Computation of Solution Spaces for Optimization-Based Trajectory Planning

Lukas Schäfer*, Stefanie Manzinger*, and Matthias Althoff

***Abstract*—The nonlinear vehicle dynamics and the non-convexity of collision avoidance constraints pose major challenges for optimization-based trajectory planning of automated vehicles. Current solutions are either tailored to specific traffic scenarios, simplify the vehicle dynamics, are computationally demanding, or may get stuck in local minima. This work presents a novel approach to address the aforementioned shortcomings by identifying collision-free driving corridors that represent spatio-temporal constraints for motion planning using set-based reachability analysis. We derive a suitable formulation of collision avoidance constraints from driving corridors that can be integrated into arbitrary nonlinear programs as well as (successive) convexification procedures. When combining our approach with existing motion planning methods based on continuous optimization, trajectories can be planned in arbitrary traffic situations in a computationally efficient way. We demonstrate the efficacy of our approach using scenarios from the CommonRoad benchmark suite.**

I. Introduction

VARIATIONAL methods for trajectory planning of automated vehicles have gained increasing interest over the past years. While discrete motion planning methods [1]–[4] are specifically suited for exploration, they may struggle to find solutions in cluttered environments due to discretization effects. In contrast, optimization-based motion planning approaches do not suffer from discretization effects as trajectories are optimized in continuous space. However, the nonlinear vehicle dynamics and the non-convexity of the set of feasible positions generally lead to a high computational burden. Some approaches require additional guidance through the solution space [5], e.g., in the form of driving corridors that represent spatio-temporal constraints.

A. Related Work

To address the above-mentioned challenges, a variety of problem reformulations is proposed in the literature. We categorize them subsequently by the fidelity of their applied vehicle model, their techniques to identify driving corridors, and their formulation of collision avoidance constraints.

Manuscript received Month Date, Year; revised Month Date, Year. This work was supported by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme SPP 1835 Cooperative Interacting Automobiles (grant number: AL 1185/4-2).

* These authors have contributed equally to this work.

The authors are with the Department of Informatics, Technical University of Munich, D-85748 Garching, Germany (e-mail: lukas.schaefer@tum.de, stefanie.manzinger@in.tum.de, althoff@in.tum.de)

1) Vehicle Dynamics: The fidelity of vehicle models for optimization-based trajectory planning ranges from very simple models [6]–[9], such as double-integrator dynamics, to rather complicated models [5], [10]–[13], such as dynamic single-track models. While the more complicated approaches demand excessive computational effort, less complicated approaches may fail in critical situations, because they neglect the non-holonomicity of the vehicle or decouple the longitudinal and lateral movement [6], [14]–[16].

2) Driving Corridors: Topological approaches for driving corridor identification have been studied for some time [12], [14], [17]–[25]. These approaches often exploit the concept of homotopy or homology to infer different maneuver variants [12], [22]–[25]. Basically, two trajectories are called homotopic if they can be continuously deformed into one another without intersecting any obstacle [26]. Topological approaches commonly decompose the collision-free regions in space-time into (convex) sub-regions: for each selected combination of sub-regions, an optimal trajectory can be planned. However, these approaches typically neglect the vehicle dynamics, and thus, cannot exclude non-drivable corridors prior to trajectory planning. Moreover, the number of distinct maneuver variants grows exponentially with the number of obstacles and some approaches are difficult to apply in dynamic environments [23], [24].

Another line of research finds driving corridors by inflating solutions from discrete motion planning methods, e.g., graph-based or sampling-based methods [27]–[33] or multi-agent partially observable Markov decision processes [34]. However, some approaches consider only static environments in their experiments [30]–[32] or only a discrete set of actions for computational tractability [34]. In general, these methods struggle in detecting narrow passageways in cluttered environments due to discretization effects.

In [35], a partitioning of the cluttered environment is obtained by means of convex lifting. The resulting partitioning of the space is utilized to obtain a reference path by graph search and to generate a driving corridor. The authors state that their algorithm can be extended to moving obstacles, but have not yet evaluated this.

Support vector machines (SVMs) have also been applied to identify driving corridors for path [36], [37] and trajectory planning [38], [39]. By assigning a passing side, i.e., either to the left or right, the SVM solver optimizes a separating surface to construct a collision-free driving corridor. However, deciding on the passing side is often the most crucial aspect.

3) Collision Avoidance Constraints: For optimization-based motion planning, obstacles are usually modeled as con-

vex shapes, e.g., polytopes [40]–[42] or ellipsoids [43]–[46], where polytopes allow one to more freely specify shapes. More general non-convex obstacles can be handled by applying (semi-) convex decomposition techniques beforehand [47].

Given a proper representation of the obstacles, mixed-integer programming is often proposed to handle non-convex collision avoidance constraints [6], [7], [48], [49]. While mixed-integer programs guarantee global optimality of feasible solutions, their computational complexity is high [7]. Another line of research uses nonlinear programming [40], [42], [44]; however, these approaches strongly rely on a suitable initial guess and are also computationally expensive.

In contrast, convex optimization problems can be efficiently solved to global optimality [50] and real-time capable solvers are available, e.g., [51], [52]. To employ convex optimization techniques, the non-convex optimization problem is typically approximated by a single [15], [16], [53]–[56] or a sequence of convex optimization problems [5], [46], [57]–[62]. This usually requires extracting a convex subset from the non-convex set of feasible positions, which we discuss next.

In [5], [11], [15], [16], [53], [63], [64], the set of admissible (lateral) positions is described by an interval of admissible deviations from a reference path. However, collision avoidance can only be guaranteed for minor deviations of the longitudinal position from the initial guess [5], [11], [53] or the vehicle dynamics is decoupled [15], [16], [63]. This issue can be circumvented by restricting the feasible positions to lie within an ellipsoid [33], [35], [65], [66]. However, ellipsoidal constraints can result in rather conservative under-approximations of the feasible set of positions. More flexible polyhedral under-approximations can be obtained by linearizing signed-distance functions [58] or potential fields [55].

The works in [42], [67], [68] propose smooth reformulations of collision avoidance constraints for polyhedral obstacles. If the collision avoidance constraints are differentiable, a convex approximation can be obtained by directly linearizing the collision avoidance constraints [31], [57], [60], [61], [69]. However, an unsuitable initial guess might cause convergence to an infeasible local minimum [43].

To obtain a larger feasible set as compared to direct-linearization techniques, the works [46], [70], [71] propose projecting the current state of the system onto the boundary of a convex keep-out zone followed by linearizing the constraint at the projection point. The approaches in [41], [72] compute polyhedral inner-approximations by growing a collision-free ellipsoid and computing the tangents to the ellipsoid where it coincides with the boundary of an obstacle. A related idea is proposed in [54], where the edges of the polyhedron are sampled and a limiting obstacle is assigned to each edge. However, all these procedures have in common that they require a collision-free initial guess.

B. Contributions

We identify collision-free driving corridors within the reachable set of an automated vehicle. This work significantly differs from our previous work on motion planning with reachable sets [63] and proposes the following innovations:

- our novel method for obtaining collision-free driving corridors enables combined longitudinal and lateral trajectory planning;
- our collision avoidance constraints are created so that arbitrary gradient- and Hessian-based solvers as well as (successive) convexification procedures can be used for trajectory planning.

Our proposed approach offers the following benefits:

- generic formulation of collision avoidance, i.e., our approach can be embedded in a wide range of different optimization-based motion planning methods, independently of the fidelity of the vehicle model;
- simplified initialization, i.e., the driving corridors facilitate the search for a suitable initial guess for nonlinear optimization problems;
- elimination of local minima induced by obstacles, i.e., the driving corridors guide the optimization-based motion planner through the collision-free solution space;
- consideration of a set of goal states, i.e., driving corridors can be constrained to end in a set of terminal states, e.g., a specific goal region or standstill in safe areas;
- applicability in arbitrary traffic scenarios, i.e., our approach is capable of handling arbitrarily cluttered scenarios involving static and dynamic obstacles;
- the computational effort of our approach typically improves with the criticality of the scenario [63], i.e., with shrinking solution space for trajectory planning.

The remainder of this paper is structured as follows: Sec. II introduces the problem statement and the solution concept. The computation of reachable sets is briefly reviewed in Sec. III. Sec. IV elaborates on the identification of driving corridors, followed by the derivation of collision avoidance constraints in Secs. V and VI. Numerical results are provided in Sec. VII and our conclusions are drawn in Sec. VIII.

II. Problem Statement and Solution Concept

To precisely formulate our problem statement and for subsequent derivations, we introduce our notation. For a set $\mathcal{S}$, let $\mathcal{S}^{\mathrm{o}}$ denotes its interior, $\partial\mathcal{S}$ its boundary, $\mathcal{S}^{\complement}$ its complement, $\mathrm{conv}(\mathcal{S})$ its convex hull, and $\mathrm{cl}(\mathcal{S})$ its closure. For a hyperplane $H^{=}_{(a,b)} := \{x | a^T x = b\}$, with $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$, let $H^{\leq}_{(a,b)}$ denote the corresponding halfspace $\{x | a^T x \leq b\}$; analogously, $H^{\geq}_{(a,b)} := \{x | a^T x \geq b\}$. If a set $\mathcal{S}$ is countable, its cardinality is denoted by $|\mathcal{S}|$. The set $\{r, r+1, \ldots, q\} \subset \mathbb{N}_0, 0 \leq r \leq q$, is denoted by $\mathcal{I}_{[r:q]}$. A sequence $\mathcal{W}$ with components $\mathcal{W}_i$, $i \in \mathcal{I}_{[r:q]}$, is denoted by $\mathcal{W} = (\mathcal{W}_i)_{i=r}^{q}$. We introduce $\square$ as the placeholder for a variable where the minimum and maximum admissible values are denoted by $\underline{\square}$ and $\overline{\square}$, respectively. We further introduce a local curvilinear coordinate frame as F^{L} aligned with a given reference path $\Gamma : \mathbb{R} \to \mathbb{R}^2$. In F^{L}, a global position $(s_x, s_y)^T$ is expressed in terms of the arc length s_ζ and the orthogonal deviation s_η from $\Gamma(s_\zeta)$.

A. Problem Statement

Let us introduce the compact set of admissible states $\mathcal{X} \subset \mathbb{R}^{n_x}$ and the set of admissible control inputs $\mathcal{U} \subset \mathbb{R}^{n_u}$ of an automated vehicle, whose motions are governed by

$$x_{k+1} = f(x_k, u_k), \tag{1}$$

where $x_k \in \mathcal{X}$ is the state, $u_k \in \mathcal{U}$ the input, and $k \in \mathbb{N}_0$ is the discrete time step corresponding to the time $t_k = k\Delta t$, $\Delta t \in \mathbb{R}^+$. The dynamics in (1) are formulated in F^{L} and $f(x_k, u_k)$ is continuously differentiable on $\mathcal{X}$. Possible state and input trajectories of the system are denoted by $x(\cdot)$ and $u(\cdot)$, respectively. Next, we define a few important sets as well as the occupancy and projection operations:

Definition 1 (Occupancy)**.** *The operator* $\mathrm{occ}(x)$ *relates the state* $x \in \mathcal{X}$ *to the set of points in the position domain occupied by the automated vehicle as* $\mathrm{occ} : \mathcal{X} \to \mathbb{P}(\mathbb{R}^2)$*, where* $\mathbb{P}(\mathbb{R}^2)$ *denotes the power set of* $\mathbb{R}^2$*.*

Definition 2 (Set of Forbidden States)**.** *Given the set* $\mathcal{O}_k \subset \mathbb{R}^2$ *of occupied positions of all obstacles (e.g., other cars and pedestrians) including the space outside of the road, the set of forbidden states of the automated vehicle at time step* k *is*

$$\mathcal{F}_k := \{x \,|\, \mathrm{occ}(x) \cap \mathcal{O}_k \neq \emptyset\}.$$

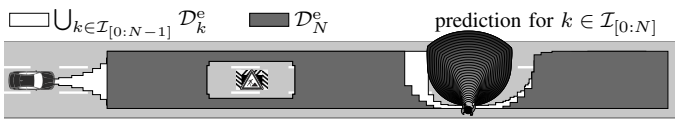
Definition 3 (One-Step Reachable Set)**.** *Let* $\mathcal{R}^{\mathrm{e}}_0 = \mathcal{X}_0$*, where* $\mathcal{X}_0$ *is the set of collision-free initial states of the automated vehicle including measurement uncertainties. The one-step reachable set* $\mathcal{R}^{\mathrm{e}}_{k+1}$ *is the set of all states that can be reached from the previous set of states* $\mathcal{R}^{\mathrm{e}}_k \subseteq \mathcal{X}$ *within one time step without intersecting* $\mathcal{F}_{k+1}$*:*

$$\mathcal{R}^{\mathrm{e}}_{k+1} := \Big\{ x_{k+1} \in \mathcal{X} \,\Big|\, \exists x_k \in \mathcal{R}^{\mathrm{e}}_k, \exists u_k \in \mathcal{U} : x_{k+1} = f(x_k, u_k) \wedge x_{k+1} \notin \mathcal{F}_{k+1} \Big\}.$$

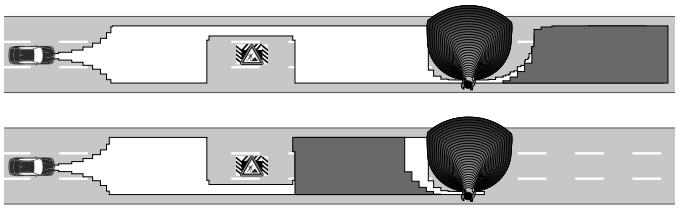
Definition 4 (Projection)**.** *The operator* $\mathrm{proj} : \mathcal{X} \to \mathbb{R}^2$ *maps the state* $x \in \mathcal{X}$ *to the* (s_ζ, s_η) *plane:* $\mathrm{proj}(x) := (s_\zeta, s_\eta)^T$*. Using the same notation, we project a set of states* $\mathcal{X}$*:* $\mathrm{proj}(\mathcal{X}) := \{\mathrm{proj}(x) \,|\, x \in \mathcal{X}\}$*. Similarly, we use* $\mathrm{proj}_\zeta : \mathcal{X} \to \mathbb{R}$ *and* $\mathrm{proj}_\eta : \mathcal{X} \to \mathbb{R}$ *to map a state* $x \in \mathcal{X}$ *to the longitudinal or lateral position domain, respectively.*

Definition 5 (Drivable Area)**.** *The drivable area* $\mathcal{D}^{\mathrm{e}}_k$ *at time step* k *is defined as* $\mathcal{D}^{\mathrm{e}}_k := \mathrm{proj}(\mathcal{R}^{\mathrm{e}}_k)$*.*

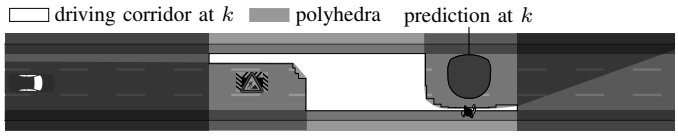
Our approach provides collision avoidance constraints for optimization-based trajectory planners. The input of our method is the current environment model that comprises the road network, the curvilinear coordinate system F^{L}, and all safety-relevant traffic participants, including their motion prediction. Our method is not tailored to any particular prediction method and only requires that $\mathcal{O}_k$ can be represented by the union of closed sets containing the future occupied positions of obstacles, including uncertainties. We aim to solve the following non-convex optimal control problem to plan the trajectory of the automated vehicle:



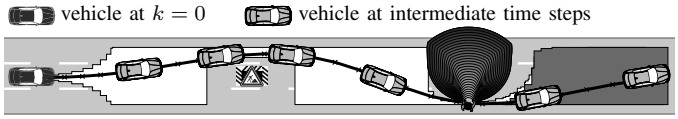
(a) Drivable area $\mathcal{D}^{\mathrm{e}}_k$ of the automated vehicle for consecutive time steps $k \in \mathcal{I}_{[0:N]}$. The drivable area at the final time step N is colored in gray.



(b) Two possible driving corridors are shown; for simplicity, we show the union of the driving corridors over time. The driving corridor at the final time step N is colored in gray.



(c) Approximation of the driving corridor complement with polyhedra depicted at a specific time step k.



(d) We extract collision avoidance constraints from the approximated driving corridor for the trajectory optimization.

Fig. 1: Illustration of the computation steps of our approach.

$$\min_{u(\cdot)} \sum_{k=0}^{N} J(x_k, u_k) \tag{2a}$$

such that

$$x_0 = \tilde{x}_0,\ x_N \in \mathcal{X}_{\mathrm{goal}}, \tag{2b}$$
$$\forall k \in \mathcal{I}_{[0:N-1]} : x_{k+1} = f(x_k, u_k), \tag{2c}$$
$$\forall k \in \mathcal{I}_{[0:N]} : \mathrm{proj}(x_k) \in \mathcal{D}^{\mathrm{e}}_k, \tag{2d}$$
$$g(x_k, u_k, k) \leq 0, \tag{2e}$$

where the cost function $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}$ is continuously differentiable. The (measured) initial state of the automated vehicle is denoted by $\tilde{x}_0$. Collision avoidance is encoded by the constraint (2d), i.e., the positions of the automated vehicle are limited to the drivable area. Additional constraints such as actuator constrains are summarized in the set of continuously differentiable, time-variant constraints $g : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{N}_0 \to \mathbb{R}^g$ in (2e).

B. Solution Concept

Our method determines a continuously differentiable approximation of the collision avoidance constraint (2d) in four steps (see Fig. 1):

1) We compute the drivable area (see Def. 5) of the automated vehicle in the current traffic scenario to explore

the dynamically reachable, collision-free solution space for trajectory planning (see Fig. 1a and Sec. III).
2) We extract dynamics-aware driving corridors from the drivable area (see Fig. 1b and Sec. IV).
3) We compute an approximation of the driving corridor complement using a fixed number of polyhedra (see Fig. 1c and Sec. V). This intermediate step allows us to use state-of-the-art methods for continuous trajectory optimization and to define the number of collision avoidance constraints in advance.
4) We derive collision avoidance constraints for the (non-) convex trajectory optimization from the approximation of the driving corridor complement (see Fig. 1d and Sec. VI).

These four steps are detailed in the subsequent sections.

III. Reachable Set Computation

Vehicle models used for motion planning usually possess nonlinear dynamics and a rather high-dimensional state space, which makes it difficult to calculate the reachable set [73]. We therefore aim to compute accurate approximations $\mathcal{R}$ of the exact reachable set $\mathcal{R}^{\mathrm{e}}$, i.e., $\mathcal{R} \approx \mathcal{R}^{\mathrm{e}}$, for computational efficiency. Among others, we realize this by approximating the vehicle dynamics by two second-order integrator models in the road-aligned coordinate system F^{L} [63]. The state $x = (s_\zeta, v_\zeta, s_\eta, v_\eta)^T$ and input $u = (a_\zeta, a_\eta)^T$ of the system are composed of the position s, velocity v, and acceleration a in the longitudinal ζ- and lateral η-directions, where both the velocity and acceleration are bounded:

$$\ddot{s}_\zeta(t) = a_\zeta(t), \quad \ddot{s}_\eta(t) = a_\eta(t), \tag{3a}$$

$$\underline{v}_\zeta \leq v_\zeta(t) \leq \overline{v}_\zeta, \quad \underline{v}_\eta \leq v_\eta(t) \leq \overline{v}_\eta, \tag{3b}$$

$$\underline{a}_\zeta \leq a_\zeta(t) \leq \overline{a}_\zeta, \quad \underline{a}_\eta \leq a_\eta(t) \leq \overline{a}_\eta. \tag{3c}$$

The approximation of the reachable set is represented as the union of base sets $\mathcal{R}_k^{(i)}$, $i \in \mathbb{N}_0$, i.e.,

$$\mathcal{R}_k^{\mathrm{e}} \approx \mathcal{R}_k \coloneqq \cup_i \mathcal{R}_k^{(i)}, \tag{4}$$

where a base set $\mathcal{R}_k^{(i)}$ is the Cartesian product of two convex polytopes in the (s_ζ, v_ζ) and (s_η, v_η) plane [74]. The projection of a base set $\mathrm{proj}(\mathcal{R}_k^{(i)})$ yields an axis-aligned rectangle $\mathcal{D}_k^{(i)}$. The union of $\mathcal{D}_k^{(i)}$ approximates the drivable area: $\mathcal{D}_k^{\mathrm{e}} \approx \mathcal{D}_k \coloneqq \cup_i \mathcal{D}_k^{(i)}$.

We use the algorithm presented in our previous works [63], [74] to compute the reachable set of the automated vehicle, which is briefly summarized below. To simplify the notation, we denote both the union and the collection of base sets $\mathcal{R}_k^{(i)}$ with $\mathcal{R}_k$; this is done analogously for the drivable area $\mathcal{D}_k$. The initial base set $\mathcal{R}_0^{(0)}$ results from the initial state $\tilde{x}_0$ to which the set of measurement uncertainties is added. The reachable set for consecutive time steps k is computed as follows:

1) Propagation: The polytopes of each base set $\mathcal{R}_k^{(i)}$ are propagated according to (3), which yields $\mathcal{R}_{k+1}^{\mathrm{prop}}$ and $\mathcal{D}_{k+1}^{\mathrm{prop}}$. At this stage, obstacles are not considered.

2) Removal of Forbidden States: We remove all colliding states from $\mathcal{R}_{k+1}^{\mathrm{prop}}$ to obtain $\mathcal{R}_{k+1}$. Since convex polytopes are not closed under set difference, we under-approximate $\mathcal{R}_{k+1}^{\mathrm{prop}} \setminus \mathcal{F}_{k+1}$ with several base sets $\mathcal{R}_{k+1}^{(j)}$.

To this end, we transform the set of obstacles $\mathcal{O}_k$ to the curvilinear coordinate frame F^{L} prior to the reachable set computation, and over-approximate the result with axis-aligned rectangles yielding the set $\tilde{\mathcal{O}}_k$. Efficient algorithms for axis-aligned rectangles benefit the remaining required steps:

a) Merging: Both sets $\mathcal{D}_{k+1}^{\mathrm{prop}}$ and $\tilde{\mathcal{O}}_{k+1}$ are merged using a sweep line algorithm [75], yielding rectilinear polygons.

b) Difference: By computing the difference between the rectilinear polygons of the propagated drivable area and the approximated obstacles, we obtain the collision-free reachable positions of the automated vehicle.

c) Partitioning: To cast the resulting collision-free reachable positions in the set representation defined in (4), we partition the set into rectangles $\mathcal{D}_{k+1}^{(j)}$ along the vertical η-direction, e.g., using a sweep line algorithm.

After we have obtained the collision-free drivable area $\mathcal{D}_{k+1}$, we determine the reachable velocities for $\mathcal{D}_{k+1}$ to obtain the reachable set $\mathcal{R}_{k+1}$ of the next time step. It is also possible to consider the shape of the automated vehicle for the removal of the forbidden states assuming that the automated vehicle is oriented along the reference path [63].

3) Update of Reachability Graph: As a consequence of the removal of forbidden states, multiple sets $\mathcal{R}_{k+1}^{(j)}$ can be reached from the same $\mathcal{R}_k^{(i)}$. For later use, we create a graph $\mathcal{G}_{\mathcal{R}}$, in which each node stores a base set $\mathcal{R}_k^{(i)}$ and its projection $\mathcal{D}_k^{(i)}$. An edge $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)})$ is added if and only if $\mathcal{R}_k^{(i)}$ can reach $\mathcal{R}_{k+1}^{(j)}$ in one time step.

IV. Identification of Driving Corridors for Coupled Dynamics

The obtained reachable sets are generally non-convex and often disconnected due to the presence of obstacles. To obtain better manageable sets, we decompose the reachable set into driving corridors:

Definition 6 (Driving Corridor)**.** *A driving corridor $\mathcal{C} \coloneqq (\mathcal{C}_k)_{k=0}^N$ is a sequence of sets $\mathcal{C}_k$ over time steps k that satisfy:*

- *C1) Reachability: $\mathcal{C}_k \subseteq \mathcal{R}_k$ and for each $\mathcal{R}_k^{(i)} \in \mathcal{C}_k$, there exists $\mathcal{R}_{k+1}^{(j)} \in \mathcal{C}_{k+1}$ such that $(\mathcal{R}_k^{(i)}, \mathcal{R}_{k+1}^{(j)}) \in \mathcal{G}_{\mathcal{R}}$;*
- *C2) Goal states: $\mathcal{C}_N \subseteq \mathcal{X}_{\mathrm{goal}}$;*
- *C3) Connectedness: $\mathrm{proj}(\mathcal{C}_k)$ is connected [76];*
- *C4) Vertical convexity: any non-empty intersection of $\mathrm{proj}(\mathcal{C}_k)$ with a vertical line is connected [77].*

A driving corridor may represent several maneuver options for each obstacle like yielding, passing, or following; nevertheless, the passing sides for obstacles are unique for all solutions in a driving corridor (see Fig. 1b). The different concepts of driving corridors and the related concept of homotopy are usually applied to obtain unique sequences of maneuvers [14], [22], [25], [34], [78]. In contrast, we have conceptualized driving corridors from an optimization point of view. By using driving corridors according to Def. 6 for trajectory planning, we are able to eliminate

1) the need for binary variables that encode collision avoidance constraints for multiple obstacles, since $\mathcal{C}_k$ is a single, connected set and already collision-free;
2) local minima due to obstacle constraints, since we assign a unique passing side for each obstacle prior to optimization (see Fig. 2).

Our approach to determine driving corridors $\mathcal{C}$ within the reachable set $\mathcal{R}$ is described next. For simplicity, we drop the projection operator when referring to $\mathcal{C}$ in the position domain.

1) Goal Region: We intersect the reachable set of the final time step with $\mathcal{X}_{\text{goal}}$, i.e., $\hat{\mathcal{R}}_N = \bigcup_i \mathcal{R}_N^{(i)} \cap \mathcal{X}_{\text{goal}}$. In case $\hat{\mathcal{R}}_N = \emptyset$, the desired goal states cannot be reached, which can be reported to a high-level planner. To remove states at earlier points in time that cannot reach $\mathcal{X}_{\text{goal}}$, we perform a graph search over $\mathcal{G}_{\mathcal{R}}$ backwards in time and remove base sets $\mathcal{R}_k^{(i)}$ that do not reach the set $\hat{\mathcal{R}}_N$.

2) Identification and Selection: We iterate over the drivable area backwards in time and decompose it into connected, vertically convex sets to extract driving corridors. As shown in Fig. 3b, the drivable area $\mathcal{D}_k$ is already vertically sliced (but not yet vertically convex nor connected) due to the removal of forbidden states. The decomposition into vertically convex sets is performed using the adjacency graph:

Definition 7 (Adjacency Graph—adapted from [23])**.** *The nodes of an adjacency graph $\mathcal{A}_k$ are sets $\mathcal{D}_k^{(i)} \in \mathcal{D}_k$ and an edge in $\mathcal{A}_k$ represents that $\mathcal{D}_k^{(i)}, \mathcal{D}_k^{(q)}$ share a border which is not just a vertex (e.g., $\mathcal{D}_k^{(0)}$ and $\mathcal{D}_k^{(1)}$ in Fig. 3b). The edges in $\mathcal{A}_k$ are directed in increasing longitudinal direction.*

By construction, $\mathcal{A}_k$ is a directed acyclic graph. Since there are no loops in $\mathcal{A}_k$, any path in $\mathcal{A}_k$ connecting a source with a sink node represents a connected, vertically convex set $\mathcal{V}_k^{(q)}$, as shown in Fig. 3b. For the identification and selection of driving corridors, we explore the driving corridor graph $\mathcal{G}_{\mathcal{C}}$ (see Fig. 3c):

Definition 8 (Driving Corridor Graph)**.** *The nodes of a driving corridor graph $\mathcal{G}_{\mathcal{C}}$ are vertically convex sets $\mathcal{V}_k^{(q)}$. An edge in $\mathcal{G}_{\mathcal{C}}$ represents that a set $\mathcal{D}_{k+1}^{(j)} \in \mathcal{V}_{k+1}^{(p)}$ is reachable from some $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ within one time step, i.e., $(\mathcal{D}_k^{(i)}, \mathcal{D}_{k+1}^{(j)}) \in \mathcal{G}_{\mathcal{R}}$.*

A path in $\mathcal{G}_{\mathcal{C}}$ represents a driving corridor $\mathcal{C}$. The exploration of $\mathcal{G}_{\mathcal{C}}$ can be performed using standard graph search algorithms like depth-first search or breadth-first search. The vertically convex sets $\mathcal{V}_N^{(q)}$ at the final time step N are sink nodes of $\mathcal{G}_{\mathcal{C}}$ and obtained through $\mathcal{A}_N$. The predecessors for a node $\mathcal{V}_k^{(q)}$ are obtained from the reachability graph $\mathcal{G}_{\mathcal{R}}$: we extract

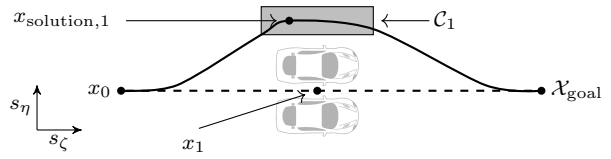
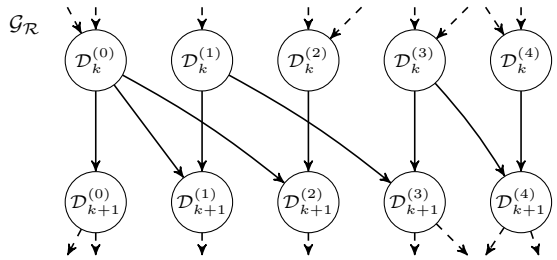
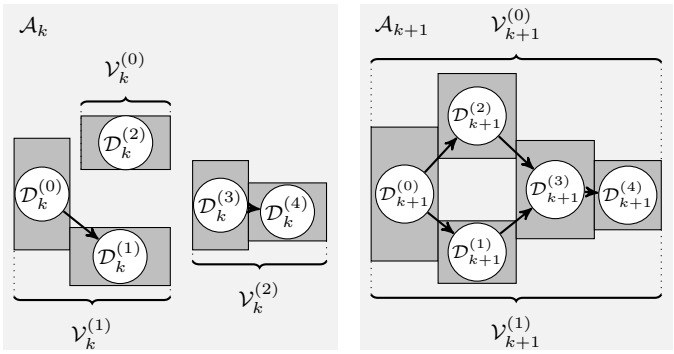


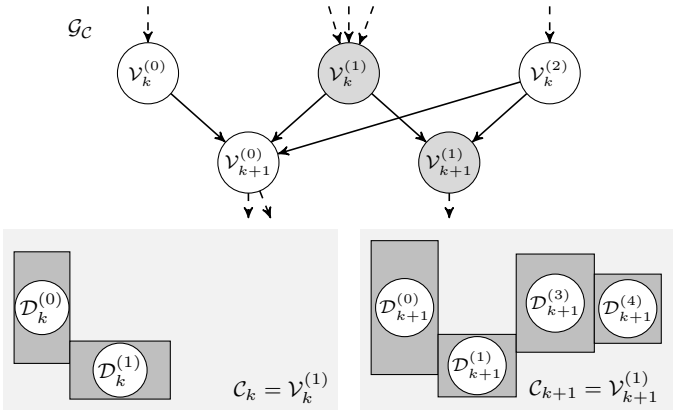
Fig. 2: Initialization (dashed line) in a (too) narrow gap close to an infeasible local minimum. Moving x_1 to either side will increase the violation of one of the constraints representing the obstacles. Using a driving corridor, a feasible solution (solid line) can be found since we rewrite the constraints as $x_1 \in \mathcal{C}_1$.



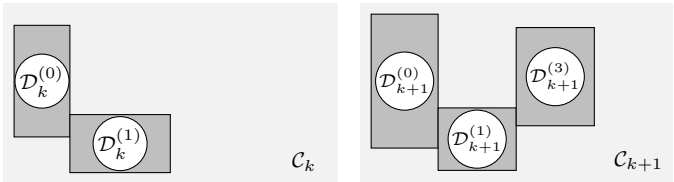
(a) Excerpt of the reachability graph $\mathcal{G}_{\mathcal{R}}$.



(b) A path in the adjacency graph $\mathcal{A}_k$ represents a connected, vertically convex set $\mathcal{V}_k^{(q)}$. $\mathcal{A}_k$ contains the sets $\mathcal{V}_k^{(0)} = \{\mathcal{D}_k^{(2)}\}$, $\mathcal{V}_k^{(1)} = \{\mathcal{D}_k^{(0)}, \mathcal{D}_k^{(1)}\}$, and $\mathcal{V}_k^{(2)} = \{\mathcal{D}_k^{(3)}, \mathcal{D}_k^{(4)}\}$. $\mathcal{A}_{k+1}$ contains the sets $\mathcal{V}_{k+1}^{(0)} = \{\mathcal{D}_{k+1}^{(0)}, \mathcal{D}_{k+1}^{(2)}, \mathcal{D}_{k+1}^{(3)}, \mathcal{D}_{k+1}^{(4)}\}$ and $\mathcal{V}_{k+1}^{(1)} = \{\mathcal{D}_{k+1}^{(0)}, \mathcal{D}_{k+1}^{(1)}, \mathcal{D}_{k+1}^{(3)}, \mathcal{D}_{k+1}^{(4)}\}$.



(c) Excerpt of the driving corridor graph $\mathcal{G}_{\mathcal{C}}$. A path in $\mathcal{G}_{\mathcal{C}}$ represents a driving corridor $\mathcal{C}$. In our example, the selected driving corridor is $\mathcal{C}_k = \mathcal{V}_k^{(1)}$ and $\mathcal{C}_{k+1} = \mathcal{V}_{k+1}^{(1)}$ at time steps k and $k+1$, respectively.



(d) After selecting a driving corridor, the forward search removes the set $\mathcal{D}_{k+1}^{(4)}$ from $\mathcal{C}_{k+1}$, since it is not reachable from $\mathcal{C}_k$ according to $\mathcal{G}_{\mathcal{R}}$.

Fig. 3: Identification of driving corridors; we only depict nodes and edges that are relevant for our example. We explore the driving corridor graph $\mathcal{G}_{\mathcal{C}}$ backwards in time to determine a driving corridor. After selecting a driving corridor, a forward search from time step $k = 0$ to N is performed to remove sets $\mathcal{D}_k^{(i)} \in \mathcal{C}_k$ that are no longer reachable.

the union $\hat{\mathcal{D}}_{k-1}$ of parents for all $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ from the reachability graph $\mathcal{G}_\mathcal{R}$ and group them to vertically convex sets $\mathcal{V}_{k-1}^{(j)}$ by identifying paths in $\mathcal{A}_{k-1}$. The exploration of $\mathcal{G}_\mathcal{C}$ can be terminated as soon as a first driving corridor is found. If more time is available, the exploration of $\mathcal{G}_\mathcal{C}$ can be continued to determine alternative driving corridors. It is also possible to assign weights $w_{j,q}$ to the edges $(\mathcal{V}_{k-1}^{(j)}, \mathcal{V}_k^{(q)})$ to guide the selection of a driving corridor when using informed search techniques.

At each time step, creating $\mathcal{A}_k$ takes $\mathcal{O}(|\mathcal{R}_k|^2)$. A path representing a set $\mathcal{V}_k^{(q)}$ in $\mathcal{A}_k$ can be computed within $\mathcal{O}(|\mathcal{R}_k| + |E_k|)$ [79], where $|E_k|$ denotes the number of edges in $\mathcal{A}_k$ that are at most $\frac{1}{2}|\mathcal{R}_k|(|\mathcal{R}_k| - 1)$. Let $r = \max_{k \in \mathcal{I}_{[0:N]}} |\mathcal{R}_k|$ and $e = \max_{k \in \mathcal{I}_{[0:N]}} |E_k|$. As we consider a time horizon of length N, the complexity of computing a first driving corridor is $\mathcal{O}(N(r+e))$ when using depth-first search.

3) Forward Search: Since we partition the predecessor sets $\hat{\mathcal{D}}_{k-1}$ of a set $\mathcal{V}_k^{(q)}$ again into vertically convex sets during the exploration of $\mathcal{G}_\mathcal{C}$, it may hold that certain sets $\mathcal{D}_k^{(i)} \in \mathcal{V}_k^{(q)}$ are not reachable when selecting a specific driving corridor in $\mathcal{G}_\mathcal{C}$. We therefore perform a forward search over the selected driving corridor: we iterate over all time steps starting from $k = 0$ and remove unreachable sets, see Fig. 3d. In the rare event that the driving corridor $\mathcal{C}_k$ becomes disconnected after the forward search due to approximation errors in the reachable set computation, we omit the corridor.

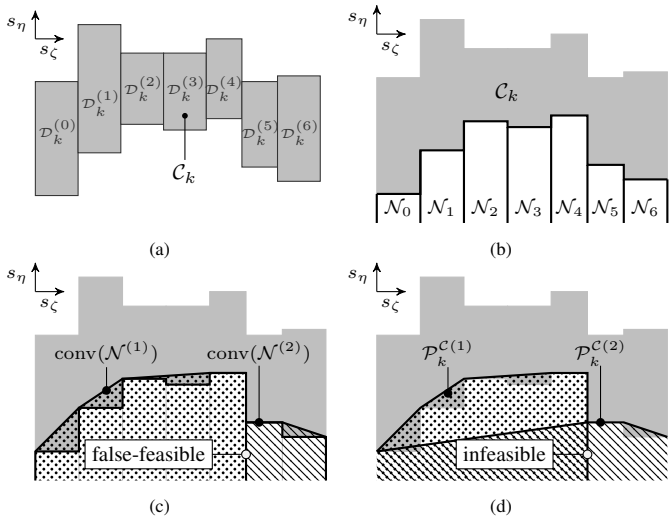


Fig. 4: (a) Reformulating $\mathrm{proj}(x_k) \in \mathcal{C}_k$ using OR statements, i.e., $\mathrm{proj}(x_k) \in \mathcal{D}_k^{(0)} \vee \mathrm{proj}(x_k) \in \mathcal{D}_k^{(1)} \vee \ldots$, requires binary variables. (b) We reformulate $\mathrm{proj}(x_k) \in \mathcal{C}_k$ without binary variables by covering the complement of the driving corridor with polyhedral keep-out zones. An exact cover of $\mathcal{C}_k^\complement$ can be obtained with polyhedra $\mathcal{N}_j$. (c) To limit the number of polyhedral keep-out zones, we partition $\mathcal{N}$ into subsequences $\mathcal{N}^{(1)}$ and $\mathcal{N}^{(2)}$. By computing the convex hull of the subsequences, we obtain polyhedral keep-out zones. However, this encoding of $\mathrm{proj}(x_k) \in \mathcal{C}_k$ is incorrect, since all positions in the relative interior of the intersection of adjacent polyhedra are not excluded from the solution space. (d) To prevent false-feasibles, we modify $\mathrm{conv}(\mathcal{N}^{(1)})$ and $\mathrm{conv}(\mathcal{N}^{(2)})$ to obtain $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$.

V. Polyhedral Approximation of the Driving Corridor Complement

Formulating collision avoidance as $\mathrm{proj}(x_k) \in \mathcal{C}_k$ is inherently difficult since the boundary of $\mathcal{C}_k$ is non-smooth and non-convex (see Fig. 4a). For example, $\mathrm{proj}(x_k) \in \mathcal{C}_k$ can be reformulated using disjunctive inequalities, where sets $\mathcal{D}_k^{(i)} \in \mathcal{C}_k$ are related by OR statements, i.e., $\mathrm{proj}(x_k) \in \mathcal{D}_k^{(0)} \vee \mathrm{proj}(x_k) \in \mathcal{D}_k^{(1)} \vee \ldots$ (see Fig. 4a), and thus, binary variables are required. As an alternative, an appropriate subset $\tilde{\mathcal{C}}_k \subset \mathcal{C}_k$ can be extracted, e.g., an ellipsoid within $\mathcal{C}_k$. However, most solutions for determining an appropriate set $\tilde{\mathcal{C}}_k$ require convex obstacles or convex decompositions of obstacles in the environment see, e.g., [41], [54], [70].

To exploit methods for smooth encodings of $\mathrm{proj}(x_k) \in \mathcal{C}_k$, see [42], [68], we approximate the complement $\mathcal{C}_k^\complement$ of the driving corridor $\mathcal{C}_k$ with polyhedral keep-out zones (see Fig. 4d). By limiting the number of polyhedral keep-out zones to $n_{\max,\mathrm{k}}$, our approach facilitates real-time capability of optimization-based motion planning. The resulting polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ follows as:

Definition 9 (Polyhedral Approximation of $\mathcal{C}_k$)**.** *We define the polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ as the intersection of the closures of the complement of all polyhedral keep-out zones $\mathcal{P}_k^{\mathcal{C}(l)} \in \mathcal{P}_k^\mathcal{C}$, $|\mathcal{P}_k^\mathcal{C}| = n_{\max,\mathrm{k}}$:*

$$\tilde{\mathcal{C}}_k := \bigcap_{l=1}^{n_{\max,\mathrm{k}}} \mathrm{cl}(\mathbb{R}^2 \setminus \mathcal{P}_k^{\mathcal{C}(l)}).$$

We say $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k$ is a proper encoding of the collision avoidance constraint $\mathrm{proj}(x_k) \in \mathcal{C}_k$ if $\tilde{\mathcal{C}}_k \subseteq \mathcal{C}_k$, i.e., $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k \implies \mathrm{proj}(x_k) \in \mathcal{C}_k$. Below, we give an overview of the computation of $\mathcal{P}_k^\mathcal{C}$.

A. Overview

Since the drivable area is computed in the road-aligned coordinate frame F^L, we compute the cover of $\mathcal{C}_k^\complement$ in the longitudinal and lateral direction separately. The polyhedra bounding the driving corridor in the longitudinal direction are immediately obtained as

$$\mathcal{P}_k^{\mathcal{C}(n_{\max,\mathrm{k}}-1)} := \{(s_{\zeta,k}, s_{\eta,k}) \in \mathbb{R}^2 \,|\, s_{\zeta,k} \geq \min\left(\mathrm{proj}_\zeta(\mathcal{C}_k)\right)\},$$
$$\mathcal{P}_k^{\mathcal{C}(n_{\max,\mathrm{k}})} := \{(s_{\zeta,k}, s_{\eta,k}) \in \mathbb{R}^2 \,|\, s_{\zeta,k} \leq \max\left(\mathrm{proj}_\zeta(\mathcal{C}_k)\right\}.$$

An exact cover of $\mathcal{C}_k^\complement$ in the lateral direction can be obtained using a sequence $\mathcal{N} := (\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_M)$ of interiorly disjoint, orthogonal polyhedra $\mathcal{N}_j$ (see Fig. 4b); for brevity, we omit the time-dependency in the notation for $\mathcal{N}$. In most cases, this leads to the number of polyhedra being different from $n_{\max,\mathrm{k}}$. We therefore partition $\mathcal{N}$ into $n_{\max,\mathrm{k}}-2$ subsequences $\mathcal{N}^{(l)} \subseteq \mathcal{N}$, from which we obtain the remaining polyhedral keep-out zones in the lateral direction.

From $\mathcal{N}^{(l)}$, a polyhedron can be computed using the convex hull see, e.g., $\mathrm{conv}(\mathcal{N}^{(1)})$ and $\mathrm{conv}(\mathcal{N}^{(2)})$ in Fig. 4c. Since we consider the generic formulation of inequality constraints $g(x_k, u_k, k) \leq 0$ and adjacent polyhedra are interiorly disjoint, the entire boundary of every $\mathrm{conv}(\mathcal{N}^{(l)})$ is considered as collision-free. However, every point in the relative interior [50, Sec. 2.1.3] of the intersection of adjacent polyhedra is outside of $\mathcal{C}_k$ (see Fig. 4c). To properly encode $\mathrm{proj}(x_k) \in \mathcal{C}_k$, we modify adjacent $\mathrm{conv}(\mathcal{N}^{(l)})$ so that the resulting polyhedra

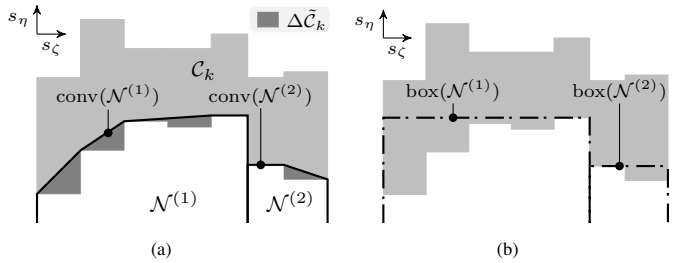


Fig. 5: (a) Approximation error $\Delta\tilde{\mathcal{C}}_k$. (b) We estimate the value of $\Delta\tilde{\mathcal{C}}_k$ by replacing the computation of $\mathrm{conv}(\mathcal{N}^{(l)})$ with $\mathrm{box}(\mathcal{N}^{(l)})$.

$\mathcal{P}_k^{\mathcal{C}(l)}$ overlap without further reducing the solution space compared to the union of $\mathrm{conv}(\mathcal{N}^{(l)})$ (see Fig. 4d). We have found empirically that the proposed approach facilitates convergence. Below, we discuss the partitioning of $\mathcal{N}$ into $n_{\max,\mathrm{k}}-2$ subsequences, followed by explaining the algorithm to obtain the polyhedra $\mathcal{P}_k^{\mathcal{C}(l)}$, $l \in \mathcal{I}_{[1:n_{\max,\mathrm{k}}-2]}$, for the lateral boundary.

B. Partitioning

We wish to partition the sequence $\mathcal{N}$ into $n_{\max,\mathrm{k}}-2$ disjoint subsequences $\mathcal{N}^{(l)} \subseteq \mathcal{N}$ such that the approximation error $\Delta\tilde{\mathcal{C}}_k$, i.e, the loss of solution space of $\tilde{\mathcal{C}}_k$ compared to $\mathcal{C}_k$, is reduced (see Fig. 5a):

$$\Delta\tilde{\mathcal{C}}_k \coloneqq \sum_{l=1}^{n_{\max,\mathrm{k}}-2} \mathrm{area}\left(\mathrm{conv}(\mathcal{N}^{(l)}) \setminus \mathcal{N}^{(l)}\right). \tag{5}$$

Minimizing (5) leads to a combinatorial optimization problem that can become difficult to solve with increasing cardinality of $\mathcal{N}$. For computational efficiency, we propose several simplifications:

1) We use a greedy algorithm that recursively partitions $\mathcal{N}$ into subsequences $\mathcal{N}^{(l)}$.
2) We replace the costly computation of the convex hull in (5) with the computation of the minimum bounding box $\mathrm{box}(\mathcal{N}^{(l)})$ (see Fig. 5b).
3) At each recursion, we continue partitioning the sequence $\mathcal{N}^{(l)}$ that reduces $\Delta\tilde{\mathcal{C}}_k$ the most. If this partition is inadmissible because too many new subsequences would have to be created, i.e., the overall number of sequences exceeds $n_{\max,\mathrm{k}}-2$, we always consider the next best one.
4) For each sequence $\mathcal{N}^{(l)}$, we consider only some of its possible partitions to evaluate the best partition. The heuristic used is described in Appendix A. This heuristic can be changed and is not the main focus of this paper.

Our algorithm can be extended to ensure the connectivity of $\tilde{\mathcal{C}}_k$ after computing the polyhedra from $\mathcal{N}^{(l)}$, $l \in \mathcal{I}_{[1:n_{\max,\mathrm{k}}-2]}$: a partition is only admissible if $\mathrm{box}(\mathcal{N}^{(l)})^{\mathrm{o}}$ is pairwise disjoint for all $\mathcal{N}^{(j)}, j \neq l$, which can be efficiently validated.

C. Computation of the Polyhedra for the Lateral Direction

After determining the subsequences $\mathcal{N}^{(l)}$ as described above, we compute their corresponding polyhedral keep-out zones $\mathcal{P}_k^{\mathcal{C}(l)}$. Let us therefore introduce the polyhedron $\tilde{\mathcal{P}}_k^{(l)} \supseteq \mathrm{conv}(\mathcal{N}^{(l)})$ that is obtained by removing both halfspaces defining the vertical edges of $\mathrm{conv}(\mathcal{N}^{(l)})$, e.g., compare $\mathrm{conv}(\mathcal{N}^{(2)})$ in Fig. 4c with $\tilde{\mathcal{P}}_k^{(2)}$ in Fig. 6a. By intersecting $\tilde{\mathcal{P}}_k^{(l)}$ with halfspaces $H_{(a_1,b_1)}^{\leq,\leftarrow}$ and $H_{(a_2,b_2)}^{\leq,\rightarrow}$ that bound $\tilde{\mathcal{P}}_k^{(l)}$ in forward and backward driving direction, respectively, we obtain $\mathcal{P}_k^{\mathcal{C}(l)}$:

$$\mathcal{P}_k^{\mathcal{C}(l)} = \tilde{\mathcal{P}}_k^{(l)} \cap H_{(a_1,b_1)}^{\leq,\leftarrow} \cap H_{(a_2,b_2)}^{\leq,\rightarrow}. \tag{6}$$

To ensure a proper encoding of $\mathrm{proj}(x_k) \in \mathcal{C}_k$ as discussed in Sec. V-A, the halfspaces are created such that (a) $\mathcal{P}_k^{\mathcal{C}(l)} \supseteq \mathrm{conv}(\mathcal{N}^{(l)})$ and (b) the relative interior of the intersection of adjacent $\mathrm{conv}(\mathcal{N}^{(l)})$ is excluded from the solution space (see Fig. 4c and Fig. 4d). Additionally, we choose the halfspaces so that (c) the solution space is not further reduced compared to the union of $\mathrm{conv}(\mathcal{N}^{(l)})$ (see Fig. 4c and Fig. 4d).

1) Algorithm: To obtain the halfspace $H_{(a_1,b_1)}^{\leq,\leftarrow}$ limiting $\tilde{\mathcal{P}}_k^{(l)}$ in backward driving direction, we apply the following approach (the computation of $H_{(a_2,b_2)}^{\leq,\rightarrow}$ works analogously, see Fig. 6c and Fig. 6d): we initialize $H_{(a_1,b_1)}^{\leq,\leftarrow}$ as $\mathbb{R}^2$ and determine the set $\mathcal{E}^{\leftarrow}$ that contains the vertices of all $\mathrm{conv}(\mathcal{N}^{(j)})$, $j \neq l$, that are encountered when traversing along $\partial\mathcal{C}_k$ in backward driving direction starting from $\mathcal{N}^{(l)}$ (see Fig. 6a). To satisfy the conditions (a) and (c), we search for a halfspace $H_{(a_1,b_1)}^{\leq,\leftarrow}$, where the corresponding hyperplane $H_{(a_1,b_1)}^{=}$ separates $\mathrm{conv}(\mathcal{N}^{(l)})$ and $\mathcal{E}^{\leftarrow}$. The first support point of $H_{(a_1,b_1)}^{=}$ is chosen as the most backward vertex of $\mathrm{conv}(\mathcal{N}^{(l)})$. The second support point is selected from $\mathcal{E}^{\leftarrow}$. To this end, we iterate over the vertices $y \in \mathcal{E}^{\leftarrow}$: if y is contained in $\tilde{\mathcal{P}}_k^{(l)} \cap H_{(a_1,b_1)}^{\leq,\leftarrow}$, indicating an additional reduction of the solution space (see the left-most vertex in Fig. 6a), we update $H_{(a_1,b_1)}^{\leq,\leftarrow}$ with y as the second support point of $H_{(a_1,b_1)}^{=}$. As

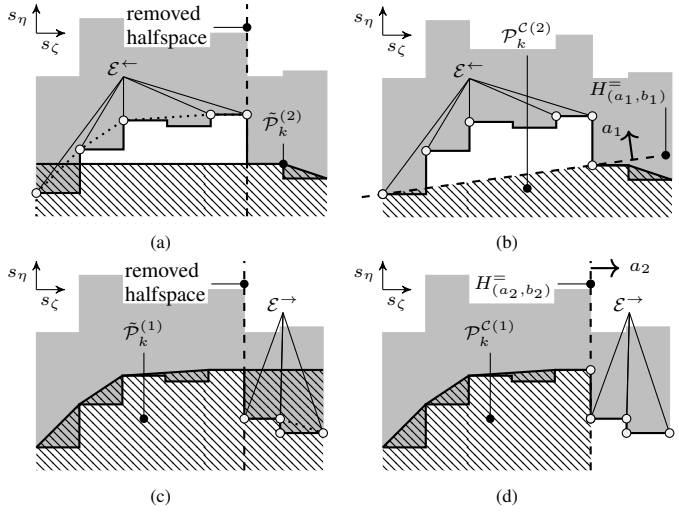


Fig. 6: Computation of the lateral polyhedra $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$: (a), (c) removal of the halfspaces defining the vertical edges of $\mathrm{conv}(\mathcal{N}^{(2)})$ and $\mathrm{conv}(\mathcal{N}^{(1)})$ to obtain $\tilde{\mathcal{P}}_k^{(2)}$ and $\tilde{\mathcal{P}}_k^{(1)}$, respectively. (b) $\tilde{\mathcal{P}}_k^{(2)}$ is intersected with $H_{(a_1,b_1)}^{\leq,\leftarrow}$ yielding $\mathcal{P}_k^{\mathcal{C}(2)}$, where $H_{(a_1,b_1)}^{=}$ separates $\mathcal{P}_k^{\mathcal{C}(2)}$ from $\mathcal{E}^{\leftarrow}$ (backward direction). (d) $\mathcal{P}_k^{\mathcal{C}(1)}$ is obtained by intersecting $\tilde{\mathcal{P}}_k^{(1)}$ with $H_{(a_2,b_2)}^{\leq,\rightarrow}$, where $H_{(a_2,b_2)}^{=}$ separates $\mathcal{P}_k^{\mathcal{C}(1)}$ from $\mathcal{E}^{\rightarrow}$ (forward direction).

an example, compare $\tilde{\mathcal{P}}_k^{(2)}$ in Fig. 6a with the polyhedron $\mathcal{P}_k^{\mathcal{C}(2)}$ in Fig. 6b. To speed up the computations, it suffices to consider only the most backward and most forward vertex of each $\mathrm{conv}(\mathcal{N}^{(j)})$ due to convexity.

Let us now consider the adjacent polyhedra $\mathcal{P}_k^{\mathcal{C}(1)}$ and $\mathcal{P}_k^{\mathcal{C}(2)}$ in Fig. 4d. Computing $H^{\leq,\leftarrow}_{(a_1,b_1)}$ in the case of $\mathcal{P}_k^{\mathcal{C}(2)}$, yields a halfspace with non-vertical hyperplane $H^{=}_{(a_1,b_1)}$ (see Fig. 6b). When computing the halfspace $H^{\leq,\rightarrow}_{(a_2,b_2)}$ in the case of $\tilde{\mathcal{P}}_k^{(1)}$, the corresponding hyperplane $H^{=}_{(a_2,b_2)}$ is vertical (see Fig. 6c and Fig. 6d). Thus, all positions in the relative interior of the intersection of $\mathrm{conv}(\mathcal{N}^{(1)})$ and $\mathrm{conv}(\mathcal{N}^{(2)})$ are excluded from the solution space, i.e., condition (c) is satisfied. Since $\mathcal{C}_k$ is connected and vertically convex (see Def. 6), the same situation occurs for every pair of adjacent $\mathrm{conv}(\mathcal{N}^{(l)})$, and therefore, the polyhedral approximation $\tilde{\mathcal{C}}_k$ of $\mathcal{C}_k$ ensures a proper encoding of $\mathrm{proj}(x_k) \in \mathcal{C}_k$.

2) Complexity: The complexity of the computation of $\mathcal{P}_k^{\mathcal{C}(l)}$ is dominated by the computation of the convex hull of $\mathcal{N}^{(l)}$: Since any $\mathcal{N}_j$ has two vertices, the number of vertices of any $\mathcal{N}^{(l)}$ is bounded by $2|\mathcal{N}|$. Thus, the complexity of computing the convex hull with respect to $|\mathcal{N}|$ is $\mathcal{O}(|\mathcal{N}|\log(|\mathcal{N}|))$ [80]. If the vertices of $\mathcal{N}^{(l)}$ are stored in a list, removing the vertical edges takes $\mathcal{O}(|\mathcal{N}|)$. Computing $H^{\leq,\leftarrow}_{(a_1,b_1)}$ and $H^{\leq,\rightarrow}_{(a_2,b_2)}$ has complexity $\mathcal{O}(|\mathcal{N}|)$, since at most $2|\mathcal{N}|$ dot and vector products in $\mathbb{R}^2$ have to be evaluated. Computing the polyhedron $\mathcal{P}_k^{\mathcal{C}(l)}$ in halfspace representation can be done in $\mathcal{O}(|\mathcal{N}|)$ if the list of vertices of $\mathcal{P}_k^{\mathcal{C}(l)}$ is ordered. Thus, the overall complexity of the approach is $\mathcal{O}((n_{\max,\mathrm{k}}-2)|\mathcal{N}|\log(|\mathcal{N}|))$.

VI. Integration Into Motion Planning Algorithms

Our approach can be integrated into a wide range of optimization-based motion planning algorithms, which is discussed next.

A. Duality-Based Reformulation

A state is collision-free if the signed distance with respect to every polyhedral keep-out zone is non-negative. However, non-convexity and non-differentiability of the signed-distance function prevent these constraints from being directly enforced [42]. To tackle this problem, Zhang et al. [42] propose a non-conservative and smooth reformulation of collision avoidance constraints for convex keep-out zones based on strong duality of convex optimization. Their results also enable one to find least-intrusive solutions in the case that a collision cannot be avoided. Below, we summarize the main results of [42], which allow us to integrate the collision avoidance constraints from driving corridors into arbitrary motion planning algorithms that rely on gradient- and Hessian-based solvers.

Consider a polyhedral keep-out zone $\mathcal{P}_k^{\mathcal{C}(l)} = \{y \,|\, y = \mathrm{proj}(x), A_k^{(l)} y - b_k^{(l)} \leq 0\}$ with matrix $A_k^{(l)}$ and vector $b_k^{(l)}$ of appropriate dimension. Using the results from [42], we encode the collision avoidance constraint $\mathrm{proj}(x_k) \notin \left(\mathcal{P}_k^{\mathcal{C}(l)}\right)^{\circ}$ as

$$(A_k^{(l)}\mathrm{proj}(x_k) - b_k^{(l)})^T \lambda_k^{(l)} \geq -\nu_k^{(l)} \tag{7a}$$

$$\|(A_k^{(l)})^T \lambda_k^{(l)}\|_2 = 1, \tag{7b}$$

$$\nu_k^{(l)} \geq 0, \lambda_k^{(l)} \geq 0, \tag{7c}$$

where the slack variable $\nu_k^{(l)} \in \mathbb{R}$ measures the penetration of $\mathcal{P}_k^{\mathcal{C}(l)}$, and $\lambda_k^{(l)}$ is the dual variable associated with the original constraint. The inequalities in (7c) apply element-wise. Since we aim to find least-intrusive solutions, the optimization problem (2) is rewritten as a soft-constrained problem by encoding the constraint (2d) using (7) and penalizing the penetration of all polyhedral keep-out zones, i.e.,

$$\min_{u(\cdot)} \sum_{k=0}^{N} J(x_k, u_k) + \kappa \sum_{k=1}^{N} \sum_{l=1}^{n_{\max,\mathrm{k}}} \nu_k^{(l)}. \tag{8}$$

If the penalty weight $\kappa \in \mathbb{R}^+$ is chosen sufficiently large, a collision-free solution can be found if one exists [42], [81].

B. Convexification of Collision Avoidance Constraints

Since the collision-avoidance constraint $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k$ is usually non-convex due to the equality constraint (7b), we sketch a convexification procedure that is based on geometrical insight into (7). We project $\mathrm{proj}(x_k)$ onto each face of $\mathcal{P}_k^{\mathcal{C}(l)}$ separately and select the closest face. Subsequently, we determine a corresponding $\lambda_k^{(l)}$ that solves the constraints (7). Any such $\lambda_k^{(l)}$ provides a supporting hyperplane $H^{=}_{(n_k^{(l)},d_k^{(l)})}$, where $n_k^{(l)} = (A_k^{(l)})^T \lambda_k^{(l)}$ and $d_k^{(l)} = (\lambda_k^{(l)})^T b_k^{(l)}$, at the associated polyhedron $\mathcal{P}_k^{\mathcal{C}(l)}$ [42] [50, Sec. 8.1]. Thus, we obtain the convexified set of collision-free positions $\mathcal{P}_k^{\epsilon}$ as

$$\mathcal{P}_k^{\epsilon} \coloneqq \bigcap_{\mathcal{P}_k^{\mathcal{C}(l)} \in \mathcal{P}_k^{\mathcal{C}}} H^{\geq}_{(n_k^{(l)},d_k^{(l)})}. \tag{9}$$

To obtain a $\lambda_k^{(l)}$ solving (7), we distinguish the following three cases:

- C1) $\mathrm{proj}(x_k) \notin \mathcal{P}_k^{\mathcal{C}(l)}$: thus, we set $\nu_k^{(l)} = 0$ and solve the convex optimization problem $y^* = \arg\min_{y \in \mathcal{P}_k^{\mathcal{C}(l)}} \|y - \mathrm{proj}(x_k)\|_2^2$, see [50, Sec. 8.1]. The supporting hyperplane at $\mathcal{P}_k^{\mathcal{C}(l)}$ in y^* with normal vector in the direction of $\mathrm{proj}(x_k) - y^*$ implicitly defines a feasible $\lambda_k^{(l)}$.
- C2) $\mathrm{proj}(x_k) \in \partial\mathcal{P}_k^{\mathcal{C}(l)}$: thus, we set $\nu_k^{(l)} = 0$. If $\mathrm{proj}(x_k)$ is not a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\lambda_k^{(l)}$ follows as the canonical basis vector that extracts the corresponding hyperplane from $A_k^{(l)}$, $b_k^{(l)}$. If $\mathrm{proj}(x_k)$ is a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\lambda_k^{(l)}$ is chosen so that $n_k^{(l)}$ is contained in the normal cone at $\mathcal{P}_k^{\mathcal{C}(l)}$ in $\mathrm{proj}(x_k)$. Moreover, the chosen $\lambda_k^{(l)}$ has to satisfy (7b) and (7c).
- C3) $\mathrm{proj}(x_k) \in (\mathcal{P}_k^{\mathcal{C}(l)})^{\circ}$: due to convexity of $\mathcal{P}_k^{\mathcal{C}(l)}$, $\mathrm{proj}(x_k)$ is not projected onto a vertex of $\mathcal{P}_k^{\mathcal{C}(l)}$. Thus, $\lambda_k^{(l)}$ is chosen as a canonical basis vector as in the first case of C2.

Note that the ambiguous cases in C2 and C3 do not admit a unique choice of $\lambda_k^{(l)}$ and, therefore, $H^{\geq}_{(n_k^{(l)},d_k^{(l)})}$. This might be resolved by comparing $n_k^{(l)}$ with the gradient of the cost function as for instance in [47, Sec. 3.3]. Since $\mathcal{P}_k^{\epsilon}$ is constructed from separating hyperplanes, see (9), it follows that $\mathrm{proj}(x_k) \in \mathcal{P}_k^{\epsilon} \subseteq \tilde{\mathcal{C}}_k$ if $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k$. While $\mathcal{P}_k^{\epsilon} \neq \emptyset$ holds if $\mathrm{proj}(x_k) \in \tilde{\mathcal{C}}_k$, $\mathcal{P}_k^{\epsilon}$ might be empty for $\mathrm{proj}(x_k) \notin \tilde{\mathcal{C}}_k$.

VII. Numerical Results

In Sec. VII-B, we demonstrate that our approach extracts driving corridors that represent semantically meaningful maneuver variants leading to a given goal region for the automated vehicle. In Sec. VII-C and VII-D, we embed our approach into two state-of-the-art motion planning algorithms. The results in Sec. VII-C show how our approach can enhance state-of-the-art planning algorithms so that local minima as in Fig. 2 are avoided. By applying our method to different scenarios from CommonRoad[1] [82] in Sec. VII-D, we demonstrate its applicability to arbitrary traffic scenarios and how the driving corridors can facilitate the search for a suitable initial guess. In Sec. VII-E we analyze the computation times of our approach, followed by a discussion on motion safety in Sec. VII-F. Since we have already shown that the computational effort of our approach typically improves with the criticality of the situation in our previous work [63], we focus on the new benefits of our approach in the evaluation.

A. Implementation Details

All computations were conducted on a laptop equipped with an Intel Core i7-10750H and 16 GB of memory. Our scenarios are modeled with the CommonRoad library. We consider a planning horizon of $5\,\mathrm{s}$ with $\Delta t = 0.1\,\mathrm{s}$ unless otherwise stated. Further implementation details are given below.

1) Driving Corridor Identification: Our first strategy for driving corridor identification referred to as DC1 demonstrates the real-time capability of our approach. Starting from the largest goal set $\mathcal{V}_N^{(*)}$, we explore the corridor graph $\mathcal{G}_\mathcal{C}$ using a depth-first search and terminate exploration once the node at time step $k = 0$ is reached. To obtain a driving corridor with nonrestrictive collision avoidance constraints, we select the largest $\mathcal{V}_k^{(*)} \in \mathcal{A}_k$ at each time step for exploration.

Our second strategy referred to as DC2 demonstrates the ability of our approach to explore complex scenarios in an anytime fashion and is employed if not otherwise stated. Strategy DC2 explores $\mathcal{G}_\mathcal{C}$ starting from every $\mathcal{V}_N^{(q)}$ and terminates as soon as at least $n_\mathcal{C}$ driving corridors for each $\mathcal{V}_N^{(q)}$ are identified. Throughout this section, we set $n_\mathcal{C} = 10$. We first explore the largest $\mathcal{V}_k^{(*)} \in \mathcal{A}_k$. All remaining $\mathcal{V}_k^{(q)} \in \mathcal{A}_k$ are sorted by their dissimilarity with respect to $\mathcal{V}_k^{(*)}$ (i.e., we compare their sets $\mathcal{D}_k^{(i)}$) and their size. This sorting strategy favors exploring different maneuver variants that reach the same goal region.

In case of DC2, we add weights to the edges of $\mathcal{G}_\mathcal{C}$ where the weight for an edge $(\mathcal{V}_{k-1}^{(j)}, \mathcal{V}_k^{(q)})$ is chosen as $w_{j,q} = \frac{1}{\mathrm{area}(\mathcal{V}_k^{(q)})}$. For each goal set $\mathcal{V}_N^{(q)}$, we determine the driving corridor $\mathcal{C}^*$ with the greatest cumulative area and an alternative driving corridor that should preferably represent a different tactical maneuver, e.g., overtaking an obstacle to the right side instead of overtaking to the left side. Therefore, we sufficiently increase the weights of the edges from $\mathcal{C}^*$ (thus, it is costly to select those edges again), and subsequently repeat the search for the driving corridor with the greatest cumulative area.

[1]https://commonroad.in.tum.de/

2) Polyhedral Approximation: For the experiments, we use the following number of polyhedra $n_{\max,\mathrm{k}}$: if $k \leq 10$, $n_{\max,\mathrm{k}} = 4$; if $11 \leq k \leq 15$, $n_{\max,\mathrm{k}} = 5$; if $16 \leq k \leq 20$, $n_{\max,\mathrm{k}} = 6$; if $21 \leq k \leq 32$, $n_{\max,\mathrm{k}} = 7$; else $n_{\max,\mathrm{k}} = 8$.

3) Trajectory Optimization: The trajectory optimization problems are modeled with CVXPY-codegen [83], which is based on the CVXPY modeling language [84], using ECOS [51] as the backend solver. For the evaluation, we approximate the occupancy of the automated vehicle as a disc. The vehicle dynamics are modeled similarly to the kinematic bicycle model in [85], but we use jerk instead of the longitudinal acceleration as input. The vehicle parameters are taken from the CommonRoad library (vehicle ID: 2).

We use a weighted combination of the cost functions proposed by CommonRoad [82] for trajectory planning:

$$J = w_{LC} J_{LC} + w_V J_V + w_O J_O + w_A J_A + w_u (J_J + J_{SR}),$$

where the (lateral) position tracking error in J_{LC} refers to the center of a given target lane. The desired velocity v_des for the partial cost J_V is extracted from the goal states of the CommonRoad scenarios and set to a constant value. The desired orientations for the partial cost function J_O are obtained by propagating the initial longitudinal position of the automated vehicle along the center of the given target lane using the desired velocity v_des.

For successive convexification procedures, we limit the maximum number of iterations by 50; if no collision-free solution can be found in time that satisfies the convergence criteria, it is considered to be infeasible.

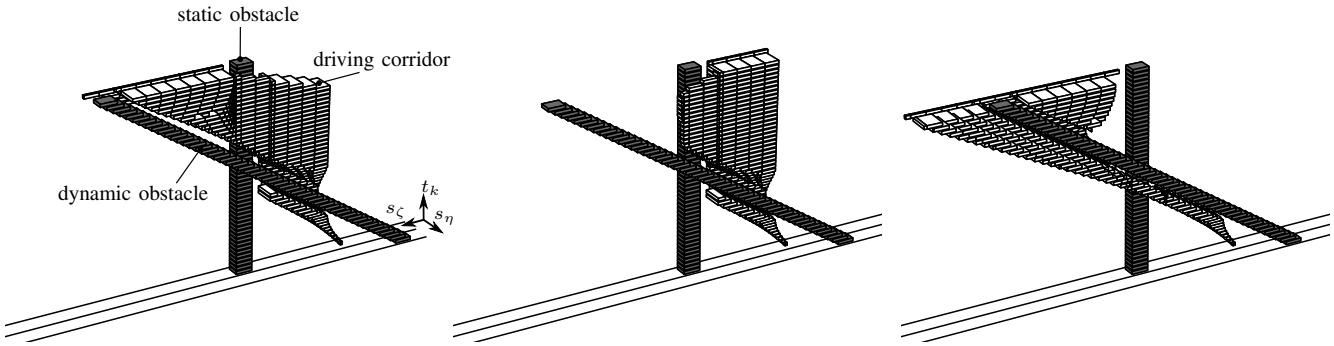
B. Driving Corridor Computation

We consider the scenario in Fig. 7a featuring a two-lane road with a static and a dynamic obstacle to determine driving corridors. The dynamic obstacle travels at a constant velocity of $17\,\mathrm{m/s}$. The goal for the automated vehicle is to return to its initial lane after $5\,\mathrm{s}$. We therefore intersect the drivable area of the automated vehicle at the final time step with the goal region.

Fig. 7 shows the results where the driving corridors are depicted as white boxes and the occupancies of the obstacles as gray boxes stacked over time. In this scenario, three semantically different maneuvers are found that lead to the desired goal region of the automated vehicle: (a) waiting until the dynamic obstacle has passed, and then, overtaking the static obstacle (see Fig. 7a); (b) staying behind the static obstacle (see Fig. 7b); and (c) passing the static obstacle before the dynamic obstacle (see Fig. 7c).

C. Augmenting Motion Planning Algorithms

In this section, we augment two state-of-the-art trajectory optimization algorithms with our proposed method. In the first experiment, we demonstrate that our method eliminates local minima occuring in optimization-based trajectory planning as illustrated in Fig. 2. The second experiment shows that our method facilitates the generalization of trajectory planners to on-road traffic situations.



(a) Waiting until the dynamic obstacle has passed, and then, overtaking the static obstacle.

(b) Staying behind the static obstacle.

(c) Passing the static obstacle before the dynamic obstacle.

Fig. 7: Driving corridors corresponding to three semantically different maneuvers. The slice of the drivable area at the top is intersected with the goal region.

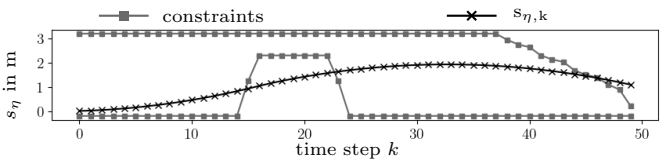
1) Elimination of Local Minima: We use the customized SQP algorithm proposed by [5] for our first experiment. At the beginning of each iteration of the successive convexification procedure in [5], the vehicle model is simulated using the input trajectory of the solution of the previous iteration (or the initial guess). The nonlinear trajectory optimization problem is then linearized around the forward-simulated trajectory to ensure dynamic feasibility of the solution.

The approach in [5] requires the passing sides of obstacles as input so that collision avoidance constraints can be represented as admissible intervals of lateral deviations $e_{\eta,k}$ from a reference path:

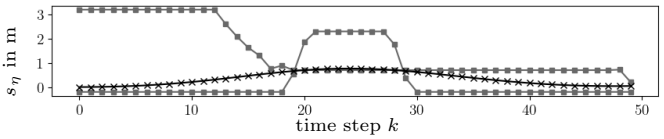
$$\underline{e}_{\eta,k}(\hat{s}_{\zeta,k}) \leq e_{\eta,k} \leq \overline{e}_{\eta,k}(\hat{s}_{\zeta,k}), \tag{10}$$

where $\hat{s}_{\zeta,k}$ denotes the longitudinal position of the forward-simulated trajectory. In simple scenarios one may argue that passing sides can be easily determined without driving corridors. However, the collision avoidance constraints in [5] are typically discontinuous with respect to the longitudinal position of the vehicle. For comparison, we therefore replace the original method for determining collision avoidance constraints with our proposed method that identifies driving corridors to obtain collision avoidance constraints, as shown in Sec. VI-B. We use the simple traffic scenario of the previous subsection and gradually raise the velocity of the dynamic obstacle starting from $10\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$ in steps of $1\,\mathrm{m/s}$. For the original version of the algorithm in [5], we specify that the static obstacle must be passed on the left side and the dynamic obstacle on the right side. As an initial guess, we set all control inputs of the vehicle model to zero and simulate the dynamics forward in time.

For velocities of $10\,\mathrm{m/s}$ to $15\,\mathrm{m/s}$ of the dynamic obstacle, the customized SQP algorithm in [5] can find a collision-free trajectory. However, the method fails if the velocity of the dynamic obstacle is between $16\,\mathrm{m/s}$ to $20\,\mathrm{m/s}$ because the algorithm gets stuck in infeasible local minima. Fig. 8 shows the initial set of position constraints for the customized SQP algorithm as well as the set at convergence if the velocity of the dynamic obstacle is set to $17\,\mathrm{m/s}$. The empty set of constraints for s_η in Fig. 8b indicates infeasibility.



(a) Constraints and lateral trajectory at the first iteration

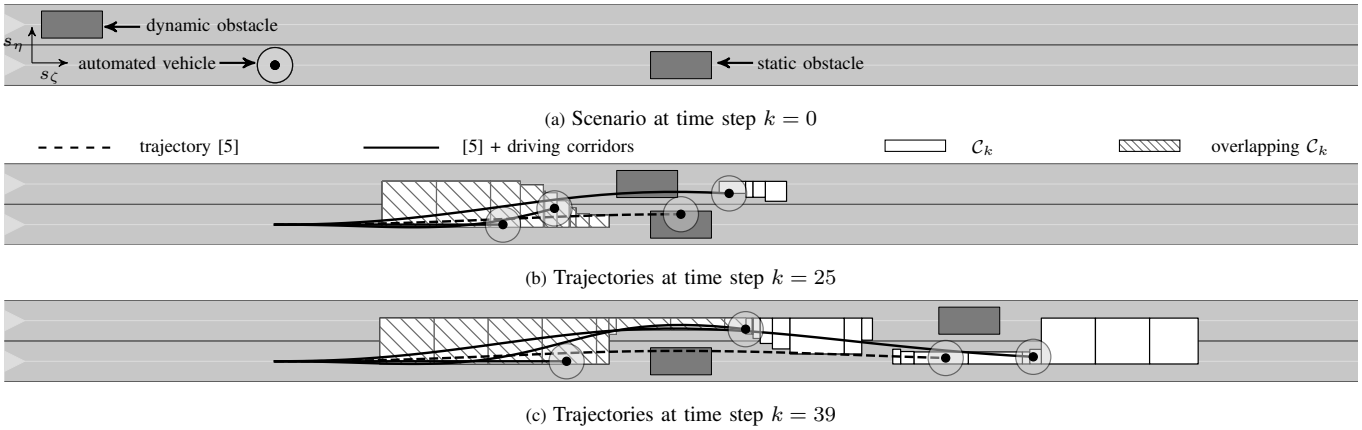


(b) Constraints and lateral trajectory at convergence

Fig. 8: Lateral position constraints used in the algorithm in [5]. Even though the initial guess in Fig. 8a seems to admit a collision-free trajectory, the algorithm converges to an infeasible local minimum shown in Fig. 8b.

In contrast to the original algorithm, the combination of the customized SQP algorithm with our computation of solution spaces provides a feasible trajectory for every identified driving corridor in each instance of the scenario. Fig. 9 provides a comparison of the results when the velocity of the dynamic obstacle is set to $17\,\mathrm{m/s}$. The trajectories that are obtained with our proposed method are depicted by the solid black lines in Fig. 9; the corresponding driving corridors are depicted in Fig. 7. Any of these found trajectories are feasible solutions of the nonlinear motion planning problem. The solution of the customized SQP algorithm in [5] is depicted by the dashed black line in Fig. 9; the infeasibility of the converged solution which was already indicated in Fig. 8b can be observed in Fig. 9b. In contrast, augmenting the algorithm from [5] with our approach yields collision-free trajectories as local minima induced by obstacles are eliminated.

2) Generalization to Traffic Scenarios: To show that our method facilitates the generalization of trajectory planners to on-road traffic situations, we combine the zeroth order hold discretization scheme [86] with the successive convexification algorithm proposed in [62] that was initially developed



(a) Scenario at time step $k = 0$

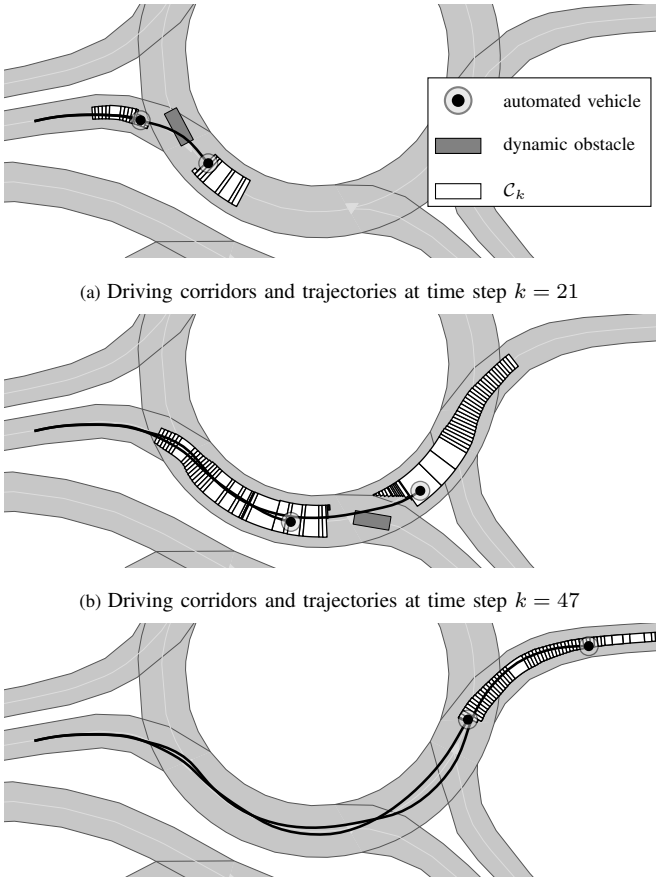
(b) Trajectories at time step $k = 25$

(c) Trajectories at time step $k = 39$

Fig. 9: Comparison of the results using the customized SQP algorithm from [5] and the combination of the algorithm with our dynamics-aware solution spaces. The circles depict the occupancy of the automated vehicle at the current time step. In Fig. 9b, it can be observed that the algorithm from [5] does not find a collision-free solution, see the dashed line. In contrast, augmenting the algorithm with our approach yields a feasible solution for the three distinct maneuvers shown in Fig. 7. The regions where the driving corridors corresponding to Fig. 7a and Fig. 7b overlap are indicated by the striped white boxes.

for powered descent guidance for extraterrestrial spaceflight. Instead of simulating the vehicle dynamics forward for linearization as in our previous experiment, [62] linearizes the nonlinear trajectory planning problem directly around the solution computed in the previous iteration of the successive convexification procedure. The resulting error in the vehicle dynamics constraint (2c) can be handled using slack variables which are referred to as virtual control inputs in [62].

Using a traffic scenario featuring a roundabout (see Fig. 10), we demonstrate that our method enables the application of the algorithm in [62] to traffic scenarios. The automated vehicle aims to enter the roundabout and take the second exit. There is an oncoming vehicle in the roundabout that must be taken into account. Since this maneuver requires longer time horizons so that the automated vehicle is able to reach the second exit, we use a planning horizon of 7 s. Our approach finds a trajectory for the automated vehicle both for the entry into the roundabout before and after the oncoming vehicle (see Fig. 10). In Sec. VII-D, we increase the difficulty of the planning problems and apply the planner to complex traffic scenarios with multiple obstacles and lanes.



(a) Driving corridors and trajectories at time step $k = 21$

(b) Driving corridors and trajectories at time step $k = 47$

(c) Driving corridors and trajectories at time step $k = N = 70$

Fig. 10: Roundabout scenario: (a), (b) our approach identifies a driving corridor for the entry before and after the obstacle in the roundabout. (c) the automated vehicle reaches the second exit using either driving corridor.

D. Applicability in Arbitrary Traffic Scenarios

Planning schemes for autonomous driving that iteratively linearize the vehicle model, such as [5], [12], usually rely on the previously computed solution as an initial guess, similar to model predictive control. In general, this initial guess is only close to the solution if the same maneuver is followed. However, if new maneuvers are initiated, implying a change to another driving corridor, the previously computed solution may be insufficient as an initial guess.

In this section, we demonstrate that driving corridors facilitate the initialization of trajectory planning methods and that our approach can be applied in arbitrary traffic scenarios with multiple obstacles. We therefore introduce different strategies for generating an initial guess for the optimization problems and compare their performance on 30 highway scenarios based on the NGSIM dataset from the CommonRoad library [82] that have five or six lanes and up to 57 obstacles (see Appendix B for the identifiers of the scenarios). After introducing different initialization strategies and the problem setup, we present the results of our experiments in Sec. VII-D3.

1) Initialization Strategies: We introduce the *solution-space-guided initialization* that works as follows: first, the goal state is projected onto the driving corridor $\mathcal{C}_N$ at the terminal time step. Afterwards, we linearly interpolate between the projected goal state and the initial state of the automated vehicle. The collision avoidance constraints are convexified at each time step $k \in \mathcal{I}_{[1:N]}$ using the procedure in Sec. VI-B and we solve N convex optimization problems of the form $y_k^* = \arg\min_{y_k} \|y_k - \mathrm{proj}(x_k)\|_2^2$, s.t. $y_k \in \mathcal{P}_k^\epsilon$. The solutions y_k^* serve as our initial guess.

For comparison, we consider the following initialization strategies:

- *Simulate*: the vehicle dynamics are simulated forward in time assuming that all control inputs are zero.
- *Initial Lane*: we translate the initial state along the centerline of the initial lane according to the desired velocity v_{des} provided by the planning problem.
- *Interpolate*: we linearly interpolate between the initial state and the goal state, e.g., as in [60], [62].

2) Problem Setup: Since the considered initialization schemes except *simulate* generally do not provide a dynamically feasible initial guess, the motion planning algorithm of [5] introduced in Sec. VII-C1 cannot be applied. We therefore use the combination of [62], [86] with our approach as in Sec. VII-C2 and exploit the virtual control inputs that handle infeasible initializations. The goal state for the initialization schemes *solution-space-guided initialization* and *interpolate* are obtained by propagating the initial longitudinal position of the automated vehicle along the center of the target lane using the desired velocity v_{des}. To assess the robustness of the initialization schemes with respect to parameter variations, we choose the mean, lower, and upper bound from the uncertain goal states of the CommonRoad scenarios as desired velocities.

3) Evaluation: Using our method, we obtain 138 driving corridors for all traffic scenarios, in which we plan trajectories for each desired velocity v_{des}, i.e., we obtain 414 combinations of driving corridors and desired velocities. Tab. I summarizes the results, which we discuss below.

In total, we are able to find a feasible trajectory in each scenario for every initialization strategy. We further analyze the success rates of the initialization strategies with regard to all combinations of driving corridors and desired velocities (see the third column in Tab. I), i.e., the success rate is 100% if for each combination of driving corridors and desired velocities a solution is found. It can be seen that the *solution-space-guided initialization* solves more instances of the motion planning problem than any other scheme, as shown in Tab. I, and the success rate is increased by 6% compared to the *simulate* initialization strategy. Furthermore, in cases where the initial guess leads to a feasible solution, the number of convex programming iterations until convergence can be reduced by an average of almost 20% with the *solution-space-guided initialization* compared to *simulate*. Some combinations of driving corridors and desired velocities were infeasible due to (a) non-connectivity of the approximated driving corridor, which might yield an initial guess that switches between connected components, and (b) non-drivability of the driving corridor with regard to the high-fidelity vehicle model used for motion planning. Therefore, a possible direction for future research is the assessment of the drivability of the driving corridors prior to trajectory planning.

The desired velocity v_{des} affects the cost function J for the trajectory optimization and the initialization schemes *solution-space-guided initialization*, *interpolate*, and *initial lane*. We therefore analyze for each driving corridor the standard deviation in the required number of convex programming iterations resulting from variations in the desired velocity. Tab. I shows the standard deviations averaged over all scenarios for each initialization scheme. The average standard deviation using the *solution-space-guided initialization* is close to the case of *simulate* where the reference velocity only affects the cost function. In contrast, *interpolate* and *initial lane* exhibit much larger variations in the number of iterations when changing the desired velocity. Moreover, if *interpolate* or *initial lane* is used for initialization, it is more likely that a variation of the desired velocity will render the optimization task infeasible, as shown in the last column of Tab. I, which lists the number of driving corridors where a feasible solution could not be found for all desired velocities.

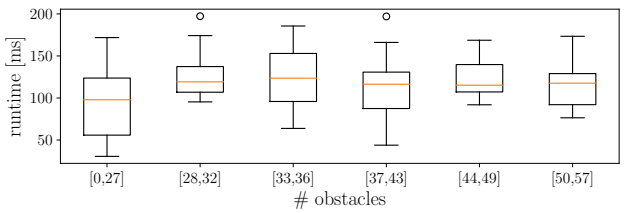
E. Computation Times

We analyze the runtime behavior of our approach in dependence of the number of obstacles in the scenarios. Fig. 11 shows boxplots of the resulting computation times for the reachable set computation, the driving corridor identification, and the computation of the polyhedral approximation, which we repeated 100 times to obtain a statistically profound example. The results shown in Fig. 11 indicate that our approach scales favorably with the number of obstacles as the median computation times vary only slightly; therefore, our method is suited to be employed in cluttered environments.

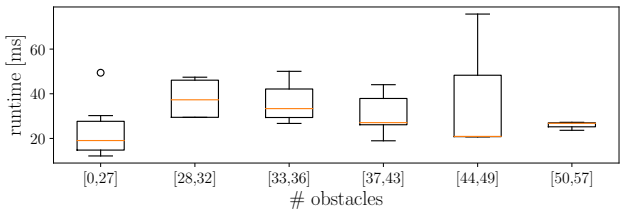
One iteration of the trajectory planning problem, i.e., convexifying the non-convex trajectory optimization problem (2) and solving the convexified optimization problem, requires $56\,\mathrm{ms}$ on average. Considering the average number of iterations given in Tab. I, the *solution-space-guided initialization* reduces the average computation time for motion planning to $347\,\mathrm{ms}$ compared to $426\,\mathrm{ms}$ for *simulate*. In conclusion, the solution-space-guided initialization provides an improvement in terms of the overall computational effort and increases robustness with respect to parameter variations only at a slightly increased effort to provide the initial guess.

F. Discussion on Motion Safety

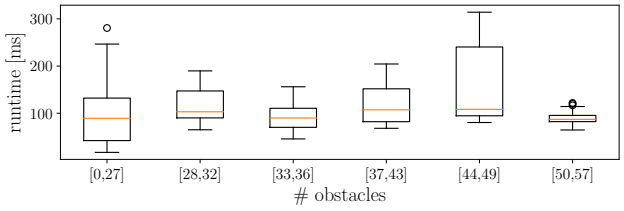
To ensure motion safety, the automated vehicle must reason over an infinite time horizon (or at least until a set of safe goal states is reached) while considering its own dynamics and the future motion of other traffic participants according to [87]. Driving corridors can be restricted to end in a set of safe goal states to ensure persistent feasibility. The missing ingredients to ensure safety are (a) that the full-dimensional vehicle shape is considered for trajectory planning, (b) that the provided predictions must be conservative (i.e., guaranteed to include the real future behavior of other road users), and (c) robustness



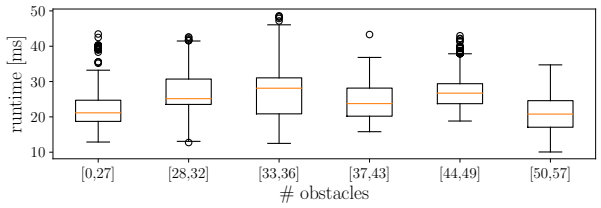
(a) Reachable set computation



(b) Driving corridor identification DC1



(c) Driving corridor identification DC2

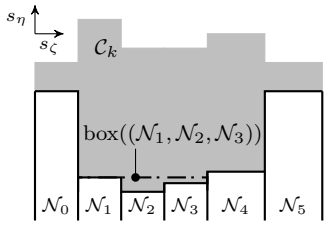


(d) Polyhedral approximation for a single corridor

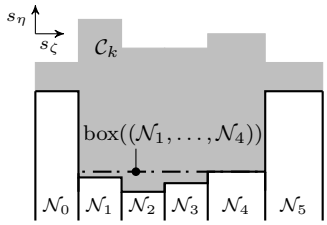
Fig. 11: Boxplots showing the runtime of our approach with respect to the number of obstacles in the scenarios listed in Appendix B. We clustered the scenarios with respect to the number of obstacles and repeated the computation of reachable sets, the identification of driving corridors, and the polyhedral approximation 100 times in each scenario.

with respect to disturbances and model uncertainties. Below, we will further discuss points (a) to (c).

In our earlier work in [63], a mathematically rigorous model for collision avoidance of the full-size vehicle in the special setup of decoupled longitudinal and lateral motion planning is given. Difficulties in generalizing the consideration of full-size vehicles arise from the fact that our approach is formulated in a road-aligned coordinate system and the assumption that the vehicle is oriented along the reference path when computing the reachable set (see Sec. III). The extension of our approach



(a) Largest sequence $(\mathcal{N}_1,\mathcal{N}_2,\mathcal{N}_3)$ for $\mathcal{N}_1$.



(b) Largest sequence $(\mathcal{N}_1,\ldots,\mathcal{N}_4)$ for $\mathcal{N}_4$.

Fig. 12: Visualization of the largest sequence $\mathcal{N}^{(i)}$ for a node $\mathcal{N}_j$ such that $\mathcal{N}_j$ limits the lateral extent of $\mathrm{box}(\mathcal{N}^{(i)})$ towards the interior of the driving corridor $\mathcal{C}_k$.

to formally correct collision-avoidance constraints for full-size vehicles in the road-aligned coordinate system is subject of future research.

Regarding (b) and (c), we suggest to integrate our approach in the online verification framework proposed in our previous works [88]–[90] to guarantee legal safety of automated vehicles and drivability of motions despite disturbances and model uncertainties. Moreover, as shown in [91], [92], our method can be extended to cooperative driving with explicit communication between groups of automated vehicles, which offers the possibility of further enhancing road safety due to reduced uncertainties regarding the intentions of others.

VIII. Conclusions

This paper provides a generalization of our previous results on combining set-based reachability analysis with optimal control by enabling the usage of vehicle models that jointly consider the longitudinal and lateral dynamics. Moreover, our approach can be combined with arbitrary existing optimization-based algorithms that rely on gradient- or Hessian-based solvers. Our results demonstrate that the proposed approach can identify different driving maneuvers in arbitrary traffic scenarios. Thereby, feasible solutions can be found in traffic scenarios that are not solvable using state-of-the-art planning algorithms. Apart from avoiding infeasible local minima, our method facilitates the generation of initial guesses for the trajectory planning problem such that the computational effort is reduced while increasing the robustness of the algorithms.

Appendix A
Heuristic for Partitioning

Considering all possible partitions of a sequence $\mathcal{N}^{(l)} \subseteq \mathcal{N}$ into subsequences can lead to a large computational overhead. Even if we only consider partitions of $\mathcal{N}^{(l)}$ into two

TABLE I: Comparison of Initialization Schemes

Initialization strategy	**Solutions**	**Success rate**	**Average # iterations**	**Avg. STD # iterations due to change of v_{des}**	**# Corridors that are not feasible for all v_{des}**
solution-space-guided	345	83%	6.2	0.55	2
interpolate	335	81%	8.6	1.44	11
simulate	317	77%	7.6	0.34	1
initial lane	326	79%	8.7	1.39	14

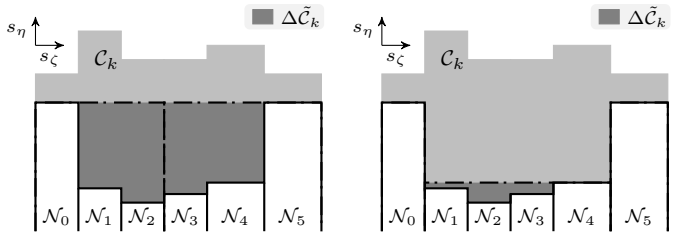


Fig. 13: (a) Any partition of $\mathcal{N}$ into two subsequences $\mathcal{N}^{(0)}$ and $\mathcal{N}^{(1)}$ cannot reduce $\Delta\tilde{\mathcal{C}}_k$ when using $\mathrm{box}(\mathcal{N}^{(l)})$ instead of $\mathrm{conv}(\mathcal{N}^{(l)})$ in (5). (b) Partitioning $\mathcal{N}$ into three subsequences, e.g., $(\mathcal{N}_0), (\mathcal{N}_1, \dots, \mathcal{N}_4)$, and $(\mathcal{N}_5)$, resolves this issue.

or three subsequences, we must already evaluate $(|\mathcal{N}^{(l)}| - 1) + \frac{(|\mathcal{N}^{(l)}|-2)(|\mathcal{N}^{(l)}|-1)}{2}$ possible partitions. Under this restriction, computing the partitions in the scenarios listed in Appendix B and using the number of polyhedra $n_{\max,\mathrm{k}}$ given in Sec. VII-A2 required $63\,\mathrm{ms}$ on average and at most $248\,\mathrm{ms}$.

We subsequently propose a heuristic that selects only $|\mathcal{N}^{(l)}|$ possible partitions of $\mathcal{N}^{(l)}$ for evaluation. These can be precomputed in $\mathcal{O}(|\mathcal{N}^{(l)}|^2)$, whereas storing and updating after each partitioning takes $\mathcal{O}(|\mathcal{N}^{(l)}|)$. The mean and maximum computation times decrease to less than 10 ms and 15 ms, respectively, in the scenarios listed in Appendix B. Compared to considering all possible combinations for partitions into two or three subsequences, our proposed heuristic yielded the same or even better results in terms of $\Delta\tilde{\mathcal{C}}_k$ in $84\,\%$ of the cases.

Let us introduce the approximation error of a set $\mathcal{N}'$ of sequences $\mathcal{N}^{(j)}$ as

$$\Delta(\mathcal{N}') \coloneqq \sum_{\mathcal{N}^{(j)} \in \mathcal{N}'} \mathrm{area}\left(\mathrm{box}(\mathcal{N}^{(j)}) \setminus \mathcal{N}^{(j)}\right). \tag{11}$$

Without loss of generality, we assume that $\mathcal{N}^{(l)} = (\mathcal{N}_0, \dots, \mathcal{N}_n)$. For each $\mathcal{N}_j \in \mathcal{N}^{(l)}$, we determine the largest subsequence $\mathcal{N}^{(i)} = (\mathcal{N}_s, \dots, \mathcal{N}_j, \dots, \mathcal{N}_e)$, $0 \leq s \leq j \leq e \leq n$, such that $\mathcal{N}_j$ limits the lateral extent of $\mathrm{box}(\mathcal{N}^{(i)})$ towards the interior of the driving corridor $\mathcal{C}_k$, as shown in Fig. 12. The resulting partition is $\mathcal{N}' = \{(\mathcal{N}_0, \dots, \mathcal{N}_{s-1}), (\mathcal{N}_s, \dots, \mathcal{N}_e), (\mathcal{N}_{e+1}, \dots, \mathcal{N}_n)\}$. Following the reasoning in [93], we consider $\mathcal{N}'$ if the relative improvement of the approximation error is above a threshold ϵ:

$$\frac{\Delta(\{\mathcal{N}^{(l)}\}) - \Delta(\mathcal{N}')}{\mathrm{area}(\mathrm{box}(\mathrm{clip}(\mathcal{N}^{(l)})))} \geq \epsilon, \tag{12}$$

where $\mathrm{clip}(\cdot)$ clips all $\mathcal{N}_j \in \mathcal{N}^{(l)}$ so that $\mathrm{area}(\mathrm{clip}(\mathrm{box}(\mathcal{N}^{(l)}))) \in \mathbb{R}^+$ has a finite value. Using this heuristic, we can consider partitions of $\mathcal{N}^{(l)}$ into two or three subsequences. Note that we consider partitions into three subsequences, since it is possible that no partition into two subsequences can reduce the approximation error due to the replacement of the convex hull in (5) with the minimum bounding box (see Fig. 13).

APPENDIX B
SELECTED SCENARIOS FOR EVALUATION IN SEC. VII-D

The scenarios are USA_US101-1_1_T-1, USA_US101-*_3_T-1 for $* \in \{5, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26\}$, USA_US101-*_4_T-1 for $* \in \{4, 6, 15, 17, 26, 28\}$, USA_US101-*_5_T-1 for $* \in \{7, 9, 10, 11, 12, 27\}$, and USA_US101-*_6_T-1 for $* \in \{2, 3, 8, 13, 14\}$.

REFERENCES

[1] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
[3] M. Werling, S. Kammel, J. Ziegler, and L. Gröll, "Optimal trajectories for time-critical street scenarios using discretized terminal manifolds," *Int. Journal of Robotics Research*, vol. 31, no. 3, pp. 346–359, 2012.
[4] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 1879–1884.
[5] A. Carvalho, Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "Predictive control of an autonomous ground vehicle using an iterative linearization approach," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2013, pp. 2335–2340.
[6] J. Nilsson, M. Brännström, J. Fredriksson, and E. Coelingh, "Longitudinal and lateral control for automated yielding maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 5, pp. 1404–1414, 2016.
[7] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2016, pp. 205–210.
[8] J. Tomas-Gabarron, E. Egea-Lopez, and J. Garcia-Haro, "Vehicular trajectory optimization for cooperative collision avoidance at high speeds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1930–1941, 2013.
[9] D. N. Godbole, V. Hagenmeyer, R. Sengupta, and D. Swaroop, "Design of emergency manoeuvres for automated highway system: obstacle avoidance problem," in *Proc. of the IEEE Conference on Decision and Control*, vol. 5, 1997, pp. 4774–4779.
[10] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, "Path planning for autonomous vehicles using model predictive control," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 174–179.
[11] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1:43 scale RC cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.
[12] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model predictive trajectory planning for automated driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 24–38, 2019.
[13] P. Liu, B. Paden, and U. Ozguner, "Model predictive trajectory optimization and tracking for on-road autonomous vehicles," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 3692–3697.
[14] M. Schmidt, C. Wissing, J. Braun, T. Nattermann, and T. Bertram, "Maneuver identification for interaction-aware highway lane change behavior planning based on polygon clipping and convex optimization," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2019, pp. 3948–3953.
[15] B. Gutjahr, L. Gröll, and M. Werling, "Lateral vehicle trajectory optimization using constrained linear time-varying MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1586–1595, 2017.
[16] C. Pek and M. Althoff, "Computationally efficient fail-safe trajectory planning for self-driving vehicles using convex optimization," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1447–1454.
[17] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 2126–2133.
[18] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 1053–1060.

[19] K. Esterle, V. Aravantinos, and A. Knoll, "From specifications to behavior: Maneuver verification in a semantic state space," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2140–2147.

[20] J. Schlechtriemen, K. P. Wabersich, and K. Kuhnert, "Wiggling through complex traffic: Planning trajectories constrained by predictions," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 1293–1300.

[21] M. Schmidt, C. Manna, J. H. Braun, C. Wissing, M. Mohamed, and T. Bertram, "An interaction-aware lane change behavior planner for automated vehicles on highways based on polygon clipping," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1876–1883, 2019.

[22] P. Bender, Ö. Ş. Taş, J. Ziegler, and C. Stiller, "The combinatorial aspect of motion planning: Maneuver variants in structured environments," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1386–1392.

[23] J. Park, S. Karumanchi, and K. Iagnemma, "Homotopy-based divide-and-conquer strategy for optimal trajectory planning via mixed-integer programming," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1101–1115, 2015.

[24] S. Kim, K. Sreenath, S. Bhattacharya, and V. Kumar, "Trajectory planning for systems with homotopy class constraints," in *Latest Advances in Robot Kinematics*, J. Lenarcic and M. Husty, Eds. Springer Netherlands, 2012, pp. 83–90.

[25] T. Gu, J. M. Dolan, and J. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2016, pp. 5474–5480.

[26] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *Annals of Mathematics and Artificial Intelligence*, vol. 84, pp. 139–160, 2018.

[27] W. Lim, S. Lee, M. Sunwoo, and K. Jo, "Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 613–626, 2018.

[28] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[29] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," in *Proc. of the European Conference on Mobile Robots*, 2015, pp. 1–6.

[30] B. Li and Y. Zhang, "Fast trajectory planning for off-road autonomous driving with a spatiotemporal tunnel and numerical optimal control approach," in *Proc. of the IEEE Conference on Advanced Robotics and Mechatronics*, 2019, pp. 924–929.

[31] X. Liu, P. Lu, and B. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[32] L. Campos-Macías, D. Gómez-Gutiérrez, R. Aldana-López, R. de la Guardia, and J. I. Parra-Vilchis, "A hybrid method for online trajectory planning of mobile robots in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 935–942, 2017.

[33] Z. Zhu, E. Schmerling, and M. Pavone, "A convex optimization approach to smooth trajectories for motion planning with car-like robots," in *Proc. of the IEEE Conference on Decision and Control*, 2015, pp. 835–842.

[34] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.

[35] D. Ioan, S. Olaru, I. Prodan, F. Stoican, and S. Niculescu, "From obstacle-based space partitioning to corridors and path planning. A convex lifting approach," *IEEE Control Systems Letters*, vol. 4, no. 1, pp. 79–84, 2020.

[36] C. Qingyang, S. Zhenping, L. Daxue, F. Yuqiang, and L. Xiaohui, "Local path planning for an unmanned ground vehicle based on SVM," *Int. Journal of Advanced Robotic Systems*, vol. 9, no. 6, 2012.

[37] Q. H. Do, S. Mita, and K. Yoneda, "Narrow passage path planning using fast marching method and support vector machine," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 630–635.

[38] M. Morsali, J. Åslund, and E. Frisk, "Trajectory planning for autonomous vehicles in time varying environments using support vector machines," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1–6.

[39] M. Morsali, J. Åslund, and E. Frisk, "Trajectory planning in traffic scenarios using support vector machines," *Proc. of the IFAC Symposium on Advances in Automotive Control*, vol. 52, no. 5, pp. 91–96, 2019.

[40] R. Soloperto, J. Köhler, F. Allgöwer, and M. A. Müller, "Collision avoidance for uncertain nonlinear systems with moving obstacles using robust model predictive control," in *Proc. of the European Control Conference*, 2019, pp. 811–817.

[41] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Springer International Publishing, 2015, pp. 109–124.

[42] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Transactions on Control Systems Technology*, vol. 29, no. 3, pp. 972–983, 2021.

[43] C. Zagaris, H. Park, J. Virgili-Llop, R. Zappulla, M. Romano, and I. Kolmanovsky, "Model predictive control of spacecraft relative motion with convexified keep-out-zone constraints," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 9, pp. 2054–2062, 2018.

[44] U. Rosolia, S. De Bruyne, and A. G. Alleyne, "Autonomous vehicle control: A nonconvex approach for obstacle avoidance," *IEEE Transactions on Control Systems Technology*, vol. 25, no. 2, pp. 469–484, 2017.

[45] M. Szmuk, C. A. Pascucci, and B. Açıkmeşe, "Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2018, pp. 1–9.

[46] A. P. Vinod, S. Rice, Y. Mao, M. M. K. Oishi, and B. Açıkmeşe, "Stochastic motion planning using successive convexification and probabilistic occupancy functions," in *Proc. of the IEEE Conference on Decision and Control*, 2018, pp. 4425–4432.

[47] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[48] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1954–1961.

[49] T. Schouwenaars, B. De Moor, E. Feron, and J. How, "Mixed integer programming for multi-vehicle path planning," in *Proc. of the European Control Conference*, 2001, pp. 2603–2608.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[51] A. Domahidi, E. Chu, and S. Boyd, "ECOS: An SOCP solver for embedded systems," in *Proc. of the European Control Conference*, 2013, pp. 3071–3076.

[52] J. Mattingley and S. Boyd, "CVXGEN: A code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[53] B. Yi, S. Gottschling, J. Ferdinand, N. Simm, F. Bonarens, and C. Stiller, "Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2016, pp. 584–589.

[54] C. Bali and A. Richards, "Robot navigation using convex model predictive control and approximate operating region optimization," in *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems*, 2017, pp. 2171–2176.

[55] S. Dixit, U. Montanaro, M. Dianati, D. Oxtoby, T. Mizutani, A. Mouzakitis, and S. Fallah, "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2310–2323, 2020.

[56] A. Britzelmeier and M. Gerdts, "A nonsmooth newton method for linear model-predictive control in tracking tasks for a mobile robot with obstacle avoidance," *IEEE Control Systems Letters*, vol. 4, no. 4, pp. 886–891, 2020.

[57] X. Liu and P. Lu, "Solving nonconvex optimal control problems by convex optimization," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 3, pp. 750–765, 2014.

[58] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *Int. Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

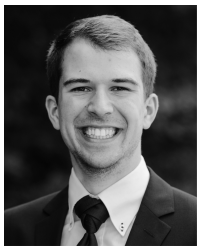
[59] Y. Mao, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems and its convergence properties," in *Proc. of the IEEE Conference on Decision and Control*, 2016, pp. 3636–3641.

[60] ——, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint arXiv:1804.06539*, 2018.

[61] R. Bonalli, A. Cauligi, A. Bylard, and M. Pavone, "GuSTO: Guaranteed sequential trajectory optimization via sequential convex programming," in *Proc. of the International Conference on Robotics and Automation*, 2019, pp. 6741–6747.

[62] T. Reynolds, D. Malyuta, M. Mesbahi, B. Açıkmeşe, and J. M. Carson, "A real-time algorithm for non-convex powered descent guidance," in *AIAA Scitech 2020 Forum*, 2020.

[63] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, 2020.
[64] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Safe driving envelopes for shared control of ground vehicles," in *Proc. of the IFAC Symposium on Advances in Automotive Control*, vol. 46, no. 21, 2013, pp. 831–836.
[65] T. Schoels, L. Palmieri, K. O. Arras, and M. Diehl, "An NMPC approach using convex inner approximations for online motion planning with guaranteed collision avoidance," in *Proc. of the International Conference on Robotics and Automation*, 2020, pp. 3574–3580.
[66] T. Schoels, P. Rutquist, L. Palmieri, A. Zanelli, K. O. Arras, and M. Diehl, "CIAO*: MPC-based safe motion planning in predictable dynamic environments," vol. 53, no. 2, 2020, pp. 6555–6562.
[67] R. B. Patel and P. J. Goulart, "Trajectory generation for aircraft avoidance maneuvers using online optimization," *Journal of Guidance, Control, and Dynamics*, vol. 34, no. 1, pp. 218–230, 2011.
[68] M. Gerdts, R. Henrion, D. Hömberg, and C. Landry, "Path planning and collision avoidance for robots," *Numerical Algebra, Control and Optimization*, vol. 2, no. 3, pp. 437–463, 2012.
[69] D. Dueri, Y. Mao, Z. Mian, J. Ding, and B. Açıkmeşe, "Trajectory optimization with inter-sample obstacle avoidance via successive convexification," in *Proc. of the IEEE Conference on Decision and Control*, 2017, pp. 1150–1156.
[70] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," in *IFAC-PapersOnLine*, 2017, vol. 50, no. 1, pp. 4063–4069.
[71] C. Y. Son, D. Jang, H. Seo, T. Kim, H. Lee, and H. J. Kim, "Real-time optimal planning and model predictive control of a multi-rotor with a suspended load," in *Proc. of the International Conference on Robotics and Automation*, 2019, pp. 5665–5671.
[72] S. Liu, M. Watterson, K. Mohta, K. Sun, S. Bhattacharya, C. J. Taylor, and V. Kumar, "Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1688–1695, 2017.
[73] A. Platzer and E. M. Clarke, "Formal verification of curved flight collision avoidance maneuvers: A case study," in *Formal Methods*, A. Cavalcanti and D. R. Dams, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 547–562.
[74] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.
[75] W. Lipski and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles," *Journal of Algorithms*, vol. 1, no. 3, pp. 235–246, 1980.
[76] H. T. Croft, K. Falconer, and R. K. Guy, *Unsolved Problems in Geometry: Unsolved Problems in Intuitive Mathematics*, 1st ed., ser. Unsolved Problems in Intuitive Mathematics. Springer-Verlag New York, 1991, vol. 2.
[77] E. Fink and D. Wood, "Generalized halfspaces in restricted-orientation convexity," *Journal of Geometry*, vol. 62, pp. 99–120, 1998.
[78] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 160–166.
[79] R. Sedgewick, *Algorithms in C, Part 5: Graph Algorithms, Third Edition*, 3rd ed. Addison-Wesley Professional, 2001.
[80] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.
[81] S.-P. Han and O. Mangasarian, "Exact penalty functions in nonlinear programming," *Mathematical Programming*, vol. 17, no. 1, pp. 251–269, 1979.
[82] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017, pp. 719–726.
[83] N. Moehle, J. Mattingley, and S. Boyd, "Embedded convex optimization with CVXPY," Talk, 2017. [Online]. Available: https://www.nicholasmoehle.com/talks/codegen.pdf
[84] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
[85] J. Kong, M. Pfeiffer, G. Schildbach, and F. Borrelli, "Kinematic and dynamic vehicle models for autonomous driving control design," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1094–1099.
[86] D. Malyuta, T. Reynolds, M. Szmuk, M. Mesbahi, B. Açıkmeşe, and J. M. Carson, "Discretization performance and accuracy analysis for the rocket powered descent guidance problem," in *AIAA Scitech 2019 Forum*, 2019.
[87] T. Fraichard, "A short paper about motion safety," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 1140–1145.
[88] C. Pek, S. Manzinger, M. Koschi, and M. Althoff, "Using online verification to prevent autonomous vehicles from causing accidents," *Nature Machine Intelligence*, vol. 2, no. 9, pp. 518–528, 2020.
[89] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.
[90] B. Schürmann, D. Heß, J. Eilbrecht, O. Stursberg, F. Köster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.
[91] S. Manzinger and M. Althoff, "Negotiation of drivable areas of cooperative vehicles for conflict resolution," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2017, pp. 1–8.
[92] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2018, pp. 444–451.
[93] M. Ghosh, N. M. Amato, Y. Lu, and J.-M. Lien, "Fast approximate convex decomposition using relative concavity," *Computer-Aided Design*, vol. 45, no. 2, pp. 494–504, 2013.



Lukas Schäfer is currently a M.Sc. student in Robotics, Cognition, Intelligence at the Technical University of Munich and was a student assistant in the Cyber-Physical Systems Group under Prof. Dr.-Ing. Matthias Althoff. He received his B.Eng. degree in Business Administration and Engineering from the Baden-Wuerttemberg Cooperative State University Stuttgart and his B.Sc. degree in Mechanical Engineering from the Technical University of Munich.

Stefanie Manzinger is a development engineer at MAN Truck & Bus SE, Germany. Before joining MAN Truck & Bus SE, she was a Ph.D. student in the Cyber-Physical Systems Group at the Technical University of Munich under Prof. Dr.-Ing. Matthias Althoff. She received her B.Sc. degree in Mechatronics and Information Technology in 2014 and her M.Sc. degree in Robotics, Cognition, Intelligence in 2016, both from the Technical University of Munich, Germany. Her research interests include automated vehicles, reachability analysis, and motion planning.

Matthias Althoff is an associate professor in computer science at the Technical University of Munich, Germany. He received his diploma engineering degree in Mechanical Engineering in 2005, and his Ph.D. degree in Electrical Engineering in 2010, both from the Technical University of Munich, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Ilmenau University of Technology, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.