot\triangledown^2}}\left({}_{j,n}\tilde{\underline{\mathbf{c}}}_{\ddot\triangledown} \otimes {}_{j,n}\tilde{\underline{\mathbf{c}}}_{\ddot\triangledown}\right) \tag{7.3.9}$$

8

Local Target Selection

The selection of LTs is guided by the terminal cost, which are integrated in the segment cost in section 7.3.3 if a node expansion exceeds the prediction horizon. This chapter defines the terminal cost such that the EV is stabilized in the GT according to the requirements in section 7.1.1. The cost impose assumptions on the other vehicles' future motion to enable far-sighted decisions at low computational cost. The GP's ability to select reasonable LTs is demonstrated subsequently in open-loop at two overtaking scene variations. The computational complexity and control performance of several GP configurations are evaluated to identify a suitable compromise. Parts of this chapter have been published in [DB24].

8.1. Terminal Cost

The terminal cost is applied as the segment cost in equation (7.3.6) if the successor node is a *final node*. The terminal cost is an estimate of the cost-to-go from the prediction horizon into the GT. According to the criteria in section 4.1.2, the terminal cost has to upper bound the transition cost between LTs in the sense of equation (4.1.3). The condition ensures the GP selects an LT on the trajectory to the GT if feasible. Simultaneously, equation (4.1.4) demands a terminal cost design, so a continuous upper bound for the terminal cost exists.

Starting at the prediction horizon, the cost-to-go estimate proposed in this section assumes that the EV approaches the GT by visiting the LTs. The cost extrapolation is a sum of the costs induced by transitions among LTs that, under simplifying assumptions, end in the GT. The LTs explicitly considered for the terminal cost calculation include the target velocity and the neighboring lane centers but not the target trajectories behind other vehicles. Problem 8.1.1 estimates the cost for the transition between the current LT to the next LT.

Problem 8.1.1 (Local target transition)**:** The minimum cost $\tilde{V}_3^* : \mathbb{R}^3 \times \mathbb{R}^3 \mapsto \mathbb{R}^+$ from the current to the next LT, considering only candidate trajectories with $\max \mathcal{X} = 3$

maximum breakpoints in the graph, is given by

$$\tilde{V}_3^*\Big(\underline{\mathbf{z}}_x(H), \underline{\mathbf{z}}_y(H)\Big) = \min_{T_x, T_y} \sum_{\triangledown = x,y} \int_0^{T_\triangledown} w_{T,\triangledown} + w_{\mathrm{j},\triangledown} \dddot{p}_5^2(t, \underline{\mathbf{z}}_{\triangledown,0}, \underline{\mathbf{z}}_{\triangledown,\mathrm{T}})\ \mathrm{d}t, \tag{8.1.1}$$

subject to

$$\begin{aligned} &\check{T}_x \leq T_x \leq H, \quad \underline{\mathbf{z}}_{x,0} = \begin{bmatrix} 0 & \mathbf{z}_{1,x}(H) & 0 \end{bmatrix}^\intercal, && \underline{\mathbf{z}}_{y,0} = \begin{bmatrix} d_{\mathrm{rc}} & 0 & 0 \end{bmatrix}^\intercal, \\ &\check{T}_y \leq T_y \leq H, \quad \underline{\mathbf{z}}_{x,\mathrm{T}} = \begin{bmatrix} p_4(x, \underline{\mathbf{z}}_{x,0}, \tilde{v}) & \tilde{v} & 0 \end{bmatrix}^\intercal, && \underline{\mathbf{z}}_{y,\mathrm{T}} = \begin{bmatrix} d_{\mathrm{mc}} & 0 & 0 \end{bmatrix}^\intercal, \\ &T_\triangledown \in \mathcal{K}_{n,\triangledown}, && \triangledown = x, y. \end{aligned} \tag{8.1.2}$$

The GP's solution trajectory ends with the *terminal node* $\underline{\mathbf{N}}_j$. When the EV reaches the state at the *terminal node* $\underline{\mathbf{N}}_j$, the state becomes the initial state of the new *initial node* $\underline{\mathbf{N}}_n$. $\tilde{V}_3^*\Big(\tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{\underline{\mathbf{z}}}_{y,j}\Big)$ is the minimum segment cost from $\underline{\mathbf{N}}_n$ to a new LT at the target velocity. Candidate trajectories can only terminate in the current or the neighboring lanes. Since the middle and right-most lanes are usually the widest, the cost of changing between them is assumed to overestimate the cost of changing to the left-most lane. The kinematic constraints from chapter 5 impose lower bounds $\check{T}_\triangledown \in [0, H]$ in the $\triangledown = x, y$ direction. For the selected boundary conditions, which demand a transition between two equilibria, a growing control horizon only reduces the trajectory's time derivative magnitudes. Section A.6 shows the property graphically. In the x-direction, the acceleration decreases with the growing control horizon, while the velocity and acceleration decrease in the y-direction. Thus, the set of kinematically feasible control horizons is bounded from above only by the prediction horizon, resulting in

$$\tilde{V}_3^*\Big(\underline{\mathbf{z}}_x(H), \underline{\mathbf{z}}_y(H)\Big) < \hat{\tilde{V}}_x\Big(\underline{\mathbf{z}}_x(H)\Big) + \hat{\tilde{V}}_y\Big(\underline{\mathbf{z}}_y(H)\Big), \tag{8.1.3}$$

$$\hat{\tilde{V}}_\triangledown\Big(\underline{\mathbf{z}}_\triangledown(H)\Big) \coloneqq \int_0^{\tilde{T}_\triangledown} w_{T,\triangledown} + w_{\mathrm{j},\triangledown} \dddot{p}_5^2(t, \underline{\mathbf{z}}_{\triangledown,0}, \underline{\mathbf{z}}_{\triangledown,\mathrm{T}})\ \mathrm{d}t, \qquad \triangledown = x, y. \tag{8.1.4}$$

Considering only GP configurations including splines with three breakpoints $\mathcal{X} = \{3\}$, equation (8.1.4) provides an upper bound on the transition cost between adjacent LTs. The maximum segment length $\tilde{T}_\triangledown \coloneqq \max \mathcal{K}_{n,\triangledown}$ is chosen as the control horizon in $\triangledown = x, y$ direction. The upper bound holds for any combination of cost weights. With a growing control horizon $T_\triangledown \to \infty$, the jerk decreases monotonously $\dddot{p}_5(t, \underline{\mathbf{z}}_{\triangledown,0}, \underline{\mathbf{z}}_{\triangledown,\mathrm{T}}) \to 0$. Thus, as $w_{T,\triangledown} \to 0$, the optimal control horizon for problem 8.1.1 increases until reaching $\tilde{T}_\triangledown$. Conversely, as $w_{\mathrm{j},\triangledown} \to 0$, the optimal control horizon approaches the minimum feasible one, which renders $\tilde{T}_\triangledown$ suboptimal. Since the splines' time derivatives only decrease with a growing $T_\triangledown$, a feasible solution at $T_\triangledown = \tilde{T}_\triangledown$ exists if one exists at $T_\triangledown < \tilde{T}_\triangledown$.
Candidate trajectories are composed of multiple segments with lower distinct cost if $\max \mathcal{X} > 3$. However, in the presence of other vehicles, a trajectory composed of multiple segments may yield a higher cost than a single polynomial. Multiple segment trajectory candidates can exceed the introduced cost bound, and opportunities for progressing toward the next LT may be missed. The upper bound is retained for splines with multiple segments by scaling the polynomial cost bound with the maximum possible number of segments. The scaling may render the cost estimation conservative, which introduces a certain robustness against other vehicles that require a multi-segment trajectory to reach the next LT. On the other hand, growing terminal cost decrease the importance of the running cost and

the control performance in convergence to an LT. The transformed terminal cost is a sum of the scaled terminal cost $\tilde{F}_\triangledown : \mathbb{R}^3 \mapsto \mathbb{R}^+$ in the $\triangledown = x, y$ direction

$$\tilde{F}(\tilde{\underline{\mathbf{z}}}_{x,j}, \tilde{\underline{\mathbf{z}}}_{y,j}) = (\max \mathcal{X} - 2)\Big(\tilde{F}_x(\tilde{\underline{\mathbf{z}}}_{x,j}) + \tilde{F}_y(\tilde{\underline{\mathbf{z}}}_{y,j})\Big) \tag{8.1.5}$$

in dependence on the state $\tilde{\underline{\mathbf{z}}}_{\triangledown,j}$ at the *terminal node* $\underline{\mathbf{N}}_j$. The directional terminal costs are defined based on the upper cost bounds (8.1.4). A control sequence from the state at each terminal node is constructed to the GT, assuming inactive collision constraints, to fulfill the terminal cost descent (4.1.3) and the upper bound (4.1.4). Four cases are distinguished, which are defined below.

Definition 8.1.1 (Right lane change to target velocity)**:** The $\underline{\mathbf{N}}_j$ *terminal node's* velocity fulfills $\tilde{\mathbf{z}}_{1,x,j} \geq \tilde{v}$, and the definitions 8.1.3 and 8.1.4 do not apply. The directional terminal costs are defined to

$$\tilde{F}_x(\tilde{\underline{\mathbf{z}}}_{x,j}) \coloneqq \hat{\tilde{V}}_x(\tilde{\underline{\mathbf{z}}}_{x,j}), \tag{8.1.6}$$

$$\tilde{F}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) \coloneqq \begin{cases} 0 & \text{if } d_{\mathrm{rc}} = \tilde{\mathbf{z}}_{0,y,j}, \\ \hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{if } d_{\mathrm{mc}} = \tilde{\mathbf{z}}_{0,y,j}, \\ 2\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{otherwise.} \end{cases} \tag{8.1.7}$$

Definition 8.1.1 applies to terminal states that reach or exceed the target velocity. Simultaneously, the EV is not in danger of violating the left-overtaking rule in definition 8.1.4, and the EV does not drive too close behind another vehicle, as in definition 8.1.3. The cost estimate (8.1.7) depends on the terminal state's associated lane. Two lane changes are required from the left-most lane to get into the right-most target lane, yielding a cost bound of $2\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,n})$. The right-most lane is reached from the middle lane with a single lane change. An exemplary scenario is depicted in figure 8.1. The terminal state $\underline{\mathbf{z}}(H)$ is located on the left-most lane, while one of the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ vehicles travels sufficiently far ahead to not interfere with the EV's motion. The vehicle's transformed state trajectory is indicated by $\underline{\mathbf{z}}_m : \mathbb{R} \mapsto \mathbb{R}^6$. The EV's transformed state prediction beyond the prediction horizon is described by $\tilde{\underline{\mathbf{z}}}_l \coloneqq \begin{bmatrix} \underline{\mathbf{z}}_x^\intercal(H + l\tilde{T}_x) & \underline{\mathbf{z}}_y^\intercal(H + l\tilde{T}_y) \end{bmatrix}^\intercal$ at multiples $l \in \mathbb{N}_0$ of the maximum segment lengths. According to the assumptions underlying definition 8.1.1, the target velocity and the middle lane are reached at $t = H + \tilde{T}_\triangledown$, $\triangledown = x, y$. After $t = H + 2\tilde{T}_\triangledown$ and the second lane change, the EV enters the GT in the right lane.

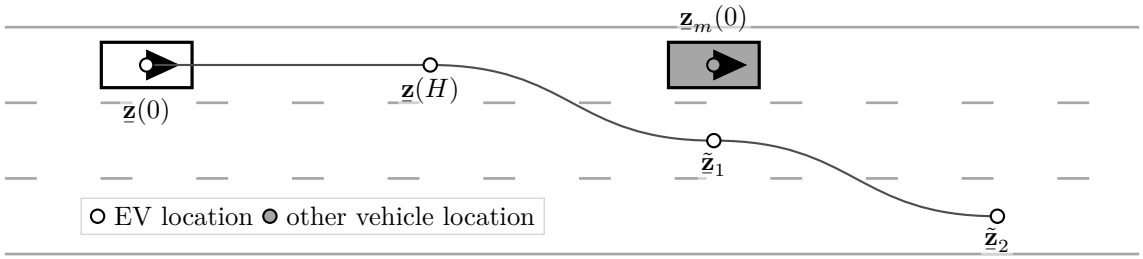


Figure 8.1.: Exemplary scenario showing the control sequence underlying the terminal cost according to definition 8.1.1.

Definition 8.1.2 (Left lane change for overtaking)**:** The *terminal node* $\underline{\mathbf{N}}_j$ is located behind a leading vehicle with $\tilde{\underline{\mathbf{z}}}_{x,j} \in \tilde{\mathcal{T}}_{x,m}$ and $\tilde{\underline{\mathbf{z}}}_{y,j} \in \tilde{\mathcal{T}}_{y,m}$ for any $m = 1, 2, \ldots, \eta_{\text{ov}}$. In addition, $\tilde{\mathbf{z}}_{1,x,j} < \tilde{v}$, $d_{\text{lc}} \neq \tilde{\mathbf{z}}_{0,y,j}$, and the impeding vehicle case in definition 8.1.4 does not apply. The directional terminal costs are defined to

$$\tilde{F}_x(\tilde{\underline{\mathbf{z}}}_{x,j}) \coloneqq \hat{\tilde{V}}_x(\tilde{\underline{\mathbf{z}}}_{x,j}), \tag{8.1.8}$$

$$\tilde{F}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) \coloneqq \begin{cases} 3\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{if } d_{\text{mc}} = \tilde{\mathbf{z}}_{0,y,j}, \\ 4\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{otherwise.} \end{cases} \tag{8.1.9}$$

Definition 8.1.2 considers scenes where the terminal state follows a leading vehicle in the middle or right lane while driving slower than the target velocity. Other vehicles are arranged, so the EV is not in danger of violating the left overtaking rule implemented with definition 8.1.4. The cost estimate assumes that the EV changes to the left-most lane, where definition 8.1.1 applies. Depending on the terminal state's associated lane, one or two lane changes are required to reach the left-most lane, followed by two additional lane changes due to definition 8.1.1. The lane change costs are accumulated in equation (8.1.9). Figure 8.2 shows a scenario where the conditions of definition 8.1.2 apply. The state at the prediction horizon $\underline{\mathbf{z}}(H)$ is located on the target trajectory behind another leading vehicle. Starting from the right-most lane, two lane changes are performed to reach the left-most lane. Because no other vehicle is located in the middle lane, definition 8.1.1 would already apply at $t = H$, so a lane change to the left-most lane is not executed.

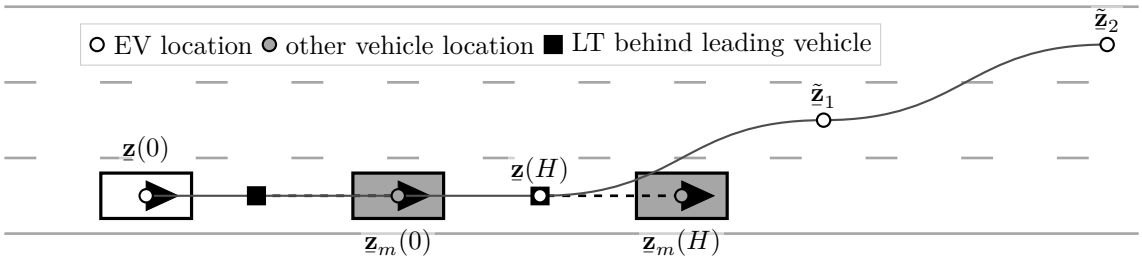


Figure 8.2.: Exemplary scenario showing the control sequence underlying the terminal cost according to definition 8.1.2.

Definition 8.1.3 (Impeding vehicle)**:** One of the $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles travels in front $z_{x,m}(H) \geq \tilde{\mathbf{z}}_{0,x,j}$ of the $\underline{\mathbf{N}}_j$ *terminal nodes'* lane $d_m(H) = \tilde{\mathbf{z}}_{0,y,j}$. In addition, the target position behind the other vehicle does not exceed the distance traveled at the target velocity $z_{\tilde{x},m}(H) \leq \tilde{v}H$, and $\dot{z}_{x,m}(H) < \tilde{v}$. With $\hat{\tilde{F}}_x \in \mathbb{R}^+$, the direction terminal cost yield

$$\tilde{F}_x(\tilde{\underline{\mathbf{z}}}_{x,j}) \coloneqq \hat{\tilde{F}}_x, \tag{8.1.10}$$

$$\tilde{F}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) \coloneqq \begin{cases} 4\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{if } d_{\text{rc}} = \tilde{\mathbf{z}}_{0,y,j}, \\ 3\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{if } d_{\text{mc}} = \tilde{\mathbf{z}}_{0,y,j}, \\ 2\hat{\tilde{V}}_y(\tilde{\underline{\mathbf{z}}}_{y,j}) & \text{otherwise.} \end{cases} \tag{8.1.11}$$

The definitions 8.1.1 and 8.1.2 provide terminal cost for situations where the terminal state does not reach the GT. However, situations can occur where another slower-driving

vehicle blocks the path along the GT. Retaining the target velocity on the target lane may be the best option according to definitions 8.1.1 and 8.1.2 since they induce zero terminal cost. Because the planning algorithm checks for collision on a receding horizon, the previously optimal trajectory can become infeasible if $z_{m,x}(H) \leq \tilde{v}H$. Depending on the candidate trajectories' degrees of freedom, the GP may find a spatially shorter trajectory into the GT to avoid a collision with the front vehicle. Consequently, the GP's result alternates between trajectories of different lengths until the definitions 8.1.1 or 8.1.2 apply. The definition 8.1.3 is introduced to resolve such situations in an anticipatory manner. If a vehicle in front of the terminal state is sufficiently close, the vehicle's target position can already be in collision $z_{\tilde{x},m}(H) \leq \tilde{v}H$, and the EV has to leave the GT. The cost in the x-direction is simplified to a constant value $\hat{\tilde{F}}_x$ to overestimate the segment cost for the transition to a new LT and the subsequent cost of the definitions 8.1.1 and 8.1.2. A scenario where definition 8.1.3 applies is illustrated in figure 8.3. The GP's candidate trajectory enters the target lane behind a slower-driving vehicle. The EV is assumed to follow the vehicle in front or to change to the left-most lane after the prediction horizon. The $m = 1, 2 \ldots, \eta_{\text{ov}}$ other vehicle states beyond the prediction horizon are described by $\tilde{\underline{\mathbf{z}}}_{m,l} \coloneqq \left[z^{(\iota)}_{m,x}(H + l\tilde{T}_x) \quad z^{(\iota)}_{m,y}(H + l\tilde{T}_y) \right]^{\mathsf{T}}_{\iota=0,1,2}$ with $l \in \mathbb{N}_0$.

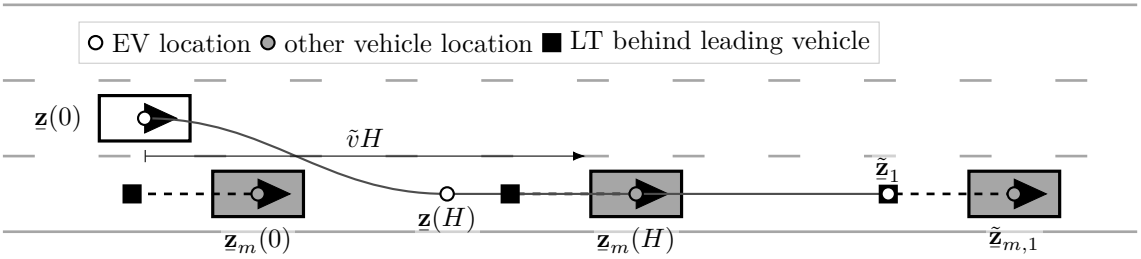


Figure 8.3.: Exemplary scenario showing the control sequence underlying the terminal cost according to definition 8.1.3.

Definition 8.1.4 (Left impeding vehicle)**:** One of the $m = 1, 2, \ldots, \eta_{\text{ov}}$ other vehicles travels in front $z_{x,m}(0) > 0$ and on one of the $\underline{\mathbf{N}}_j$ *terminal nodes'* left lanes $d_m(H) > \tilde{\mathbf{z}}_{0,y,j}$. In addition, target position behind the same leading vehicle's does not exceed the distance traveled at the target velocity $z_{\tilde{x},m}(H) \leq \tilde{v}H$, and $\dot{z}_{x,m}(H) < \tilde{v}$. The directional terminal costs are given by the equations (8.1.10) and (8.1.11).

The definition 8.1.4 ensures that other vehicles are overtaken on the left side. The conditions are similar to definition 8.1.3 but require the other vehicles to be on the *terminal nodes'* left lanes instead of the current one. In contrast to definition 8.1.3, where another vehicle is in front of the terminal position, another vehicle is in front of the initial position $z_{x,m}(0) > 0$. If the reference position beind a leading vehicle is closer than $z_{\tilde{x},m}(H) \leq \tilde{v}H$, the trajectory resulting from a constant target velocity passes the target position and is in danger of overtaking the vehicle on the right side. A scenario fulfilling the conditions is shown in figure 8.4. The costs are the same as in definition 8.1.3 because the assumptions for resolving the situation are the same.

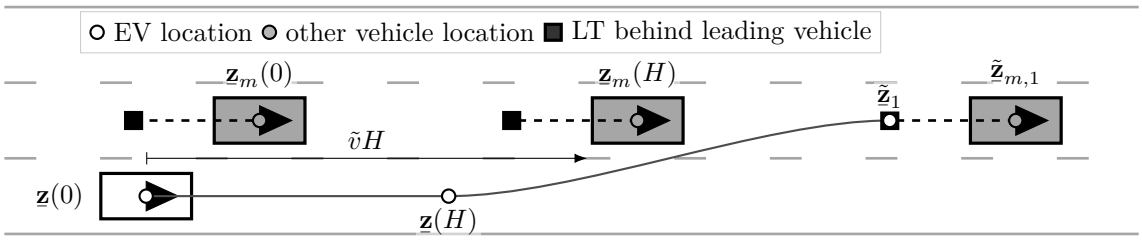


Figure 8.4.: Exemplary scenario showing the control sequence underlying the terminal cost according to definition 8.1.4.

8.2. Complexity and Performance Analysis

The application of the GP requires a sufficiently dense discretization of the state space and time. In addition, the trajectory parameterization must have sufficient degrees of freedom. Since the GP shall be part of a hierarchical algorithm that includes the LP in the second stage, searching a graph for the best solution close to the optimal one in the continuous space is not required. Instead, the GP shall identify LTs enabling progress toward the GT and a corresponding feasible initial guess for the LP. This section analyzes the influence of the breakpoint number and the spline segment lengths on the GP's control performance and complexity. The aim is to find a configuration to likely progress toward the GT at a reasonable computational effort.

The GP's result trajectories are analyzed in different scenes at the starting point *s2*, shown in figure 3.2, to design the discretization. While a closed-loop analysis provides more accurate inference, it is more computationally demanding. Additional factors, such as the cost design and the accuracy of the other vehicles' predictions would also influence the results. Two scene variations, consisting of twenty-nine highway scenes each, are considered for the following analysis. The $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ vehicles' positions $z_{x,m}(0)$ and velocities $\dot{z}_{x,m}(0)$ corresponding to each scene are drawn uniformly from a set of positions and velocities $\mathcal{Z}_{v,x,m} \subset \mathbb{R}^+$. The vehicles' initial position intervals for both scene variations are marked along the lane centers in figure 8.5. The first scene variation, called *SC-3V*, includes three vehicles. The EV is placed in the middle lane, considering the right-most lane at $\tilde{v} = 122\,\mathrm{km\,h^{-1}}$ the GT. *Vm* labels the other vehicles. Vehicle *V1* is placed behind or in front of the EV in the left lane and travels at a velocity between $100\,\mathrm{km\,h^{-1}}$ and $120\,\mathrm{km\,h^{-1}}$. Thus, *V1* is faster than the EV but slower than the EV's target velocity. Vehicle *V2* starts at different locations in front of the EV in the middle lane while driving with the same initial velocity. The second scene variation, *SC-2V*, includes two vehicles in total. The initial lane, the target lane, and the target velocity of *SC-3V* are retained. *V3* is in the right lane, driving slower and at different positions in front of the EV. The scenes are selected such that many of the GP's candidate trajectories are likely to collide with the other vehicles' predictions, allowing to observe the effect of the discretization and the degrees of freedom on the open-loop control performance. The vicinity to the other vehicles is crucial when assessing the performance advantage due to additional breakpoints since a polynomial trajectory is optimal in the absence of obstacles.

The discretization of time, y-position and x-velocity are chosen, so a suitable performance is expected in the considered scene variations. The time is discretized in the $\nabla = x, y$

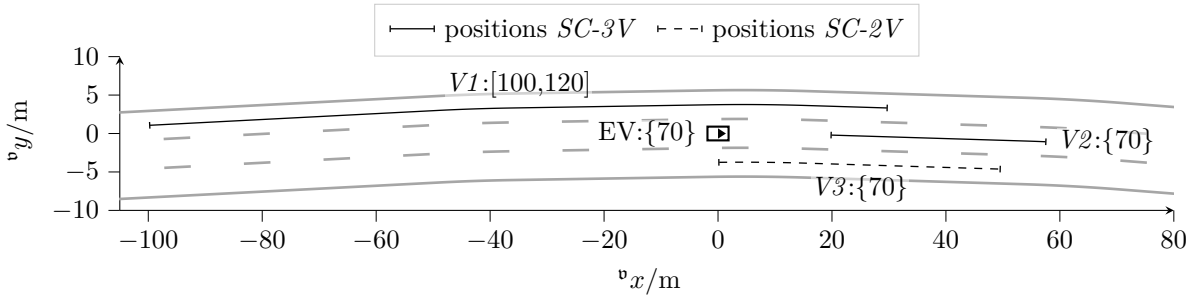


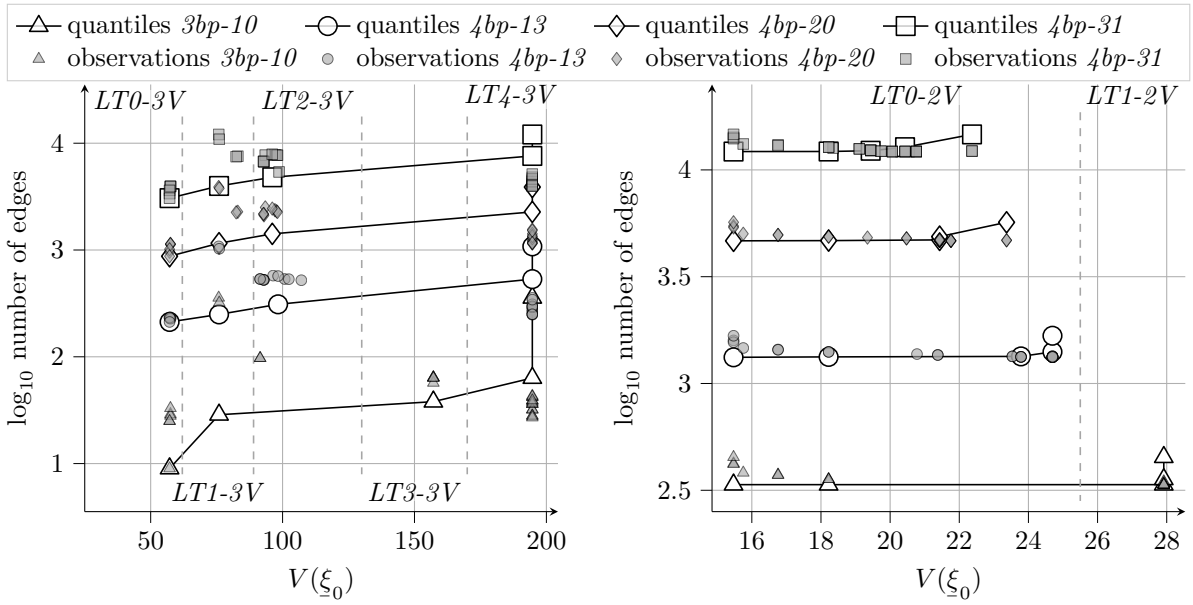
Figure 8.5.: Road layout, the initial position, and the initial velocity intervals for the $m = 1, 2, 3$ vehicles $Vm : \mathcal{Z}_{v,x,m}/\mathrm{km\,h^{-1}}$ for the generation of the scene variations *SC-3V* and *SC-2V*. The vehicles *V1* and *V2* belong to *SC-3V*. *V3* is used in *SC-2V*. The EV, which is part of all scene variations, is indicated by EV: $\dot{z}_x/\mathrm{km\,h^{-1}}$.

direction to $\mathcal{K}_\triangledown = \{0\,\mathrm{s}, 1\,\mathrm{s}, \ldots, 10\,\mathrm{s}\}$. The y-position's discretization includes the current and neighboring lane centers. The x-velocity is discretized equidistantly with $14.25\,\mathrm{km\,h^{-1}}$ between the minimum feasible and the target velocity to obtain $\mathcal{Z}_{\dot{x}}$ with five elements. The spline segment lengths in table 8.1 defines the feasible breakpoint sequences. In total, four different configurations are considered. The configuration *3bp-10* allows trajectory candidates with two or three breakpoints. The two-breakpoint sequence connects the initial nodes directly with the terminal nodes and is only considered by the GP if the EV is located in the terminal set. The initial segment length of candidate trajectories with three breakpoints is contained in $\mathcal{I}_3$. In total, ten possible breakpoint sequences exist in each direction. Three additional configurations consider four breakpoints with intervals from $\mathcal{I}_4$ for the first two segments. The number of feasible breakpoint sequences grows with the feasible segment lengths. As the configuration complexity increases, the discrete set of candidate trajectories becomes increasingly refined. Thus, each configuration performs at least as well as those with fewer breakpoint sequences. Simultaneously, the complexity of the graph search for the same scene grows with an increase in the number of breakpoints.

Table 8.1.: Ranges of the allowed breakpoint intervals and the resulting number of candidate breakpoint sequences.

configurations	$\mathcal{I}_3$/s	$\mathcal{I}_4$/s	breakpoint sequence number
3bp-10	[1,9]	$\emptyset$	10
4bp-13	[1,9]	[4,5]	13
4bp-20	[1,9]	[3,6]	20
4bp-31	[1,9]	[2,7]	31

The number of edges added to the graph in the search process is used as a hardware-independent complexity metric. A node expansion, described in algorithm 7.2.1, essentially includes the determination of potential successor nodes, evaluating the corresponding edges' feasibility and cost, and performing the update of predecessors and cost. The constraint evaluation, which involves the matrix multiplications for spline coefficient transformations, is terminated early if a segment is infeasible. If the current segment is feasible, the



(a) Control performances and edge counts in scene variations *SC-3V*.

(b) Control performances and edge counts in scene variations *SC-2V*.

Figure 8.6.: Shows the open-loop control performance and complexity metrics for all twenty-nine scenes from *SC-3V* and *SC-2V* and for all variations of the GP configuration. The data are partitioned into four groups between the 0 %, 25 %, 50 %, 75 %, and 100 % quantiles. In addition, the observations are grouped into the LTs selected by the GP.

constraint evaluation is the most computationally expensive step. Only feasible edges are added to the graph, requiring the evaluation of the cost and processing all constraints. Thus, the computational complexity is the same for all feasible edges except for the terminal segments with their reduced constraint set. In contrast, the number of infeasible edges is not considered since they are discarded mostly early in the constraint evaluation due to their kinematic infeasibility without checking the expensive collision constraints. The performance is assessed via the cost of the solution trajectory $V(\underline{\xi}_0)$. The open-loop cost allows a conclusion to the closed-loop performance only in scenarios where the assumptions underlying the cost design are fulfilled, and the terminal cost is a tight upper bound on the minimum cost-to-go.

Figures 8.6a and 8.6b show the cost distributions and the number of edges for each configuration and scene variation. Four groups partition the data at 25 % intervals to summarize the quantitative properties of the cost and complexity distributions. In addition, the LT corresponding to each observation is indicated. The number of observations for each LT and GP configuration is summarized in tables 8.2a and 8.2b to complement the partially overlapping marks in figures 8.6a and 8.6b. In general, one can observe a monotonous increase in the number of edges with the growing configuration complexity. However, the growing complexity does not necessarily result in a cost reduction.

The cost is complexity invariant in the lower quartile for both scene variations. The optimal trajectory is a single polynomial segment, mostly ending in the feasible LT of minimum terminal cost. The respective LT (*LT0-3V*) in *SC-3V*, which is selected in six cases, is at the target velocity and requires overtaking to the left lane. Simultaneously,

Table 8.2.: Provides an overview on the number of selected LTs in figure 8.6 for each GP configuration.

(a) Counts of LT selections for each GP configuration in scene variations *SC-3V*.

LTs	*3bp-10*	*4bp-13*	*4bp-20*	*4bp-31*
LT0-3V	6	6	6	6
LT1-3V	2	2	4	4
LT2-3V	2	10	8	9
LT3-3V	5	0	0	0
LT4-3V	14	11	11	10

(b) Counts of LT selections for each GP configuration in scene variations *SC-2V*.

LTs	*3bp-10*	*4bp-13*	*4bp-20*	*4bp-31*
LT0-2V	21	29	29	29
LT1-2V	8	0	0	0

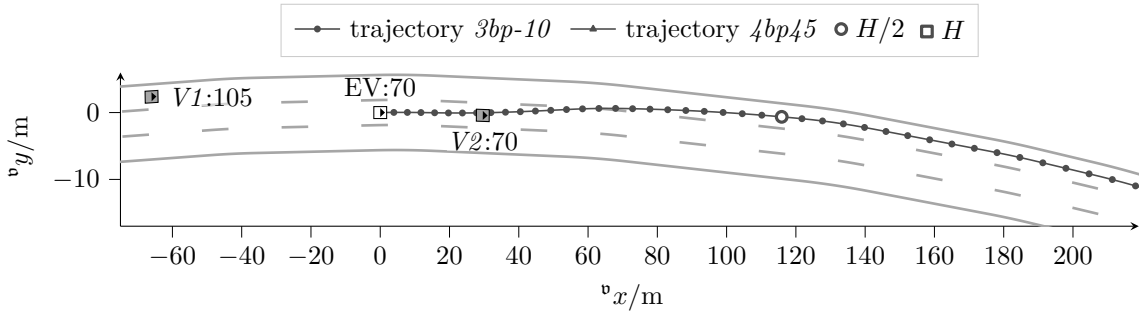
it is located in front of *V1*. An exemplary scene is depicted in figure 8.7a. In two cases, the GP selects *V1* as the leading vehicle (*LT1-3V*), which is the option with the second lowest terminal cost. All trajectories from the lower quartile in *SC-2V* terminate in the GT on the right-most lane and end at the target velocity (*LT0-2V*). Figure 8.8a shows a corresponding scene.

All fourteen result trajectories of maximum cost in the *SC-3V* scenes due to *3bp-10* target the right-most lane (*LT4-3V*) and are prone to violating the right overtaking rule. One scene is shown in figure 8.7b. An additional breakpoint enables the GP to follow *V2* (*LT2-3V*) instead in four cases with *4bp-31* and three cases with *4bp-13* and *4bp-20*. The corresponding costs are distributed close to the configurations' respective median.

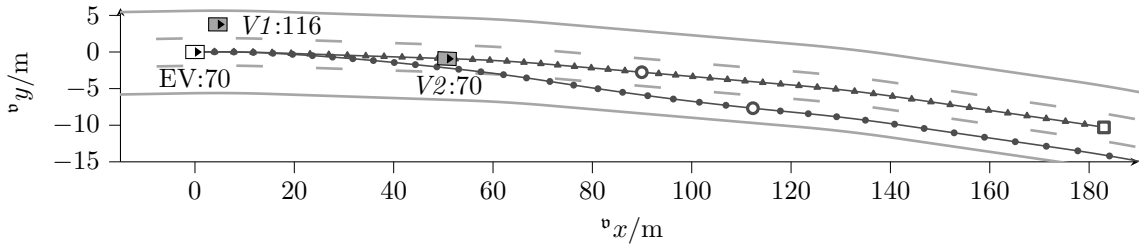
A fourth breakpoint also improves the *3bp-10* results between the upper and lower quartiles. In five scenes with the *3bp-10* configuration, the GP targets the left lane behind *V1* at the target velocity (*LT3-3V*), like in figure 8.7c. *V1* is sufficiently close to be classified as an impeding vehicle, resulting in the second-highest terminal cost. *LT2-3V* is selected in two additional cases. All four breakpoint configurations select *LT2-3V* instead of *LT3-3V*. However, only *4bp-31* and *4bp-20* can reduce the cost in the two additional cases by targeting *LT1-3V* instead of *LT2-3V*.

In contrast to the *SC-3V* scenes, the cost beyond the lower quartile from the *SC-2V* scene variations continuously improve with increasing configuration complexity. In the twenty-one corresponding cases, the GP with a *3bp-10* configuration chooses a trajectory toward the middle lane (*LT1-2V*), shown in figure 8.8b. In the same scenes, all four breakpoint configurations plan a trajectory to *LT0-2V*, resulting in lower overall cost. Although all four breakpoint configurations share the same LT, their increasing complexity reduces the overall cost due to lower running cost.

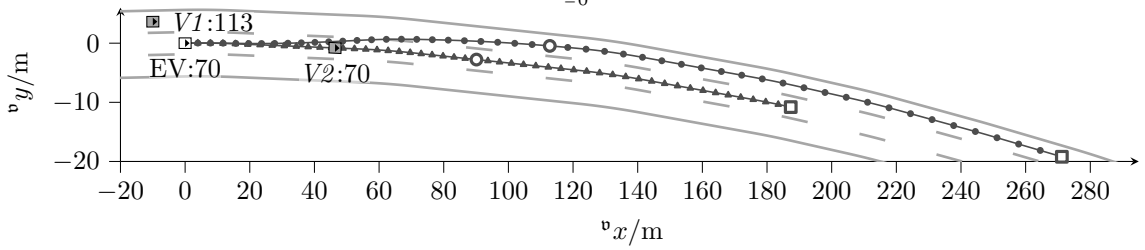
Because of the prioritized shortest-path search, the number of graph edges depends on the number of feasible candidate trajectories and their cost distribution in the current scene. If the number of edges added is low, the optimal solution may be among the branches with the lowest cost. Also, many branches can become infeasible early in the search process if the feasible set is small. The overall number of edges is greater in figure 8.6b than in figure 8.6a for the same breakpoint configurations. Since *SC-2V* only considers a single other vehicle, the graph has a higher number of feasible branches to explore. For the same reason, figure 8.6b shows a higher number of edges for the low-cost cases. The other vehicle



(a) Scene showing a left lane change to the target velocity, with minimum cost $V(\underline{\xi}_0) = 57.4$ for all configurations



(b) Scene showing a right lane change inducing left impeding cost and $V(\underline{\xi}_0) = 194.6$ for *3bp-10*. The *4bp45* configuration follows *V2* inducing $V(\underline{\xi}_0) = 92.8$.

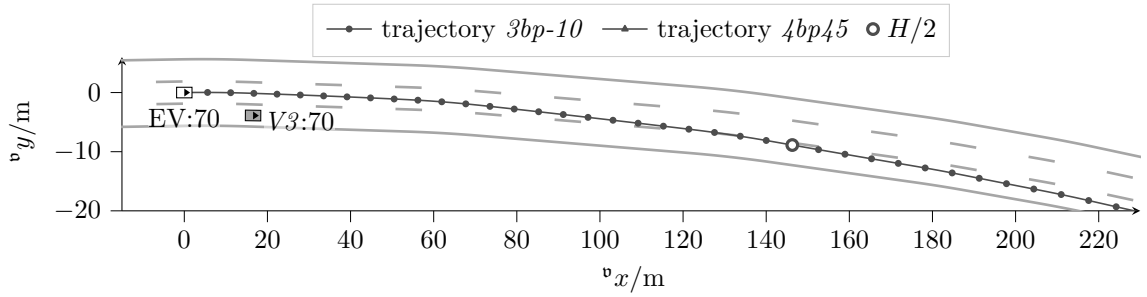


(c) Scene showing a left lane change resulting in impeding vehicle cost with $V(\underline{\xi}_0) = 157.1$ for *3bp-10*. The *4bp45* configuration follows *V2* inducing $V(\underline{\xi}_0) = 98.1$.

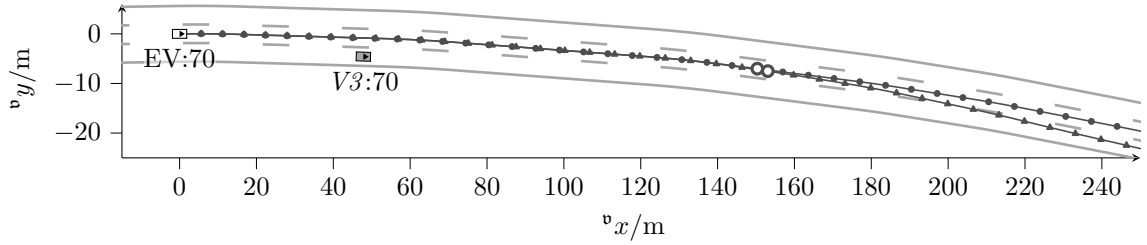
Figure 8.7.: Exemplary scenes from *SC-3V*. The EV is indicated by EV: $\dot{z}_x/\mathrm{km\,h^{-1}}$ and the $m = 1, 2$ other vehicles with $Vm : \dot{z}_{x,m}/\mathrm{km\,h^{-1}}$.

is located on the right lane close in front of the EV, interfering with fewer trajectories than in the cases of higher cost. In contrast, the variation in the number of edges is higher in figure 8.6a, as two vehicles can render more trajectory candidates infeasible. The lowest edge count is observed in the minimum and maximum cost cases. Either the minimum cost trajectory toward the target lane is found quickly, or most trajectory candidates are eliminated early.

The highest increase in the median complexity is observed in *SC-3V* scenes from *3bp-10* to *4bp-13* with 8.13 times more edges. The increase saturates as the number of breakpoints increases with 4.57 times between *4bp-13* and *4bp-20*, and 3.39 times between *4bp-20* and *4bp-31*. In contrast, the median complexity increase in *SC-2V* scenes is lower since more splines that are composed of three breakpoints are feasible candidates. The increase between the configurations of lowest complexity amounts to 3.98 times. Between the four breakpoint configurations, the edge count increases by 3.47 and 2.63 times.



(a) Scene showing a right lane change to the target velocity, with minimum cost $V(\underline{\xi}_0) = 18.2$ for all configurations



(b) Scene showing lane keeping to the target velocity and $V(\underline{\xi}_0) = 27.8$ for *3bp-10*. The *4bp45* configuration approaches the GT with $V(\underline{\xi}_0) = 24.5$.

Figure 8.8.: Exemplary scenes from *SC-2V*. The EV is indicated by EV: $\dot{z}_x/\mathrm{km\,h^{-1}}$ and the other vehicle with *V3*: $\dot{z}_{x,m}/\mathrm{km\,h^{-1}}$.

In conclusion, the computational complexity of the graph search grows for all scenes with increasing configuration complexity. The increase in complexity decreases with a growing number of candidate breakpoints. Simultaneously, an increasing traffic density also increases the change in the number of edges between the different configurations. Since the polynomial segments are optimal for the running cost, the open-loop performance only benefits from the increased complexity if other vehicles render trajectories with three breakpoints infeasible. While an increase in complexity among the four breakpoint configurations reduces the running cost for *SC-2V*, a benefit for the terminal cost is observed only in about 10 % of the *SC-3V* scenes in figure 8.6a. In total, the *4bp-13* configuration already reduces the terminal cost in 16 scenes compared to *3bp-10*. Simultaneously, the *4bp-45* achieves a terminal cost reduction in only one additional scene. Therefore, *4bp-13* is chosen as a suitable performance-complexity compromise among the four breakpoint configurations for the evaluation in the next chapter.

9

Analysis of Two-Stage Trajectory Planning in Highway Scenarios

The LP and the GP are combined in a two-stage hierarchical algorithm. The GP considers the combinatorial nature of problem 4.3.2 and provides a feasible but suboptimal trajectory and an LT to the LP. The LP either refines the solution from the previous time step or the GP's solution. The properties of the distinct planning algorithms are evaluated in the previous chapters. The following evaluation aims to analyze the hierarchical algorithm's closed-loop performance, the stability, and the control performance's sensitivity to the trajectories' degrees of freedom. Therefore, two planner configurations resulting in different computational complexities are applied in closed-loop to several highway scenario variations. Finally, the planning algorithm's ability to implement the development criteria from section 4.1 is assessed.

9.1. Two-Stage Trajectory Planner

The GP and the LP are planning algorithms with complementary properties. A combination of both algorithms in a hierarchical manner is proposed, in line with the automated driving control hierarchy in figure 1.2, to mitigate their respective disadvantages. Subsequently, the properties of the distinct planning algorithms and those of the hierarchical combination are discussed, followed by the implementation of the hierarchical planner. Parts of this section have been published in [DB24].

9.1.1. Planner Properties

The GP searches over a discrete set of candidate trajectories and yields the global optimal trajectory available in the resolution of the state space and time discretization. The resulting trajectory is feasible considering the constraints introduced in chapter 5. Due to its discrete nature, the GP approximates the solution to problem 4.3.2 and decides implicitly on the next maneuver, the number of spline breakpoints, and their ordering. The choice of the solution trajectory is based on the running and terminal cost, which guide the EV into the GT. However, the state space decomposition is not designed for recursively feasible results. In addition, the GP likely returns a suboptimal trajectory due

to its limited resolution.

In contrast, the LP returns the local optimal trajectory for a given LT by solving the NLP in problem 6.2.1. The nonlinear optimization relies on an initial guess for the optimization variables, which reduces the iterations until convergence with a decreasing distance from the optimal solution. A feasible initial guess increases the likelihood of a successful convergence because the iterations can approach a stationary infeasible point otherwise. The LP uses a shrinking horizon toward the LT to ensure recursive feasibility in the nominal case. Disturbances can invalidate the recursive feasibility and may require selecting another LT.

The combination of the GP and the LP is performed hierarchically. The GP selects the global optimal trajectory among its candidates. The LP either improves its previous solution or improves the feasible GP solution locally. Thus, the EV converges toward the GT while the control performance during the LT convergence is invariant to the GP's resolution. The LP preserves recursive feasibility, so the currently selected maneuver is not aborted due to insufficient state space or time discretization. Moreover, if disturbances invalidate the recursive feasibility, the GP likely provides a feasible initial guess.

9.1.2. Planner Combination

The structure of the two-stage planning algorithm is depicted in figure 9.1. The GP is located at the top of the hierarchy and passes its solution trajectory to the initial selection module. In addition to the GP's trajectory, the breakpoint adaption provides the transformed trajectory from the previous time step. The initial selection module passes the feasible trajectory of minimum cost to the LP and to the solution selection module. The LP solves problem 6.2.1, starting with an initial guess on the optimization variables based on the given trajectory. Subsequently, the solution selection module receives the optimal LP trajectory and the initial trajectory. Like the initial selection, the solution selection decides on the solution trajectory considering the trajectories' cost and their feasibility.

Initial selection module chooses the transformed trajectory as the initial trajectory if the trajectory is feasible for the constraints (6.2.2) and its cost (4.3.8) is lower than the cost of the GP solution trajectory. The transformed trajectory is likely selected if the disturbances are low and the current maneuver is retained, even if the GP decides on a different one. Depending on the discretization and the scenario, the GP may switch between LTs frequently. Thus, the initial selection contributes to the consistency of the two-stage planner's solution. If the GP does not find a solution or none of the available trajectories is feasible, the transformed trajectory is also selected. The initial selection chooses the GP trajectory if it yields lower cost or disturbances render the transformed trajectory infeasible. The former case likely applies if a new LT of lower terminal cost than the current one becomes reachable in the GP's graph. Thereby, the two-stage planner still enables convergence toward the GT. In addition, it provides a suboptimal backup solution which the LP subsequently improves.

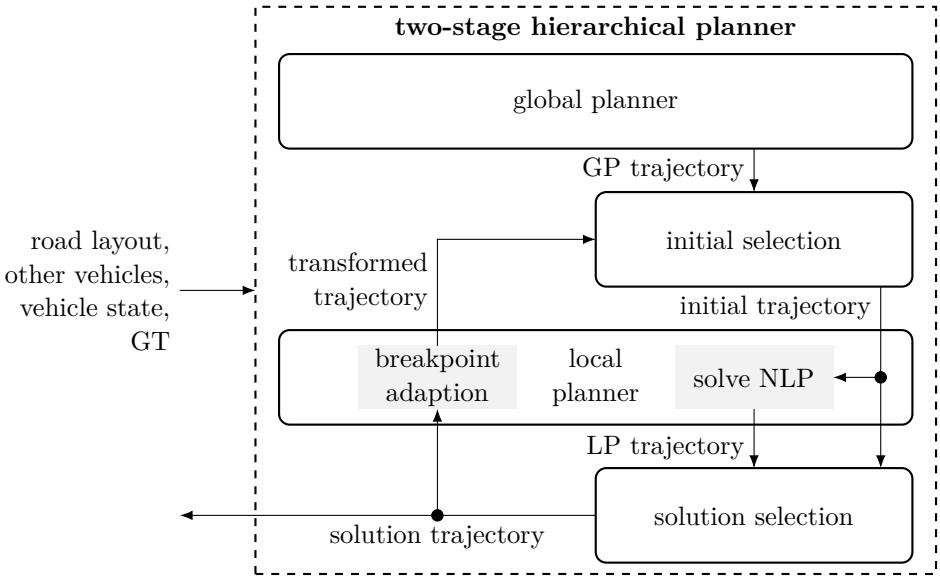


Figure 9.1.: Overview on the modules of the two-stage planning algorithm.

Solution selection module returns the LP trajectory only if it is feasible and costs less than the initial trajectory or if the initial trajectory is infeasible. Otherwise, the initial trajectory is selected if feasible. Thus, the solution selection handles a failed IPOPT convergence, which is essential if the optimization algorithm's iterations are limited. The initial guess guarantees convergence toward the LT if the recursive feasibility applies. If it does not apply, the GP supplies a suboptimal solution for the LP to improve in the next step. In contrast to the initial selection, the solution selection module only returns a solution if a feasible one is available. Otherwise, the planning algorithm might trigger a backup strategy to retain the vehicle's safety.

9.2. Experiment Setup

The two-stage planning algorithm is applied to five scenarios of 85 s duration each. The initial x-positions, the initial velocities, and the desired velocities of six other vehicles and the EV are drawn independently from continuous uniform distributions. The distributions are visualized in figure 9.2. The EV uses the starting point *s1*, which marks also the target lane on the three-lane highway in figure 3.2. The target velocity is $\tilde{v} = 122\,\mathrm{km\,h^{-1}}$. The six other vehicles are distributed such that two are in each lane center. The distribution of each vehicle's initial and target velocity is the same. The vehicles on the left-most lane drive at the highest velocity beyond the EV's desired speed. The vehicles in the middle lane are slower than the EV's target velocity but faster than the vehicles in the right-most lane. All vehicles from the middle and the right-most lane start in front of the EV.
The other vehicles' placement and velocities require the EV to leave its initial lane and overtake via the left-most lane to reach the GT. Vehicle *V2* starts behind the EV. Thus, the planning algorithm must decide whether to perform the lane change to the left-most lane in front or behind *V2*. The inter-vehicle gaps in the middle and right-most lanes vary among the scenarios, depending on the difference between the vehicles' desired velocities.

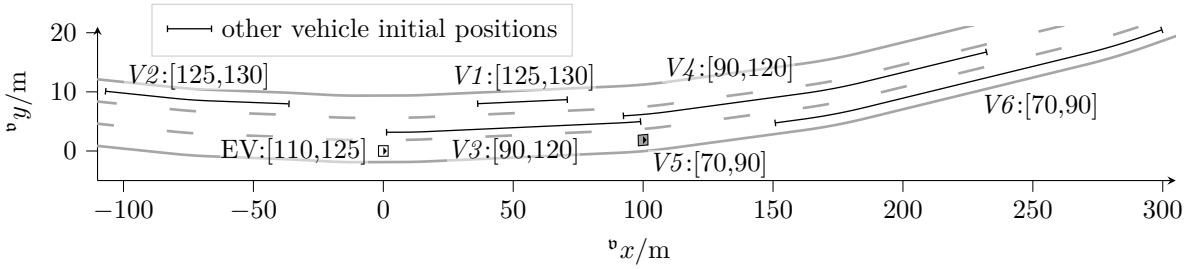
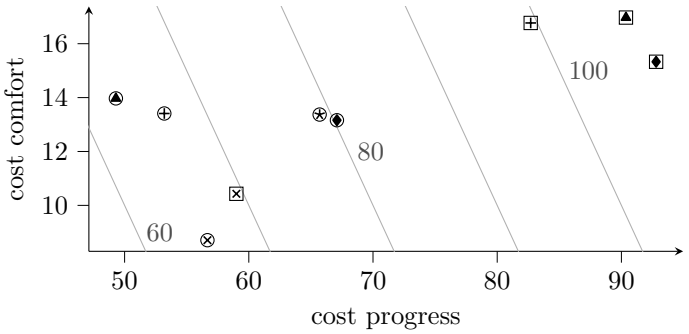


Figure 9.2.: Initial velocity and position intervals along the lane centers. The EV is indicated by EV: $\mathcal{Z}_{v,x,0}/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \mathcal{Z}_{v,x,m}/\mathrm{km\,h^{-1}}$. Deterministic positions are marked with the actual vehicle shape.

If the gaps are sufficiently large, the EV merges in between to reach the GT quicker. Each scenario can be decomposed into three phases. The first phase, called *initial lane changes*, denotes the time from the scenario start until reaching the left-most lane. The second one, called *left transition*, involves passing the other vehicles via the left-most and the middle lane. The final sequence of lane changes toward the GT are the *final lane changes*.
In the following, two configurations of the two-stage planning algorithm are evaluated. The configurations *3bp-10* and *4bp-13* are the same as in the section 8.2 with the same prediction horizon of $H = 10\,\mathrm{s}$. The *4bp-13* configuration is identified in section 8.2 as a good performance-complexity trade-off for the GP. *4bp-13* is of higher complexity than *3bp-10* since it considers trajectories with four breakpoints in addition to those generated with *3bp-10*. The IPOPT iteration limit $i_{\mathrm{ter}} \in \mathbb{R}^+$ is set to $i_{\mathrm{ter}} = 20$, which is found a suitable compromise between the complexity of the local optimization and the resulting control performance gain. An analysis of the achieved cost reduction and feasibility rate, depending on the maximum iterations, is provided in section A.7.

9.3. Evaluation

The configurations *3bp-10* and *4bp-13* are applied to the same five scenarios generated with the initial state distributions in figure 9.2. An overall impression of the control performance is provided in figure 9.3. The closed-loop cost is computed as in section 6.4 by integration of the transformed running cost (4.2.7) evaluated at the closed-loop trajectory over time. In contrast, the time integral ends as the GT is reached the first and the second time. The cost is reported in section 6.4 as the sum of the comfort and progress costs achieved before reaching the GT for the first time. The marks are a combination of two mark sets. The inner marks denote the scenario. The algorithm configuration is indicated by the enclosing circle or square shape. Both configurations reach the GT before the simulation ends at least once, except from *3bp-10* in scenario 2. In scenario 0, the EV leaves the GT after reaching it the first time to achieve convergence a second time later in the simulation. In all scenarios, the *4bp-13* configuration results in lower closed-loop comfort, and progress costs than *3bp-10*. Reaching the GT earlier with *4bp-13* mainly contributes to the cost differences. The highest difference is observed for scenario 3, and the lowest one for scenario 0.
The marks of *4bp-13* can be separated into a set of low-cost scenarios below a closed-loop



	3bp-10		*4bp-13*	
	□ 1st	2nd	○ 1st	2nd
× scenario 0	69.43	114.1	65.38	110.2
+ scenario 1	99.47	-	66.61	-
⋆ scenario 2	-	-	79.07	-
▲ scenario 3	107.30	-	63.27	-
♦ scenario 4	108.10	-	80.26	-

Figure 9.3.: Closed-loop cost values for comfort and progress when reaching the GT. In addition, the contours of the closed-loop cost are shown. The table on the right side provides the closed-loop cost when reaching the GT the first and the second time as a sum of comfort and progress.

cost of 70 and a set of high-cost scenarios beyond 70. Scenario 2 is described in detail as an example of a high-cost scenario. Afterward, the qualitative difference between scenario 2 and the low-cost scenarios is pointed out.

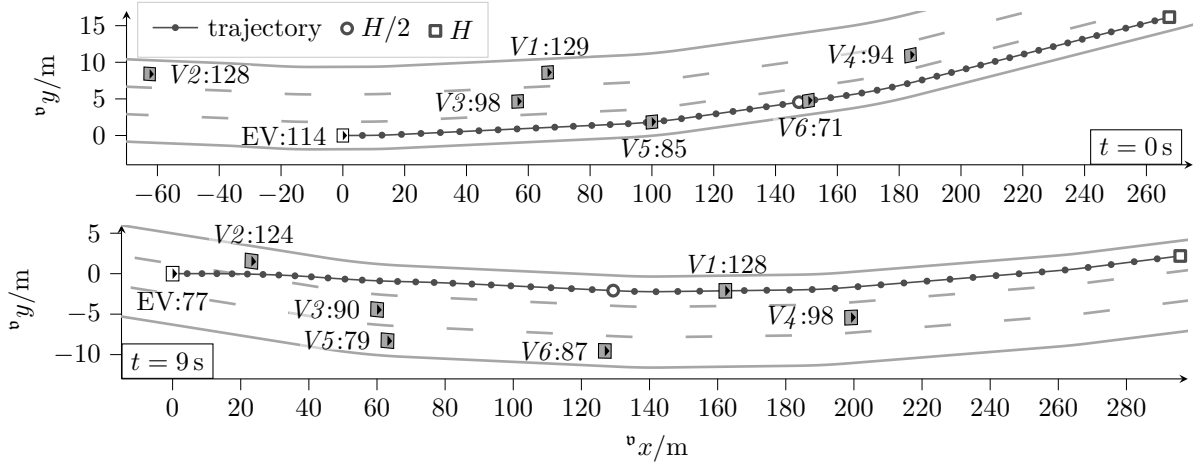


Figure 9.4.: Initial double lane change from the right-most to the left-most lane during the *initial lane changes* phase in scenario 2 with application of *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.

In scenario 2, the *initial lane changes* start with the EV on the right-most lane behind the slower *V5*. Figure 9.4 shows two scenes from the *initial lane changes* phase. In the first 4 s, the planning algorithm decides to follow *V5*, resulting in a deceleration. Simultaneously, the distance toward *V3* on the middle lane is reduced. At $t = 4\,\mathrm{s}$, the planning algorithm initiates a lane change because following *V3* yields a lower overall cost than following *V5*. As soon as the EV enters the middle lane at $t = 7.1\,\mathrm{s}$, the algorithm further reduces the terminal cost by changing to the left-most lane. *V2* travels faster than the target velocity. Thus, the optimal trajectory reaching the target velocity behind *V2* yields zero terminal cost in the x-direction. The left-most lane center is reached at $t = 12.5\,\mathrm{s}$.
The *left transition* phase starts on the left-most lane, where the target velocity is finally reached at 16 s. Two scenes from the phase are visualized in figure 9.5. *V4* maintains a higher velocity than *V3*. Thus, the gap between the two vehicles grows over time. At

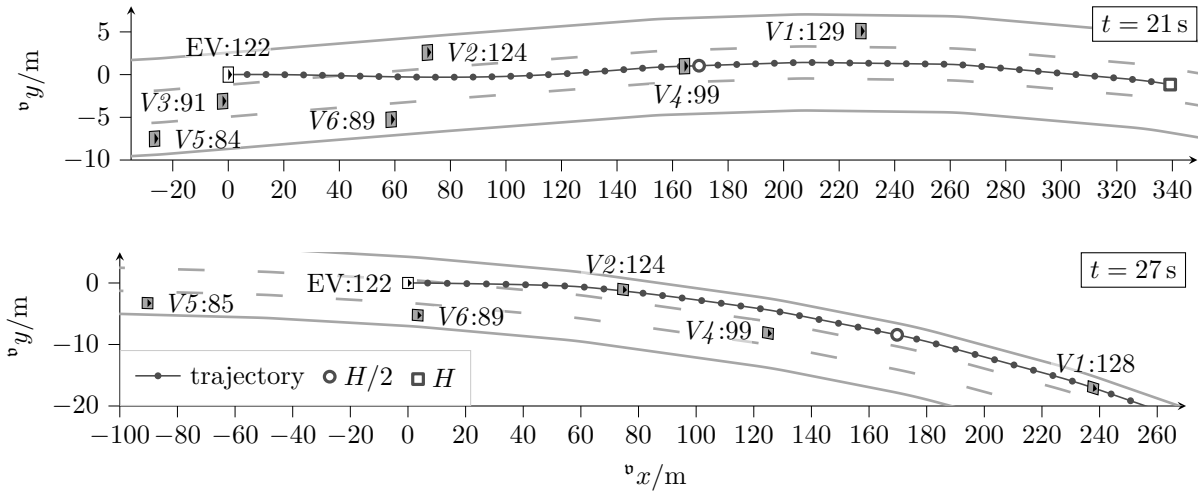
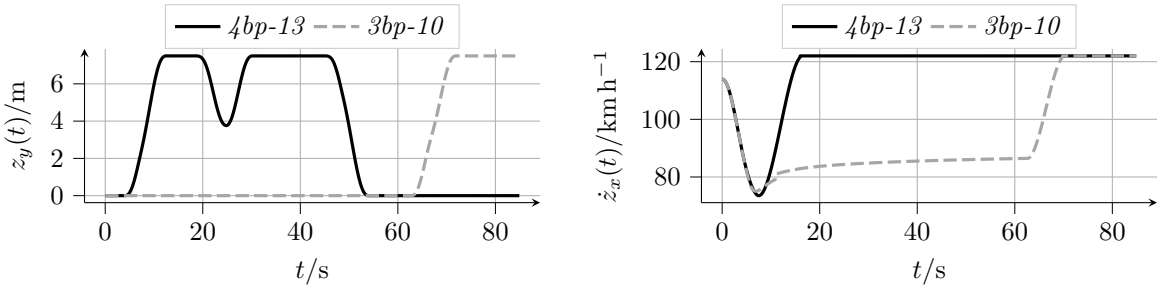


Figure 9.5.: Lane changes between the left-most lane and the middle lane during the *left transition* phase with application of *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.

$t = 18.6\,\mathrm{s}$, the gap is sufficiently large to initiate a lane change to the middle lane. The EV cuts in front of *V3* with a time headway of about 0.51 s, so *V3* decelerates to maintain its desired time headway. According to the terminal cost assumptions, only one additional lane change from the middle to the right-most lane is required to reach the GT. However, the right lane is still occupied by *V6*. Simultaneously, the EV approaches *V4* until it is considered an impeding vehicle at $t = 24.6\,\mathrm{s}$. A lane change to the left is the best maneuver in this situation since continuing to the target lane either causes a collision with *V6* or violates the left overtaking rule. The EV reenters the left-most lane center at $t = 30.2\,\mathrm{s}$ behind *V2*. The left-most lane is tracked until the end of the *left transition* phase at $t = 45\,\mathrm{s}$. The target velocity is retained during the changes between the left-most and the middle lane, as shown in figure 9.6b. In contrast, the velocity due to *3bp-10* is lower because the EV follows *V5* for an extended duration.



(a) y-position trajectory.

(b) x-velocity trajectory.

Figure 9.6.: Closed-loop trajectories of the EV in scenario 2.

The planning algorithm initiates the *final lane changes* phase at $t = 45.1\,\mathrm{s}$. In the following seconds, the EV cuts in front of *V4* with a time headway of 0.47 s and plans a trajectory into the GT as it reaches the middle lane. The target lane is finally reached at $t = 53.9\,\mathrm{s}$ and tracked until the scenario ends. Scenes from the second-last and the last lane change

are show in figure 9.7.

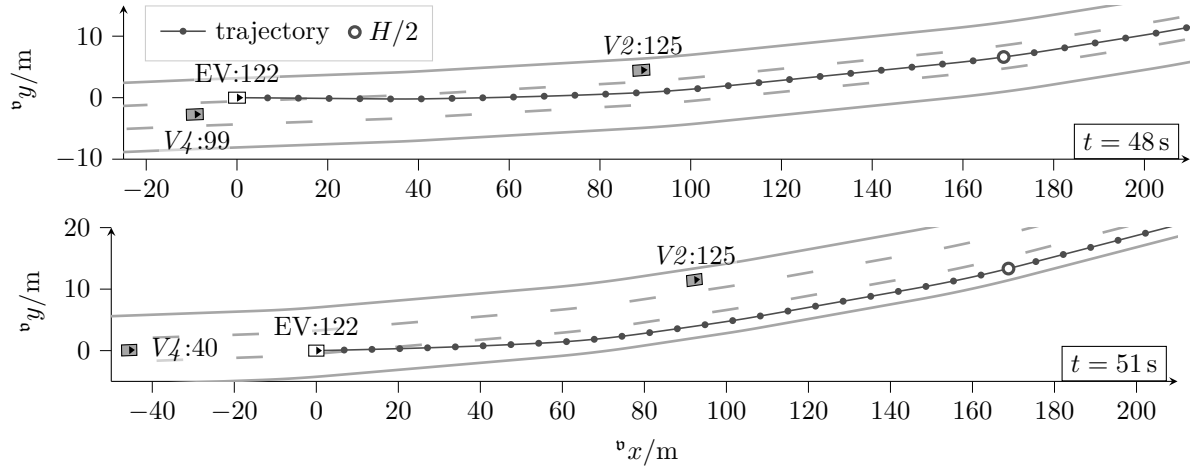


Figure 9.7.: Double lane change to the GT during the *final lane changes* phase in scenario 2 with application of *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.

One of the main qualitative differences between the high-cost scenarios 2 and 4 is observed during the *left transition* phase. In scenario 4, the EV does not switch to the middle lane but stays in the left-most lane until the phase ends. The reason is that *V4* starts 88 m closer in front of the EV, so the gap between *V3* and *V4* is too small for the EV to merge into. Another difference to scenario 2 is a 2.2 s earlier lane change initiation to the middle lane in the *initial lane changes* phase. However, the EV spends more time following *V3* until *V4* overtakes the EV on the left-most lane. The increased cost in scenario 4 is due to the higher jerk in the *initial lane changes* phase and the phase's extension by 2.5 s.
In all three *4bp-13* lower-cost scenarios, the EV reaches the GT earlier. In scenario 1, *V4* starts 81 m closer to the EV compared to scenario 2. Simultaneously, *V4* drives slower than in the scenarios 2 and 4. Consequently, the EV overtakes the vehicles in the middle lane more quickly, so the *final lane changes* phase to the target lane ends 12.1 s earlier than in scenario 2. During the scenarios 0 and 3 the gap between the *V3* and *V4* grows sufficiently large for the EV to merge into it. In contrast to scenario 2, *V6* travels at a lower velocity while the EV decelerates less during the *initial lane changes* phase. Thus, the EV overtakes *V6* before changing from the left-most to the middle lane. Simultaneously, *V4* converges to an $8\,\mathrm{km\,h^{-1}}$ higher velocity in scenario 0 and an $18\,\mathrm{km\,h^{-1}}$ higher velocity in scenario 3 compared to scenario 2. Consequently, *V4* is not considered impeding while the EV changes to the middle lane. The circumstances reduce the duration of the *left transition* phase from 32.4 s in scenario 2 to 14.4 s in scenario 0 and 23.7 s in scenario 3. Thus, the EV reaches the target lane 12.8 s and 16.1 s earlier than in scenario 2 without overtaking *V4*.
In contrast to the other scenarios, the EV reaches the GT a second time in scenario 0. Figure 9.8 shows a scene at the beginning of the second *initial lane changes* phase at $t = 44.5\,\mathrm{s}$. Because *V4* still cruises at a lower speed than the EV's desired one, the EV catches up. The left overtaking rule requires another double lane change to overtake *V4*. Afterward, the remaining time is sufficient to reach the GT a second time at $t = 83.2\,\mathrm{s}$.

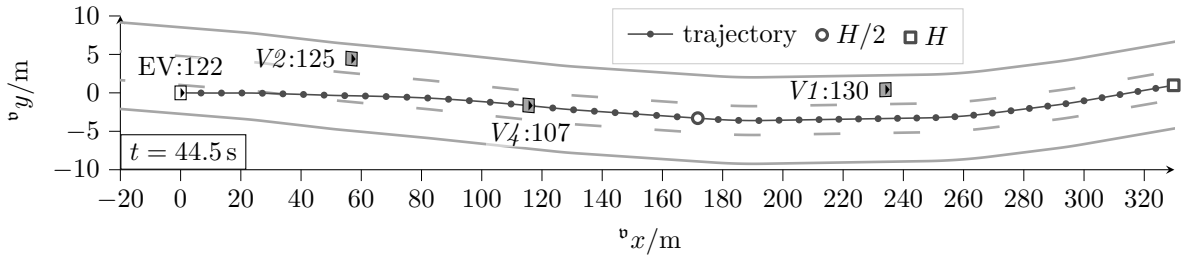


Figure 9.8.: Shows the *initial lane changes* phase after first reaching the GT in scenario 0 with application of *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.

In all scenarios, the EV accelerations are feasible and stay within $|a_x(t)| \leq 2.83\,\mathrm{m\,s^{-2}}$ and $|a_y(t)| \leq 1.35\,\mathrm{m\,s^{-2}}$. Most of the time, the EV time headway toward its leading vehicle always exceeds 1.9 s. When the EV merges behind *V2* at the end of the *initial lane changes* phase the time headway drops to 0.47 s in all scenarios. The current cost descent factor (6.4.1) always stays positive except for three situations. In the first situation, the EV follows the initial leading vehicle *V5* in the scenarios 2 and 4. The acceleration performed by *V5* impedes the convergence toward its target trajectory over a limited time. The second situation occurs when the vehicle in front is considered impeding, as with *V3* in scenario 2. In this case, a cost increase encourages a lane change or following the impeding vehicle. In the third situation observed in scenario 4 the planning algorithm attempts a lane change during the *initial lane changes* phase in front of *V2*. However, the lane change is aborted because the trajectory onto the left-most lane is rendered infeasible 0.4 s after the lane change initiation. Instead, the EV continues to follow *V3* and terminates the *initial lane changes* phase behind *V2*.

Applying the *3bp-10* configuration increases the closed-loop cost in all scenarios compared to *4bp-13*. The reason in all scenarios, except for scenario 0, is an extension of the *initial lane changes* phase due to a delayed change to the middle lane. The delay ranges between 5.9 s and 58.7 s compared to the respective scenarios using the *4bp-13* configuration. The highest one is observed in scenario 2, preventing the convergence to the GT in 85 s. Figure 9.6a shows the y-position in scenario 2. The EV stays 58.7 s longer on the initial lane due to the *3bp-10* configuration because the planning algorithm cannot find a feasible trajectory to follow *V3* at the desired time headway. Simultaneously, the impeding vehicle terminal cost applies to the trajectories that reach the target velocity behind *V3*. Thus, the EV follows *V5* until *V3* is sufficiently far in front to no longer be considered impeding. In figure 9.6b, one can also observe the resulting lower velocity. The time until the distance to *V3* is sufficient depends on the velocity difference between *V3* and *V5*. The difference is maximum in scenario 3, resulting in the lowest delay. Afterward, the EV tracks the middle lane until *V3* is considered impeding. The cost increase triggers the lane change to the left-most lane. The EV behavior in the scenarios 0 and 3 is in line with the *4bp-13* configuration. The gap between *V3* and *V4* is sufficiently large for the EV to merge in between, followed by a lane change to the target lane. In the other scenarios, *V4* must also be overtaken to reach the GT.

The *3bp-10* configuration performs closest to *4bp-13* in scenario 0. *V3* starts at a distance

of 77 m in front of the EV, which is the longest distance among all scenarios. Consequently, the GP's optimal trajectory at $t = 0\,\mathrm{s}$ follows *V3*, and no delay occurs from following *V5*. Still, the reduced degrees of freedom and feasible set delay the subsequent lane changes. The GT is reached for the first time with a 1.5 s delay and a second time 0.4 s later compared to the *4bp-13* in scenario 0.

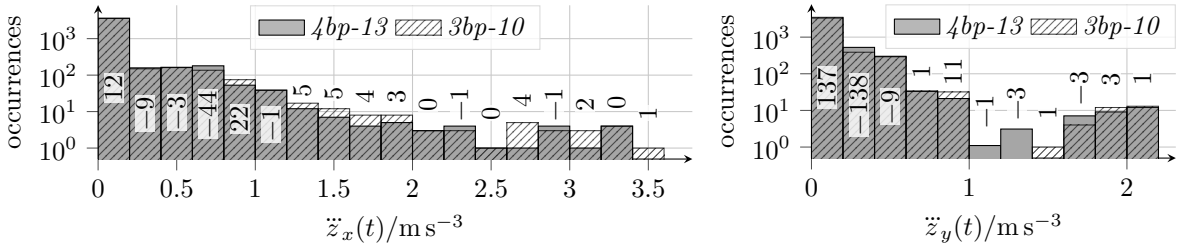
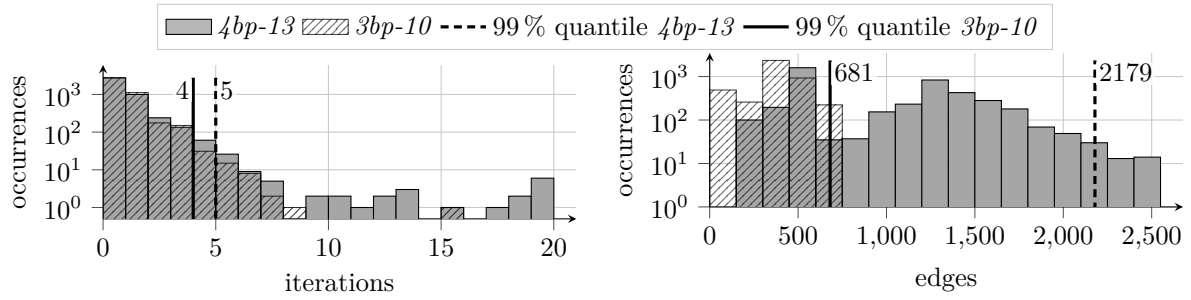


Figure 9.9.: Occurrences of jerks in the x- and y-directions. Each pair of bars is labeled with the difference between the occurrences from the *3bp-10* and the *4bp-13* configurations.

The *4bp-13* configuration performs better in terms of progress and comfort than *3bp-10*. However, the comfort advantage is less profound in comparison. The comfort cost increases for the *3bp-10* configuration results from the EV settling behind the leading vehicles *V3* and *V5* during the *initial lane changes* phase. The settling between the maneuvers due to the *3bp-10* configuration require a higher jerk compared to a quick sequence of *initial lane changes*, as performed by *4bp-13*. Also, following *V5* for an extended period increases the difference between the minimum and maximum velocity in two scenarios. While the jerk stays below $0.2\,\mathrm{m\,s^{-1}}$ in most cases also with *3bp-10*, figure 9.9 shows a tendency toward higher jerk values in the x-direction. The execution of the distinct phases requires a higher jerk in the y-direction in most cases with both configurations. However, the difference is too small to induce a clear difference in the occurrences shown in figure 9.9. Thus, the main comfort difference is due to the jerk increase in the x-direction. The acceleration from the EV perspective stays within $|a_x(t)| \leq 3.2\,\mathrm{m\,s^{-2}}$ and $|a_y(t)| \leq 1.52\,\mathrm{m\,s^{-2}}$ for all scenarios and configurations. The extreme accelerations for the *4bp-13* configuration are about $0.37\,\mathrm{m\,s^{-2}}$ and $0.17\,\mathrm{m\,s^{-2}}$ lower than for *3bp-10* in the x- and y-direction, respectively.
The *3bp-10* configuration tends to keep a higher time headway toward the leading vehicles than *4bp-13*. The effect prominently occurs when merging behind *V2*, resulting in the minimum time headway of 1.02 s. In contrast, the *4bp-13* configuration leads to a minimum time headway of 0.48 s. The time headway toward the rear vehicles does not drop below 0.69 s with *3bp-10* and 0.47 s with *4bp-13*. The reason for the increased distances is twofold. First, the EV arrives later on the left-most lane, so the distance to *V2* is larger. Second, the lower breakpoint number reduces the feasible set. Thus, lane changes are performed with a greater distance toward the other vehicles.
In three scenarios, the EV settles with *3bp-10* in the LT behind *V5*, while this is observed with *4bp-13* in only one scenario. In the vicinity of the LT, the current cost descent factor turns negative for a limited time due to the acceleration of *V5*. Still, the convergence into the terminal set is not prevented. Otherwise, no situations with a negative current cost descent factor are observed.
The complexity of the LP and the GP are assessed via the number of performed iterations and the number of edges in the graph after the termination of the Dijkstra algorithm.



(a) Occurrences of the IPOPT iterations with a maximum of 15 for *3bp-10* and 20 for *4bp-13*.

(b) Occurrences of the final graphs' edge count with a maximum of 738 for *3bp-10* and 2516 for *4bp-13*.

Figure 9.10.: Occurrences of complexity measures accumulated over all scenarios.

Their occurrences in all scenarios are shown in figure 9.10. The *4bp-13* configuration requires more iterations and adds more edges to the graph in most cases. The maximum iterations increase by a factor of 1.33, while the maximum number of edges is 3.41 times higher compared to *3bp-10*. The increased number of breakpoints also increases the runtime complexity per iteration and graph node. If an additional breakpoint is used, the NLP due to *4bp-13* includes more optimization variables and constraints, requiring more operations to find a new search direction. In addition, the number of possible successor edges increases, demanding more feasibility checks and cost evaluations. An iteration count near zero is frequently observed due to an almost optimal initial guess from the previous time step. Thus, the 99 % quantile is about four times smaller than the $i_{\mathrm{ter}} = 20$ maximum iterations. The highest iteration counts occur only in scenes where the GP solution is selected to perform a new maneuver. In contrast to the iterations, the number of edges is more evenly distributed, so the 99 % quantile is closer to the maximum. For the *4bp-13* configuration, two modes are visible at about 500 and 1300 edges. The first mode results from scenes yielding a trajectory terminating in the GT. Generally, more edges are required if the resulting trajectory has a non-zero terminal cost, which is the case for the occurrences around the second mode.

9.4. Discussion of Development Criteria

This work aims to develop a trajectory planning algorithm capable of considering the automated driving objectives. The objectives include progress toward a target, comfort, and obeying the traffic rules. Simultaneously, the algorithm shall stabilize the EV in the GT, which may not be reachable within the planning horizon. The required computations shall be low while providing a sufficient control performance.

Driving Objectives

Generally, the proposed planning algorithm considers most of the automated driving objectives. Progress and comfort are preferences and therefore encoded in the cost function, which impacts the transition between LTs. Although additional breakpoints can improve the overall closed-loop performance, the performance is mostly invariant to the number of

breakpoints in convergence to an LT. The result is explained by the running cost design in combination with the optimal trajectory parameterization, which is used in several related works in automated driving applications. The progress and the comfort measure improve with the planning algorithm's ability to select a suitable LT. The evaluation provides evidence that the planning algorithm makes reasonable maneuver decisions for the proposed terminal cost when using multiple polynomial segments. The terminal cost resembles an advantage compared to the related work. It allows the planning algorithm to optimally choose the LT without tuning parameters in the running cost. Instead, sufficient conditions are formulated, the terminal cost has to fulfill, to achieve convergence to the GT. Due to the LP's recursive feasibility and integration with the GP, time-consistent planning results are achieved. In addition, the optimization in the LP refines the GP's results despite a limited number of iterations, which is further analyzed in open-loop in section A.7. However, the improvements achieved in open-loop do not fully translate to the closed-loop performance, which is investigated further in section A.8. The consistent maneuver execution allows quick transitions between LTs, so the EV spends most of its time in the terminal set, where maximum longitudinal progress to a distant destination is achieved with maximum comfort. Still, the maneuver decisions are limited by the terminal cost's accuracy. The limitation becomes apparent especially with low degrees of freedom. The high terminal cost due to an impeding vehicle on the left lane prevents progress toward the GT despite the availability of feasible lane change trajectories. The evaluation also shows the planning algorithm's capability to consider the traffic rules. The maximum velocity and the left overtaking rule are not violated in closed-loop. Since the latter rule is implemented via the terminal cost, it does not reduce the feasible space. While the EV keeps a sufficient distance from other vehicles most of the time, the observed minimum time headway toward the leading and trailing vehicles is problematic for perceived and actual safety. The problem could be addressed by a velocity-dependent vehicle shape inflation at the cost of control performance.

Stability

The terminal set is designed with the automated driving objectives in mind, so they are fulfilled if the EV is stabilized in the terminal set. The constraints considered in the OCP are derived from a dynamic vehicle model, rendering the planned trajectory likely feasible for an actual vehicle. An advantage over related work is the feasible inner approximation of the semi-infinite nonlinear constraints with a finite number of polynomial functions to achieve dynamically feasible trajectories in the Frénet frame. Another advantage is the LP's recursive feasibility, which guarantees the convergence to an LT in the nominal case. Disturbances can impede the convergence, which occurs when the constant velocity prediction does not hold. Assuming the other vehicles thrive to reach a steady-state, the convergence might be impeded only for a limited time. In the scenarios evaluated in section 9.3, the terminal cost enables finite-time convergence toward the GT, although the underlying assumptions are not always fulfilled. Thus, the terminal cost is attested stabilizing properties under nominal conditions, but also a certain robustness. Even if no progress is achieved toward the GT, the vehicle's safety is not compromised as long as the EV tracks one of the LTs. Still, the restrictive terminal set design can be

problematic in dense traffic if low degrees of freedom prevent a sufficient exploration of LTs. Such configurations would benefit from a more sophisticated impeding vehicle cost and a terminal region about the target velocity and lane centers. Werling et al. [Wer+12] introduce an extended terminal set enclosing their LTs. The extended terminal set could be applied to the problem at hand if the terminal cost also estimates the cost for reaching the selected LT and the GT. The hierarchical combination of the GP and the LP improves the overall planning algorithm's robustness. Even if the LP does not find a feasible solution, the GP can provide a suboptimal one. Conversely, due to the LP's recursive feasibility, it is likely to retain a feasible trajectory toward its LT, even if the GP should fail. Still, additional tests in more adversarial scenarios are necessary to determine the limitations of the algorithm design.

Complexity

Although the planning algorithms are designed for computational efficiency, enhancements are necessary to enable a real-time application. Considering the time increment used in the closed-loop simulation, the LP and the GP combination must surpass a running time of 100 ms, which is also assumed by [Kuw+09]. An impression of the running time[1] is provided in section A.9. In 99 % of the cases, the running time of the hierarchical combination stays below 210 ms with the *4bp-13* configuration and below 72 ms with the *3bp-10* configuration. While the latter complies with the time limit often, there are still extreme cases where the limit is exceeded, requiring the reduction of the algorithms' worst case complexity. Instead of IPOPT, applying an SQP algorithm seems promising in combination with a proper iteration limit. Mercy, Parys, and Pipeleers [Mer+18b] report an eleven times reduced average running time of blockSQP [Jan15] compared to IPOPT, which are used to solve their spline-parameterized trajectory planning problem. The observation may apply to the problem at hand since an SQP can generally better leverage a warm start [NW06, pp. 416, 550]. The NLP can be simplified by removing the breakpoints from the optimization variables. Thus, computing gradients for the B-splines would be avoided, which is currently required in the majority of the equality constraints. Although a worse control performance can be expected when choosing a new maneuver and in cases of disturbances, the result may be sufficient most of the time, as indicated by section A.8. Applying the A* algorithm with a cost-to-go heuristic seems a straightforward improvement for the shortest-path search. However, finding an accurate and admissible heuristic of low computational complexity that also considers the terminal cost is not straightforward. Still, limiting the number of node expansions in the Anytime A* algorithm [Lik+03], which is designed to find a suboptimal solution quickly, can reduce the worst-case complexity. The hierarchical algorithm could cope with the suboptimal solution if the disturbances are sufficiently low and the LTs are still explored sufficiently.

[1] The running time measurements were carried out on Ubuntu 22.04 with AMD Ryzen 5 3600 CPU at 3.6 GHz and 16 GB RAM.

10
Conclusion and Outlook

The problem of decision-making and motion planning for automated driving is frequently understood as an OCP. It allows compromising objectives while considering the vehicle's state and input limitations. Sampling-based and numerical optimization-based solution algorithms applied in an MPC fashion provide sufficient control performance in many scenarios. The sampling algorithms are capable of approximating the global optimal solution to the OCP, considering the problem's combinatorial nature. However, a highly accurate solution generally requires a high computational effort. In contrast, most numerical optimization algorithms yield a local optimal solution, which is achieved efficiently and with high accuracy. The dichotomy of properties motivates a two-stage hierarchical combination of both algorithm classes to fulfill the behavior and motion planning modules' tasks. However, the OCP is usually neither developed systematically to achieve convergence toward a distant target nor is the planned motion guaranteed to be feasible. In this work, a two-stage planning algorithm is developed to fully leverage the advantages of both algorithm classes and implement driving objectives, stability criteria, and computational complexity criteria. The hierarchical algorithm is composed of two separate planning algorithms. The LP adopts numerical optimization, and the GP uses a sampling strategy. Both algorithms are based on the same OCP which is transformed via a spline trajectory parameterization to a SIP. The spline convex hull property enables an approximation of the SIP, so solution feasibility is retained. The distinct planning algorithms are evaluated to verify their intended properties. The performance of their hierarchical combination is validated in simulated highway scenarios. A summary of the concepts and results is given below.

Local Planner

The LP relies on the IPOPT numerical optimization algorithm to provide high control performance during convergence to a given LT defined by a lane center and a target velocity or leading vehicle. An NLP is formulated providing a feasible approximate solution to the SIP. A set of terminal constraints requires the trajectory to reach the LT. The differential flatness of a dynamic vehicle model is exploited, enabling the optimal spline parameterization of the planned trajectory in the Frénet frame. The splines' breakpoints and coefficients are subject to optimization. The LP compromises comfort and progress via a linear quadratic minimum-time cost function, where the time in the x- and y-directions

are optimized individually. The constraints resulting from limitations on the model input, road boundaries, and other vehicles are formulated as polynomial functions of the spline trajectory. Consequently, the constraints are spline functions, allowing the exploitation of the convex hull property to enforce semi-infinite constraints in a finite-dimensional optimization problem. A breakpoint adaption strategy is proposed to ensure recursive feasibility toward the LT in the absence of disturbances. The splines' nonuniformity prevents determining the involved spline coefficient transformations prior to optimization. The transformations are implemented efficiently by coupling additional constraint splines with equality constraints.
The convergence of the LP-controlled vehicle is analyzed in two highway scenarios. The scenarios show recursive feasibility and practical stability despite a mismatch between other vehicle predictions and their actual motion in the longitudinal direction. In addition, the influence of the breakpoint number on the complexity and control performance is evaluated. Adding breakpoints to the spline trajectory considerably increases the NLP complexity. On the other hand, additional breakpoints improve the open-loop control performance in situations close to occupied regions of the state space. However, improvements in open-loop performance may not equally carry over to the closed-loop performance.

Global Planner

While the LP is designed for high control performance toward an LT, it cannot decide on a suitable LT itself to make progress toward a possibly distant GT. The LTs, which also include the GT, form a disconnected terminal set. The terminal cost is introduced as the upper bound on the minimum cost-to-go to guide the selection of LTs. The EV progresses to the GT if the assumptions underlying the terminal cost design are met. The GP approximates the solution to the SIP by finding the optimal trajectory with the Dijkstra shortest-path search algorithm in a graph. The state space and nonuniform time discretizations implicitly define the graph. In line with the LP, the candidate trajectories are parameterized by spline functions, which are selected using the same constraints and running cost considered by the LP. The constraint and cost evaluation is performed efficiently segment-wise with precomputed coefficient transformations and B-spline evaluations.
Four GP configurations of different complexity are compared in several highway scenario variations. The observed open-loop control performance and complexity measures are used to select a suitable compromise for the application in a hierarchical combination. The evaluation shows that candidate trajectories with four breakpoints outperform those with three breakpoints in open-loop control performance since additional maneuver options become available. However, the performance saturates with increasing breakpoint combinations while the complexity increases further.

Hierarchical Planner

The LP and the GP are combined in a two-stage hierarchical planning algorithm to leverage the advantages of both algorithms. The GP shall approximate the global optimal solution to the SIP, making discrete decisions on the number and order of breakpoints, and the LT. Instead of passing the GP's solution to the LP directly, it is compared with the LP's solution from the previous time step. The LP receives the feasible trajectory of

minimum cost and the corresponding LT, which are utilized to warm start IPOPT. Thus, the recursive feasibility property is preserved. Simultaneously, a new LT is selected if it yields a lower cost. If disturbances render the previous time step's solution infeasible, the GP still provides a feasible backup trajectory. Subsequently, the LP solution is compared with the initial trajectory to account for the possibility of an infeasible result due to a failed convergence of the optimization algorithm.
The capabilities of the hierarchical planning algorithm are demonstrated in closed-loop simulation. In the simulated highway scenario variations, the EV has to overtake several other vehicles to reach the GT. The results are analyzed for compliance with the driving objectives, the ability to stabilize the vehicle in the GT, and the compromise of computational effort and control performance attained. The consideration of candidate trajectories with four breakpoints achieves convergence toward the GT with consistent planning results and comprehensible maneuver decisions. The planning algorithm implements a suitable compromise between progress and comfort while obeying the traffic rules most of the time. Exceptions occur while merging in front or behind other vehicles, where the time headway can drop below 0.9 s. A breakpoint configuration with only three breakpoints considerably reduces the computational effort. On the other hand, the restrictive terminal set and the terminal cost design diminish progress toward the GT. Still, the desired stability properties are retained regardless of the configuration and despite a tight limit on the maximum IPOPT iterations.

Outlook

The results show the applicability of the proposed trajectory planning algorithm to control an automated vehicle in highway scenarios. Still, several enhancements and experiments are necessary to ensure stability and sufficient control performance in various scenarios for an implementation on an automated vehicle. The current implementation cannot provide a sufficiently tight bound on the computational effort suitable for a real-time application. Reducing the number of optimized breakpoints in combination with an SQP seems promising for reducing the LP's computational complexity. Simultaneously, the worst-case complexity of the GP can be reduced with an upper limit on the expanded nodes in combination with an anytime shortest-path search. Although the expansion limit may avert a solution from being found, the planning algorithm could be applied safely in the hierarchical combination. Another issue apparent in the results is the low time headway distance while merging in front or behind other vehicles. Although an obstacle shape inflation is used already, it does not implement a sufficient safety margin in closed-loop. A more sophisticated design of the obstacle shapes can provide robustness guarantees by imposing worst-case assumptions on the other vehicles' future motion. The challenge is to find a representation that still enables a solution of sufficient control performance. Since the planning algorithms provide an inner approximation to the SIP solution, they yield a feasible but suboptimal trajectory. Breakpoint insertion, degree elevation [Loo+15a], and a MINVO basis [TH22b] are options to reduce the approximation error. Currently, the recursive feasibility guarantee relies on a constant velocity prediction of the other vehicles. The limitation can be addressed by enforcing the constraints along the terminal segment with a local reduction or the convex hull property with a subsequent hull refinement.

The assumptions underlying the terminal cost enable a reasonable maneuver choice if the trajectories' degrees of freedom are sufficiently high. However, the progress toward the GT is sensitive to the number of breakpoints, especially in the presence of impeding vehicles on the neighboring lanes. A more accurate cost estimation and a less restrictive terminal set would improve the control performance in these cases. The enhancements require validation in additional experiments. The experiments should consider different sources of uncertainty commonly encountered in automated highway driving. Common sources of uncertainty are vehicle perception, other traffic participants, and vehicle model inaccuracy.

Bibliography

[Ada+22] **V. K. Adajania, A. Sharma, A. Gupta, H. Masnavi, K. M. Krishna, and A. K. Singh:** "Multi-Modal Model Predictive Control Through Batch Non-Holonomic Trajectory Optimization: Application to Highway Driving". In: *IEEE Robotics and Automation Letters* 7.2 (Apr. 2022), pp. 4220–4227. DOI: 10.1109/LRA.2022.3148460.

[Aja+18] **Z. Ajanović, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn:** "Search-Based Optimal Motion Planning for Automated Driving". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2018, pp. 4523–4530. DOI: 10.1109/IROS.2018.8593813.

[AL17] **F. Altché and A. de La Fortelle:** "An LSTM network for highway trajectory prediction". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017, pp. 353–359. DOI: 10.1109/ITSC.2017.8317913.

[AM00] **G. Alefeld and G. Mayer:** "Interval analysis: theory and applications". In: *Journal of Computational and Applied Mathematics* 121.1 (Sept. 2000), pp. 421–464. DOI: 10.1016/S0377-0427(00)00342-3.

[Ame+00] **P. R. Amestoy, I. S. Duff, and J.-Y. L'Excellent:** "Multifrontal parallel distributed symmetric and unsymmetric solvers". In: *Computer Methods in Applied Mechanics and Engineering* 184.2 (Apr. 2000), pp. 501–520. DOI: 10.1016/S0045-7825(99)00242-X.

[And+18] **J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl:** "CasADi: a software framework for nonlinear optimization and optimal control". In: *Mathematical Programming Computation* 11.1 (July 2018), pp. 1–36. DOI: 10.1007/s12532-018-0139-4.

[Aze+22] **M. M. R. Azer, M. Keyser, C. R. Satish, A. Schmees, and S. Subramanian:** *Creating a Close-To-Reality Simulation Environment for Automated Driving*. Project Final Report. Dortmund, Germany: Technical University Dortmund, Apr. 2022.

[Ban+19] **M. Bansal, A. Krizhevsky, and A. Ogale:** "ChauffeurNet: Learning to Drive by Imitating the Best and Synthesizing the Worst". In: *Robotics: Science and Systems XV*. June 2019. DOI: 10.15607/RSS.2019.XV.031.

[Bas+16] **H. Bast et al.:** "Route Planning in Transportation Networks". In: *Algorithm Engineering: Selected Results and Surveys*. Lecture Notes in Computer Science. Nov. 2016, pp. 19–80. DOI: 10.1007/978-3-319-49487-6_2.

[Bas+17] **C. Basu, Q. Yang, D. Hungerman, M. Singhal, and A. D. Dragan:** "Do You Want Your Autonomous Car To Drive Like You?" In: *HRI '17: Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. New York, NY, USA, Mar. 2017, pp. 417–425. DOI: 10.1145/2909824.3020250.

[Bel18] **H. Bellem:** "Comfort in Automated Driving: Analysis of Driving Style Preference in Automated Driving". Dissertation. Chemnitz, Germany: Chemnitz University of Technology, 2018. URL: https://core.ac.uk/download/pdf/159353504.pdf.

[Bel21] **R. Bellan:** *Cruise launches driverless robotaxi service in San Francisco*. TechCrunch. Nov. 3, 2021. URL: https://techcrunch.com/2021/11/03/cruise-launches-driverless-robotaxi-service-for-employees-in-san-francisco/ (visited on 10/12/2025).

[Bel54] **R. Bellman:** "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515. DOI: 10.1090/S0002-9904-1954-09848-8.

[Ben+15] **P. Bender, Ö. S. Tas, J. Ziegler, and C. Stiller:** "The combinatorial aspect of motion planning: Maneuver variants in structured environments". In: *2015 IEEE Intelligent Vehicles Symposium (IV)*. June 2015, pp. 1386–1392. DOI: 10.1109/IVS.2015.7225909.

[Boo01] **C. de Boor:** *A Practical Guide to Splines*. 1st ed. Vol. 27. Applied Mathematical Sciences. New York, NY, USA: Springer, Nov. 2001. URL: https://link.springer.com/book/9780387953663.

[Boo72] **C. de Boor:** "On calculating with B-splines". In: *Journal of Approximation Theory* 6.1 (July 1972), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9.

[Brü+18] **T. Brüdigam, M. Olbrich, M. Leibold, and D. Wollherr:** "Combining Stochastic and Scenario Model Predictive Control to Handle Target Vehicle Uncertainty in an Autonomous Driving Highway Scenario". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2018, pp. 1317–1324. DOI: 10.1109/ITSC.2018.8569909.

[BW12] **C. Büskens and D. Wassel:** "The ESA NLP Solver WORHP". In: *Modeling and Optimization in Space Engineering*. Vol. 73. Springer Optimization and Its Applications. New York, NY, USA, 2012, pp. 85–110. DOI: 10.1007/978-1-4614-4469-5_4.

[Car+13] **A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli:** "Predictive control of an autonomous ground vehicle using an iterative linearization approach". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. Oct. 2013, pp. 2335–2340. DOI: 10.1109/ITSC.2013.6728576.

[Che+16] **J. Chen, T. Liu, and S. Shen:** "Online generation of collision-free trajectories for quadrotor flight in unknown cluttered environments". In: *2016 IEEE International Conference on Robotics and Automation (ICRA)*. May 2016, pp. 1476–1483. DOI: 10.1109/ICRA.2016.7487283.

[Cla+20] **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 (May 2020), pp. 1826–1848. DOI: 10.1109/TITS.2019.2913998.

[CS66] **H. B. Curry and I. J. Schoenberg:** "On Pólya frequency functions IV: The fundamental spline functions and their limits". In: *Journal d'Analyse Mathématique* 17.1 (Dec. 1966), pp. 71–107. DOI: 10.1007/BF02788653.

[DB24] **P. Dorpmüller and T. Bertram:** "Two-Stage Hierarchical Motion Planning with Basis-Splines in Highway Environments". In: *2024 IEEE 18th International Conference on Advanced Motion Control (AMC)*. Feb. 2024. DOI: 10.1109/AMC58169.2024.10505654.

[Deo+23] **S. Deolasee, Q. Lin, J. Li, and J. M. Dolan:** "Spatio-temporal Motion Planning for Autonomous Vehicles with Trapezoidal Prism Corridors and Bézier Curves". In: *2023 American Control Conference (ACC)*. May 2023, pp. 3207–3214. DOI: 10.23919/ACC55779.2023.10155930.

[Deu13a] **Deutscher Bundestag:** "Verordnung über die Erteilung einer Verwarnung, Regelsätze für Geldbußen und die Anordnung eines Fahrverbotes wegen Ordnungswidrigkeiten im Straßenverkehr (Bußgeldkatalog-Verordnung – BKatV)". In: *Bundesgesetzblatt Teil I* 14 (Mar. 2013), pp. 498–546. URL: http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl113s0498.pdf.

[Deu13b] **Deutscher Bundestag:** "Verordnung zur Neufassung der Straßenverkehrs-Ordnung (StVO)". In: *Bundesgesetzblatt Teil I* 12 (Mar. 2013), pp. 367–427. URL: http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl113s0367.pdf.

[Deu78] **Deutscher Bundestag:** "Verordnung über eine allgemeine Richtgeschwindigkeit auf Autobahnen und ähnlichen Straßen (Autobahn-Richtgeschwindigkeits-V)". In: *Bundesgesetzblatt Teil I* 64 (Nov. 1978), p. 1824. URL: http://www.bgbl.de/xaver/bgbl/start.xav?startbk=Bundesanzeiger_BGBl&jumpTo=bgbl178s1824.pdf.

[Die+06] **M. Diehl, H. G. Bock, H. Diedam, and P.-B. Wieber:** "Fast Direct Multiple Shooting Algorithms for Optimal Robot Control". In: *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*. Vol. 340. Lecture Notes in Control and Information Sciences. Berlin and Heidelberg, Germany, Jan. 1, 2006, pp. 65–93. DOI: 10.1007/978-3-540-36119-0_4.

[Die+23] **C. Diehl, T. S. Sievernich, M. Krüger, F. Hoffmann, and T. Bertram:** "Uncertainty-Aware Model-Based Offline Reinforcement Learning for Automated Driving". In: *IEEE Robotics and Automation Letters* 8.2 (Feb. 2023), pp. 1167–1174. DOI: 10.1109/LRA.2023.3236579.

[Dij59] **E. W. Dijkstra:** "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (Dec. 1959), pp. 269–271. DOI: 10.1007/BF01386390.

[Din+19a] **W. Ding, W. Gao, K. Wang, and S. Shen:** "An Efficient B-Spline-Based Kinodynamic Replanning Framework for Quadrotors". In: *IEEE Transactions on Robotics* 35.6 (Dec. 2019), pp. 1287–1306. DOI: 10.1109/TRO.2019.2926390.

[Din+19b] **W. Ding, L. Zhang, J. Chen, and S. Shen:** "Safe Trajectory Generation for Complex Urban Environments Using Spatio-Temporal Semantic Corridor". In: *IEEE Robotics and Automation Letters* 4.3 (July 2019), pp. 2997–3004. DOI: 10.1109/LRA.2019.2923954.

[Do+13] **Q. H. Do, S. Mita, H. T. N. Nejad, and L. Han:** "Dynamic and Safe Path Planning Based on Support Vector Machine among Multi Moving Obstacles for Autonomous Vehicles". In: *IEICE Transactions on Information and Systems* E96.D.2 (Feb. 2013), pp. 314–328. DOI: 10.1587/transinf.E96.D.314.

[Dol+08] **D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel:** "Practical Search Techniques in Path Planning for Autonomous Driving". In: *Proceedings of the First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR-08)*. June 2008. URL: https://ai.stanford.edu/~ddolgov/dolgov08gppSTAIR.html.

[Don+20] **Z. Dong, X. Xu, X. Zhang, X. Zhou, X. Li, and X. Liu:** "Real-time Motion Planning Based on MPC With Obstacle Constraint Convexification For Autonomous Ground Vehicles". In: *2020 3rd International Conference on Unmanned Systems (ICUS)*. Nov. 2020, pp. 1035–1041. DOI: 10.1109/ICUS50048.2020.9274881.

[Dor+21] **P. Dorpmüller, M. Keller, and T. Bertram:** "Optimization of B-Spline parameterized Trajectories for On-Road Vehicle Guidance". In: *AmE 2021 - Automotive meets Electronics; 12th GMM-Symposium*. Mar. 2021. URL: https://ieeexplore.ieee.org/document/9399729.

[Dor+22] **P. Dorpmüller, T. Schmitz, M. Keller, and T. Bertram:** "Interpretable Approximation of Optimal Trajectories for Lateral Vehicle Guidance". In: *Automatisiertes Fahren 2022*. Apr. 2022, pp. 25–39. DOI: 10.1007/978-3-658-44797-7_3.

[Dor+23] **P. Dorpmüller, T. Schmitz, N. Bejagam, and T. Bertram:** "Time-Optimal Trajectory Planning in Highway Scenarios Using Basis-Spline Pa-

rameterization". In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. Sept. 2023, pp. 596–601. DOI: 10.1109/ITSC57777.2023.10422490.

[Dor+24] **P. Dorpmüller, T. Schmitz, N. Bejagam, and T. Bertram:** "Application of Basis-Splines for Trajectory Planning in Highway Scenarios". In: *AmEC 2024 – Automotive meets Electronics & Control; 14. GMM Symposium*. Mar. 2024, pp. 18–23. URL: https://ieeexplore.ieee.org/abstract/document/10564564.

[Dos+17] **A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun:** "CARLA: An Open Urban Driving Simulator". In: *Proceedings of the 1st Annual Conference on Robot Learning*. Vol. 78. Proceedings of Machine Learning Research. Nov. 2017, pp. 1–16. URL: https://proceedings.mlr.press/v78/dosovitskiy17a.html.

[Due+17] **D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açikmeşe:** "Trajectory optimization with inter-sample obstacle avoidance via successive convexification". In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. Dec. 2017, pp. 1150–1156. DOI: 10.1109/CDC.2017.8263811.

[Elb+15] **M. Elbanhawi, M. Simic, and R. Jazar:** "In the Passenger Seat: Investigating Ride Comfort Measures in Autonomous Cars". In: *IEEE Intelligent Transportation Systems Magazine* 7.3 (July 2015), pp. 4–17. DOI: 10.1109/MITS.2015.2405571.

[Fan+18] **H. Fan et al.:** "Baidu Apollo EM Motion Planner". July 2018. arXiv: 1807.08048 [cs.RO].

[Fas+17] **D. Fassbender, B. C. Heinrich, T. Luettel, and H.-J. Wuensche:** "An optimization approach to trajectory generation for autonomous vehicle following". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Sept. 2017, pp. 3675–3680. DOI: 10.1109/IROS.2017.8206213.

[Fed24] **Federal Statistical Office of Germany:** *Accidents registered by the police by type of damage/location*. July 2024. URL: https://www.destatis.de/EN/Themes/Society-Environment/Traffic-Accidents/Tables/accidents-registered-police.html (visited on 09/03/2024).

[FK14] **D. J. Fagnant and K. M. Kockelman:** "The travel and environmental implications of shared autonomous vehicles, using agent-based model scenarios". In: *Transportation Research Part C: Emerging Technologies* 40 (Mar. 2014), pp. 1–13. DOI: 10.1016/j.trc.2013.12.001.

[Fli+92] **M. Fliess, J. Lévine, P. Martin, and P. Rouchon:** "On differentially flat nonlinear systems". In: *Nonlinear Control Systems Design 1992*. IFAC Symposia Series. Oxford, June 1992, pp. 159–163. DOI: 10.1016/S1474-6670(17)52275-2.

[Fli+95] **M. Fliess, J. Lévine, P. Martin, and P. Rouchon:** "Flatness and defect of non-linear systems: introductory theory and examples". In: *International Journal of Control* 61.6 (June 1995), pp. 1327–1361. DOI: 10.1080/00207179508921959.

[Föl94] **O. Föllinger:** *Optimale Regelung und Steuerung*. 3rd ed. Methoden der Regelungs- und Automatisierungstechnik. Berlin, Germany and Boston, MA, USA: Oldenbourg Wissenschaftsverlag, Dec. 1994. DOI: 10.1515/9783486787306.

[FX23] **V. Freire and X. Xu:** "Flatness-Based Quadcopter Trajectory Planning and Tracking With Continuous-Time Safety Guarantees". In: *IEEE Transactions on Control Systems Technology* 31.6 (Nov. 2023), pp. 2319–2334. DOI: 10.1109/TCST.2023.3250954.

[Gao+18] **F. Gao, W. Wu, Y. Lin, and S. Shen:** "Online Safe Trajectory Generation for Quadrotors Using Fast Marching Method and Bernstein Basis Polynomial". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 344–351. DOI: 10.1109/ICRA.2018.8462878.

[GK11] **C. J. Green and A. Kelly:** "Toward Optimal Sampling in the Space of Paths". In: *Robotics Research*. Vol. 66. Springer Tracts in Advanced Robotics (STAR) Vol. 66. Berlin and Heidelberg, Germany, 2011, pp. 281–292. DOI: 10.1007/978-3-642-14743-2_24.

[Gon+16] **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 (Apr. 2016), pp. 1135–1145. DOI: 10.1109/TITS.2015.2498841.

[Göt+15] **C. Götte, M. Keller, C. Hass, K.-H. Glander, A. Seewald, and T. Bertram:** "A model predictive combined planning and control approach for guidance of automated vehicles". In: *2015 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*. Nov. 2015, pp. 69–74. DOI: 10.1109/ICVES.2015.7396896.

[Göt+17] **C. Götte, M. Keller, T. Nattermann, C. Haß, K.-H. Glander, and T. Bertram:** "Spline-Based Motion Planning for Automated Driving". In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 9114–9119. DOI: 10.1016/j.ifacol.2017.08.1709.

[GP11] **L. Grüne and J. Pannek:** *Nonlinear Model Predictive Control. Theory and Algorithms*. 1st ed. Communications and Control Engineering (CCE). London, UK: Springer, Apr. 2011. DOI: 10.1007/978-0-85729-501-9.

[Har+68] **P. E. 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 (July 1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[Hat01] **A. Hatcher:** *Algebraic topology*. New York, NY, USA: Cambridge University Press, Dec. 2001. URL: https://pi.math.cornell.edu/~hatcher/AT/AT+.pdf.

[Hen+83] **H. V. Henderson, F. Pukelsheim, and S. R. Searle:** "On the history of the kronecker product". In: *Linear and Multilinear Algebra* 14.2 (1983), pp. 113–120. DOI: 10.1080/03081088308817548.

[Het86] **R. Hettich:** "An implementation of a discretization method for semi-infinite programming". In: *Mathematical Programming* 34.3 (Apr. 1986), pp. 354–361. DOI: 10.1007/BF01582235.

[HJ78] **R. Hettich and H. Th. Jongen:** "Semi-infinite programming: Conditions of optimality and applications". In: *Optimization Techniques*. Vol. 7. Lecture Notes in Control and Information Sciences. Berlin and Heidelberg, Germany, 1978, pp. 1–11. DOI: 10.1007/BFb0006502.

[HK93] **R. Hettich and K. O. Kortanek:** "Semi-Infinite Programming: Theory, Methods, and Applications". In: *SIAM Review* 35.3 (Sept. 1993), pp. 380–429. DOI: 10.1137/1035089.

[HL06] **D. Henrion and J.-B. Lasserre:** "LMIs for constrained polynomial interpolation with application in trajectory planning". In: *Systems & Control Letters* 55.6 (June 2006), pp. 473–477. DOI: 10.1016/j.sysconle.2005.09.011.

[Hti+20] **Z. Htike, G. Papaioannou, E. Velenis, and S. Longo:** "Motion Planning of Self-driving Vehicles for Motion Sickness Minimisation". In: *2020 European Control Conference (ECC)*. May 2020, pp. 1719–1724. DOI: 10.23919/ECC51009.2020.9143789.

[Hua+22] **Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen:** "A Survey on Trajectory-Prediction Methods for Autonomous Driving". In: *IEEE Transactions on Intelligent Vehicles* 7.3 (Sept. 2022), pp. 652–674. DOI: 10.1109/TIV.2022.3167103.

[Hun+08] **F. von Hundelshausen, M. Himmelsbach, F. Hecker, A. Mueller, and H.-J. Wuensche:** "Driving with tentacles: Integral structures for sensing and motion". In: *Journal of Field Robotics*. Special Issue on the 2007 DARPA Urban Challenge, Part II 25.9 (Aug. 2008), pp. 640–673. DOI: 10.1002/rob.20256.

[Int97] **International Organization for Standardization:** *Mechanical vibration and shock — Evaluation of human exposure to whole-body vibration — Part 1: General requirements*. ISO 2631-1:1997. May 1997. URL: https://www.iso.org/standard/7612.html.

[Ise22] **R. Isermann:** "Lateral Vehicle Behavior". In: *Automotive Control: Modeling and Control of Vehicles*. Berlin and Heidelberg, Germany, 2022, pp. 147–207. DOI: 10.1007/978-3-642-39440-9_7.

[Jan15] **D. Janka:** *blockSQP user's manual*. GitHub. Oct. 2015. URL: https://github.com/djanka2/blockSQP/blob/master/doc/manual/manual.pdf (visited on 01/27/2025).

[KD16] **F. Kunz and K. Dietmayer:** "Hybrid discrete-parametric optimization for trajectory planning in on-road driving scenarios". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 802–807. DOI: 10.1109/ITSC.2016.7795647.

[Kel+14] **M. Keller, F. Hoffmann, C. Hass, T. Bertram, and A. Seewald:** "Planning of Optimal Collision Avoidance Trajectories with Timed Elastic Bands". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 9822–9827. DOI: 10.3182/20140824-6-ZA-1003.01143.

[Kel+15] **M. Keller, C. Hass, A. Seewald, and T. Bertram:** "A Model Predictive Approach to Emergency Maneuvers in Critical Traffic Situations". In: *2015 IEEE 18th International Conference on Intelligent Transportation Systems*. Nov. 2015, pp. 369–374. DOI: 10.1109/ITSC.2015.69.

[KF10] **S. Karaman and E. Frazzoli:** "Incremental Sampling-based Algorithms for Optimal Motion Planning". In: *Proceedings of Robotics: Science and Systems VI*. 2010. DOI: 10.15607/RSS.2010.VI.034.

[KF17] **H. Kano and H. Fujioka:** "Velocity and acceleration constrained trajectory planning by smoothing splines". In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. June 2017, pp. 1167–1172. DOI: 10.1109/ISIE.2017.8001410.

[KF18] **H. Kano and H. Fujioka:** "Spline Trajectory Planning for Path with Piecewise Linear Boundaries". In: *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation, EUROSIM 2016, The 57th SIMS Conference on Simulation and Modelling SIMS 2016*. Dec. 2018, pp. 439–445. DOI: 10.3384/ecp17142439.

[Kra+18] **R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein:** "The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2018, pp. 2118–2125. DOI: 10.1109/ITSC.2018.8569552.

[Kuw+09] **Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How:** "Real-Time Motion Planning With Applications to Autonomous Urban Driving". In: *IEEE Transactions on Control Systems Technology* 17.5 (Sept. 2009), pp. 1105–1118. DOI: 10.1109/TCST.2008.2012116.

[Lam+15] **S. K. Lam, A. Pitrou, and S. Seibert:** "Numba: a LLVM-based Python JIT compiler". In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. LLVM '15. Nov. 2015. DOI: 10.1145/2833157.2833162.

[Lan18] **A. Lange:** "Gestaltung der Fahrdynamik beim Fahrstreifenwechselmanöver als Rückmeldung für den Fahrer beim automatisierten Fahren". Dissertation. Munich, Germany: Technical University Munich, 2018. URL: https://mediatum.ub.tum.de/1366901.

[LaV98] **S. LaValle:** *Rapidly-exploring random trees: a new tool for path planning.* Ames, IA, USA: Iowa State University, Oct. 1998. URL: https://lavalle.pl/papers/Lav98c.pdf.

[Le+19] **D. Le, Z. Liu, J. Jin, K. Zhang, and B. Zhang:** "Historical Improvement Optimal Motion Planning with Model Predictive Trajectory Optimization for On-road Autonomous Vehicle". In: *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society.* Oct. 2019, pp. 5223–5230. DOI: 10.1109/IECON.2019.8927189.

[Len+11] **S. Lengagne, N. Ramdani, and P. Fraisse:** "Planning and Fast Replanning Safe Motions for Humanoid Robots". In: *IEEE Transactions on Robotics* 27.6 (Dec. 2011), pp. 1095–1106. DOI: 10.1109/TRO.2011.2162998.

[Lew+10] **B. Lewis-Evans, D. De Waard, and K. A. Brookhuis:** "That's close enough—A threshold effect of time headway on the experience of risk, task difficulty, effort, and comfort". In: *Accident Analysis & Prevention* 42.6 (Nov. 2010), pp. 1926–1933. DOI: 10.1016/j.aap.2010.05.014.

[LF09] **M. Likhachev and D. Ferguson:** "Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles". In: *Proceedings of Robotics: Science and Systems IV.* 2009. DOI: 10.15607/RSS.2008.IV.028.

[Li+16] **X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu:** "Real-Time Trajectory Planning for Autonomous Urban Driving: Framework, Algorithms, and Verifications". In: *IEEE/ASME Transactions on Mechatronics* 21.2 (Apr. 2016), pp. 740–753. DOI: 10.1109/TMECH.2015.2493980.

[Li+20] **B. Li et al.:** "On-road Trajectory Planning with Spatio-temporal RRT* and Always-feasible Quadratic Program". In: *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE).* Aug. 2020, pp. 942–947. DOI: 10.1109/CASE48305.2020.9217044.

[Li+23] **T. Li, X. Zhang, S. Zhang, X. Zhang, and X. Chen:** "STUNS-Planner: a Spatiotemporal Motion Planner with Unbending and Consistency Awareness for Quadrotors in Unknown Environments". In: *Journal of Intelligent & Robotic Systems* 107.7 (Jan. 2023). DOI: 10.1007/s10846-022-01773-3.

[Lie22] **C. Lienke:** "Trajectory Planning for Automated Driving in Dynamic Environments: An Integrated Approach to Spline-based Motion Planning using Hierarchical Trajectory Optimization". Dissertation. Dortmund, Germany: Technical University Dortmund, 2022. DOI: 10.17877/DE290R-22910.

[Lie25] **M. Liedtke:** *Waymo's robotaxis to start carrying passengers in Atlanta, expanding Uber partnership*. BNN Bloomberg. June 24, 2025. URL: https://www.bnnbloomberg.ca/business/2025/06/24/waymos-robotaxis-to-start-carrying-passengers-in-atlanta-expanding-uber-partnership/ (visited on 10/12/2025).

[Lik+03] **M. Likhachev, G. Gordon, and S. Thrun:** "ARA*: anytime A* with provable bounds on sub-optimality". In: *Proceedings of the 16th International Conference on Neural Information Processing Systems*. NIPS'03. Dec. 2003, pp. 767–774. URL: https://proceedings.neurips.cc/paper_files/paper/2003/file/ee8fe9093fbbb687bef15a38facc44d2-Paper.pdf.

[Lim+18] **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 (Feb. 2018), pp. 613–626. DOI: 10.1109/TITS.2017.2756099.

[Lim+21] **W. Lim, S. Lee, M. Sunwoo, and K. Jo:** "Hybrid Trajectory Planning for Autonomous Driving in On-Road Dynamic Scenarios". In: *IEEE Transactions on Intelligent Transportation Systems* 22.1 (Jan. 2021), pp. 341–355. DOI: 10.1109/TITS.2019.2957797.

[Loo+15a] **W. van Loock, G. Pipeleers, and J. Swevers:** "B-spline parameterized optimal motion trajectories for robotic systems with guaranteed constraint satisfaction". In: *Mechanical Sciences* 6.2 (Sept. 2015), pp. 163–171. DOI: 10.5194/ms-6-163-2015.

[Loo+15b] **W. van Loock, G. Pipeleers, and J. Swevers:** "Optimal motion planning for differentially flat systems with guaranteed constraint satisfaction". In: *2015 American Control Conference (ACC)*. July 2015, pp. 4245–4250. DOI: 10.1109/ACC.2015.7171996.

[Lou+10] **C. Louembet, F. Cazaurang, and A. Zolghadri:** "Motion planning for flat systems using positive B-splines: An LMI approach". In: *Automatica* 46.8 (Aug. 2010), pp. 1305–1309. DOI: 10.1016/j.automatica.2010.05.001.

[Luo+22] **J. Luo, X. Xu, H. Pu, J. Ma, M. Yuan, and L. Ding:** "An Adaptive Trajectory Planning for Autonomous Driving Using Trapezoidal Decomposition". In: *2022 China Automation Congress (CAC)*. Nov. 2022, pp. 4943–4948. DOI: 10.1109/CAC57257.2022.10055875.

[Ma+15] **L. Ma, J. Xue, K. Kawabata, J. Zhu, C. Ma, and N. Zheng:** "Efficient Sampling-Based Motion Planning for On-Road Autonomous Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 16.4 (Aug. 2015), pp. 1961–1976. DOI: 10.1109/TITS.2015.2389215.

[Mer+17] **T. Mercy, W. van Loock, and G. Pipeleers:** "Real-time motion planning in the presence of moving obstacles". In: *2016 European Control Conference (ECC)*. Jan. 2017, pp. 1586–1591. DOI: 10.1109/ECC.2016.7810517.

[Mer+18a] **T. Mercy, E. Hostens, and G. Pipeleers:** "Online motion planning for autonomous vehicles in vast environments". In: *2018 IEEE 15th International Workshop on Advanced Motion Control (AMC)*. Mar. 2018, pp. 114–119. DOI: 10.1109/AMC.2019.8371072.

[Mer+18b] **T. Mercy, R. van Parys, and G. Pipeleers:** "Spline-Based Motion Planning for Autonomous Guided Vehicles in a Dynamic Environment". In: *IEEE Transactions on Control Systems Technology* 26.6 (Nov. 2018), pp. 2182–2189. DOI: 10.1109/TCST.2017.2739706.

[Mer23] **Mercedes-Benz:** *Automated driving revolution: Mercedes-Benz announces U.S. availability of DRIVE PILOT – the world's first certified SAE Level 3 system for the U.S. market*. MBUSA Newsroom. Sept. 27, 2023. URL: http://media.mbusa.com/releases/automated-driving-revolution-mercedes-benz-announces-us-availability-of-drive-pilot-the-worlds-first-certified-sae-level-3-system-for-the-us-market (visited on 09/02/2024).

[Mey15] **M. Meywerk:** *Vehicle dynamics*. Automotive series. Chichester, UK: Wiley, Apr. 2015. URL: https://www.wiley.com/en-us/Vehicle+Dynamics-p-9781118971369.

[MM93] **H. Michalska and D. Mayne:** "Robust receding horizon control of constrained nonlinear systems". In: *IEEE Transactions on Automatic Control* 38.11 (Nov. 1993), pp. 1623–1633. DOI: 10.1109/9.262032.

[Mog+21] **M. Moghadam, A. Alizadeh, E. Tekin, and G. H. Elkaim:** "A Deep Reinforcement Learning Approach for Long-term Short-term Planning on Frenet Frame". In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*. Aug. 2021, pp. 1751–1756. DOI: 10.1109/CASE49439.2021.9551598.

[Moo66] **R. E. Moore:** *Interval analysis*. Prentice-Hall series in automatic computation. Englewood Cliffs, NJ, USA: Prentice-Hall, 1966.

[Nat+21] **R. Natarajan, H. Choset, and M. Likhachev:** "Interleaving Graph Search and Trajectory Optimization for Aggressive Quadrotor Flight". In: *IEEE Robotics and Automation Letters* 6.3 (July 2021), pp. 5357–5364. DOI: 10.1109/LRA.2021.3067298.

[Nat+24] **R. Natarajan, S. Mukherjee, H. Choset, and M. Likhachev:** "PINSAT: Parallelized Interleaving of Graph Search and Trajectory Optimization for Kinodynamic Motion Planning". Jan. 2024. arXiv: 2401.08948 [cs.RO].

[Nau+20] **M. Naumann, L. Sun, W. Zhan, and M. Tomizuka:** "Analyzing the Suitability of Cost Functions for Explaining and Imitating Human Driving Behavior based on Inverse Reinforcement Learning". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. May 2020, pp. 5481–5487. DOI: 10.1109/ICRA40945.2020.9196795.

[Nil+15] **J. Nilsson, P. Falcone, M. Ali, and J. Sjöberg:** "Receding horizon maneuver generation for automated highway driving". In: *Control Engineering Practice* 41 (Aug. 2015), pp. 124–133. DOI: 10.1016/j.conengprac.2015.04.006.

[Nil+16] **J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson:** "If, When, and How to Perform Lane Change Maneuvers on Highways". In: *IEEE Intelligent Transportation Systems Magazine* 8.4 (Nov. 2016), pp. 68–78. DOI: 10.1109/MITS.2016.2565718.

[Nil+17] **J. Nilsson, M. Brannstrom, E. Coelingh, and J. Fredriksson:** "Lane Change Maneuvers for Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 18.5 (May 2017), pp. 1087–1096. DOI: 10.1109/TITS.2016.2597966.

[NW06] **J. Nocedal and S. J. Wright:** *Numerical Optimization*. 2nd ed. Springer Series in Operations Research and Financial Engineering. New York, NY, USA: Springer, Dec. 2006. DOI: 10.1007/978-0-387-40065-5.

[Pad+16] **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 (Mar. 2016), pp. 33–55. DOI: 10.1109/TIV.2016.2578706.

[Par+15] **J. Park, S. Karumanchi, and K. Iagnemma:** "Homotopy-Based Divide-and-Conquer Strategy for Optimal Trajectory Planning via Mixed-Integer Programming". In: *IEEE Transactions on Robotics* 31.5 (Oct. 2015), pp. 1101–1115. DOI: 10.1109/TRO.2015.2459373.

[Pis21] **B. Pishue:** *2021 INRIX Global Traffic Scorecard*. INRIX, Dec. 2021. URL: https://inrix.com/press-releases/2021-traffic-scorecard-de/ (visited on 01/27/2025).

[Ple17] **M. G. Plessen:** "Trajectory planning of automated vehicles in tube-like road segments". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. Oct. 2017. DOI: 10.1109/ITSC.2017.8317585.

[Pol+17] **P. Polack, F. Altche, B. d'Andrea-Novel, and A. De La Fortelle:** "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 812–818. DOI: 10.1109/IVS.2017.7995816.

[Pon+62] **L. Pontryagin, V. Boltyanski, R. Gamkrelidze, and E. Miscenko:** *The Mathematical Theory of Optimal Processes*. Chichester, UK: Wiley, 1962.

[PP17] **R. van Parys and G. Pipeleers:** "Spline-Based Motion Planning in an Obstructed 3D Environment". In: *IFAC-PapersOnLine* 50.1 (July 2017), pp. 8668–8673. DOI: 10.1016/j.ifacol.2017.08.1525.

[Qia+16a] **X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle:** "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 205–210. DOI: 10.1109/ITSC.2016.7795555.

[Qia+16b] **X. Qian, I. Navarro, A. de La Fortelle, and F. Moutarde:** "Motion planning for urban autonomous driving using Bézier curves and MPC". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. Nov. 2016, pp. 826–833. DOI: 10.1109/ITSC.2016.7795651.

[Raj06] **R. Rajamani:** *Vehicle dynamics and control*. Mechanical Engineering Series. New York, NY, USA: Springer, 2006. DOI: 10.1007/0-387-28823-6.

[Rat+15] **C. Rathgeber, F. Winkler, X. Kang, and S. Müller:** "Optimal Trajectories for Highly Automated Driving". In: *International Journal of Mechanical and Mechatronics Engineering* 9.6 (2015), pp. 969–975. DOI: 10.5281/zenodo.1106815.

[Rat16] **C. Rathgeber:** "Trajektorienplanung und -folgeregelung für assistiertes bis hochautomatisiertes Fahren". Dissertation. Berlin, Germany: Technical University Berlin, 2016. DOI: 10.14279/DEPOSITONCE-5506.

[Raw+22] **J. B. Rawlings, D. Q. Mayne, and M. Diehl:** *Model Predictive Control: Theory, Computation, and Design*. 2nd ed. Santa Barbara, CA, USA: Nob Hill Publishing, 2022. URL: https://sites.engineering.ucsb.edu/~jbraw/mpc/MPC-book-2nd-edition-4th-printing.pdf.

[RD09] **G. van Rossum and F. L. Drake:** *Python 3 Reference Manual*. Scotts Valley, CA, USA: CreateSpace, 2009. URL: https://dl.acm.org/doi/book/10.5555/1593511.

[Ree91] **R. Reemtsen:** "Discretization methods for the solution of semi-infinite programming problems". In: *Journal of Optimization Theory and Applications* 71.1 (Oct. 1991), pp. 85–103. DOI: 10.1007/BF00940041.

[RG98] **R. Reemtsen and S. Görner:** "Numerical Methods for Semi-Infinite Programming: A Survey". In: *Semi-Infinite Programming*. Vol. 25. Nonconvex Optimization and Its Applications (NOIA). Boston, MA, USA, 1998, pp. 195–275. DOI: 10.1007/978-1-4757-2868-2_7.

[Rou+19] **G. Rousseau, C. Stoica Maniu, S. Tebbani, M. Babel, and N. Martin:** "Minimum-time B-spline trajectories with corridor constraints. Appli-

cation to cinematographic quadrotor flight plans". In: *Control Engineering Practice* 89 (Aug. 2019), pp. 190–203. DOI: 10.1016/j.conengprac.2019.05.022.

[Sad+16] **D. Sadigh, S. Sastry, S. A. Seshia, and A. D. Dragan:** "Planning for Autonomous Cars that Leverage Effects on Human Actions". In: *Proceedings of Robotics: Science and Systems XII*. June 2016. DOI: 10.15607/rss.2016.xii.029.

[SAE21] **SAE International:** *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. SAE J3016_202104. Apr. 2021. DOI: 10.4271/J3016_202104.

[Sch+13] **J. Schulman, J. Ho, A. Lee, I. Awwal, H. Bradlow, and P. Abbeel:** "Finding Locally Optimal, Collision-Free Trajectories with Sequential Convex Optimization". In: *Proceedings of Robotics: Robotics: Science and Systems IX*. June 2013. DOI: 10.15607/RSS.2013.IX.031.

[Sch+16] **J. Schlechtriemen, K. P. Wabersich, and K.-D. Kuhnert:** "Wiggling through complex traffic: Planning trajectories constrained by predictions". In: *2016 IEEE Intelligent Vehicles Symposium (IV)*. June 2016, pp. 1293–1300. DOI: 10.1109/IVS.2016.7535557.

[Sch+18] **W. Schwarting, J. Alonso-Mora, L. Paull, S. Karaman, and D. Rus:** "Safe Nonlinear Trajectory Generation for Parallel Autonomy With a Dynamic Vehicle Model". In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (Sept. 2018), pp. 2994–3008. DOI: 10.1109/TITS.2017.2771351.

[Sch+20] **T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl:** "An NMPC Approach using Convex Inner Approximations for Online Motion Planning with Guaranteed Collision Avoidance". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. May 2020, pp. 3574–3580. DOI: 10.1109/ICRA40945.2020.9197206.

[Sch+23] **L. Schäfer, S. Manzinger, and M. Althoff:** "Computation of Solution Spaces for Optimization-based Trajectory Planning". In: *IEEE Transactions on Intelligent Vehicles* (Jan. 2023), pp. 216–231. DOI: 10.1109/TIV.2021.3077702.

[Sco+99] **P. Scokaert, D. Q. Mayne, and J. B. Rawlings:** "Suboptimal model predictive control (feasibility implies stability)". In: *IEEE Transactions on Automatic Control* 44.3 (1999), pp. 648–654. DOI: 10.1109/9.751369.

[Set99] **J. A. Sethian:** *Level set methods and fast marching methods*. 2nd ed. Cambridge, MA, USA: Cambridge University Press, 1999. URL: https://math.berkeley.edu/~sethian/2006/Publications/Book/2006/OnLine/book_sethian.html.

[She24] **D. Shepardson:** *US agency seeks answers from Waymo in self-driving vehicle probe*. Reuters. June 12, 2024. URL: https://www.reuters.com/technology/us-agency-seeks-answers-waymo-self-driving-vehicle-probe-2024-06-12/ (visited on 10/12/2025).

[Sho06] **D. C. Shoup:** "Cruising for parking". In: *Transport Policy*. Parking 13.6 (Nov. 2006), pp. 479–486. DOI: 10.1016/j.tranpol.2006.05.005.

[SK22] **A. Steinke and U. Konigorski:** "Trajectory Planning considering Motion Sickness and Head Movements". In: *IFAC-PapersOnLine* 55.14 (2022), pp. 113–119. DOI: 10.1016/j.ifacol.2022.07.592.

[Sni06] **M. Sniedovich:** "Dijkstra's algorithm revisited: the dynamic programming connexion". In: *Control and Cybernetics* 35.3 (Jan. 2006), pp. 599–620. URL: http://matwbn.icm.edu.pl/ksiazki/cc/cc35/cc3536.pdf.

[Sto+15] **F. Stoican, I. Prodan, and D. Popescu:** "Flat trajectory generation for way-points relaxations and obstacle avoidance". In: *2015 23rd Mediterranean Conference on Control and Automation (MED)*. June 2015, pp. 695–700. DOI: 10.1109/MED.2015.7158827.

[Sto+16] **F. Stoican, V.-M. Ivănuşcă, I. Prodan, and D. Popescu:** "Obstacle avoidance via B-spline parametrizations of flat trajectories". In: *2016 24th Mediterranean Conference on Control and Automation (MED)*. June 2016, pp. 1002–1007. DOI: 10.1109/MED.2016.7536053.

[Sto+17] **F. Stoican, I. Prodan, D. Popescu, and L. Ichim:** "Constrained trajectory generation for UAV systems using a B-spline parametrization". In: *2017 25th Mediterranean Conference on Control and Automation (MED)*. July 2017, pp. 613–618. DOI: 10.1109/MED.2017.7984185.

[Sun+21] **Y. Sun, C. Zhang, and C. Liu:** "Collision-free and dynamically feasible trajectory planning for omnidirectional mobile robots using a novel B-spline based rapidly exploring random tree". In: *International Journal of Advanced Robotic Systems* 18.3 (June 2021). DOI: 10.1177/17298814211016609.

[Sur+10] **F. Suryawan, J. De Dona, and M. Seron:** "On splines and polynomial tools for constrained motion planning". In: *18th Mediterranean Conference on Control and Automation, MED'10*. June 2010, pp. 939–944. DOI: 10.1109/MED.2010.5547747.

[Tak+89] **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*. Sept. 1989, pp. 392–397. DOI: 10.1109/IROS.1989.637936.

[Tan+19] **L. Tang, H. Wang, P. Li, and Y. Wang:** "Real-time Trajectory Generation for Quadrotors using B-spline based Non-uniform Kinodynamic Search".

In: *IEEE International Conference on Robotics and Biomimetics*. Dec. 2019, pp. 1133–1138. DOI: 10.1109/ROBIO49542.2019.8961485.

[Tan+21] **L. Tang, H. Wang, Z. Liu, and Y. Wang:** "A real-time quadrotor trajectory planning framework based on B-spline and nonuniform kinodynamic search". In: *Journal of Field Robotics* 38.3 (May 2021), pp. 452–475. DOI: 10.1002/rob.21997.

[TH22a] **J. Tordesillas and J. P. How:** "MADER: Trajectory Planner in Multiagent and Dynamic Environments". In: *IEEE Transactions on Robotics* 38.1 (Feb. 2022), pp. 463–476. DOI: 10.1109/TRO.2021.3080235.

[TH22b] **J. Tordesillas and J. P. How:** "MINVO Basis: Finding Simplexes with Minimum Volume Enclosing Polynomial Curves". In: *Computer-Aided Design* 151.103341 (Oct. 2022). DOI: 10.1016/j.cad.2022.103341.

[Tor+19] **J. Tordesillas, B. T. Lopez, and J. P. How:** "FASTER: Fast and Safe Trajectory Planner for Flights in Unknown Environments". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2019, pp. 1934–1940. DOI: 10.1109/IROS40897.2019.8968021.

[Tre+00] **M. Treiber, A. Hennecke, and D. Helbing:** "Congested Traffic States in Empirical Observations and Microscopic Simulations". In: *Physical Review E* 62.2 (Feb. 2000), pp. 1805–1824. DOI: 10.1103/PhysRevE.62.1805.

[Ulb+17] **F. Ulbrich, D. Goehring, T. Langner, Z. Boroujeni, and R. Rojas:** "Stable timed elastic bands with loose ends". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 186–192. DOI: 10.1109/IVS.2017.7995718.

[Val+17] **C. Vallon, Z. Ercan, A. Carvalho, and F. Borrelli:** "A machine learning approach for personalized autonomous lane change initiation and control". In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. June 2017, pp. 1590–1595. DOI: 10.1109/IVS.2017.7995936.

[Vaz+01] **A. Vaz, E. Fernandes, and M. Gomes:** "Discretization methods for semi-infinite programming". In: *Investigação Operacional 21*. 2001, pp. 37–46. DOI: 10.1007/BF00940041.

[Vaz+04] **A. I. F. Vaz, E. M. G. P. Fernandes, and M. P. S. F. Gomes:** "Robot trajectory planning with semi-infinite programming". In: *European Journal of Operational Research*. EURO Young Scientists 153.3 (Mar. 2004), pp. 607–617. DOI: 10.1016/S0377-2217(03)00266-2.

[VL91] **E. Verriest and F. Lewis:** "On the linear quadratic minimum-time problem". In: *IEEE Transactions on Automatic Control* 36.7 (July 1991), pp. 859–863. DOI: 10.1109/9.85066.

[Wad+10] **T. Wada, S. Fujisawa, K. Imaizumi, N. Kamiji, and S. Doi:** "Effect of Driver's Head Tilt Strategy on Motion Sickness Incidence". In: *IFAC Proceedings Volumes*. 11th IFAC/IFIP/IFORS/IEA Symposium on Analysis,

Design, and Evaluation of Human-Machine Systems 43.13 (2010), pp. 192–197. DOI: 10.3182/20100831-4-FR-2021.00035.

[WB05] **A. Wächter and L. T. Biegler:** "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (Apr. 2005), pp. 25–57. DOI: 10.1007/s10107-004-0559-y.

[Wer+12] **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 (Mar. 2012), pp. 346–359. DOI: 10.1177/0278364911423042.

[WG23] **L. Wang and Y. Guo:** "Speed Adaptive Robot Trajectory Generation Based on Derivative Property of B-Spline Curve". In: *IEEE Robotics and Automation Letters* 8.4 (Apr. 2023), pp. 1905–1911. DOI: 10.1109/LRA.2023.3241812.

[Win+23] **K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau:** "Standards for passenger comfort in automated vehicles: Acceleration and jerk". In: *Applied Ergonomics* 106.103881 (Jan. 2023). DOI: 10.1016/j.apergo.2022.103881.

[WL12] **M. Werling and D. Liccardo:** "Automatic collision avoidance using model-predictive online optimization". In: *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. Dec. 2012, pp. 6309–6314. DOI: 10.1109/CDC.2012.6426612.

[Xie+22] **S. Xie, J. Hu, P. Bhowmick, Z. Ding, and F. Arvin:** "Distributed Motion Planning for Safe Autonomous Vehicle Overtaking via Artificial Potential Field". In: *IEEE Transactions on Intelligent Transportation Systems* 23.11 (Nov. 2022), pp. 21531–21547. DOI: 10.1109/TITS.2022.3189741. (Visited on 10/18/2024).

[YC15] **Y. Yan and G. S. Chirikjian:** "Closed-form characterization of the Minkowski sum and difference of two ellipsoids". In: *Geometriae Dedicata* 177.1 (Aug. 2015), pp. 103–128. DOI: 10.1007/s10711-014-9981-3.

[Yi+16] **B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller:** "Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers". In: *2016 IEEE Intelligent Vehicles Symposium (IV): 19-22 June 2016*. June 2016, pp. 584–589. DOI: 10.1109/IVS.2016.7535446.

[Yi+19] **B. Yi, P. Bender, F. Bonarens, and C. Stiller:** "Model Predictive Trajectory Planning for Automated Driving". In: *IEEE Transactions on Intelligent Vehicles* 4.1 (Mar. 2019), pp. 24–38. DOI: 10.1109/TIV.2018.2886683.

[Zha+24] **D. Zhang, C. Liang, X. Gao, K. Wu, and Z. Pan:** "Provably Feasible Semi-Infinite Program Under Collision Constraints via Subdivision". In: *IEEE*

Transactions on Robotics 40 (2024), pp. 2602–2619. DOI: 10.1109/TRO.2024.3391649.

[Zho+19] **B. Zhou, F. Gao, L. Wang, C. Liu, and S. Shen:** "Robust and Efficient Quadrotor Trajectory Generation for Fast Autonomous Flight". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 3529–3536. DOI: 10.1109/LRA.2019.2927938.

[Zie+08] **J. Ziegler, M. Werling, and J. Schroder:** "Navigating Car-like Robots in Unstructured Environments Using an Obstacle Sensitive Cost Function". In: *2008 IEEE Intelligent Vehicles Symposium*. June 2008, pp. 787–791. DOI: 10.1109/IVS.2008.4621302.

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

[Zie+14b] **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.

[ZS09] **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*. Oct. 2009, pp. 1879–1884. DOI: 10.1109/IROS.2009.5354448.

[ZS10] **J. Ziegler and C. Stiller:** "Fast collision checking for intelligent vehicle motion planning". In: *2010 IEEE Intelligent Vehicles Symposium*. June 2010, pp. 518–522. DOI: 10.1109/IVS.2010.5547976.

Peer-Reviewed Publications

P. Dorpmüller and T. Bertram: "Two-Stage Hierarchical Motion Planning with Basis-Splines in Highway Environments". In: *2024 IEEE 18th International Conference on Advanced Motion Control (AMC)*. Feb. 2024. DOI: 10.1109/AMC58169.2024.10505654.

P. Dorpmüller, T. Schmitz, N. Bejagam, and T. Bertram: "Application of Basis-Splines for Trajectory Planning in Highway Scenarios". In: *AmEC 2024 – Automotive meets Electronics & Control; 14. GMM Symposium*. Mar. 2024, pp. 18–23. URL: https://ieeexplore.ieee.org/abstract/document/10564564.

P. Dorpmüller, T. Schmitz, N. Bejagam, and T. Bertram: "Time-Optimal Trajectory Planning in Highway Scenarios Using Basis-Spline Parameterization". In: *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. Sept. 2023, pp. 596–601. DOI: 10.1109/ITSC57777.2023.10422490.

P. Dorpmüller, T. Schmitz, M. Keller, and T. Bertram: "Interpretable Approximation of Optimal Trajectories for Lateral Vehicle Guidance". In: *Automatisiertes Fahren 2022*. Apr. 2022, pp. 25–39. DOI: 10.1007/978-3-658-44797-7_3.

P. Dorpmüller, M. Keller, and T. Bertram: "Optimization of B-Spline parameterized Trajectories for On-Road Vehicle Guidance". In: *AmE 2021 - Automotive meets Electronics; 12th GMM-Symposium*. Mar. 2021. URL: https://ieeexplore.ieee.org/document/9399729.

Supervised Theses

F. Funk: "Erweiterung eines Trajektorienplaners zur Vergrößerung des verfügbaren Lösungsraumes in Autobahnszenarien". Master's Thesis. Dortmund, Germany: Technical University Dortmund, Jan. 2024.

A. Chahine: "Vergleich von Algorithmen zur Trajektorienplanung mittels quadratischer Programmierung". Bachelor's Thesis. Dortmund, Germany: Technical University Dortmund, Nov. 2022.

D. A. Desai: "Comparative Analysis of Methods for Collision Avoidance". Master's Thesis. Dortmund, Germany: Technical University Dortmund, May 2021.

H. Mei: "Fußgängerkennung in der Simulationsumgebung Carla". Master's Thesis. Dortmund: Technical University Dortmund, Aug. 2021.

A. Fotti: "Regelung und Modellierung der Fahrdynamik zur Kollisionsvermeidung". Master's Thesis. Dortmund, Germany: Technical University Dortmund, July 2020.

G. Li: "B-spline parametrized Trajectory Planning". Master's Thesis. Dortmund, Germany: Technical University Dortmund, Nov. 2020.

Supervised Student Project Groups

A. Kanduri et al.: *Automated Driving*. Project Group Final Report. Dortmund, Germany: Technical University Dortmund, Mar. 2021.

A. S. Raghava et al.: *Machine Learning for Automated Driving*. Project Group Final Report. Dortmund: Technical University Dortmund, Dec. 2021.

A

Appendix

A.1. Frénet Frame Velocities

This section provides additional steps to calculate the Frénet frame velocities introduced in section 3.4.2. The time argument is omitted for brevity. The first time derivative of the Frénet frame y-position ${}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_y = \frac{\mathrm{d}}{\mathrm{d}t}\left[{}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^{\intercal}\left(\gamma^*\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left(\gamma^*\right)\right)\right]$ is expanded and simplified:

$$\begin{aligned}
{}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_y &= {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{\mathbf{n}}}^{\intercal}(\gamma^*)\Big({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma^*)\Big) + {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^{\intercal}(\gamma^*)\Big({}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}} - {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{p}}(\gamma^*)\Big) \\
&= -\kappa_{\mathrm{p}}(\gamma^*)\, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x \underbrace{{}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}(\gamma^*)\Big({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma^*)\Big)}_{=0} + {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^{\intercal}(\gamma^*)\, {}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}} - \underbrace{{}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}}^{\intercal}\, {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}(\gamma^*)}_{=0}\, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x \\
&= v_{\mathrm{v}} \sin {}^{\mathfrak{p}}_{\mathrm{v}}\psi.
\end{aligned} \tag{A.1.1}$$

Equation (A.1.2) is a consequence of the vehicle projection ${}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left(\gamma^*\right)$ always being the position of the shortest distance from ${}^{\mathfrak{v}}_{\mathrm{v}}\underline{p}$. Thus, the x-position's first time derivative becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\left[{}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}\left(\gamma^*\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left(\gamma^*\right)\right)\right] = 0, \tag{A.1.2}$$

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}t}\left[{}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}\left(\gamma^*\right)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}\left(\gamma^*\right)\right)\right] = {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{\mathbf{t}}}^{\intercal}(\gamma^*)\left({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma^*)\right) + {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}(\gamma^*)\left({}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}} - {}^{\mathfrak{v}}_{\mathrm{p}}\dot{\underline{p}}(\gamma^*)\right) \\
=&\kappa_{\mathrm{p}}(\gamma^*)\, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x\, {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{n}}^{\intercal}(\gamma^*)\Big({}^{\mathfrak{v}}_{\mathrm{v}}\underline{p} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{p}(\gamma^*)\Big) + {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}(\gamma^*)\, {}^{\mathfrak{v}}_{\mathrm{v}}\dot{\underline{p}} - {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}^{\intercal}(\gamma^*)\, {}^{\mathfrak{v}}_{\mathrm{p}}\underline{\mathbf{t}}(\gamma^*)\, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x \\
=&\kappa_{\mathrm{p}}(\gamma^*)\, {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x\, {}^{\mathfrak{p}}_{\mathrm{v}}p_y + v_{\mathrm{v}} \cos {}^{\mathfrak{p}}_{\mathrm{v}}\psi - {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x = 0,
\end{aligned} \tag{A.1.3}$$

$$\Leftrightarrow\ {}^{\mathfrak{p}}_{\mathrm{v}}\dot{p}_x = \frac{v_{\mathrm{v}} \cos {}^{\mathfrak{p}}_{\mathrm{v}}\psi}{1 - \kappa_{\mathrm{p}}\, {}^{\mathfrak{p}}_{\mathrm{v}}p_y}. \tag{A.1.4}$$

A.2. Intelligent Driver Model

The IDM is a microscopic traffic and car-following model [Tre+00]. It determines the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles' acceleration $a_{\mathrm{idm},m} : \mathbb{R} \mapsto \mathbb{R}$ in dependence on their current velocity and the gap $\Delta z_{x,m} : \mathbb{R} \mapsto \mathbb{R}^+$ to the respective leading vehicle on the

same lane:

$$a_{\mathrm{idm},m}(t) = \tilde{a}\left[1 - \left(\frac{\dot{z}_{x,m}(t)}{\tilde{v}_m}\right)^e - \left(\frac{\Delta\tilde{z}_{x,m}\Big(\dot{z}_{x,m}(t), \Delta\dot{z}_{x,m}(t)\Big)}{\Delta z_{x,m}(t)}\right)\right], \quad \text{(A.2.1)}$$

$$\Delta\tilde{z}_{x,m}\Big(\dot{z}_{x,m}(t), \Delta\dot{z}_{x,m}(t)\Big) = \Delta\tilde{z}_x + \tilde{t}_{\mathrm{hw}}\dot{z}_{x,m}(t) + \frac{\dot{z}_{x,m}(t)\Delta\dot{z}_{x,m}(t)}{2\sqrt{\tilde{a}\tilde{a}_{\mathrm{d}}}}. \quad \text{(A.2.2)}$$

$a_{\mathrm{idm},m}(t)$ results from scaling the desired acceleration $\tilde{a} \in \mathbb{R}^+$ depending on the ratios of the respective vehicle's state to the target velocity $\tilde{v}_m \in \mathbb{R}^+$ and the desired gap $\Delta\tilde{z}_{x,m} : \mathbb{R}^+ \times \mathbb{R}^+ \mapsto \mathbb{R}^+$. The acceleration exponent $e \in [1, \infty)$ determines the acceleration's decrease when approaching $\tilde{v}_m$ for a leading vehicle far ahead. If $e \to \infty$, the acceleration is linear. If $e = 1$, the $\tilde{v}_m$ is approached asymptotically. The desired gap $\Delta\tilde{z}_{x,m} : \mathbb{R}^+ \times \mathbb{R}^+ \mapsto \mathbb{R}^+$ includes the jam distance $\Delta\tilde{z}_x \in \mathbb{R}^+$, the desired time headway $\tilde{t}_{\mathrm{hw}} \in \mathbb{R}^+$, and the desired deceleration $\tilde{a}_{\mathrm{d}} \in \mathbb{R}^+$ as additional parameters. For this work, the model is extended by

$$\ddot{z}_{x,m}(t) = \begin{cases} \check{a}_{\mathrm{idm},m} & \text{if } a_{\mathrm{idm},m}(t) < \check{a}_{\mathrm{idm},m}, \\ \hat{a}_{\mathrm{idm},m} & \text{if } a_{\mathrm{idm},m}(t) > \hat{a}_{\mathrm{idm},m}, \\ a_{\mathrm{idm},m}(t) & \text{otherwise}, \end{cases} \quad \text{(A.2.3)}$$

to keep the acceleration within the bounds $\check{a}_{\mathrm{idm},m} \in \mathbb{R}$ and $\hat{a}_{\mathrm{idm},m} \in \mathbb{R}$. The used parameter values are listed in table A.1. The target velocity is drawn from a uniformly distributed set $\tilde{v}_m \in \mathcal{Z}_{v,x,m}$ that is specified in sections 6.4 and 9.2.

Table A.1.: Parameters of the IDM for the $m = 1, 2, \ldots, \eta_{\mathrm{ov}}$ other vehicles.

description	variable	unit	value
minimum acceleration	$\check{a}_{\mathrm{idm},m}$	$\mathrm{m\,s^{-2}}$	-8
maximum acceleration	$\hat{a}_{\mathrm{idm},m}$	$\mathrm{m\,s^{-2}}$	3
acceleration exponent	e	1	4
desired time headway	$\tilde{t}_{\mathrm{hw}}$	s	2
jam distance	$\Delta\tilde{z}_x$	m	9
desired acceleration	$\tilde{a}$	$\mathrm{m\,s^{-2}}$	0.73
desired deceleration	$\tilde{a}_{\mathrm{d}}$	$\mathrm{m\,s^{-2}}$	1.67

A.3. Nonlinear Optimization Algorithm

Problem 6.2.1 is solved with the IPOPT algorithm [WB05]. The primal-dual interior-point algorithm is designed to solve large-scale nonlinear optimization problems. It handles inequality constraints via a barrier function. The barrier problem is solved with a damped Newton's method, which involves solving a linear equation system to obtain the direction for the next iteration. MUMPS [Ame+00] solves the linear equation system in this work. The solver obtains a solution efficiently by leveraging the problem's sparsity and the parallelization of computations. The C++ implementation of IPOPT is invoked via the Python interface of the CasADi toolbox [And+18]. The latter supplies the exact sparse

gradient information. Table A.2 lists the termination criteria for the IPOPT algorithm. An issue in the application of interior-point algorithms is their limited warm-starting

Table A.2.: IPOPT non-default parameters for algorithm termination. If not stated otherwise, the *max_iter* parameter is set to the given value.

description	parameter	value
abs. tol. acc. compl. conditions	*acceptable_compl_inf_tol*	$1 \cdot 10^{4}$
abs. tol. acc. constraint violation	*acceptable_constr_viol_tol*	$1 \cdot 10^{-4}$
abs. tol. acc. dual infeasibility	*acceptable_dual_inf_tol*	$1 \cdot 10^{4}$
rel. tol. acc. convergence	*acceptable_tol*	$1 \cdot 10^{-4}$
abs. tol. compl. conditions	*compl_inf_tol*	$1 \cdot 10^{4}$
abs. tol. constraint violation	*constr_viol_tol*	$1 \cdot 10^{-4}$
rel. tol. convergence	*tol*	$1 \cdot 10^{-4}$
abs. tol. dual infeasibility	*dual_inf_tol*	1
acc. iterates before termination	*acceptable_iter*	5
maximum number of iterations	*max_iter*	500

capabilities. Since the barrier functions require the optimization variables to obey their bounds strictly, IPOPT requires the initial guess to keep a minimum distance to the bounds. The parameters in table A.3 keep any modifications to the initial guess and the constraints low, so only a few iterations are needed to find the optimal solution from a good initial guess. Further non-default parameters are listed in table A.4.

Table A.3.: IPOPT non-default parameters that are important for warm-starting.

description	parameter	value
initial bound relaxation factor	*bound_relax_factor*	$1 \cdot 10^{-9}$
initial barrier parameter	*mu_init*	$1 \cdot 10^{-9}$
rel. distance primal variables	*warm_start_bound_frac*	$1 \cdot 10^{-9}$
abs. distance primal variables	*warm_start_bound_push*	$1 \cdot 10^{-9}$
abs. distance dual variables	*warm_start_mult_bound_frac*	$1 \cdot 10^{-9}$
rel. distance slack variables	*warm_start_slack_bound_frac*	$1 \cdot 10^{-9}$
abs. distance slack variables	*warm_start_slack_bound_push*	$1 \cdot 10^{-9}$
apply warm start	*warm_start_init_point*	yes

Table A.4.: IPOPT additional non-default parameters.

description	parameter	value
heuristic for infeasibility detection	*expect_infeasible_problem*	no
update strategy barrier parameter	*mu_strategy*	adaptive
maximum gradient after NLP scaling	*nlp_scaling_max_gradient*	1

A.4. Configurations

This section specifies the planning algorithms' configurations used in the previous chapters. Table A.5 lists the values for the bounds on the EV states and the reference path configuration, which are used in chapter 5 for the polynomial constraint formulation. The polynomial constraints are designed to approximate the state and input constraints conservatively while preserving a feasible space.

Table A.5.: Bounds on reference path configuration and vehicle states.

description	variable	unit	value
symmetric bound reference path curvature	$\bar{\kappa}_\mathrm{p}$	m^{-1}	$1.39 \cdot 10^{-3}$
symmetric bound reference path curvature rate	$\bar{\kappa}'_\mathrm{p}$	m^{-2}	$5 \cdot 10^{-6}$
symmetric bound vehicle y-acceleration	$\bar{a}_y$	$\mathrm{m\,s}^{-2}$	4
lower bound vehicle x-acceleration	$\check{a}_x$	$\mathrm{m\,s}^{-2}$	-8
upper bound vehicle x-acceleration	$\hat{a}_x$	$\mathrm{m\,s}^{-2}$	3.5
symmetric bound Frenét yaw angle	$\bar{\xi}_\psi$	$^\circ$	8.2
lower bound vehicle x-velocity	$\check{\xi}_v$	$\mathrm{km\,h}^{-1}$	60
upper bound vehicle x-velocity	$\hat{\xi}_v$	$\mathrm{m\,s}^{-2}$	130
symmetric bound Frenét y-position	$\bar{z}_y$	m	9.375

The parameters in table A.6 are used in the design of problem 6.2.1 and the terminal set. The target velocity $\tilde{v}$ deviates from $\hat{\xi}_v$ to avoid many infeasible trajectory candidates in the GP when the EV state is close to the target velocity.

Table A.6.: Parameters used in the problem 6.2.1 formulation.

description	variable	unit	value
target velocity	$\tilde{v}$	$\mathrm{m\,s}^{-1}$	122
prediction horizon	H	s	10
desired time headway	$\tilde{t}_\mathrm{hw}$	s	2.5
weight squared jerk in x-direction	$w_{\mathrm{j},x}$	$\mathrm{m}^2\,\mathrm{s}^{-5}$	1
weight squared jerk in y-direction	$w_{\mathrm{j},y}$	$\mathrm{m}^2\,\mathrm{s}^{-5}$	1
weight squared jerk in x-direction	$w_{T,x}$	s^{-1}	1
weight squared jerk in y-direction	$w_{T,y}$	s^{-1}	1
minimum breakpoint interval	$\Delta\check{\mathbf{k}}$	s	0.21

The configuration of the GP's time and state space decomposition is provided in table A.7. The parameters aim at a sparse discretization while providing a sufficient control performance in highway scenarios.

Table A.7.: Parameters used by the GP.

description	variable	unit	value
transformed terminal cost upper bound in x-direction	$\hat{\breve{F}}_x$	1	100
time samples x-direction	$\breve{K}_x$	1	10
time samples y-direction	$\breve{K}_y$	1	10
x-velocity samples	η_x	1	5
three breakpoint interval lengths	$\mathcal{I}_3$	s	[1, 9]
four breakpoint interval lengths	$\mathcal{I}_4$	s	config. dependent
breakpoint sequence lengths	$\mathcal{X}$	1	config. dependent

The vehicle shapes are modeled in section 4.2.4 with axis-aligned ellipses. The semi-axis lengths of the EV, of the other vehicles, and the dimension of the uncertainty ellipses are given in table A.8. The semi-axis lengths in the y-direction are designed so that two vehicles can drive next to each other on neighboring lanes. The semi-axes in the x-direction still enable the EV to follow a leading vehicle at the desired time headway.

Table A.8.: Parameters defining the EV and the other vehicles' shapes.

description	variable	unit	value
EV semi-major axis	Δ_x	m	5.21
EV semi-minor axis	Δ_y	m	1.3
other vehicles' semi-major axis	$\Delta_{x,m}$	m	3.77
other vehicles' semi-minor axis	$\Delta_{y,m}$	m	1.3
uncertainty semi-minor axis	$\overset{\circ}{\Delta}_y$	m	1.8
uncertainty semi-major axis time headway	$\overset{\circ}{t}_{\mathrm{hw}}$	s	1

A.5. Highway Heading Angles

The environment model describes the vehicle shapes with axis-aligned ellipses, assuming a limited heading angle deviation from the reference path. The HighD dataset [Kra+18], which includes recordings of 110 000 vehicle trajectories along straight highway segments, is used to choose a suitable limit. The occurrences of heading angles are reported in figure A.1. The choice $\psi_{\mathrm{ov}} = 7°$ overestimates the observed heading angles and retains a sufficient feasible set.

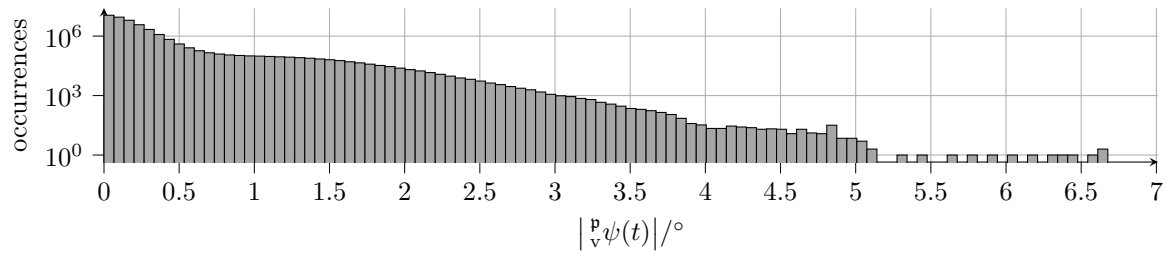


Figure A.1.: Heading angle occurrences in the HighD dataset at velocities beyond $60\,\mathrm{km\,h^{-1}}$.

A.6. Polynomial Derivatives

The terminal cost introduced in section 8.1 relies on an overestimation of the time integral over the running cost for transitions between LTs. It is assumed that kinematic constraints only yield a lower bound on the transition time. The property is fulfilled, because the LTs resemble velocity or acceleration equilibria. Thus, the higher order time derivatives converge to zero with an increasing time interval. This section depicts the properties of polynomials of degrees five and four. The boundary conditions on the first and second order time derivatives of a fifth degree polynomial are zero in figure A.2. In addition, the elements in the derivative coefficient transformation in equation (3.5.9) only decrease with a growing breakpoint difference. Thus, the polynomials' time derivatives have to converge to zero with a growing transition time.

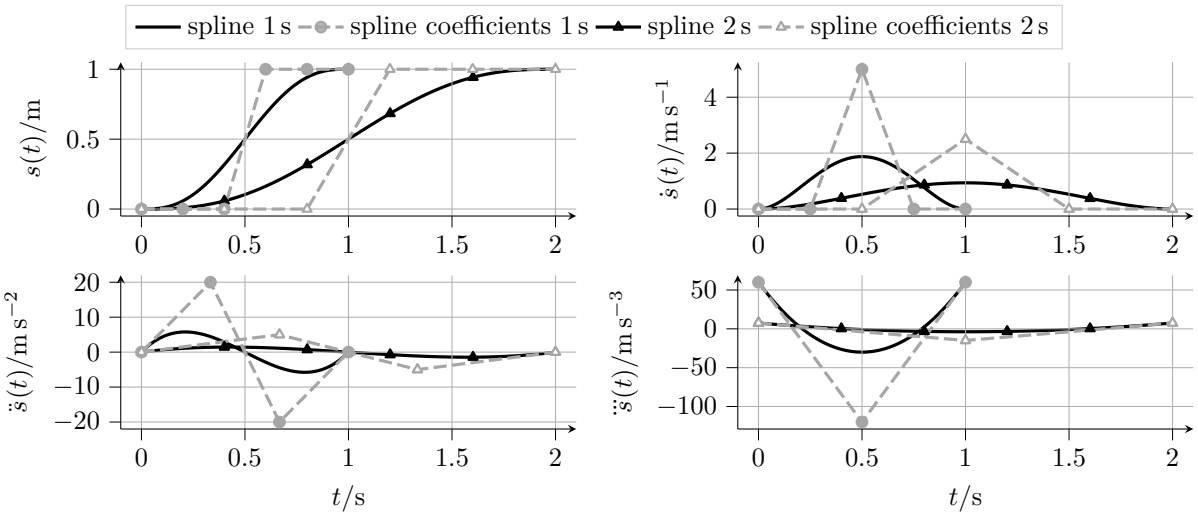


Figure A.2.: Polynomial time derivatives in basis form with boundary conditions at zero starting from the first time derivative.

In the case of the degree four polynomial, the boundary conditions on the second time derivative are zero, shown in figure A.3. Thus, time derivates beyond the first one approach zero as the segment length diverges.

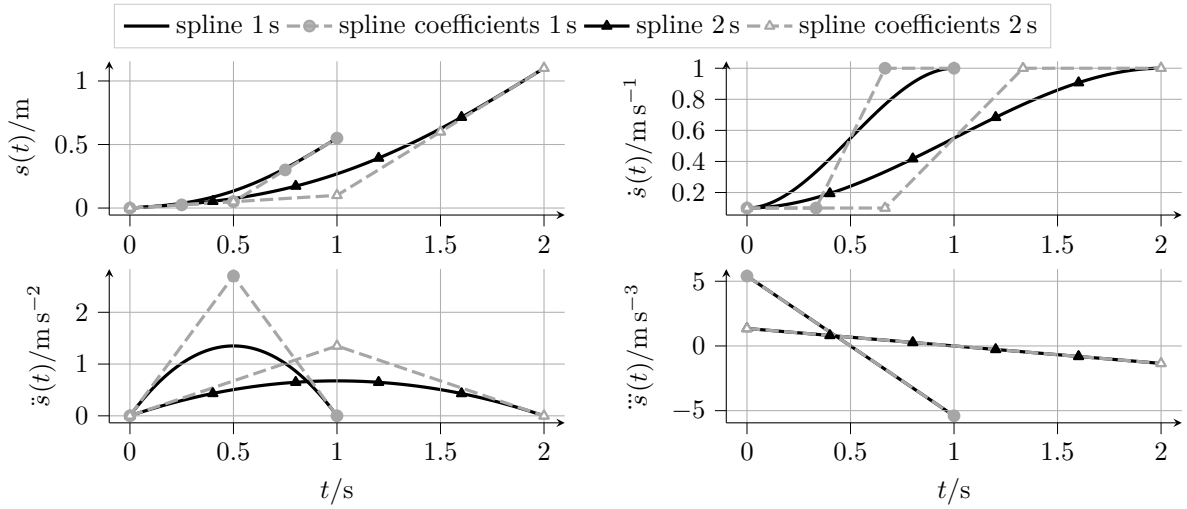


Figure A.3.: Polynomial time derivatives in basis form with boundary conditions at zero starting from the second time derivative.

A.7. Two-Stage Trajectory Planner Open-Loop Analysis

The LP improves the GP's result trajectory in the two-stage planning algorithm described in section 9.1. In addition, the GP provides a feasible backup trajectory if IPOPT fails to converge to a feasible solution. Thus, the maximum iterations i_{ter} can be safely reduced for a lower computational complexity. However, the benefit of the LP may vanish if i_{ter} becomes too low. This section analyses the influence of i_{ter} on the LP's ability to improve the GP's solution in open-loop and identifies a suitable iteration limit. The subsequent evaluation is performed on the scenes from section 8.2 with the *4bp-13* configuration. In contrast to the two-stage combination, the GP's solution is unconditionally passed to the LP as the initial guess for this experiment. A variation of the maximum iterations $i_{\text{ter}} \in \{10, 15, 20, 25, 30\}$ is assessed for all scenes.

The IPOPT algorithm can violate constraints during the optimization. Although a feasible initial guess is always provided, a termination before meeting the convergence criteria may return an infeasible trajectory. No large cost difference is observed among the feasible solutions in the same scene. Thus, the rate of feasible solutions is selected as the performance metric to compare the maximum iterations. The rates of infeasible and feasible solutions from all fifty-eight scenes are reported in figure A.4 for different breakpoint numbers. Trajectories with three breakpoints in each direction are referred to by *6bp*. The ones with three breakpoints in one direction and four breakpoints in the other are denoted by *7bp*. *8bp* refers to trajectories with four breakpoints in each direction.

In general, the feasible solution rate increases with i_{ter}. For $i_{\text{ter}} = 10$, none of the *8bp* trajectories result in a feasible solution, and most of the *7bp* ones are infeasible. However, a higher control performance benefit might be achieved due to the local optimization with a growing number of breakpoints. The number of successfully solved *8bp* problems is nonzero at $i_{\text{ter}} = 15$ and keeps growing together with the *7bp* trajectories at $i_{\text{ter}} = 20$. At $i_{\text{ter}} = 25$, all *6bp* problems are solved successfully. On the other hand, no improvement is achieved for the *8bp* problems. The improvement in solved problems continues for $i_{\text{ter}} = 30$, where only a portion of the *8bp* trajectories remains infeasible.

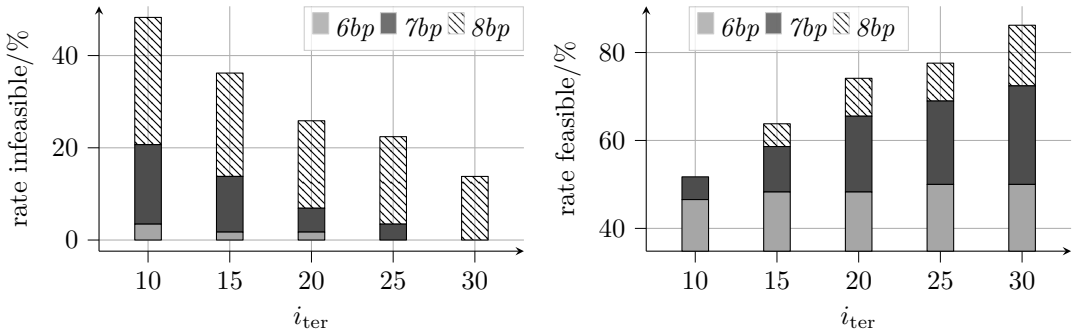


Figure A.4.: Rate of infeasible and feasible IPOPT solutions in 58 scenes due to different maximum iterations. The rates are categorized according to the sum of breakpoints in the x- and y-direction into six (*6bp*), seven (*7bp*) and eight (*8bp*) breakpoints.

With growing i_{ter}, the rate of solved problems saturates. The best compromise between the number of solved problems and i_{ter} is chosen using the quotient of the number of solved *8bp* and *7bp* problems per maximum iteration. The *6bp* are not considered because they are expected to yield comparably minor performance improvements due to the local optimization. The quotient is maximum at $i_{\text{ter}} = 20$ with 1.25. The lower iteration limits of $i_{\text{ter}} = 15$ and $i_{\text{ter}} = 10$ result in 0.6 and 0.3. The ratio decreases also to 0.64 and 0.7 for the higher limits of $i_{\text{ter}} = 25$ and $i_{\text{ter}} = 30$, respectively.

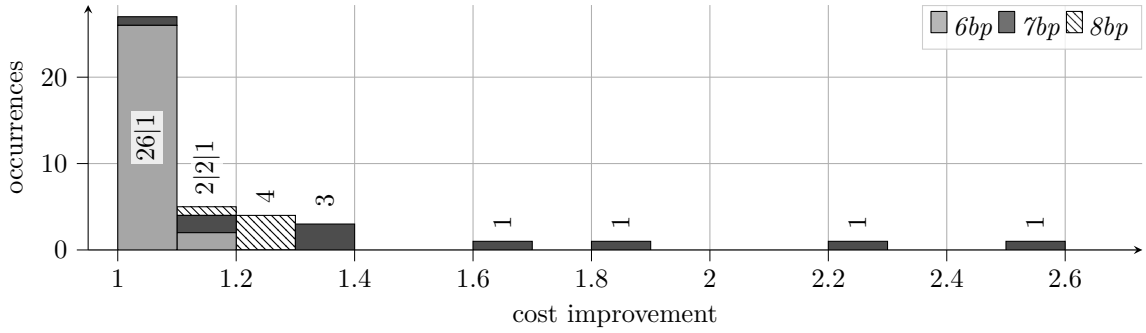


Figure A.5.: Cost improvement over the initial guess for $i_{\text{ter}} = 20$. The improvement is categorized according to the sum of breakpoints in the x- and y-direction into six (*6bp*), seven (*7bp*) and eight (*8bp*) breakpoints.

The cost improvement achieved with $i_{\text{ter}} = 20$ over the initial guess for all 43 feasible solutions is visualized in figure A.5. The *7bp* trajectories result in the highest improvement with a maximum of 2.55. The majority of trajectories are composed of six breakpoints, which achieve a high control performance already before the local optimization. Although the *8bp* trajectories have the most degrees of freedom, they rarely exceed the cost improvement due to *7bp*. The observation is explained by the limited scene variations considered. The highest gain in open-loop control performance is observed in scenes where the solution trajectory follows a leading vehicle. However, the solutions in those scenes do not benefit from an additional breakpoint in the y-direction. If a lane change is required while following a leading vehicle, it can always be performed directly with a single polynomial segment.
Figure A.6 provides a qualitative example of a *8bp* trajectory with a 1.2 times cost

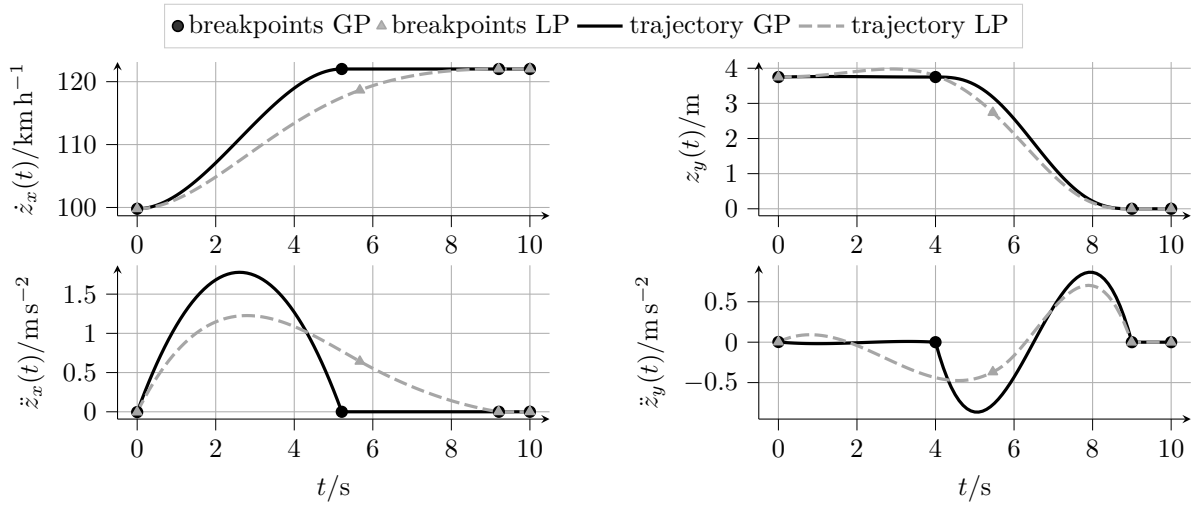


Figure A.6.: Trajectory with 1.2 times cost improvement corresponding to the scene in figure A.7.

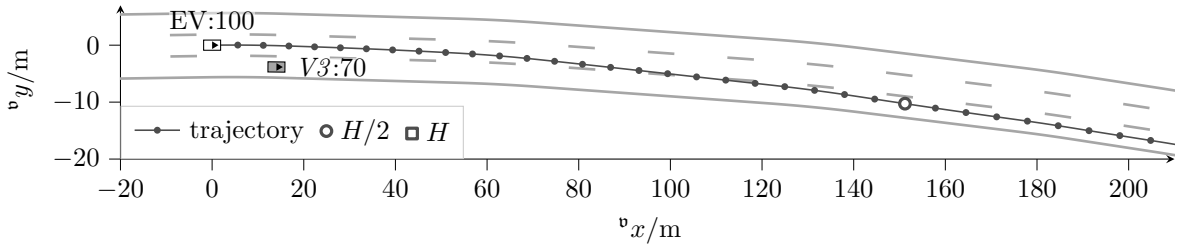


Figure A.7.: Scene with 1.2 times cost improvement. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the other vehicle with *V3*: $\dot{z}_{x,3}(t)/\mathrm{km\,h^{-1}}$.

improvement. The corresponding scene from *SC-2V* is shown in figure A.7, where the EV overtakes a slower-driving vehicle in the right-most lane. The acceleration in both directions has to be zero at the second breakpoint of the GP solution. The corresponding y-position has to follow one of the lane centers. In contrast, the LP can move the second breakpoint continuously in the state space and in time. Thus, a trajectory of lower acceleration and jerk is achieved while the control horizon is mostly retained.

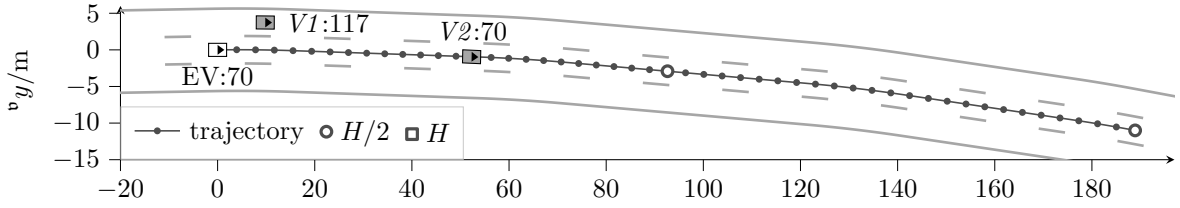


Figure A.8.: Scene with 2.55 times cost improvement. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2$ other vehicles with Vm : $\dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.

The scene from *SC-3V* with the highest observed cost improvement of 2.55 times is depicted in figure A.8. The EV follows the vehicle in front since the vehicle on the left-most lane prevents a lane change to the left. A high terminal cost prevents overtaking on the right-most lane. Figure A.9 only shows the velocity and acceleration in the x-direction

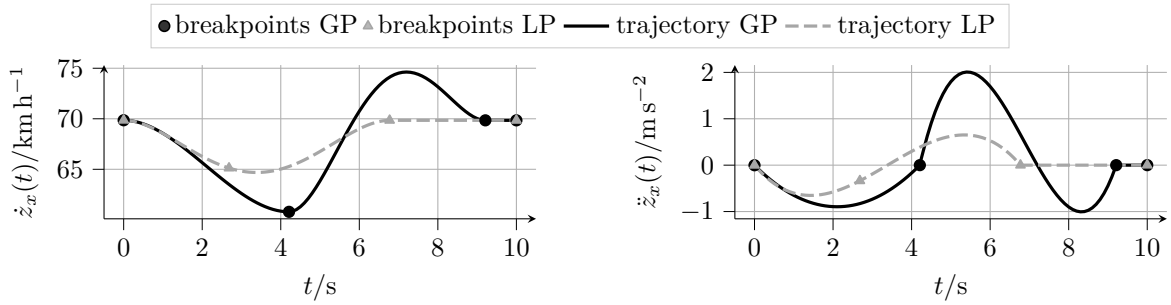
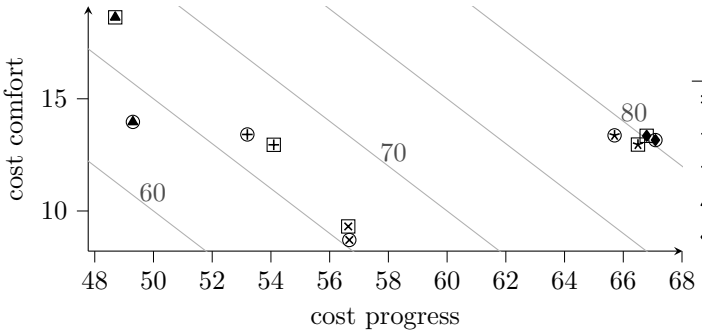


Figure A.9.: Trajectory with 2.55 times cost improvement corresponding to the scene in figure A.8.

because the planned trajectory stays on the middle lane. According to the GP solution trajectory, the EV has to slow down to $50\,\mathrm{km\,h^{-1}}$, followed by an acceleration to $75\,\mathrm{km\,h^{-1}}$ to reach the desired time headway behind vehicle 2. The LP finds a trajectory of lower acceleration and reduces the control horizon by 2.44 s.

A.8. Disabled Local Optimization

Section A.7 shows that the LP improves the GP's solution trajectory if IPOPT converges in the iteration limit. This section analyzes how far the open-loop improvement carries over to the closed-loop performance. Therefore, the evaluation in section 9.3 is extended by a configuration with disabled local optimization. The configuration *4bp-13-0it* is the same as *4bp-13*, except for $i_{\mathrm{ter}} = 0$. Thus, the LP does not improve the GP's solution. Either the GP's solution trajectory or the trajectory from the previous time step is returned by the two-stage planning algorithm. The configuration is applied to the same five scenarios from section 9.2. The corresponding closed-loop cost is reported in figure A.10 and compared with the *4bp-13* configuration. Except for scenario 4 and the second GT arrival in scenario 0, the cost is always higher for *4bp-13-0it*. The cost increase is small compared to the cost due to the *3bp-10* configuration in figure 9.3.



	4bp-45-0it		*4bp-13*	
	□ 1st	2nd	○ 1st	2nd
× scenario 0	65.94	109.80	65.38	110.20
+ scenario 1	67.05	-	66.61	-
⋆ scenario 2	79.46	-	79.07	-
▲ scenario 3	67.33	-	63.27	-
♦ scenario 4	80.15	-	80.26	-

Figure A.10.: Closed-loop comfort and progress cost when reaching the GT the first time. In addition, the contours of the closed-loop cost are shown. The table on the right side provides the closed-loop cost as sum of comfort and progress.

The largest cost difference is observed in scenario 3. The lower cost of the *4bp-13* configuration results mainly from a reduced jerk in the first 8.8 s of the *initial lane changes*

phase. The absence of improvement due to the LP leads to a deviation from the *4bp-13* closed-loop trajectory, yielding a higher acceleration and jerk in both directions, which is visible in figure A.12. At $t = 5.3\,\mathrm{s}$, the planning algorithms select the initial guess from the GP because the previously planned trajectory is infeasible. A discontinuity is visible in the x-acceleration trajectory with *4bp-13-0it* in figure A.12a. The missing optimization induces a subsequent increase in cost and a negative current cost descent factor. On the other hand, the LP with *4bp-13* enables a lower performance drop.

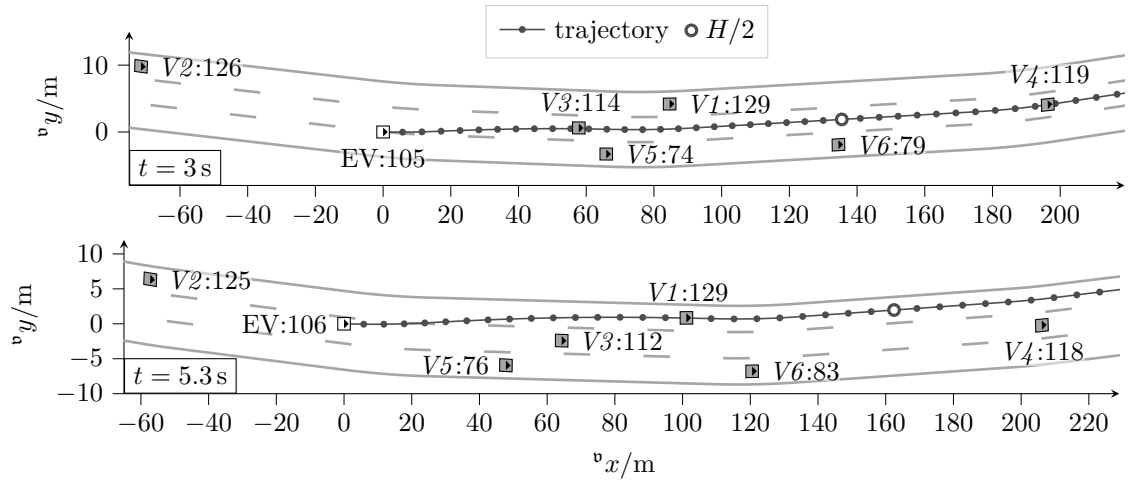
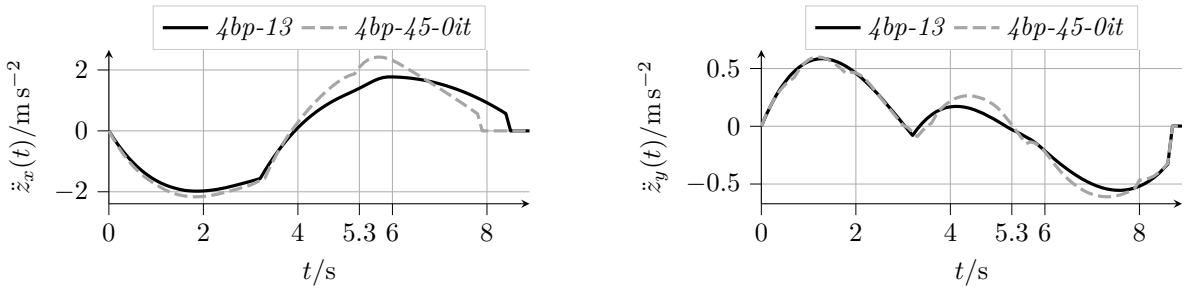


Figure A.11.: Scenes from the *initial lane changes* phase of scenario 3 using *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.



(a) y-position in the target lane center's Frénet frame.

(b) x-velocity in the target lane center's Frénet frame.

Figure A.12.: EV closed-loop trajectory in the first 8.8 s of scenario 3.

Two scenarios are observed where *4bp-13-0it* achieves lower cost than *4bp-13*. The trajectory refinement of the LP with *4bp-13* even contributes to the cost increase in scenario 0. At $t = 44.5\,\mathrm{s}$, depicted in figure A.13, both planning algorithms track vehicle 4 on the middle lane, which would be considered impeding otherwise. The planned velocities are shown in figure A.14a. The open-loop cost due to *4bp-13* is lower than the cost from *4bp-13-0it*, but the EV has to leave the target velocity. However, the middle lane is reached 0.8 s later, and a lane change to the left-most lane is initiated while the solution trajectory reaches the target velocity again. During the 0.8 s interval, the acceleration in figure A.14b induces nonzero comfort cost and a time penalty for not staying at the target velocity, which increases the closed-loop cost for *4bp-13*.

In scenario 4, no such suboptimal behavior is observed. The closed-loop cost due to the *4bp-13* configuration is smaller most of the time. Still, both configurations reach the GT simultaneously. However, the jerk induced by the dual mode strategy, which is described in section 6.3 and sets the vehicle into the terminal set, is higher with *4bp-13* and causes a higher cost at the simulation end. The results might change for a different dual mode activation threshold. Thus, for cost differences in the order of 10^{-1}, it is unclear whether the planning algorithm configuration causes the performance differences.

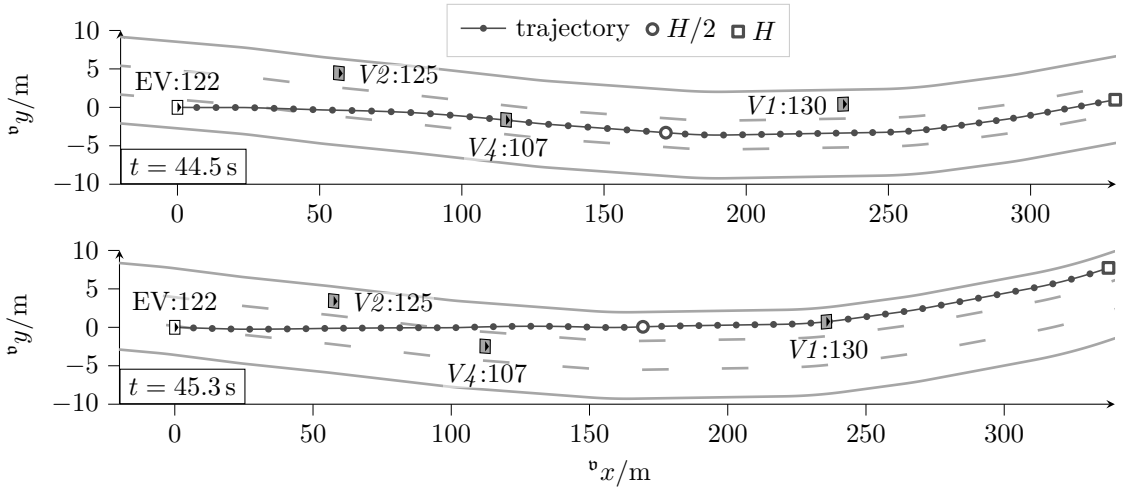
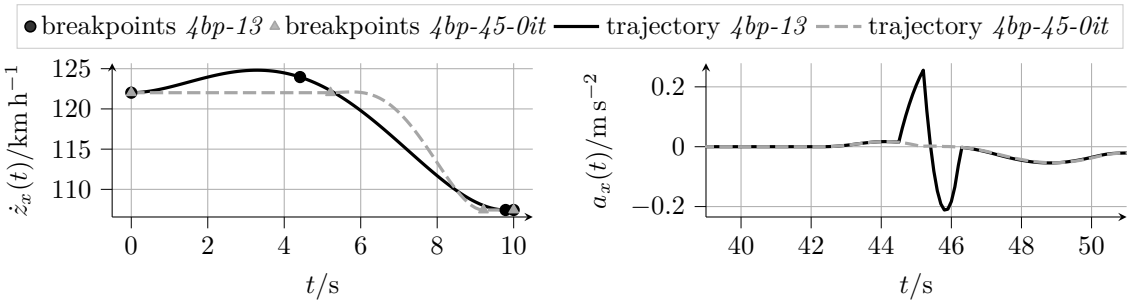


Figure A.13.: Scenes from the *initial lane changes* phase of scenario 3 using *4bp-13*. The EV is indicated by EV: $\dot{z}_x(t)/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}(t)/\mathrm{km\,h^{-1}}$.



(a) open-loop x-velocity at $t = 44.5\,\mathrm{s}$

(b) closed-loop x-acceleration

Figure A.14.: Open-loop and closed-loop trajectories showing a suboptimal outcome of the local optimization in scenario 0.

In all scenarios, the maximum observed absolute accelerations increase by no more than $0.03\,\mathrm{m\,s^{-2}}$ and the absolute jerk by no more than $0.09\,\mathrm{m\,s^{-3}}$ compared to *4bp-13*. The difference in the minimum time headway from *4bp-13* is below 0.02 s. An overall increase or decrease in the time headway is not present. A negative cost decent factor, not observed with the *4bp-13* configuration, occurs with *4bp-13-it* in scenario 0 while following vehicle 3, which is shown in figure A.15. The LP's lower cost trajectories keep the current cost descent factor in figure A.16 positive but below one, while the metric turns negative for a limited time with the *4bp-13-0it* configuration.

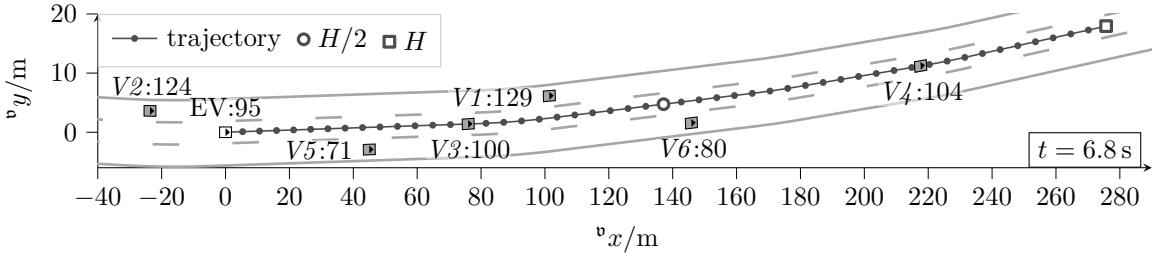


Figure A.15.: Scene while following vehicle 3 in the *initial lane changes* phase of scenario 0 using *4bp-13*. The EV is indicated by EV: $\dot{z}_x/\mathrm{km\,h^{-1}}$ and the $m = 1, 2, \ldots, 6$ other vehicles with $Vm : \dot{z}_{x,m}/\mathrm{km\,h^{-1}}$.

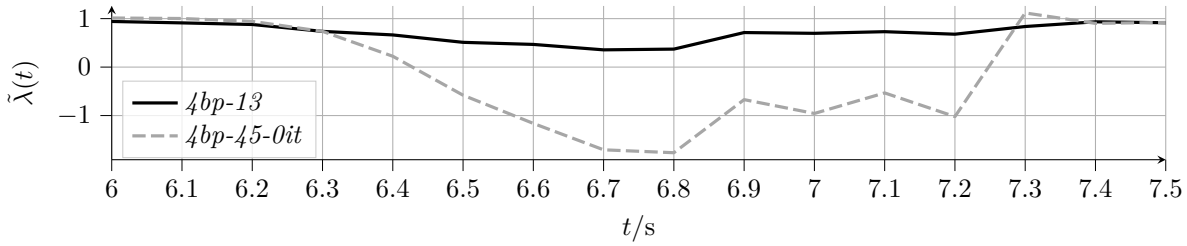
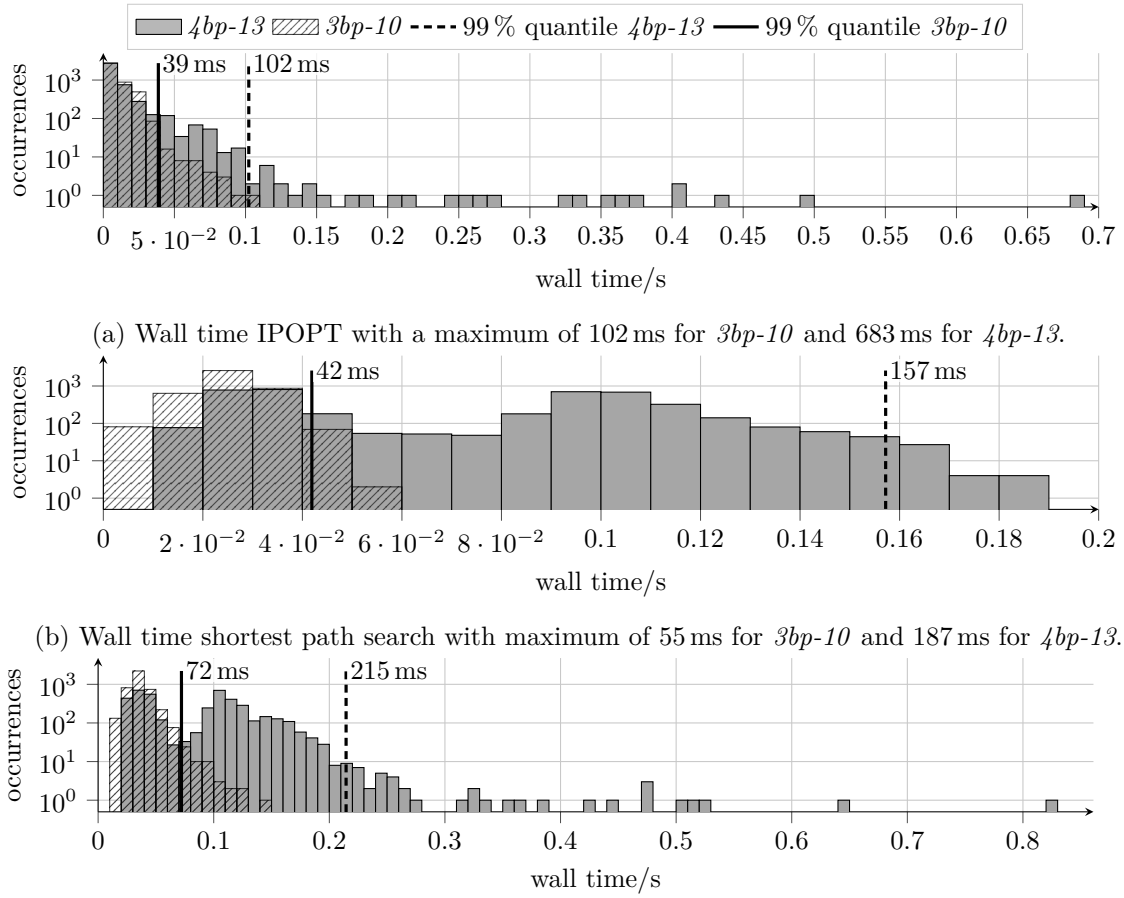


Figure A.16.: Cost descent while following vehicle 3 in scenario 0.

In summary, the *4bp-13-0it* configuration achieves a similar behavior and quantitative performance compared with *4bp-13*. Still, the results indicate a disadvantage if disturbances are present since the planning algorithm has to rely on the GP's suboptimal solution in cases where the previous solution becomes infeasible. While overall stability and progress toward the GT are barely affected, acceleration and jerk can increase compared to the results from the *4bp-13* configuration in the same situation. However, the performance difference might increase if the disturbances become more severe. A higher cost difference can also be expected if the trajectory parameterization is not the optimal solution to problem 4.3.1. Nevertheless, a LP of lower computational complexity than the *4bp-13* configuration might yield a similar control performance. For instance, the LP must not optimize all breakpoints but could focus on refining the spline coefficients.

A.9. Time Complexity

Figure A.17 provides an impression of the observed running times in connection with the five scenarios from the evaluation in section 9.3. Figure A.17a shows occurrences of wall times spent in IPOPT during the optimization in the LP. Figure A.17b depicts the wall times required for the GP's shortest-path search algorithm. The sum of the wall times from each simulation time step is shown in figure A.17b. The measurements were carried out on Ubuntu 22.04 with AMD Ryzen 5 3600 CPU at 3.6 GHz and 16 GB RAM. However, the measurements should be interpreted cautiously because no repetitions are performed. Also, the measurements consider only the time of the spend in IPOPT and in the shortest-path search algorithm. The preparations and postprocessing steps performed in the planning algorithms are not included.



(a) Wall time IPOPT with a maximum of 102 ms for *3bp-10* and 683 ms for *4bp-13*.

(b) Wall time shortest path search with maximum of 55 ms for *3bp-10* and 187 ms for *4bp-13*.

(c) Sum of IPOPT and shortest path search wall times assuming sequential execution.

Figure A.17.: Wall time occurrences for solving the trajectory planning problems occurring in the five scenarios in section 9.3 with the *4bp-13* and *3bp-10* configurations.

Most of the running times resulting from the *4bp-13* and *3bp-10* configurations in figure A.17a are below 100 ms. However, the worst cases due to *4bp-13* are exceptionally high. Generally, the higher number of constraints and optimization variables due to *4bp-13* increase the average running time per iteration by about 2 times at the 99 % quantile. Based on the iteration counts reported in figure 9.10a, the average running time per iteration is 20.4 ms. In contrast, the average running time per iteration amounts to 9.75 ms for the *3bp-10* configuration.

The shortest-path search algorithm is implemented with the Numba toolbox [Lam+15], which compiles the Python implementation just in time. The implementation does not leverage parallel computations. In most cases, the graph search is more time-intensive than the local optimization, but it it requires less time in the worst cases.

The sum of the two distinct planning algorithms' running times is a lower bound on the running times achievable with a two-stage combination, if further computations are neglected. The distributions of the combined running times' occurrences are reported in figure A.17c At the 99 % quantile, the overall running times due to the *4bp-13* configuration are about three times higher than those due to the *3bp-10* configuration. The factor lies between the ones from the LP with 2.6 and the GP with 3.7. During closed-loop simulation,

an additional breakpoint in the x-direction is used more frequently than in the y-direction. Thus, limiting the number of breakpoints in the y-direction from four to three could resemble a good compromise between a reduced GP control performance and a reduced running time.

A.10. Usage of generative AI - Affidavit

☐ not at all

☒ for correcting, optimizing, or restructuring the entire work (This eliminates the need for explicit marking of individual passages or sections, as this type of usage refers to the entire written work. Explicit marking in the text is not necessary, as this serves as the global indication.)

☐ Code optimization: Optimization or restructuring of software function

☐ Code generation: Creating entire software functions from a detailed functional description.

☐ Substance generation in code: Generating entire software source code

☐ Media optimization: Correction, optimization, or restructuring of entire passages

☐ Media generation: Creating entire passages from given content.

☐ Substance generation in media: Generating entire sections

☐ More, namely:

I assure that I have provided all usages completely. Missing or incorrect information may be considered an attempt to deceive.

place, date

Philip Dorpmüller