Two-Layer MPC Architecture for Efficient Mixed-Integer-Informed Obstacle Avoidance in Real-Time

Alexander L. Gratzer, Maximilian M. Broger, Alexander Schirrer, and Stefan Jakubek

***Abstract*— Safe and efficient obstacle avoidance in complex traffic situations is a major challenge for real-time motion control of connected and automated vehicles (CAVs). Limited processing power leads to a trade-off between real-time capability and maneuver efficiency, especially for trajectory planning in highly dynamic traffic environments like urban intersections. Addressing this problem, we propose a novel two-layer model predictive control (MPC) architecture utilizing a differentially flat representation of the kinematic single-track vehicle model for optimal control. While a real-time capable quadratic programming-based MPC ensures local obstacle avoidance at every time step, its problem formulation is asynchronously updated by the globally optimal solution of a computationally more expensive mixed-integer MPC formulation. Both optimization problems are computed in parallel and incorporate position predictions of surrounding traffic participants available via vehicle-to-everything (V2X) communication. Collision-free and efficient obstacle avoidance in real time under realistic model errors is validated via high-fidelity co-simulations of typical urban intersection and highway scenarios with the traffic simulator CARLA.**

***Index Terms*— Obstacle avoidance, model predictive control, flatness-based control, mixed-integer programming, motion planning, trajectory planning, nonlinear control, single-track model.**

I. Introduction

AUTOMATED driving can reduce conflicts between individual traffic participants' needs, safety, efficiency, and environmental impact at urban traffic nodes [1], [2], [3]. Collision-free and efficient obstacle avoidance (OA) in real-time is a key aspect of automated driving and poses a computationally expensive task. Especially in complex road scenarios consisting of a multitude of traffic participants, it is crucial to develop a well-informed motion planner that guarantees collision-free and efficient maneuvers while considering the trade-off between real-time capability and fidelity. In this work, we seek to tackle this challenge by proposing a novel two-layer obstacle avoidance model-predictive control (MPC) architecture for optimal and collision-free vehicle control in real-time that incorporates motion predictions of surrounding traffic participants.

Recent methods suitable for trajectory generation for connected and automated vehicles (CAVs) can be categorized as machine-learning-based, sampling-based, geometry-based, and optimization-based approaches [4], [5]. This work focuses on optimization-based motion planning, specifically model-predictive control methods. MPC allows to inherently consider input and state constraints and predictions of other traffic participants in an optimization problem formulated over a receding prediction horizon. This optimal control concept proves highly useful to realize efficient obstacle avoidance with collision safety guarantees while exploiting vehicle-to-everything (V2X) communication capabilities [6]. The computational efforts for online optimization conducted in MPC algorithms are high and increase with the complexity of the used prediction models, (collision avoidance) constraints, and the number of considered obstacles.

While nonlinear MPC (NMPC) methods utilize high-fidelity (nonlinear) prediction models and obstacle avoidance constraints, the solutions to the emanating nonlinear programming (NLP) problems are not guaranteed to be globally optimal due to the employed local minimum search algorithms. Further, NLP-based trajectory optimization implies high computational effort which renders NMPC-based obstacle avoidance approaches problematic for real-time application in complex scenarios.

One method to reduce the computational load induced by NMPC is successive system linearization which results in a linear time-variant MPC (LTV-MPC) formulation [7]. However, the LTV approximation of the original NMPC problem introduces linearization errors and requires a decent initialization of the local search algorithm that might converge to a locally optimal rather than a globally optimal solution. This means, for example, that the decision to take another route around an obstacle, an evident case of efficient obstacle avoidance, is fundamentally not covered by LTV-MPC implementations.

Certain nonlinear prediction models, which possess the property of differential flatness, can be *exactly* linearized [8] through nonlinear system transformations. The resulting equivalent linear system representation can be used for flatness-based linear time-invariant MPC (LTI-MPC). LTI-MPC outperforms LTV-MPC in tracking performance [9] and guarantees a globally optimal solution, understood with respect to the transformed (flat) problem description. Achieving global

Manuscript received 30 May 2023; revised 9 March 2024; accepted 14 May 2024. Date of publication 30 May 2024; date of current version 4 October 2024. This work was supported in part by the Austrian Research Promotion Agency (FFG) via the Research Project Intelligent Intersection [Information and Communications Technologies (ICT) of the Future] under Grant 880830 and in part by Vienna University of Technology (TU Wien) Bibliothek through its Open Access Funding Program. The Associate Editor for this article was L. Li. *(Corresponding author: Alexander L. Gratzer.)*

The authors are with the Institute of Mechanics and Mechatronics, TU Wien, 1060 Vienna, Austria (e-mail: alexander.gratzer@tuwien.ac.at; maximilian.broger@tuwien.ac.at; alexander.schirrer@tuwien.ac.at; stefan.jakubek@tuwien.ac.at).

Digital Object Identifier 10.1109/TITS.2024.3402559

optimality is vital for realizing efficient and agile obstacle avoidance. Examples of differentially flat systems related to automated driving are the well-known dynamic (holonomic) [10] and kinematic (non-holonomic) [11] single-track vehicle models. The *kinematic* single-track model, despite not considering tire slip, is widely used in trajectory planning for automated vehicles [12], [13], [14], [15], [16], [17]. It turns out to be useful for obstacle avoidance modeling at low and moderate speeds since it represents the relevant vehicle dynamics with sufficient accuracy as shown in [18].

The structure of the emanating optimization problem is determined not only by the properties of the prediction model applied but also by the formulation of the implemented (obstacle avoidance) constraints. Obstacle avoidance can be realized by formulating linear half-space constraints and utilizing the Big-M method [19], resulting in a mixed-integer programming (MIP) formulation. Hereby the obstacle shapes are approximated by convex polygonal regions with binary variables defining the edges for the relevant exclusion constraints [20]. The MIP problem formulation provides *globally* optimal solutions to the non-convex obstacle avoidance motion planning problem, but it is still an NP-hard problem and therefore does not provide useful worst-case bounds on solving effort, which renders it problematic for real-time applications in complex traffic scenarios [20], [21].

A. Related Work

Recent model-based obstacle avoidance approaches are collected in Table I, whereby we consider an algorithm real-time capable if its maximum core solver time lies below 0.05 s. These concepts can be roughly categorized regarding the utilized prediction model for the ego dynamics, the employed obstacle avoidance constraints (which comprise the representations of the ego vehicle and obstacle shapes), the resulting MPC implementation, and real-time capability.

All mentioned works employ nonlinear models to predict the ego vehicle's motion, whereby the dynamic [15], [22], [24] and kinematic [13], [14], [15], [16], [17] single-track models are most commonly used. Models with higher fidelity are used in [23], [25], and [26].

The formulation of the obstacle avoidance constraints heavily influences the optimal control problems' (OCPs) complexity and calculation time. In [24] the ego and obstacle shapes are represented by sets of circles which captures the effect of vehicle rotation and results in an NMPC problem formulation that is not real-time capable and relies on an initial trajectory provided by a higher-level planner. Reference [25] considers the space occupied by the ego vehicle as a polygonal region while obstacles are represented as static points obtained by a LIDAR sensor. While the resulting linear obstacle avoidance constraints enable fast numerical computation of the emanating convex quadratic programming (CQP) problem, obstacle predictions are disregarded. Essentially real-time-capable obstacle avoidance is achieved in [22] by approximating the obstacles by ellipsoidal avoidance constraints. These are simplified to be aligned axis-parallel with the ego reference path. The resulting NLP problem is solved by a local sequential quadratic programming (SQP) approach. Real-time capable NMPC algorithms for avoiding static obstacles are proposed in [14] and [23]. Both approaches consider obstacles by a reduced lateral track width in Frenet coordinates, making them not applicable for complex multi-agent traffic scenarios. Obstacle avoidance in real-time is realized in [13] utilizing intention predictions of other traffic participants whose shapes are approximated as superellipses. The emanating NLP optimization problem is approximated by a convex quadratic program (QP) formulation resulting in an LTV-MPC implementation with its aforementioned drawbacks. The computational effort scales cubically with the number of obstacles considered which may jeopardize real-time computation in dense urban traffic scenarios. In [16] the authors use successive linearization to transform originally non-convex obstacle avoidance constraints into linear ones whereas the initial trajectory is obtained by an A*-reshaping algorithm which requires a fully discretized configuration space. The proposed algorithm shows a high success rate in finding feasible trajectories for static environments but lacks real-time capability. An adaptive Lagrange discretization and hybrid obstacle avoidance constraints (elliptic and dynamical linear ones) are proposed in [26]. While real-time computation could not be achieved, the authors aim at converting the sequential numerical solving process into a parallel one for utilizing parallel processors. In [17] and [28] obstacle avoidance is realized by constructing a spatial-temporal corridor around an initial trajectory derived by a sampling-and-search algorithm (dynamic programming) and formulating "within-corridor" constraints. The emanating NLP control problem is iteratively solved via a local search solver. A gradient descent-based obstacle avoidance MPC implementation is proposed and investigated with respect to the computational capabilities of automotive electronic control units (ECUs) in [15]. It is shown that the dynamic single-track model proves problematic for real-time applications while the kinematic single-track model captures vehicle motions very well in normal driving conditions, but leads to more aggressive control actions. Static circular obstacle shapes on a two-lane road are considered.

All concepts discussed up to this point utilize local search algorithms to solve the emanating NLP problems which result due to a combination of more or less detailed prediction models and obstacle avoidance constraint formulations. As a result, these concepts yield only *locally* optimal motion trajectories, which may lead to inefficient maneuvers in complex intersection traffic scenarios as illustrated in Fig. 1.

An overview of recent MIP-based motion control concepts that provide *globally* optimal solutions to the non-convex obstacle avoidance problem at the expense of an emanating NP-hard OCP formulation can be found in [21]. Compared to the broad research found on local approaches as summarized above, significantly fewer recent obstacle avoidance publications utilize mixed-integer-based formulations. Addressing the computational complexity of mixed-integer quadratic programming (MIQP), reference [29] proposes an online algorithm that exploits the optimization problem's structure by reducing it to a neural network evaluation and a linear system solution. A benchmark motion planning example

TABLE I
RELATED WORK

Ref.	ego shape	ego prediction	obstacle shape	obst. pred.	misc.	problem type	real-time
[22]	rectangle	particle model / dyn. single-track	ellipse	const. accel. on ref. path	obstacles axis parallel aligned to ego ref. path	NLP (SQP)	yes
[23]	point	nonlinear	red. track-width	static	high-fidelity model	NLP (NMPC)	yes
[14]	point	kin. single-track	red. track-width	static	given obstacle allocation	NLP (NMPC)	yes
[13]	2 circles	kin. single-track	superellipse	intentions	cooperative planning	NLP (LTV-MPC)	yes
[15]	circle	kin./dyn. single-tr.	circle	static	two-lane road	NLP (NMPC)	yes/no
[24]	m circles	dyn. single-track	circle	static	initial trajectory provided	NLP (NMPC)	no
[25]	conv. polygon	nonlinear	point-cloud	static	LIDAR point cloud	CQP (CMPC)	no
[16]	m circles	kin. single-track	rectangle	static	initial traj. provided by A*	NLP (iter. OCP)	no
[17]	2 circles	kin. single-track	rectangle	dyn. corridor	initial traj. provided by DP	NLP (iter. OCP)	no
[26]	point	high-fidelity	ellipse	on ref. path	hybrid OA constraints	NLP (NMPC)	no
[27]	circle	kin. single-track	rectangle	const. vel.	obstacles axis parallel aligned to ego ref. path, $T_\mathrm{s} = 1\,\mathrm{s}$	MIQP (MIP-DM)	yes

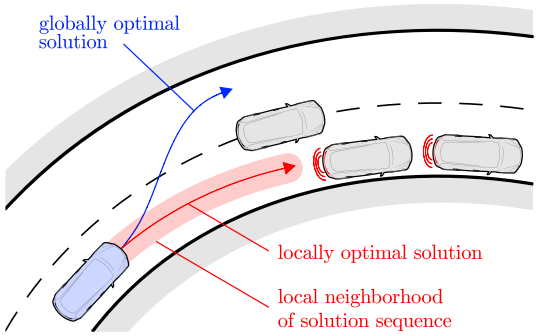

Fig. 1. MIQP concept illustration: Global versus local optimality in the obstacle avoidance context. After detecting the decelerating cars in the right lane a local search algorithm may not be able to find the global optimal trajectory because it lies outside the local search neighborhood.

with avoidance of static obstacles realized via the Big-M formulation [19] shows speedups from two to three orders of magnitude compared to Gurobi [30], a state-of-the-art MIQP solver. A MIP-based MPC decision maker (MIP-DM) for automated driving is developed in [27] which uses a linear vehicle model in road-aligned coordinates, including obstacle avoidance, lane-change decisions, and traffic rules via mixed-integer inequalities. The controller evaluates in real-time in various low-speed (1 m/s) traffic scenario simulations using the dedicated solvers *BB-ASIPM* [31] and Gurobi. Its limited OA fidelity and coarse sampling time of $T_\mathrm{s} = 1\,\mathrm{s}$, however, could render the concept problematic in complex and highly dynamic traffic situations such as crowded intersections.

These recent findings show that promising MIQP-based obstacle avoidance concepts are on the verge of becoming real-time capable while solving the OA problems in a globally optimal way, thus effectively avoiding deadlocks.

B. Research Gap & Contributions

To the best of the authors' knowledge, there exists no obstacle avoidance control concept that enables globally optimal automated driving in real-time while exploiting V2X communication to multiple moving and/or static obstacles of arbitrary shape and orientation. Limited available processing power leads to a trade-off between achieving global optimality or real-time capability. It is desirable to enable detailed and efficient motion control while aiming for global optima and real-time computation of the obstacle avoidance problem. It is this research gap that we attempt to close with the obstacle avoidance control architecture proposed in this work.

The main contribution of this work is the development of a novel two-layer LTI-MPC architecture to solve the obstacle avoidance and automated driving control problem. We extend our work presented in [20] to design an upper-level MIQP-based obstacle avoidance controller that provides globally optimal performance and agility by exploiting the differential flatness property of the used single-track vehicle model. Even though sufficiently efficient on average, since MIQP problems are generally NP-hard, however, no reasonable worst-case runtime bounds can be stated [21]. Hence, this MPC alone does not guarantee real-time solvability. The proposed lower-level QP-MPC is designed as a closely related conservative convex real-valued QP, informed by the last known upper-level MIQP solution structure. It provides acceptable runtime guarantees and is ensured to produce a stabilizing and collision-free solution. Global optimality in the obstacle avoidance optimal control problem sense is automatically recovered whenever the upper-level MIQP-MPC computation time is sufficiently fast. The proposed solution approach combines this global optimality aspect with safety and feasibility guarantees and the direct integration of other road participants' motion predictions, providing excellent situational awareness and efficiency in partially and fully automated road traffic scenarios. The novel two-layer obstacle avoidance MPC (TL-OA-MPC) architecture is analyzed and tested in typical complex urban intersection and highway scenarios, and its excellent performance is validated in a high-fidelity co-simulation study with the CARLA Simulator [32].

The contributions of this work are summarized as follows:

1) A two-layer LTI-MPC architecture is developed to solve the obstacle avoidance and automated driving control problem utilizing a differentially flat prediction model.
2) An upper-level OA-MPC provides globally optimal performance by exploiting an MIQP problem formulation.
3) A lower-level controller provides collision-free solutions in real-time via a QP-MPC formulation which is asynchronously updated with upper-level solutions to recover global optimality whenever computationally possible.

4) The control architecture aims to provide safe, feasible, and globally optimal maneuvers in real-time by integrating other road participants' motion predictions.
5) The control concept is validated in a high-fidelity co-simulation study.

The remainder of this work is organized as follows: The problem formulation is given in Sec. II. The proposed two-layer OA-MPC control architecture is presented in Sec. III and discussed in Sec. IV. The performance of the proposed control concept is validated in a realistic co-simulation study in Sec. V while Sec. VI concludes this paper.

II. Problem Formulation

This work considers the motion planning and control of a single CAV in urban intersection traffic scenarios with an emphasis on dynamic obstacle avoidance.

A. Control Goals

The following control goals need to be addressed when realizing obstacle avoidance in road traffic: (i) Collision safety against static and dynamic obstacles, (ii) stability, and (iii) feasibility with respect to the vehicle dynamics have to be guaranteed at all times, which requires (iv) the control problem to evaluate in real-time. (v) The resulting maneuver should be efficient while (vi) maximizing passenger comfort and (vii) obeying traffic regulations.

The two-layer OA-MPC architecture proposed in this work addresses all mentioned aspects efficiently. The novel combination of mixed-integer and quadratic programming combines the advantages of both approaches and yields globally respective locally optimal trajectories in real-time.

B. Assumptions

The control problem is based on the following assumptions:

A1) A pre-defined reference path, usually mid-lane, is available (e.g., provided by an environmental perception unit).
A2) Backward vehicle motions are disregarded.
A3) The shape of the ego vehicle is represented as a circle centered at the rear axle with radius r.
A4) The planar poses and shapes of surrounding traffic participants together with their...
A5) deterministic motion predictions in the form of position trajectories over a defined prediction horizon are available.
A6) An external perception module detects, classifies, and observes relevant traffic participants (and possibly orders them by criticality).

CAV predictions are received via V2X communication while HDV predictions are provided either via V2X communication (e.g., prediction done by intelligent infrastructure, collective perception) or an onboard prediction module. As in [27], [33], and [34] we assume the prediction and communication modules are present. We propose a deterministic MPC design, yet the consideration of uncertainties will be discussed in Sec- IV.

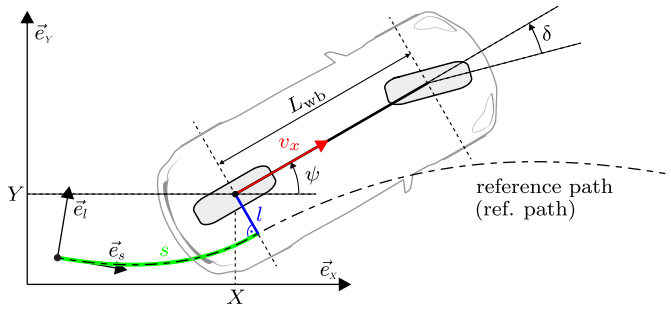

Fig. 2. Kinematic single-track vehicle model incl. Frenet coordinates (s, l) with respect to a given reference path.

C. Vehicle Model

The vehicle dynamics used for the control design are modeled according to a kinematic single-track model (non-holonomic, zero slip, also referred to as bicycle model) depicted in Fig. 2. The equations of motion are formulated as (cf. [20])

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{\psi} \\ \dot{v}_x \end{bmatrix} = \begin{bmatrix} v_x \cos\psi \\ v_x \sin\psi \\ v_x \tan(\delta)/L_{\mathrm{wb}} \\ a_x \end{bmatrix}, \tag{1}$$

with the state vector $\boldsymbol{x} = \left[X,\ Y,\ \psi,\ v_x\right]^{\mathrm{T}}$ comprising the global Cartesian coordinates X and Y, the heading angle ψ, and the longitudinal velocity $v_x \geq 0$ (compare assumption A2). Note that, here, the lateral velocity vanishes, i.e. $v_y \equiv 0$. The input vector $\boldsymbol{u} = \left[a_x,\ \delta\right]^{\mathrm{T}}$ contains the longitudinal acceleration and the steering angle, respectively. The parameter $L_{\mathrm{wb}} > 0$ represents the wheelbase distance. Although (1) by definition does not consider tire slip, the model yields consistent results for limited lateral vehicle accelerations as discussed in [18] and observed in Sec. V.

By exploiting the differential flatness property of the kinematic single-track model, (1) can be transformed to flat coordinates, producing an exactly linearized LTI system comprised of two decoupled double integrators

$$\dot{\boldsymbol{z}} = \underbrace{\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\boldsymbol{A}_{\mathrm{c}}} \boldsymbol{z} + \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\boldsymbol{B}_{\mathrm{c}}} \boldsymbol{v}, \tag{2}$$

with the flat state vector $\boldsymbol{z}$ and virtual input vector $\boldsymbol{v}$ depending on the particular choice of flat outputs $\boldsymbol{y}$. In this work, the Frenet coordinates (s, l), with s being the arc length and l being the lateral deviation with respect to a defined differentiable reference path, see Fig. 2, are chosen as flat outputs

$$\boldsymbol{y} = \left[s,\ l\right]^{\mathrm{T}}, \tag{3}$$

which yields the flat state and virtual input vectors

$$\boldsymbol{z} = \left[s, \quad \dot{s}, \quad l, \quad \dot{l}\right]^{\mathrm{T}}, \quad \boldsymbol{v} = \left[\ddot{s}, \quad \ddot{l}\right]^{\mathrm{T}}. \tag{4}$$

The reference path is assumed to be provided by an environmental perception module (not discussed in this work).

Since no analytical transformation between the physical and flat Frenet states is possible for general (arbitrarily curved) reference paths, the numerical transformation methods `global2frenet` ($\mathcal{F}$) respectively `frenet2global` ($\mathcal{F}^{-1}$) as provided by the MATLAB® environment [35] are used to map vertices into flat coordinates and vice versa. By representing the vehicle dynamics in (flat) Frenet coordinates it is possible to decouple longitudinal and lateral vehicle dynamics control [35] which allows straightforward implementation of car-following strategies and lateral lane geometry constraints [20]. In particular, $v_x = \dot{s}$ and $a_x = \ddot{s}$ hold for nominal motion along the reference path. While this representation is not applicable in unstructured road environments and the mapping to Cartesian coordinates introduces distortions for tightly curved roads [17], the Frenet frame is commonly utilized for autonomous driving in (semi-)structured road environments [4], [13], [14], [23], [36], [37]. The LTI vehicle model representation (2) facilitates its straightforward application for linear MPC design, as shown in the next section.

III. Two-Layer Obstacle Avoidance Control

This section presents the two-layer obstacle avoidance control architecture for optimal real-time control of a CAV in urban traffic. It consists of two MPC formulations of similar structure which only differ in their formulation of the obstacle avoidance constraints, leading to different types of optimization problems. On one hand, a mixed-integer quadratic programming MPC (MIQP-MPC) formulation realizes globally optimal obstacle avoidance. On the other, a convex quadratic programming MPC (QP-MPC) ensures real-time capability with reduced OA fidelity. Both controllers are evaluated asynchronously in parallel. The QP-MPC realizes real-time control, being flexibly informed by the MIQP-MPC once it produces a new solution. After presenting the general MPC setup and the obstacle constraint variants, the control architecture is completed by proposing a robustifying solution selection concept.

A. Generic Flatness-Based OA-MPC Formulation

The generic OA-MPC formulation used for both, the MIQP- and the QP-MPC designs is introduced as in [20]. The linearized LTI system dynamics (2) is solved and expressed in discrete time with sampling time T_s under the zero-order hold assumption:

$$\boldsymbol{z}_{k+1} = \underbrace{\begin{bmatrix} 1 & T_\mathrm{s} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_\mathrm{s} \\ 0 & 0 & 0 & 1 \end{bmatrix}}_{\boldsymbol{A}_\mathrm{d}} \boldsymbol{z}_k + \underbrace{\begin{bmatrix} T_\mathrm{s}^2/2 & 0 \\ T_\mathrm{s} & 0 \\ 0 & T_\mathrm{s}^2/2 \\ 0 & T_\mathrm{s} \end{bmatrix}}_{\boldsymbol{B}_\mathrm{d}} \boldsymbol{v}_k \,. \tag{5}$$

It is used as the prediction model for the ego vehicle dynamics. The discrete-time OCP at time step $t_k = k\,T_\mathrm{s}$ with $k \in \mathbb{N}$ is to find the optimal transformed input sequence $\boldsymbol{\mathcal{V}}_k^* = \left[\boldsymbol{v}_k^*, \boldsymbol{v}_{k+1}^*, \ldots, \boldsymbol{v}_{k+N_\mathrm{p}}^*\right]$ and flat state sequence $\boldsymbol{Z}_k^*$ which minimize the convex quadratic objective function (6a) subject to the constraints (6b)–(6e) explained below:

$$\min_{\boldsymbol{\mathcal{V}},\boldsymbol{Z}} \quad \left(J + \boldsymbol{r}_\mathrm{s}^\mathrm{T}\boldsymbol{s} + J_\mathrm{term}\right) \tag{6a}$$

$$\text{s.t.} \quad \boldsymbol{z}_{k+j+1} = \boldsymbol{A}_\mathrm{d}\,\boldsymbol{z}_{k+j} + \boldsymbol{B}_\mathrm{d}\,\boldsymbol{v}_{k+j}, \tag{6b}$$

$$\boldsymbol{v}_{k+j} \in \mathcal{W}, \tag{6c}$$

$$\boldsymbol{z}_{k+j+1} \in \mathcal{Z}, \tag{6d}$$

$$\boldsymbol{s} \geq \boldsymbol{0}, \tag{6e}$$

with $j = 0, 1\ldots, N_\mathrm{p} - 1$. The cost function J is defined in Sec. III-A1, and optional terminal costs J_term are discussed in Sec. IV. The slack cost weight $\boldsymbol{r}_\mathrm{s} = r_\mathrm{s}\,\mathbb{1}_{n_\mathrm{s}\times 1} \gg \boldsymbol{0}$ is chosen sufficiently large to enforce collision safety while securing problem feasibility. The transformed input set $\mathcal{W}$ is defined in Sec. III-A2, and the flat state set $\mathcal{Z}$, realizing lane keeping, speed limits, and obstacle avoidance, is discussed in Sec. III-A3 and Sec. III-B.

1) Cost Function: The cost function J is defined as

$$J = \sum_{j=0}^{N_\mathrm{p}-1} \left(\boldsymbol{e}_{k+j+1}^\mathrm{T}\,\boldsymbol{Q}\,\boldsymbol{e}_{k+j+1} + \boldsymbol{v}_{k+j}^\mathrm{T}\boldsymbol{R}\,\boldsymbol{v}_{k+j}\right), \tag{7}$$

with the tracking error $\boldsymbol{e} = [e_s,\ e_l]^\mathrm{T}$ and tuning matrices $\boldsymbol{Q} = \mathrm{diag}(q_s, q_l)$ and $\boldsymbol{R} = \mathrm{diag}(r_1, r_2)$.

Remark: Increasing the acceleration input weighting r_1 and optionally adding a penalty term for the acceleration jerk $\dot{a}_x$ to the cost function (7) yields a more conservative, fuel-efficient driving behavior (not focused on here).

The model representation in Frenet coordinates allows the decoupling of lateral and longitudinal control and therefore the essentially independent weighting of the longitudinal resp. lateral cost functionals and virtual inputs (4). The lateral error reads

$$e_{l,k+j+1} := l_{\mathrm{ref},k+j+1} - l_{k+j+1}\,, \tag{8}$$

with $\boldsymbol{l}_{\mathrm{ref},k} = \left[l_{\mathrm{ref},k}, l_{\mathrm{ref},k+1}, \ldots, l_{\mathrm{ref},k+N_\mathrm{p}}\right]$ usually set to zero (compare assumption A1). The longitudinal error is chosen to realize one of several typical control modes. We show two typical modes, namely velocity tracking and time-gap tracking:

a) Velocity tracking: Tracking of a desired reference velocity signal $\boldsymbol{v}_{\mathrm{ref},k} = \left[v_{\mathrm{ref},k}, v_{\mathrm{ref},k+1}, \ldots, v_{\mathrm{ref},k+N_\mathrm{p}}\right]$ is achieved by defining the longitudinal position error as

$$e_{s,k+j+1} := \left(\sum_{i=0}^{j} T_\mathrm{s}\,v_{\mathrm{ref},k+i}\right) - \left(s_{k+j+1} - s_k\right)\,. \tag{9}$$

To limit lateral vehicle accelerations a_n on curved roads, $\boldsymbol{v}_{\mathrm{ref},k+i}$ is reduced depending on the local reference path curvature $\kappa_\mathrm{ref}(s)$ by

$$v_{\mathrm{ref},k+i} = \begin{cases} \min\left(v_\mathrm{ref}, \sqrt{\dfrac{a_{n,\max}}{\left|\kappa_\mathrm{ref}(s^*_{k-1+i})\right|}}\right) & \text{if } \kappa_\mathrm{ref}(s^*_{k-1+i}) \neq 0 \\ v_\mathrm{ref} & \text{otherwise}\,, \end{cases} \tag{10}$$

utilizing the flat state trajectory of the last time step $\boldsymbol{Z}_{k-1}^*$. Alternatively, the lateral accelerations can be limited by implementing corresponding (soft) constraints. As demonstrated in [14], the kinematic single-track model describes vehicle motion accurately up to a maximal lateral acceleration of about $a_{n,\max} \approx 4\,\mathrm{m/s^2}$ in dry road conditions.

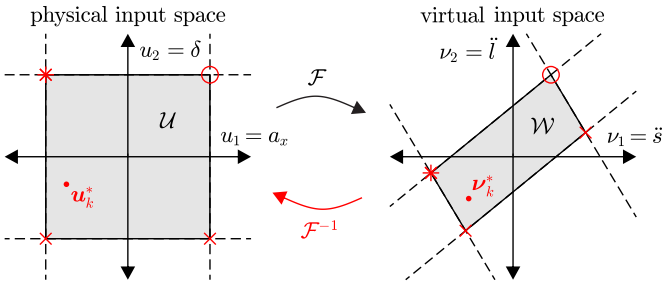

Fig. 3. The input set $\mathcal{U}$ in physical coordinates is mapped to the virtual input space while the obtained optimal virtual inputs $\boldsymbol{\mathcal{V}}_k^*$ are transformed back to physical inputs $\boldsymbol{U}_k^*$, here visualized for a curved ref. path and sample k.

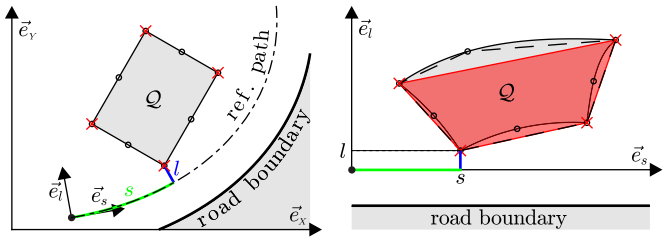

Fig. 4. Obstacle $\mathcal{Q}$ with $n_\mathrm{e} = 4$ edges represented in global Cartesian and Frenet coordinates incl. applied convex hull approximation in red, respectively. Increasing the number of mapped vertices (circles) results in a finer approximation, but increases the face count and thus computational load.

b) Time-gap tracking: Direct access to s and $\dot{s}$ enables the straightforward implementation of time-gap tracking policies that aim to track an inter-vehicle distance of $h \cdot v$, where h is a chosen fixed time span and v the ego velocity. With $v \approx \dot{s}$ the longitudinal error reads

$$e_{s,k+j+1} := \underbrace{\sum_{i=0}^{j+1} (h\,\dot{s}_{k+i}) + s_0}_{\text{time-gap}} - \underbrace{\left(s_{k+j+1}^{\text{pre}} - s_{k+j+1} - L_\text{ego}\right)}_{\Delta s_{k+j+1}} . \tag{11}$$

Therein, s_0 represents the desired standstill distance and Δs the distance to the predecessor vehicle with L_ego being the length of the ego vehicle [6]. The (rear end) position of the predecessor projected on the ego vehicle's reference path is denoted as s^pre. The parameters h, s_0 are defined on the tactical level, including the selection of the predecessor vehicle to be followed. One method to realize an overtaking maneuver, for example, is to drop the time-gap tracking objective (11) and switch to the velocity tracking mode (9). Tracking a higher reference velocity than the predecessor and the realized OA capability automatically induces a safe overtaking maneuver.

Remark: In this work, the actual selection of the control mode is specified externally by a decision module (out of scope here). Directly including tactical decisions in the OCP formulation is currently under development, compare Sec. VI.

2) Input Constraints: The input set in physical coordinates

$$\mathcal{U} = \left\{\boldsymbol{u} : \boldsymbol{G}_u \boldsymbol{u} \leq \boldsymbol{f}_u\right\} \tag{12}$$

typically comprises interval constraints on $\boldsymbol{u}$ of the form

$$\mathcal{U} = \{\boldsymbol{u} : \boldsymbol{u}_\text{min} \leq \boldsymbol{u} \leq \boldsymbol{u}_\text{max}\} . \tag{13}$$

These box constraints are then mapped to the virtual inputs corresponding to the flat coordinates using the state and input trajectories of the last time step's solution $\boldsymbol{X}_{k-1}^*$ and $\boldsymbol{U}_{k-1}^*$, respectively. This allows the formulation of constraints for the transformed inputs $\boldsymbol{\nu}$ (see Fig. 3) according to

$$\mathcal{W} = \left\{\boldsymbol{\nu} : \boldsymbol{G}_\nu \boldsymbol{\nu} \leq \boldsymbol{f}_\nu\right\} . \tag{14}$$

3) State Constraints: The problem formulation in flat Frenet coordinates and the resulting decoupling of the longitudinal and lateral dynamics allow the direct formulation of state constraints in the flat coordinate space. We formulate the longitudinal velocity constraint as $\dot{s} \leq v_\text{max}$ (soft), lane boundary constraints $l_{lhs} \leq l \leq l_{rhs}$ (soft), and $0 \leq \dot{s}$ (hard). Soft constraints are utilized to ensure the solvability of the optimization problem in all cases. Here, soft constraints are formed by re-defining a "hard" inequality constraint of the form $\boldsymbol{g}^\mathrm{T}\boldsymbol{u} \leq f$ to $\boldsymbol{g}^\mathrm{T}\boldsymbol{u} - s \leq f$, with $s \geq 0$ and high penalty cost on s, in which $\boldsymbol{u}$ and s are decision variables.

B. Obstacle Avoidance Constraints

The key idea of obstacle avoidance is to avoid any overlap of the ego vehicle's spatial footprint with any of the modeled "obstacle regions" at any time. Obstacles with time-varying postures and shapes are commonly called "dynamic obstacles". We represent such an obstacle i in terms of its *convex* obstacle region $\mathcal{O}_{i,k}$ at time index k based on its known or predicted position and rotation, as well as its shape inflated by chosen buffer distances. All these quantities are considered known in the scope of this work. Here, the ego vehicle shape is not considered explicitly but accounted for by increased obstacle size.

The main problem in the mathematical treatment of the resulting obstacle avoidance constraint $\boldsymbol{y}_{k+j} \notin \mathcal{O}_{i,k+j}$ is that this exclusion renders the problem landscape non-convex. By utilizing a suitable mixed-integer formulation with auxiliary binary decision variables, a reasonably efficient optimization problem is attained which can be solved to global optimality with modern solver algorithms. MIQP problems are generally NP-hard, so that no useful (polynomial) worst-case runtime bounds can be given [38]. However, it becomes evident that on today's hardware, the investigated OA-MIQP-MPC problems can be carefully formulated and solved on average in times similar to the required sampling times. This observation spawned the idea of the two-layer OA-MPC algorithm structure presented in this paper, as detailed in Sec. III-C.

While the MIQP formulation enables globally optimal maneuvers, the heuristic QP formulation ensures real-time capability with local optimality.

1) MIQP Formulation: The well-known Big-M method is utilized to formulate the constraints to prevent the ego vehicle 2D-position $\boldsymbol{y}$ from entering a convex obstacle region [19], [39]. Let a bounded convex polygonal obstacle region $\mathcal{Q}$ with n_e edges, compare Fig. 4, be given in the flat coordinates $\boldsymbol{y}$ as

$$\mathcal{Q} = \{\boldsymbol{y} : \boldsymbol{G}\boldsymbol{y} \leq \boldsymbol{f}\} \tag{15}$$

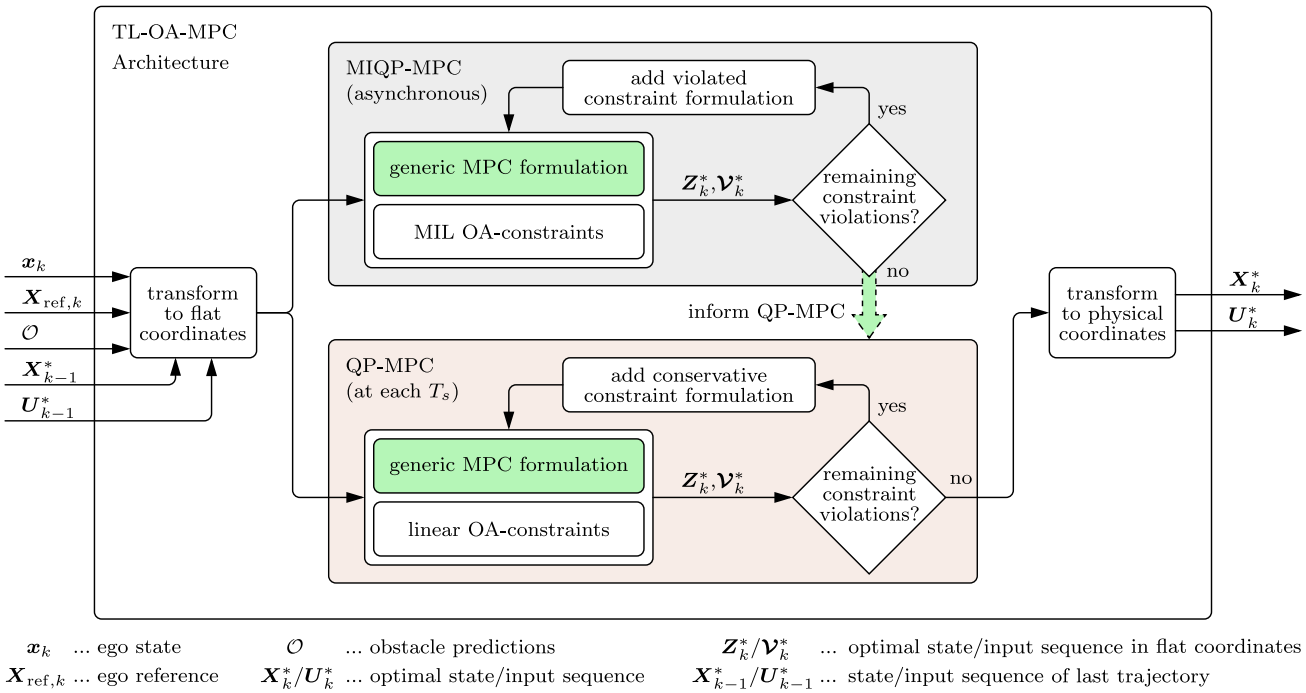

Fig. 5. Two-layer obstacle avoidance MPC (TL-OA-MPC) architecture. The QP-MPC is evaluated at every time step and its obstacle avoidance constraints are informed/updated asynchronously with the globally optimal MIQP-MPC solution.

with coefficients $\boldsymbol{G}_{n_e \times 2}$ and $\boldsymbol{f}_{n_e \times 1}$. The Big-M method is utilized to express the exclusion $\boldsymbol{y} \notin \mathcal{Q}$ by introducing a large constant scalar M (interpreted as a constraint relaxation distance), binary decision variables $\boldsymbol{\delta}_{n_e \times 1} \in \{0,1\}^{n_e}$ and the exclusion constraints

$$\boldsymbol{f} - \boldsymbol{G}\boldsymbol{y} + (\gamma - s)\,\mathbb{1} \leq M\,(\mathbb{1} - \boldsymbol{\delta})\,, \tag{16a}$$

$$s \geq 0\,, \tag{16b}$$

$$\mathbb{1}^{\mathrm{T}}\,\boldsymbol{\delta} \geq 1\,. \tag{16c}$$

These are realized in a soft (slacked) formulation. $\gamma, s \in \mathbb{R}$ represent a buffer distance and slack variable, respectively. The binary variables $\boldsymbol{\delta}$ taking value 1 indicate which of the edge constraints in (15) are *violated*, which has to hold true for at least one edge due to (16c) with $\mathbb{1}_{n_e \times 1} = \underbrace{[1,\ 1,\ \ldots,\ 1]^{\mathrm{T}}}_{n_e}$. It is evident that when formulating (16) for all known obstacles at all time steps in the problem (6), the globally optimal, feasible and collision-free ego trajectory is obtained if a collision-free solution exists and if assumptions A1-A6 in Sec. II-B are fulfilled. However, this approach produces a very large MIQP problem formulation with many binary decision variables, requiring high computational effort. Typical trajectories of course do not interfere with all obstacles at each time step, but only a few of these constraints are actually relevant. This sparsity is easily exploited by only formulating those OA constraints which are violated otherwise, and re-solving. This iterative approach leads to significantly faster total MIQP solver times.

To test a solution for OA constraint violations, and to quantify the severity of the violations, the following constraint residuals are determined for each obstacle i and each time step $k+j+1$, $j = 0, \ldots, N_p - 1$:

$$r_{i,k+j+1}^{\mathrm{viol}} := \max\left[-\max\left(\boldsymbol{G}_{i,k+j+1}\boldsymbol{y}_{k+j+1} - \boldsymbol{f}_{i,k+j+1}\right) + \gamma,\, 0\right]. \tag{17}$$

Positive values indicate violations. The corresponding OA constraints are added to the OCP formulation, and it is re-solved. This is repeated until no OA constraint violations occur. This sequence of sparsely populated MIQP problems eventually terminates with a collision-free solution if one exists. The OA constraints that had to be considered in the solution we denote as *relevant* obstacles / OA constraints.

2) QP Formulation: Solving the outlined MIQP problem yields the globally optimal solution but still requires considerable, variable computational effort and can take longer than the chosen sampling time. To achieve a real-time-capable control concept, we also solve a simplified problem in less (and bounded) computation time — asynchronously and in parallel to solving the MIQP-MPC problem: First, the OA constraint structure (given by the values of $\boldsymbol{\delta}$) is initialized from the last known MIQP-MPC-solution and updated as needed via tailored heuristics (see below). Then, the remaining convex QP-MPC problem is solved, which is typically accomplished well within each sampling period. If this solution shows further OA constraint violations, the constraints are updated via the heuristics again, and the QP-MPC is re-solved. This iteration procedure is done until no further OA constraint violations are predicted. This method yields a collision-free, feasible, and well-informed solution which is sub-optimal, but constructed typically quickly enough to meet real-time requirements. The key ideas of the constraint update heuristics are proposed as follows for two situations:

a) New relevant obstacle detected: If the current solution violates OA constraints with a new obstacle, its critical face is determined and formulated as a half-space inequality constraint. The critical face is that whose normal vector aligns best with the predicted ego vehicle orientation (velocity vector). This strategy yields a simplified, efficient, and safe OA constraint for the new obstacle.

b) Informing OA constraints from prior solution: If an obstacle that has already been considered causes new violations at different points in time, the corresponding constraints are derived from the previously known OA constraints utilizing their formulated faces: if earlier OA constraints are known, the closest earlier OA constraint is formulated at the current time step. Otherwise, i.e, if later OA constraints are known, the closest later OA constraint is formulated at the current time step. This heuristic algorithm proved useful to extend OA constraints to neighbouring time steps if necessary. These heuristic rules are summarized in Alg. 1. The resulting problem is then solved and the obstacle violations are checked again. If any new violations occur, the described procedure is repeated which results in the iterative solution of the obstacle avoidance problem.

Algorithm 1 QP-MPC Constraint Updating Heuristics

Input: obstacle violations

if violations of new obstacle detected $\mathcal{Q}_i \notin \mathcal{O}$ **then**
  identify earliest violation
  identify critical face
  formulate corresponding half-space constraint (16) for first violated sample
else if new violations for known obstacle $\mathcal{Q}_i \in \mathcal{O}$ **then**
  if preceding constraints formulated **then**
    assign to nearest preceding constraint
  else if no preceding constraints formulated **then**
    assign to nearest subsequent constraint
  end if
end if
return integer variables

c) Fail-safe controller variant: For the proposed iterative MIQP and the QP variants above, the number of iterations could be high. Conceptually, a specific *fail-safe* variant of the QP-MPC with a strongly limited number of maximum iterations is proposed here. It is constructed as a standard QP-MPC variant of (6), however with simplified OA constraints with at most 1 iteration per considered obstacle with the aim of achieving guaranteed, small computation times. We assume that such a controller can be formulated and proven to be stabilizing (without constraints) and feasible (with admissible OA constraints). To do so, J_{term} in (6a) can be chosen as Riccati terminal costs for the constant-velocity steady-state case. To limit the number of iterations (at the expense of increased conservativism), we propose to start the fail-safe QP-MPC-formulation without OA constraints. The OA violations are determined, and for each obstacle for which any violations occur, the critical-face half-space constraints are formulated for *all* time steps, for all obstacles with violations. Fig. 6 illustrates an exemplary, conservative OA constraint utilized to find the fail-safe QP-MPC. This approach leads to a maximum of $|\mathcal{Q}|$ (number of obstacles) iterations.

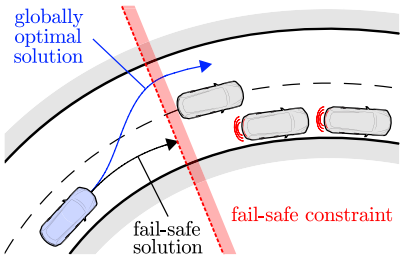

Fig. 6. Exemplary illustration of a conservative fail-safe constraint formulation.

3) Ego and Obstacle Shape Representations: When evaluating the collision constraints, the shape of the ego vehicle is approximated by a circle with radius r_{ego} centered at the rear axle. All spatial obstacles and constraints are inflated by this radius, so as to condense the ego shape to a point in the 2D plane, compare Fig. 2. Note that choosing the reference point at the front axle is possible by utilizing feed-forward control and adapting the mapping algorithms. However, for illustrative purposes, we stick to the reference point on the rear axle, noting that this choice may be suboptimal for high-precision obstacle avoidance when using reduced buffer distances. Solutions considering the ego vehicle's shape, however also by implying higher computational complexity are found in [24] and [25].

The obstacle shapes are approximated by convex polygons as depicted in red in Fig. 4. As a result of a possible curvature of the reference path, originally straight lines appear bent in the Frenet coordinate frame. This effect is compensated by inflating the hull approximations of the transformed obstacle shapes with the buffer distance γ similar to [40]. Alternatively, a finer convex shape approximation can be obtained by increasing the number of mapped vertices (compare dashed approximation in Fig. 4), albeit increasing the number of binary variables and therefore computational load.

C. Control Architecture

On the one hand, an MIQP-formulation of the OA problem is solved to global optimality. To ensure realtime-capability, a suboptimal convex QP-MPC variant in which the mixed-integer OA constraint formulation is simplified, informed by a known MIQP solution, and corrected by heuristics. As a result, the QP-MPC can be solved, fulfilling the real-time computation requirements in each time step, and its solution is also a valid but a suboptimal solution to the MIQP problem. The proposed two-layer obstacle avoidance control architecture comprising the MIQP- and QP-MPC is depicted in Fig. 5: First, the current state $\boldsymbol{x}_k$, reference $\boldsymbol{X}_{\text{ref},k}$, and current and predicted vertices of the obstacle regions $\mathcal{Q}_i \in \mathcal{O}$ are transformed from local to flat coordinates. Then, the input constraints (13) are mapped into flat coordinates utilizing the state and input trajectory of the last time step as described in Sec. III-A2.

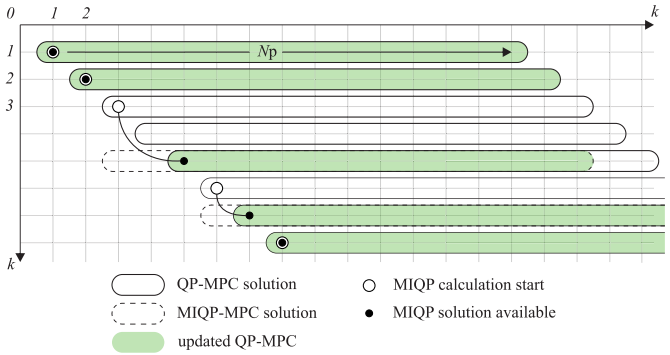

Fig. 7. The QP-MPC evaluates at every time step and updates its considered obstacles and obstacle faces as soon as a new MIQP-MPC solution (which needs to be shifted) is available.

The solution of the MIQP-MPC is obtained by iterative testing for collisions with obstacles: First, the optimization problem is solved without the consideration of any obstacles. Then the relevant obstacle shapes, that need to be avoided, are identified and successively added to the optimization problem by formulating the corresponding exclusion constraints. This is done by cycling through $\mathcal{O}$ and testing for collisions via the iterative evaluation of the respective interior constraints (15) for each obstacle $\mathcal{Q}_i$ at each time step $t_k, t_{k+1}, \ldots, t_{k+N_\mathrm{P}}$. Once added, the respective obstacle avoidance constraints (16) remain in the current OCP formulation to ensure convergence to a globally optimal solution and avoid infinite loops.

The real-time capable QP-MPC combines the conservative OA heuristic described in Sec. III-B2 together with the global information and binary variables provided by the MIQP-MPC solution to update its obstacle avoidance constraints accordingly. The QP problem is evaluated asynchronously in parallel to the MIQP problem and provides (depending on the MIQP update) the locally resp. globally optimal solution $\boldsymbol{\mathcal{V}}_k^*$ at each time step. As a consequence, the closed-loop dynamics are saved and we do not need to give up on optimality with this real-time capable two-layer OA control architecture.

Finally, the obtained optimal virtual input sequence $\boldsymbol{\mathcal{V}}_k^*$ is transformed back to physical inputs, and the first control action $\boldsymbol{U}_k^*(1) = \boldsymbol{u}_k^* = \left[a_{x,k}^*,\ \delta_k^*\right]^\mathrm{T}$ is applied to the controlled vehicle.

The data exchange between the two OA-MPCs is illustrated in Fig. 7: While the QP-MPC provides a locally optimal solution at every time step, the evaluation of the global optimal MIQP-MPC may take more time. E.g., the solution of the MIQP formulated at $k = 3$ is available at $k = 5$. Therefore, at $k = 5$ the relevant obstacle faces of the QP-MPC are updated with the ones selected by the MIQP solution by fixing the binary variables ($\boldsymbol{\delta}_\mathrm{MIQP} \rightarrow \boldsymbol{\delta}_\mathrm{QP}$). If the solution of the MIQP-MPC is obtained in real-time, it is basically directly applied, e.g., at $k \in \{1, 2, 8\}$ in Fig. 7.

D. Solution Selection Strategy to Retain Optimality

We discuss the idea of global optimality and feasibility with respect to the following verification strategy. Let

$$\mathcal{P}_\mathrm{verif}: \quad J \text{ as defined in (7)} \tag{18a}$$

$$\text{s.t. (6b)–(6d)}, \tag{18b}$$

$$\boldsymbol{y}_{k+j+1} \notin \mathcal{Q}_{i,k+j+1}, \tag{18c}$$

with $j = 0, 1, \ldots, N_\mathrm{p} - 1$ and $\mathcal{Q}_i \in \mathcal{O}$ define the *verification problem* with cost function J and the essential system, input, and output constraints (18b), and obstacle-avoidance constraints (18c) based on the *current* problem (and obstacle) information at time step k. Any solution $\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{Z}}$ defined compatibly with the generic OA OCP in Sec. III-A can hence be tested against the constraints of $\mathcal{P}_\mathrm{verif}$, (18b)–(18c), and its cost J expressed according to (18a). Hence, if a solution fulfills the constraints (18b)–(18c) it is admissible and avoids any collisions under the assumptions that the obstacle predictions are correct. The solution's performance is quantified by (18a).

A real-time-capable, safe, and efficient solution selection strategy (at each time step k) is proposed in Algorithm 2. The algorithm employs a two-stage selection process that utilizes the verification problem as a unified criterion to grade the solutions. In the first step, all solutions that fulfill the verification constraints (18b)–(18c) are selected and their verification objective costs (18a) are computed. The comparison of the verification objective cost then allows the identification of the best solution in the second algorithm step.

Algorithm 2 Collision-Free and Efficient OCP Solution Selection

Input: Failsafe solution $(\mathcal{V}, \mathcal{Z})_\mathrm{fs}$ that fulfills (18b)–(18c)
Input: Available solution set $\mathcal{L} = \{l : (\mathcal{V}, \mathcal{Z})_l \text{ available}\}$

```
I ← {fs}
for l ∈ L do
    if (V, Z)_l fulfils (18b)–(18c) then
        I ← I ∪ {l}
        J_l ← (18a)                          ▷ calculate cost
    end if
end for
l* ← arg min_{l∈I} J_l                       ▷ select best solution
return v_k* of selected solution l*
```

IV. Control System Discussion

In this section, important aspects, features, and limitations of the proposed control architecture are discussed.

A. Real Time Capability

The motivation of the proposed two-layer control architecture lies in the lack of a (useful) upper bound on MIQP-MPC solution complexity. Even though the MIQP-MPC can often be solved in a reasonable time, its worst-case runtime, representing an NP-hard mixed-integer problem, cannot be guaranteed. Instead, the convex (fail-safe) QP-MPC setting is utilized to ensure control law computation in bounded time. Still, this QP-MPC-problem is iteratively refined: if predicted violations are detected, a conservative heuristics is utilized to add relevant obstacle constraints, and the QP-MPC-problem is solved again, until no violations are predicted anymore. The number of these iterations can be limited by the design of the constraint inclusion heuristics. It is noted here that this solution does not solve the MIQP-MPC problem optimally anymore, but

its solution is a feasible solution for the MIQP-MPC setting. The real-time capability of the fail-safe QP-MPC problem is guaranteed if

$$T_s > N_{\text{it,max}} \cdot t_{\text{solve,max}} \tag{19}$$

whereby $N_{\text{it,max}}$ is the upper bound of the number of iterations per time step, and $t_{\text{solve,max}}$ is the upper bound of the computation time of a single iteration of the QP-MPC solution.

B. Optimality

This simple strategy always provides collision safety (under the assumption that the fail-safe OA OCP solution is always feasible) and utilizes better (lower-cost) solutions whenever they are safe and available. It is evident that this strategy recovers globally optimal OA performance if a sequence of globally optimal solutions is available (i.e. if they are available with sufficiently small computational effort), but automatically falls back to lower-performance solutions if necessary. Also, this strategy is highly generic in that it allows to exploit any heuristic attempt to improve solution quality by including these in the set of available solutions $\mathcal{L}$.

C. Stability & Feasibility

The control problems (6) are designed to be always feasible because all state constraints (except the hard constraint ensuring $v_x \geq 0$) are formulated as soft constraints with slacks. Since the involved MPC problems are expressed in linear time-invariant coordinates, standard LTI-MPC stabilization arguments (terminal set constraint and Riccati terminal cost terms) are readily available [41] if needed to guarantee the closed-loop stability of the constrained problems under uncertainty, external disturbances [42], and suitable infinite-horizon regularity conditions. An approach to formulating the (optional) terminal costs J_{term} in (6a) is given in Appendix A. The feasibility of the hard-constrained control problem can fundamentally be destroyed by critical or malicious obstacles if a collision cannot be avoided by the ego vehicle's control authority. However, in these cases, the proposed soft-constrained problem formulation yields a solution that can be deemed as a sensible trade-off. For example, maximum braking would be employed to minimize collision penalty cost in an unavoidable head-on collision, hence also reducing collision severity in reality.

A comment on the key aspects to ensuring closed-loop stability and recursive feasibility can be sketched as follows: First, we note that always realizing the fail-safe solution is assumed to be Lyapunov stable in an appropriate sense with the cost function (18a) being a Lyapunov function for the verification problem under the fail-safe control law. This can be achieved by designing the fail-safe solution under reasonable feasibility conditions via standard MPC stabilization methods [41], i.e. formulating the fail-safe control problem as a local LTI MPC problem with nominal closed-loop stability guarantees with conservative OA constraints and with cost function (18a). Building on this foundation, if we realize a control solution with an even smaller cost (18a), for example, achieved by the MIQP-MPC with detailed OA constraints, this solution would be selected and realized as outlined in Alg. 2. Consequently, the cost under this control law is still a Lyapunov function of the problem (because it is bounded above by the fail-safe cost value), hence also ensuring closed-loop stability. We omit a rigorous proof here but refer to Sec. V for a co-simulation-based validation of closed-loop stability and performance.

D. Robustness of Collision Safety

The main motivation for formulating an obstacle avoidance MPC is to solve the vehicle control problem with collision safety requirements in the best-informed way possible. Two aspects need to be resolved in this regard:

(I) The fact that the formulated collision constraints only approximate the collision problem setting requires a suitable choice of (small) additional safety buffer distances. Using soft constraints (i.e., the possibility of marginal violation) requires sufficiently high penalty costs. Several major sources of relevant model errors are shape inaccuracies, disregarding the ego shape, the fact that the constraints are formulated only at the sample instants (and not in between), as well as model errors in input-output dynamics of the ego vehicle (such as uncertain braking reaction / dead times, or unmodeled drivetrain dynamics affecting the realization of demanded accelerations/decelerations). These require the choice of reasonable spatial safety buffers, realized by enlarging the modeled obstacles at all sides. These distances are adjusted via simulation studies.

(II) The case that obstacles do not move according to their available predictions, but rather perform unforeseen actions such as sudden braking requires specific safety distances to attain collision safety in this uncertain context. This could be realized by larger (virtual) obstacle shapes at the far end of the prediction horizon to depict the higher uncertainties associated to longer predictions, potentially informed by stochastic metrics derived from corresponding estimations. However, this extension is out of scope of this work. In this work, we assume to have access to deterministic (possibly pre-processed) obstacle predictions compare also Sec. IV-F5.

E. Limitations of the Frenet Frame Transformation

While the formulation in Frenet coordinates facilitates a convenient controller design procedure, some limitations of the coordinate transformation need to be considered, especially in the case of tightly curved reference paths.

1) Transformation Stability: $\mathcal{F}$ is unique if $|l| < |1/\kappa_{\text{ref}}|$ holds [17], [43]. While this is usually the case for four-legged intersections (compare Fig. 9), for tightly curved roads, a transformation curve that does not exhibit singularities in the feasible driving region is proposed in [44]. Further, $\mathcal{F}^{-1}$ shows a singularity for $\dot{s} = 0$, which can be dealt with by applying the methodology described in [35].

2) Transformation Continuity: It is known that a curvature-continuous trajectory in the Frenet frame may become curvature-discontinuous after being converted back into the global Cartesian frame if the used reference path is curvature-discontinuous [17]. Herein, this can lead to

abrupt changes in the steering angle solely caused by the transformation from the virtual inputs $\boldsymbol{\mathcal{V}}_k^*$ to the physical control inputs $\boldsymbol{U}_k^*$. This issue is solved by providing an alternative transformation curve with a minimized change of curvature similar to [20].

3) Transformation Distortion: As already discussed in Sec. III-B3 the obstacle shapes appear distorted in the Frenet frame for curved reference paths, compare Fig. 4. As illustrated therein, transforming additional points on the boundary of the obstacles allow to reduce the effect of these distortions. Very long obstacles, e.g., buses, can be split into two obstacle shapes to reduce distortions caused by tightly curved reference paths even further at the cost of additional computation effort (not shown in this work).

F. Further Limitations

Further limitations of the TL-OA-MPC architecture in its current form are summarized and discussed as follows.

1) Vehicle Model: The kinematic single-track model (and its flatness-based treatment) utilized here is valid in the low-acceleration driving regime, but it does not capture wheel slip, side slip, or drifting. In highly dynamic driving situations, such as fast cornering, however, a significant model error arises. The MPC feedback action compensates for this error partially, but tracking accuracy is reduced in high-acceleration situations. As a possible extension to our OA-MPC concept, the so-called *kinodynamic* vehicle model [45], which also admits a flat representation, could be incorporated instead. This would only increase complexity moderately (more states) but allows describing wheel slip in an over-actuated vehicle setting, thus extending the applicability of the concept to accurate high-performance, high-dynamic maneuvers. This, however, is out of the scope of this work, and model error tolerance is showcased in the co-simulation example (C) in Sec. V-D.

2) Deterministic MPC: The proposed concept, even though being a deterministic design, is formulated to address uncertainties at several levels: (i) soft (slack) formulations of the obstacle-avoidance constraints serve as a basic and effective robustification against marginal model and obstacle prediction uncertainties. (ii) The design weightings are chosen to balance nominal control performance vs. robustness through weight adjustments and by cross-validating with co-simulations.

3) Environmental Disturbances: The proposed control architecture does not consider uncertainties such as road inclination, aerodynamic drag, wind, slippery roads, different loads, and drivetrain parameter variations explicitly. These environmental disturbances are essentially offloaded to the subordinate controllers at a drivetrain level, which is aided by the interpretability of the flat parametrization (out of the scope of this work). However, besides robust and stochastic MPC methods, sophisticated and efficient approaches that directly address vehicle and environmental uncertainties are proposed in [46], [47], [48], and [49] for longitudinal platooning and off-road vehicle control while [50] combines stochastic MPC with Taguchi's robustness strategy to ensure both robustness and reliability.

4) Communication: Communication faults are not considered in this work since the high degree of robustness of MPC with respect to temporary communication drops has already been investigated by the authors in [6]. Also, the control architecture can utilize simplified proxy models for obstacle motion prediction (computed at the ego vehicle, thus independent of V2V communication).

5) Obstacle Predictions: The deterministic position predictions, received either via V2X communication or obtained by a prediction module, can be interpreted as the mean values of stochastic occupation probabilities and the obstacle shapes can be inflated in correlation to the variance. Uncertain predictions can be taken into account by, e.g., increasing the safety margins to the surrounding traffic participants and lane boundaries over the prediction horizon in relation to the confidence level of the assumed prediction modules [33] (out of scope of this work).

Further, the re-planning nature of receding horizon MPC allows changes in the perceived environment to be accounted for at each time instance, which makes the TL-OA-MPC architecture robust to prediction errors and uncertainty [33]. This is verified in Sec. V where the control concept is tested under several model errors.

G. Alternative Problem Formulation in Cartesian Coordinates

This work shows the generic MPC problem formulation in flat Frenet coordinates. However, an alternative formulation in flat Cartesian coordinates, as described in [20], is possible as well by choosing the flat outputs as $\boldsymbol{y} = \left[X,\ Y\right]^{\mathrm{T}}$ which yields the flat state and virtual input vector $\boldsymbol{z} = \left[X,\ v_x\cos\psi,\ Y,\ v_x\sin\psi\right]^{\mathrm{T}}$ and $\boldsymbol{v} = \left[\ddot{X},\ \ddot{Y}\right]^{\mathrm{T}}$, respectively. The main advantages of this formulation are (i) the existing analytical transformation with simple handling of $v_x = 0$ as described in [20], (ii) the exact mapping of obstacle shapes, and (iii) its applicability to unstructured environments (no reference paths needed). On the other hand, the formulation and individual weighting of decoupled lateral and longitudinal control goals, as shown in Sec. III-A1, is not possible and the implementation of lane boundary constraints proves cumbersome.

V. Co-Simulation Based Validation

The proposed control concept is validated by realistic co-simulation of typical highly dynamical intersection and highway scenarios utilizing the traffic simulator for autonomous driving research *CARLA* [32].

A. Co-Simulation Architecture

The vehicle dynamics, approximated for the control design by (5), are computed in CARLA with higher fidelity, which includes the simulation of (i) longitudinal tire slip, (ii) lateral tire slip, and (iii) drive train dynamics. The robustness of the proposed control architecture with respect to these modeling errors is shown by utilizing the co-simulation architecture depicted in Fig. 8.

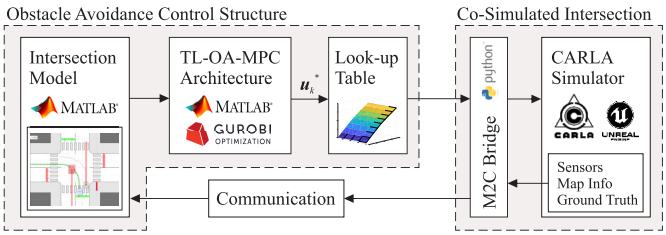

Fig. 8. Co-simulation architecture with the TL-OA-MPC architecture as described in Fig. 5 and a look-up table that maps $\boldsymbol{u}_k^*$ to normalized brake, throttle, and steering inputs. The intersection model [51] provides the information necessary for the optimization problems.

CARLA (Car Learning to Act) is an open-source hyper-realistic (traffic) simulator that enables the design and validation of autonomous driving systems [32], [52], [53], [54]. It uses Unreal Engine 4 to run the simulation and OpenDRIVE standard 1.4 to define roads and urban settings [55]. CARLA is designed as a server-client system and control over the simulation is granted through a Python API that is constantly developed by the CARLA research team. Vehicles are represented by the standard Unreal Engine 4 vehicle model (`PhysXVehicles`) with adjusted kinematic parameters for realism. Detailed vehicle dynamics are simulated by coupling the components engine, clutch, gears, differential, wheels, tires, suspensions, and chassis. In CARLA vehicles are controlled by the commands of steering, accelerating, and braking. An inbuilt basic controller that governs vehicle behavior, namely lane following, respecting traffic lights, speed limits, and decision-making at intersections, can be activated. Vehicles and pedestrians can detect and avoid each other. In this work, CARLA version 0.9.14 is used.

Interfacing between CARLA and MATLAB is established by a self-developed MATLAB2CARLA (M2C) bridge that utilizes the provided client API in combination with Python™ 3.8.10 to communicate with the CARLA server.

Communication: We assume perfect communication in the scope of this work.

Intersection Model: The multi-agent model architecture developed in [51] serves as a digital intersection twin by representing the traffic participants and their position predictions in the MATLAB environment and providing the necessary information for the optimization problems.

Optimization software: The OA-MPC optimization problems (6) are formulated and solved by MIP resp. QP utilizing the *untuned* commercial solver Gurobi® Optimizer version 10.0.0. The transformation and computation of the MPC control actions are carried out in MATLAB® R2023b. Although parameter tuning could significantly reduce calculation time, especially for MIP, as stated in [30], we use the untuned solver with default settings to allow for easy benchmarking.

Vehicle control interface: The control inputs $\boldsymbol{u}_k^*$ are mapped to normalized brake, throttle, and steering inputs via identified look-up tables as an alternative to a low-level PI controller. All vehicles are represented by electric cars to avoid model errors due to switching operations. The Tesla Model 3 vehicle model from the CARLA standard vehicle library is used with adjusted braking torque and engine/drivetrain damping with the clutch engaged, as listed in Table II, Appendix B.

The two-layer obstacle avoidance control architecture is tested in three traffic situations: Scenario (A) represents a left-turn maneuver in dense traffic, scenario (B) demonstrates the collision safety capability in an emergency braking maneuver due to a suddenly appearing obstacle, and the complex highway scenario (C) stress-tests the proposed control concept beyond its intended use case. Table II, Appendix B summarizes the simulation parameters applied. The co-simulated environment is sampled with 100 Hz while the vehicle controllers use a sampling time of $T_\mathrm{s} = 0.05\,\mathrm{s}$. We omit J_term in the following. The presented co-simulation scenarios are specifically designed to highlight the main features of the proposed control concept, namely (i) precise and efficient obstacle avoidance, (ii) in complex and critical traffic scenarios (iii) under severe model errors. This includes the configuration of vehicle spawn positions, maneuvers, reference velocities and paths, traffic light phase plans, and an aggressive TL-OA-MPC controller tuning.

B. Left Turn in Dense Traffic (A)

The TL-OA-MPC architecture is assessed with respect to several model errors in an unprotected left-turn maneuver at an urban intersection with dense traffic.

Fig. 9 depicts four selected time instances of the co-simulated scenario. All traffic participants spawn with v_ref, track their pre-assigned reference paths, and communicate their position predictions obtained under the kinematic single-track model assumption via V2V communication. The ego vehicle ① implements the TL-OA-MPC architecture while all other traffic participants track pre-assigned reference paths utilizing the Intelligent Driver Model [56] (a simple car-following model) for longitudinal control and the Stanley controller [57] (a geometrical path-tracking controller) for lateral control, comp [51]. The perspective of the ego vehicle, which enters the intersection from the South, is highlighted in Fig. 9 while Fig. 10 shows the corresponding time-series data. It tracks the reference path (shown in the first snapshot) while avoiding potential obstacles that are detected inside its detection cone. The ego vehicle evades the cars ② and ③ that perform sudden emergency braking maneuvers with a_min at $t = 1.25\,\mathrm{s}$ and $t = 3\,\mathrm{s}$, respectively. Finally, the ego vehicle exploits the gap between vehicle ④ and vehicles ⑤ and ⑥ to perform a left turn without hindering vehicles ⑤ and ⑥ maneuvers.

The proposed TL-OA-MPC architecture enables a collision-free and efficient maneuver in real time under several model errors, namely (i) the mismatch between the modeled and co-simulated vehicle kinematics, (ii) the non-modeled drive train dynamics, and (iii) inaccurate obstacle predictions that are based on the kinematic single-track model assumption. These discrepancies between the modeled and actual environment manifest in increased calculation times as observed in simulation studies (not shown here).

The calculation times of the MIQP- and QP-MPC problems are shown in Fig. 11. While the MIQP problem shows solver times up to 0.408 s, the majority of MIQP solutions are obtained in less than real-time. The QP-MPC always evaluates

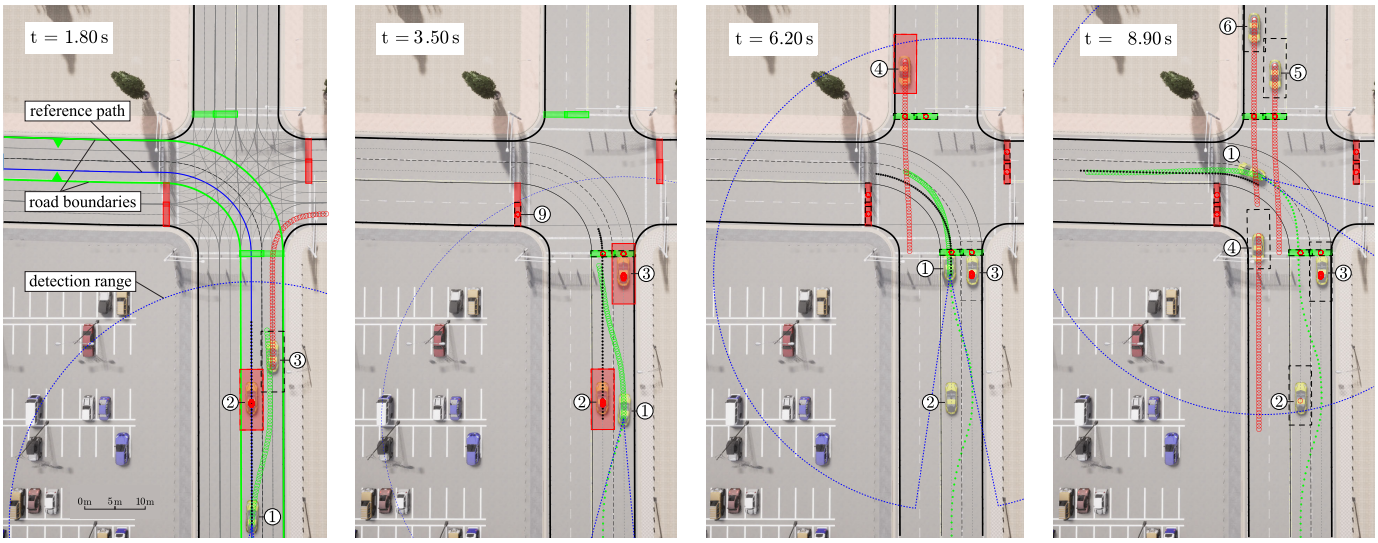

Fig. 9. Scenario A: The reference path and road boundary constraints of the ego vehicle ① are highlighted in the first snapshot while the predictions of detected traffic participants $\in \mathcal{O}$ inside the dashed blue detection range of the ego vehicle are visualized with red circles. The inflated shapes of traffic participants that need to be actively avoided are highlighted in red. The position prediction and history of the ego vehicle are visualized with green circles and green dots, respectively. Traffic lights are also detected and, depending on their phase plan predictions, considered as (static) obstacles.

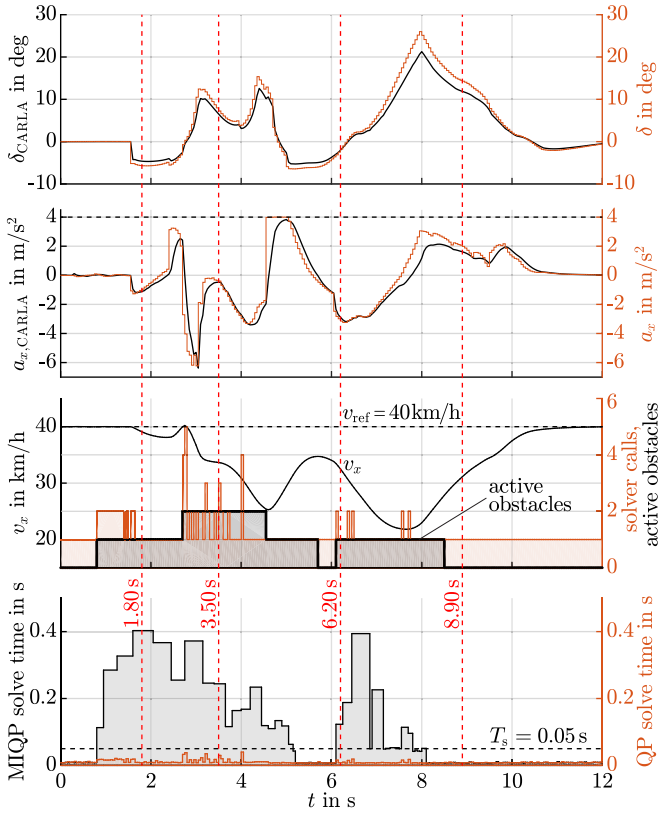

Fig. 10. Scenario A: TL-OA-MPC control signals (δ, a_x) and control inputs actually realized in CARLA (δ_{CARLA}, $a_{x,\text{CARLA}}$), longitudinal vehicle velocity v_x, QP-MPC solver calls (orange), active obstacles (gray), and computation times of the MIQP (gray) and QP (cumulated, orange) problems of the ego vehicle ① incl. snapshot times of Fig. 9.

in real-time with mean and max. calculation times of 0.010 s and 0.041 s, respectively. The application of the conservative fail-safe QP solution is not needed here.

Finally, we observe, that the kinematic single-track model (1) in combination with the reference shaping (10) is

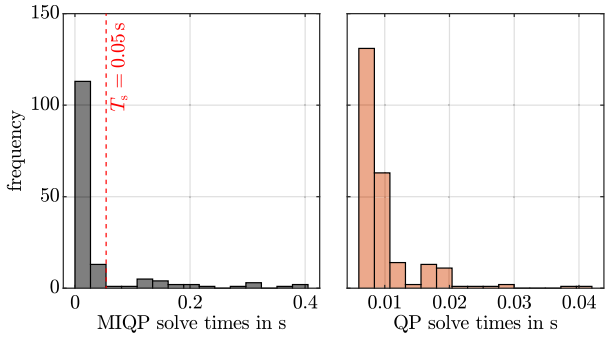

Fig. 11. Scenario A: Solver times of MIQP- and QP-MPC without overhead.

well-suited as a prediction model for model-based obstacle avoidance applications in low-speed urban intersection traffic scenarios.

Remark: The TL-OA-MPC architecture considers traffic lights (like ⑨ in Fig. 9) as immobile infrastructure agents that are assigned to specific routes and only affect approaching vehicles traveling on these reference paths. A relevant red traffic light is considered in the OCP formulation (6) via spatially fixed half-space constraints (16a). Instead of position predictions a so-called *Time to Activate (TTA)* encodes the signal phase plan and indicates when the traffic light next turns red and thus locks the respective half-space [51]. The TTA state, which is received via V2X communication, can accordingly be interpreted to optimally decide whether to plan a stopping maneuver or still pass the intersection.

Remark: While the traffic scenario shown appears rather simple, it is important to mention that the used multi-agent model architecture [51] is capable of handling arbitrary intersection topologies, obstacle shapes, and traffic participant types. Additionally, the co-simulation framework allows to include the simulation of V2X communication (out of scope here). We on purpose only displayed the relevant agents for clarity, Sec. V-D presents a more cluttered scenario consisting

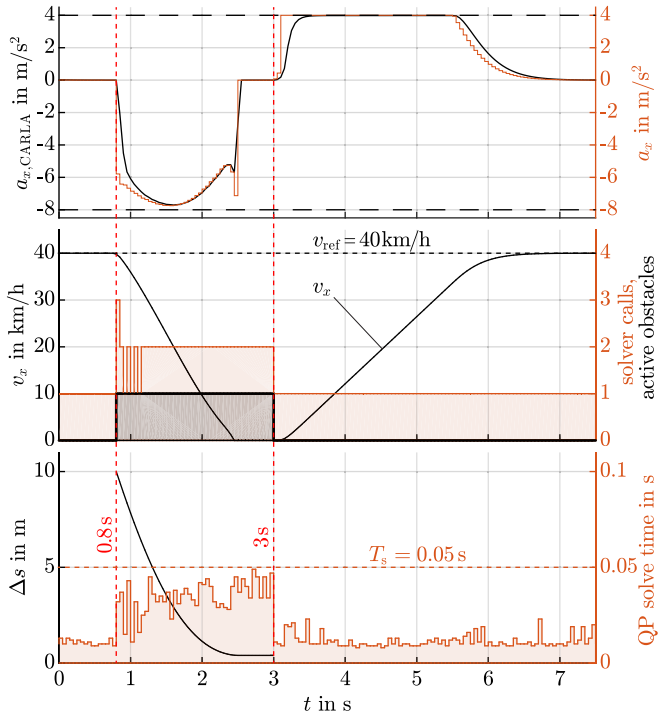

Fig. 12. Scenario B: TL-OA-MPC control signal (a_x) and control input actually realized in CARLA ($a_{x,\mathrm{CARLA}}$), longitudinal vehicle velocity v_x, QP-MPC solver calls (orange), active obstacles (gray), and cumulated computation times of the QP-MPC problem of the ego vehicle incl. longitudinal distance to nearest obstacle Δs.

of a multitude of agents in a 5-lane highway setting. Only active obstacles contribute to increased MIQP calculation times. The scalability of the TL-OA-MPC architecture is discussed in Sec. V-D which presents a more dense traffic scenario consisting of a multitude of agents driving in a 5-lane highway setting.

C. Emergency Braking Due to Sudden Obstacle Detection (B)

The collision safety and real-time capability of the QP-MPC heuristic without any MIQP updates, are assessed for an emergency braking scenario due to a sudden, unforeseeable obstacle appearance in front of the ego vehicle.

The same controller parameters as in scenario (A) are used, with the difference, that MIQP-updates are deactivated. The corresponding time series incl. the longitudinal distance to the closest obstacle Δs are plotted in Fig. 12. The ego vehicle drives along a straight road with v_{ref}. At $t = 0.8\,\mathrm{s}$ an obstacle suddenly appears 10 m in front of the ego vehicle and forces it into an emergency braking maneuver with a maximum deceleration of $7.7\,\mathrm{m/s^2}$. The ego vehicle comes to a standstill at $t = 2.45\,\mathrm{s}$ with $\Delta s = 0.4\,\mathrm{m}$. The obstacle disappears at $t = 3\,\mathrm{s}$ and the ego vehicle continues to track v_{ref}.

The sudden appearance of an obstacle just in front of the ego vehicle is unlikely in reality (perception module, compare assumption A6) but is intended to demonstrate the robustness of the heuristic QP formulation which avoids any collisions and evaluates with a maximum solver time of 47.0 ms and a mean solver time of 17.8 ms, compare Fig. 13. Slightly

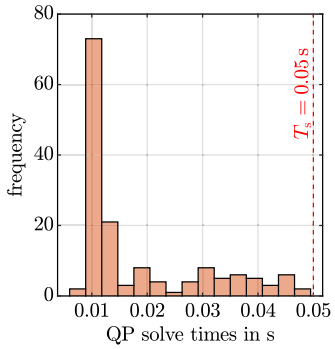

Fig. 13. Scenario B: Solver times of QP-MPC without overhead.

increased calculation times at standstill are related to the provisional handling of the singularity of the inverse Frenet transformation $\mathcal{F}^{-1}$ at $\dot{s} = 0$, compare Sec. IV-E1. Since the QP-MPC evaluates in less than real-time, the application of the conservative fail-safe QP solution is also not required in this maneuver.

D. Dense Highway Traffic Scenario (C)

The final co-simulation case studies the TL-OA-MPC architecture's capabilities and limitations in an overly dynamic "stress-test" maneuver: a complex high-speed 5-lane highway scenario with multiple interacting vehicles, high accelerations, as well as difficult obstacle avoidance decisions. The algorithm's tolerance to excessive model errors, its scalability, and its ability to recover global optimality are tested in a situation when the slip-free assumptions of the kinematic single-track vehicle model are significantly violated.

Different from Table II and due to the high reference velocity of the ego vehicle, the detection radius is increased to $r_{\mathrm{det}} = 120\,\mathrm{m}$ while the weight of the lateral position errors is decreased to $q_l = 0.3$ to facilitate maneuvers over all five available lanes. Eight scenario time instances are depicted in Fig. 14 while the corresponding time-series data is shown in Fig. 15. The ego vehicle ① tracks the middle lane (lane 3 of 5) with $v_{\mathrm{ref}} = 100\,\mathrm{km/h}$ which is dynamically adapted in the curved segments via (10) to limit lateral accelerations. The stationary vehicle convoy ③ and vehicles ④ and ⑤ in lanes 1 and 2 form an L-shaped dead-end. The ego vehicle plans to evade the stationary vehicle convoy ③ on the left side and recovers from the trap (local optimum) as soon as a MIQP-MPC update is available at $t = 1.70\,\mathrm{s}$. At $t = 3.25\,\mathrm{s}$ vehicle ② is added to the active obstacles and the ego vehicle adapts its speed slightly. When vehicles ② and ⑥ suddenly reduce their speed to 30 km/h and 12 km/h, respectively, the ego vehicle plans to pass them on the left two lanes. Due to severe model errors due to tire slip on the curved road, the vehicle shortly violates its soft-constrained a_{min} with a maximal deceleration of $9.45\,\mathrm{m/s^2}$. A similar phenomenon is observed around $t = 9.30\,\mathrm{s}$. From $t = 9.80\,\mathrm{s}$ on the ego vehicle increases its speed to reach v_{ref} and successfully avoids stationary and slower driving surrounding cars.

This scenario highlights the main advantage of the proposed TL-OA-MPC architecture: MIQP updates that recover global optimal maneuvering whenever possible. To emphasize this

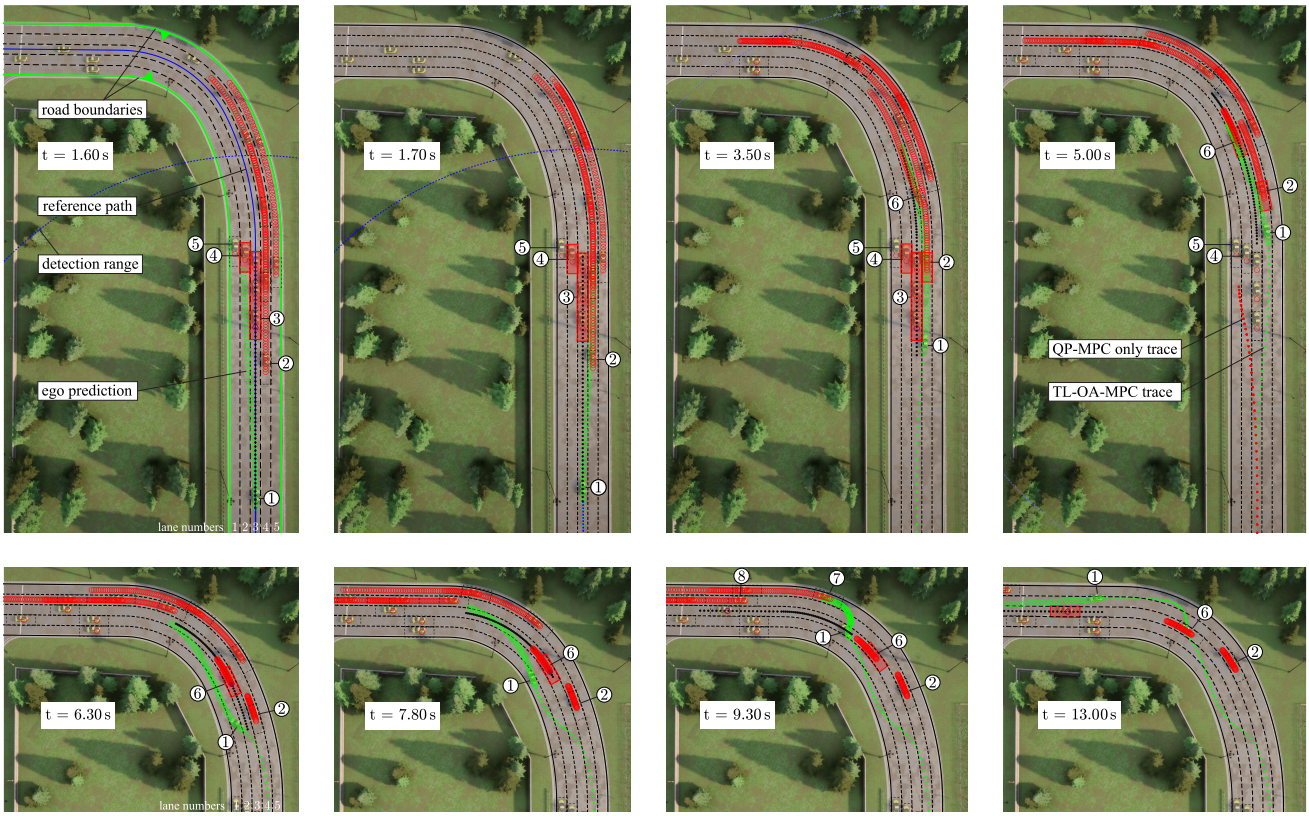

Fig. 14. Scenario C: The ego vehicle ① successfully recovers from the local optimum discovered at $t = 1.50\,\mathrm{s}$ via an MIQP-MPC update at $t = 1.50\,\mathrm{s}$. Without such an update the QP-MPC gets trapped in the local optimum and has to stop, position trajectory shown in red in snapshot 4 for $t = 5.00\,\mathrm{s}$.

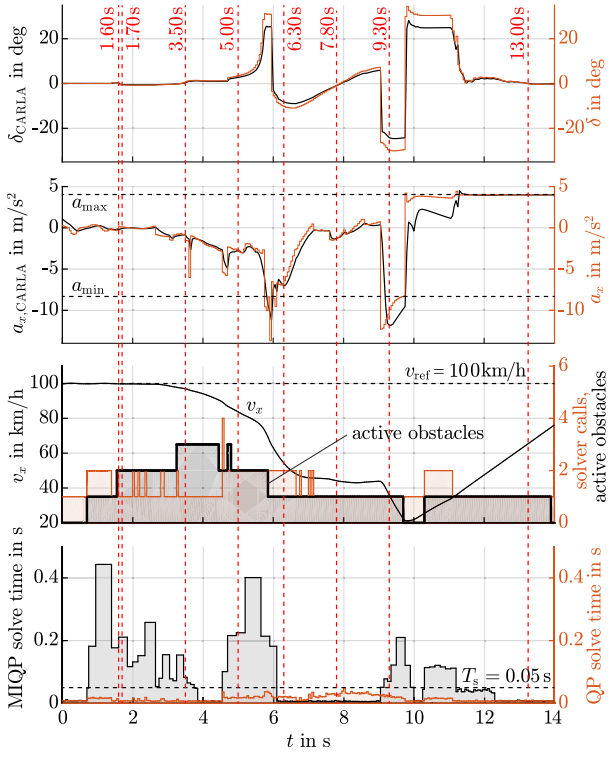

Fig. 15. Scenario C: TL-OA-MPC control signals (δ, a_x) and control inputs actually realized in CARLA (δ_{CARLA}, $a_{x,\mathrm{CARLA}}$), longitudinal vehicle velocity v_x, QP-MPC solver calls (orange), active obstacles (gray), and computation times of the MIQP (gray) and QP (cumulated, orange) problems of the ego vehicle ① incl. snapshot times of Fig. 14.

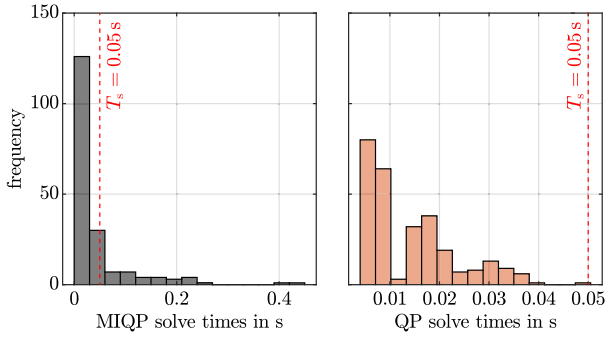

Fig. 16. Scenario C: Solver times of MIQP- and QP-MPC without overhead.

aspect, the position trajectory for the case that the QP-MPC does not receive any MIQP update after $t = 1.50\,\mathrm{s}$ is shown in red in snapshot 4 of Fig. 14. The QP-MPC is trapped and has to stop behind vehicles ④ and ⑤, which could have been averted with a single MIQP-MPC update.

The TL-OA-MPC architecture is able to safely and efficiently handle the complex high-speed traffic scenario with a curved layout and 16 dynamic and stationary obstacles under severe model errors. Up to three active obstacles are considered in the OCP formulations at the same time, the maximal peak MIQP-MPC solve time is 0.44 s, and the QP-MPC evaluates in real-time with a maximal solve time of 0.049 s, compare Fig. 16. Again, the application of the conservative fail-safe QP solution is not needed here. The control concept considers all detected traffic participants, but only *active* obstacles need to be actively avoided and therefore considered in the MIQP and QP OCP formulations. The maximal possible number of active obstacles can be

estimated for each road layout and speed/prediction horizon configuration. To favor real-time computation in extremely complex scenarios, an upper bound on the number of active obstacles could be defined similarly to [27], trading maneuver fidelity against real-time computation guarantees.

Using the kinematic single-track model, the proposed control concept is not intended for high-acceleration driving maneuvers, which lead to significant model errors, but this limitation could be overcome by utilizing the kinodynamic vehicle model [45] as discussed in Sec. IV-F.

VI. Conclusion

The main motivation of this work is to make the advantage of a MIP-based obstacle avoidance MPC formulation accessible to real-time applications. Therefore we developed a two-layer obstacle avoidance MPC architecture that enables collision-free and efficient automated driving in complex traffic situations in real-time. Asynchronously updating/informing a QP-MPC problem with globally optimal MIQP-based MPC solutions allows to retain global optimality while guaranteeing efficient maneuvers and collision safety in real time. Both controllers are based on a generic LTI-MPC formulation in flat Frenet coordinates, exploit V2X communication, and differ only in the implementation of the respective obstacle-avoidance constraints. A concluding validation with realistic vehicle dynamics co-simulations showed excellent performance of the proposed control architecture and its successful realization under severe model imperfections.

The two-layer obstacle avoidance MPC architecture is well-suited for parallel computing and parallel processor applications, which will be investigated in our future research. Additionally, we will focus on the incorporation of tactical decisions, e.g., lane changing and overtaking, in the MPC problem formulation. Another relevant research direction is the consideration of uncertain position predictions of the surrounding traffic participants and therefore dropping assumption A5.

Appendix A
Terminal Cost Term

The terminal cost term J_{term}, compare (6a), is derived by solving the discrete-time algebraic Riccati equation (DARE) for the following general quadratic terminal cost formulation

$$J_{\text{N}_\text{p}}(z_{\text{N}_\text{p}}) = \left(z_{\text{N}_\text{p}} - z^*_{\text{N}_\text{p}}\right)^{\text{T}} \boldsymbol{P} \left(z_{\text{N}_\text{p}} - z^*_{\text{N}_\text{p}}\right) \tag{20}$$

$$+ \left(\boldsymbol{v}_{\text{N}_\text{p}} - \boldsymbol{v}^*_{\text{N}_\text{p}}\right)^{\text{T}} \boldsymbol{R} \left(\boldsymbol{v}_{\text{N}_\text{p}} - \boldsymbol{v}^*_{\text{N}_\text{p}}\right). \tag{21}$$

which simplifies with $\boldsymbol{v}_{\text{N}_\text{p}} - \boldsymbol{v}^*_{\text{N}_\text{p}} = \boldsymbol{v}_{\text{N}_\text{p}}$ (desired virtual input $\boldsymbol{v}^*_{\text{N}_\text{p}} = \boldsymbol{0}$, no consideration of air drag and other resistances) and

$$\boldsymbol{e}_{z,\text{N}_\text{p}} = z_{\text{N}_\text{p}} - z^*_{\text{N}_\text{p}} = \begin{bmatrix} e_s \\ \dot{s} - \dot{s}^* \\ e_l \\ \dot{l} - \dot{l}^* \end{bmatrix}_{\text{N}_\text{p}} = \begin{bmatrix} e_s \\ \dot{s} - v_{\text{ref}} \\ e_l \\ \dot{l} \end{bmatrix}_{\text{N}_\text{p}} \tag{22}$$

to

$$J_{\text{N}_\text{p}}(z_{\text{N}_\text{p}}) = \boldsymbol{e}^{\text{T}}_{z,\text{N}_\text{p}} \boldsymbol{P}\, \boldsymbol{e}_{z,\text{N}_\text{p}} + \boldsymbol{v}^{\text{T}}_{\text{N}_\text{p}} \boldsymbol{R}\, \boldsymbol{v}_{\text{N}_\text{p}}. \tag{23}$$

The desired terminal states are chosen to $\dot{s}^* = v_{\text{ref}}$ and $\dot{l}^* = 0$. The terminal cost weighting matrix $\boldsymbol{P}$ is obtained by solving the DARE

$$\boldsymbol{P} = \boldsymbol{A}^{\text{T}} \boldsymbol{P} \boldsymbol{A} - \left(\boldsymbol{A}^{\text{T}} \boldsymbol{P} \boldsymbol{B}\right) \left(\boldsymbol{R} + \boldsymbol{B}^{\text{T}} \boldsymbol{P} \boldsymbol{B}\right)^{-1} \left(\boldsymbol{B}^{\text{T}} \boldsymbol{P} \boldsymbol{A}\right) + \boldsymbol{Q} \tag{24}$$

online at each time step, whereby $\boldsymbol{P}$, $\boldsymbol{Q}$, and $\boldsymbol{R}$ are symmetric positive definite matrices [58]. Finally, the terminal cost term reads

$$J_{\text{term}} = \sum_{N_\text{p}+1}^{\infty} \boldsymbol{e}_k^{\text{T}} \boldsymbol{Q} \boldsymbol{e}_k = \boldsymbol{e}^{\text{T}}_{z,\text{N}_\text{p}} \boldsymbol{P}\, \boldsymbol{e}_{z,\text{N}_\text{p}}. \tag{25}$$

Note that J_{term} was not used in the co-simulation scenarios shown in Sec. V and that in this formulation, we assume a free reference path without any additional interference with obstacles outside the detection radius/cone of the ego vehicle. Further assumptions are $v_{\text{ref},k+N\text{p}} = v_{\text{ref}} = \text{const.}$ for $k \geq 0$.

Appendix B
Co-Simulation Parameters

The co-simulation and control parameters used in this work are listed in Table II. The control parameters have been selected manually here.

TABLE II
Control and Simulation Parameters

Global Co-Simulation			
$T_{\text{s,cos}}$	0.01	s	sampling time co-sim (CARLA)
T_s	0.05	s	sampling time control
N_p	60	samples	prediction horizon
r_{det}	40	m	detection radius
v_{ref}	40	km/h	reference velocity
a_{max}	4	m/s^2	maximum acceleration
a_{min}	-8	m/s^2	maximum deceleration
δ_{max}	± 30	°	steering angle saturation
L_{wb}	2.8	m	wheel base
Tesla Model 3 (original → adapted value)			
700 →	2000	N m	maximum brake torque per tire
2 →	0.15	/	damping rate for zero throttle
TL-OA-MPC Architecture (implemented on vehicle ①)			
l_{ref}	0	m	lateral offset to reference path
$a_{n,\text{max}}$	4	m/s^2	maximum lateral acceleration
q_s, q_l	$0.5, 2$	/	output weighting
r_1, r_2	$0.2, 0.2$	/	input weighting
r_s	10^{12}	/	slack weight
M	100	/	Big M
γ	1	m	soft constr. dist. from obstacle
r_{ego}	2	m	radius of ego shape approximation
n_e	4	edges	number of mapped obstacle edges
$\mathcal{Q}_{i,\text{infl}}$	10×3.8	m	inflated shape measures

References

[1] J. Guo, U. Kurup, and M. Shah, "Is it safe to drive? An overview of factors, metrics, and datasets for driveability assessment in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 8, pp. 3135–3151, Aug. 2020.

[2] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, May 2020.

[3] D. Shen, Y. Chen, and L. Li, "State-feedback switching linear parameter varying control for vehicle path following under uncertainty and external disturbances," in *Proc. IEEE 25th Int. Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2022, pp. 3125–3132.

[4] Z. Han et al., "An efficient spatial–temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 1797–1814, Feb. 2024.

[5] B. Li et al., "Fast trajectory planning for AGV in the presence of moving obstacles: A combination of 3-dim A* search and QCQP," in *Proc. 33rd Chin. Control Decis. Conf. (CCDC)*, May 2021, pp. 7549–7554.

[6] A. L. Gratzer, S. Thormann, A. Schirrer, and S. Jakubek, "String stable and collision-safe model predictive platoon control," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 19358–19373, Oct. 2022.

[7] P. Falcone, F. Borrelli, J. Asgari, H. E. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *IEEE Trans. Control Syst. Technol.*, vol. 15, no. 3, pp. 566–580, May 2007.

[8] M. Fliess, J. Lévine, P. Martin, and P. Rouchon, "Flatness and defect of non-linear systems: Introductory theory and examples," *Int. J. Control*, vol. 61, no. 6, pp. 1327–1361, Jun. 1995.

[9] Z. Wang, J. Zha, and J. Wang, "Flatness-based model predictive control for autonomous vehicle trajectory tracking," in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 4146–4151.

[10] S. Fuchshumer, K. Schlacher, and T. Rittenschober, "Nonlinear vehicle dynamics control—A flatness based approach," in *Proc. 44th IEEE Conf. Decis. Control*, Dec. 2005, pp. 6492–6497.

[11] R. M. Murray and S. S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Trans. Autom. Control*, vol. 38, no. 5, pp. 700–716, May 1993.

[12] M. Wang, Z. Wang, J. Talbot, J. Christian Gerdes, and M. Schwager, "Game theoretic planning for self-driving cars in competitive scenarios," in *Robotics: Science and Systems*, vol. 37, no. 4. San Francisco, CA, USA: Robotics: Science and Systems Foundation, Jun. 2021, pp. 1313–1325.

[13] J.-H. Pauls, M. Boxheimer, and C. Stiller, "Real-time cooperative motion planning using efficient model predictive contouring control," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2022, pp. 1495–1503.

[14] D. Kloeser, T. Schoels, T. Sartor, A. Zanelli, G. Prison, and M. Diehl, "NMPC for racing using a singularity-free path-parametric model with obstacle avoidance," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14324–14329, 2020.

[15] P. Dini and S. Saponara, "Processor-in-the-loop validation of a gradient descent-based model predictive control for assisted driving and obstacles avoidance applications," *IEEE Access*, vol. 10, pp. 67958–67975, 2022.

[16] C. Sun, Q. Li, B. Li, and L. Li, "A successive linearization in feasible set algorithm for vehicle motion planning in unstructured and low-speed scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 4, pp. 3724–3736, Apr. 2022.

[17] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous driving on curvy roads without reliance on frenet frame: A Cartesian-based trajectory planning method," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15729–15741, Sep. 2022.

[18] P. Polack, F. Altché, B. d'Andréa-Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jun. 2017, pp. 812–818.

[19] F. Janeček, M. Klaučo, M. Kalúz, and M. Kvasnica, "OPTIPLAN: A MATLAB toolbox for model predictive control with obstacle avoidance," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 531–536, Jul. 2017.

[20] A. L. Gratzer, M. M. Broger, A. Schirrer, and S. Jakubek, "Flatness-based mixed-integer obstacle avoidance MPC for collision-safe automated urban driving," in *Proc. 9th Int. Conf. Control, Decis. Inf. Technol. (CoDIT)*, Jul. 2023, pp. 1844–1849.

[21] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annu. Rev. Control*, vol. 51, pp. 65–87, Oct. 2021.

[22] T. Weiskircher, Q. Wang, and B. Ayalew, "Predictive guidance and control framework for (semi-)autonomous vehicles in public traffic," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 6, pp. 2034–2046, Nov. 2017.

[23] J. V. Frasch et al., "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. Eur. Control Conf. (ECC)*, Jul. 2013, pp. 4136–4141.

[24] M. Brown and J. C. Gerdes, "Coordinating tire forces to avoid obstacles using nonlinear model predictive control," *IEEE Trans. Intell. Vehicles*, vol. 5, no. 1, pp. 21–31, Mar. 2020.

[25] Z. Wang, G. Li, H. Jiang, Q. Chen, and H. Zhang, "Collision-free navigation of autonomous vehicles using convex quadratic programming-based model predictive control," *IEEE/ASME Trans. Mechatronics*, vol. 23, no. 3, pp. 1103–1113, Jun. 2018.

[26] F. Gao, Y. Han, S. Eben Li, S. Xu, and D. Dang, "Accurate pseudospectral optimization of nonlinear model predictive control for high-performance motion planning," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 2, pp. 1034–1045, Feb. 2023.

[27] R. Quirynen, S. Safaoui, and S. Di Cairano, "Real-time mixed-integer quadratic programming for vehicle decision making and motion planning," 2023, *arXiv:2308.10069*.

[28] B. Li and Y. Zhang, "Fast trajectory planning in Cartesian rather than frenet frame: A precise solution for autonomous driving in complex urban scenarios," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 17065–17070, 2020.

[29] D. Bertsimas and B. Stellato, "Online Mixed-Integer optimization in milliseconds," *INFORMS J. Comput.*, vol. 34, no. 4, pp. 2229–2248, Jul. 2022.

[30] *Gurobi Optimizer Reference Manual*, Gurobi Optimization LLC, Beaverton, OR, USA, 2021.

[31] R. Quirynen and S. Di Cairano, "Tailored presolve techniques in branch-and-bound method for fast mixed-integer optimal control applications," *Optim. Control Appl. Methods*, vol. 44, no. 6, pp. 3139–3167, Nov. 2023.

[32] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. 1st Annu. Conf. Robot Learn.*, Nov. 2017, pp. 1–16.

[33] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, "Lane change maneuvers for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1087–1096, May 2017.

[34] J. Nilsson, M. Brännström, J. Fredriksson, and E. Coelingh, "Longitudinal and lateral control for automated yielding maneuvers," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 5, pp. 1404–1414, May 2016.

[35] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenet frame," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 987–993.

[36] S. Joos, R. Bruder, T. Specker, M. Bitzer, and K. Graichen, "Kinematic real-time trajectory planning with state and input constraints for the example of highly automated driving," in *Proc. 23rd Int. Conf. Syst. Theory, Control Comput. (ICSTCC)*, Oct. 2019, pp. 779–784.

[37] R. Reiter, A. Nurkanović, J. Frey, and M. Diehl, "Frenet-Cartesian model representations for automotive obstacle avoidance within nonlinear MPC," *Eur. J. Control*, vol. 74, Nov. 2023, Art. no. 100847.

[38] L. A. Wolsey, *Integer Programming*, 2nd ed. Hoboken, NJ, USA: Hoboken, NJ, USA: Wiley, 2020.

[39] B. Alrifaee, "Networked model predictive control for vehicle collision avoidance," Ph.D. dissertation, RTWH Aachen, Faculty Mech. Eng., Inst. Autom. Control, Aachen, Germany, 2017.

[40] B. Li et al., "Embodied footprints: A safety-guaranteed collision-avoidance model for numerical optimization-based trajectory planning," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 2046–2060, Feb. 2024.

[41] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, Jun. 2000.

[42] D. Shen, L. Li, Y. Chen, and F.-Y. Wang, "Cascade LPV control for automated vehicle trajectory tracking considering parametric uncertainty and varying velocity," in *Proc. Austral. New Zealand Control Conf. (ANZCC)*, Nov. 2022, pp. 176–181.

[43] H. Wang, J. Kearney, and K. Atkinson, "Robust and efficient computation of the closest point on a spline curve," in *Proc. 5th Int. Conf. Curves Surf.*, 2002, pp. 397–405.

[44] R. Reiter and M. Diehl, "Parameterization approach of the Frenet transformation for model predictive control of autonomous vehicles," in *Proc. Eur. Control Conf. (ECC)*, Jun. 2021, pp. 2414–2419.

[45] Z. Wang, J. Zha, and J. Wang, "Autonomous vehicle trajectory following: A flatness model predictive control approach with hardware-in-the-loop verification," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 9, pp. 5613–5623, Sep. 2021.

[46] D. Shen, J. Yin, X. Du, and L. Li, "Distributed nonlinear model predictive control for heterogeneous vehicle platoons under uncertainty," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 3596–3603.

[47] Z. Ju, H. Zhang, and Y. Tan, "Distributed stochastic model predictive control for heterogeneous vehicle platoons subject to modeling uncertainties," *IEEE Intell. Transp. Syst. Mag.*, vol. 14, no. 2, pp. 25–40, Mar. 2022.

[48] J. Yin, Z. Hu, Z. P. Mourelatos, D. Gorsich, A. Singh, and S. Tau, "Efficient reliability-based path planning of off-road autonomous ground vehicles through the coupling of surrogate modeling and RRT," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 12, pp. 15035–15050, Dec. 2023.

[49] S. Baldi, D. Liu, V. Jain, and W. Yu, "Establishing platoons of bidirectional cooperative vehicles with engine limits and uncertain dynamics," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 5, pp. 2679–2691, May 2021.

[50] J. Yin, D. Shen, X. Du, and L. Li, "Distributed stochastic model predictive control with Taguchi's robustness for vehicle platooning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 15967–15979, Sep. 2022.

[51] A. L. Gratzer, A. Schirrer, and S. Jakubek, "Agile multi-agent model architecture for intelligent intersection traffic simulation," *IFAC-PapersOnLine*, vol. 55, no. 27, pp. 89–95, 2022.

[52] R. Gutierrez, J. F. Arango, C. Gomez-Huelamo, L. M. Bergasa, R. Barea, and J. Araluce, "Validation method of a self-driving architecture for unexpected pedestrian scenario in CARLA simulator," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Jul. 2021, pp. 1144–1149.

[53] D. R. Morais and A. P. Aguiar, "Model predictive control for self driving cars: A case study using the simulator CARLA within a ROS framework," in *Proc. IEEE Int. Conf. Auto. Robot Syst. Competitions (ICARSC)*, Apr. 2022, pp. 124–129.

[54] Z. Zhou, C. Rother, and J. Chen, "Event-Triggered model predictive control for autonomous vehicle path tracking: Validation using CARLA simulator," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 6, pp. 3547–3555, Jun. 2023.

[55] P. Kaur, S. Taghavi, Z. Tian, and W. Shi, "A survey on simulators for testing self-driving cars," in *Proc. 4th Int. Conf. Connected Auto. Driving (MetroCAD)*, Apr. 2021, pp. 62–70.

[56] M. Treiber and A. Kesting, *Traffic Flow Dynamics Data, Models and Simulation*. Cham, Switzerland: Springer, 2013.

[57] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing," in *Proc. Amer. Control Conf.*, Jul. 2007, pp. 2296–2301.

[58] P. Lancaster and L. Rodman, *Algebraic Riccati Equations*. Oxford, U.K.: Clarendon Press, 1995.

Alexander L. Gratzer received the M.Sc. degree in mechanical engineering from TU Wien, Vienna, Austria, in 2019, where he is currently pursuing the Ph.D. degree. Since 2019, he has been a member of the Project Team, Institute of Mechanics and Mechatronics, TU Wien. His research interests include modeling, control, and optimization of complex systems, with a recent focus on automated driving and intelligent transportation systems (ITS).

Maximilian M. Broger received the M.Sc. degree in mechanical engineering from TU Wien, Vienna, Austria, in 2023. He has been a member of the Project Team, Institute of Mechanics and Mechatronics, TU Wien, for one year, where he is currently working in the field of automation technology.

Alexander Schirrer received the M.S., Ph.D., and Habilitation degrees in mechanical engineering from TU Wien, Vienna, Austria, in 2007, 2011, and 2018, respectively. Since 2011, he has been a Post-Doctoral Researcher and a Teacher of graduate-level lectures with the Institute of Mechanics and Mechatronics, TU Wien. His research interests include modeling, simulation, optimization, and control of complex and distributed-parameter systems.

Stefan Jakubek received the M.S., Ph.D., and Habilitation degrees in mechanical engineering from TU Wien, Vienna, Austria, in 1997, 2000, and 2007, respectively. From 2007 to 2009, he was the Head of Development for Hybrid Powertrain Calibration and Battery Testing Technology, AVL List GmbH, Graz, Austria. He is currently a Professor with the Institute of Mechanics and Mechatronics, TU Wien. His research interests include fault diagnosis and system identification.