Robust and Efficient Curvilinear Coordinate Transformation with Guaranteed Map Coverage for Motion Planning

Gerald Würsching and Matthias Althoff

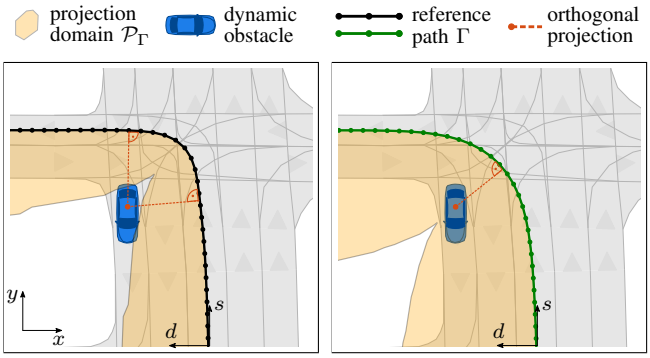
***Abstract*— Curvilinear coordinate frames are a widespread representation for motion planners of automated vehicles. In structured environments, the required reference path is often extracted from map data, e.g., by linearly interpolating the center points of lanes. Often, these reference paths are not directly suited for curvilinear frames, as the representation of points is not guaranteed to be unique for relevant parts of the road. Artifacts arising from faulty coordinate conversions can impede the robustness of downstream planning tasks and may result in safety-critical situations. We present an iterative procedure to adapt a reference path, ensuring a unique representation of all points within a provided subset of a map. Our numerical experiments demonstrate the efficacy of our method when combined with two motion planning tasks: Computing the reachable set of the ego vehicle and planning trajectories using a sampling-based approach.**

I. Introduction

Curvilinear coordinate frames (often referred to as Frenet frames) are commonly applied in motion planning [1], prediction [2] and control [3] for automated road vehicles. Because curvilinear coordinate systems can be aligned with the road geometry (see Fig. 1), they exhibit several desireable properties for motion planning in structured environments: We can formulate the nonlinear collision avoidance constraints from the road boundaries as linear constraints. Also, the nonlinear vehicle dynamics can be linearized around the reference path of the curvilinear frame [4].

Despite the aformentioned benefits, curvilinear coordinate frames inherently entail drawbacks. A major problem is the singularity of the transformation for regions which exceed the radius of the oscullating circle [5]–[7]. As shown in Fig. 1, points within these regions cannot be uniquely represented in the curvilinear coordinate frame. This is problematic, as obstacles in these areas would not be transformed correctly for collision checking and continuous trajectories passing through these regions would exhibit discontinuities after transformation [8]. This issue becomes particularly prevalent for reference paths with large curvatures relative to the width of the road (e.g., in intersections). Apart from that, the reference path needs to be at least C^2 continuous to ensure curvature continuity of trajectories represented in the curvilinear coordinate frame [1]. In most works, the existence of an appropriate reference path, e.g., the lane center points, is implicitly assumed. However, in arbitrary, real-world scenarios, this reference path might not be directly suitable for constructing the coordinate frame, as the transformation can be non-unique in parts of the road network due to the occurring singular regions. In this study, we address both aforementioned issues by adapting reference paths to ensure that all points within a provided subset of the road network can be uniquely represented in the curvilinear coordinate frame and that the resulting reference path is C^2 continuous.

All authors are with the School of Computation, Information and Technology at the Technical University of Munich, 85748 Garching, Germany.
{gerald.wuersching,althoff}@tum.de



(a) Original reference path (b) Modified reference path

Fig. 1: Scenario showing a reference path for a turn at a wide intersection. a) For the original path, using the center points of the lane, the dynamic obstacle is outside of the unique projection domain. b) Our approach, which modifies the reference path, ensures that the position (x, y) can be represented uniquely in the curvilinear frame (s, d).

A. Related Work

1) Motion planning within curvilinear frames: Various types of motion planners apply curvilinear coordinate representations: Optimization-based approaches [4], [7], [9]–[11] utilize curvilinear coordinate frames to simplify collision avoidance constraints and linearize vehicle dynamics around the reference path. Planning approaches using discrete sampling [1], [12]–[14] generate trajectories by sampling around the reference path of the curvilinear coordinate system. None of the mentioned works, however, directly addresses singularities in the transformation originating from the geometry of the reference path. Only some works circumvent this issue by discarding trajectories that are outside the unique projection domain [15], manually correcting trajectories after transformation [16], or resorting to planning in Cartesian space for high-curvature scenarios [8].

2) Reference path generation: Previous studies propose to generate a curvature-continuous reference path for planning in structured environments. The center points of the lanes form a polyline which can be pre-processed via optimization [17] and spline-based interpolation and smoothing [18]. Similarly, [13], [19] generate smooth reference paths, which consider static obstacles. Optimization-based offline process-

ing approaches for race track applications are proposed in [20], [21], which generate an optimized racing line. Therein, the existence of a reference path with a unique transformation is assumed a-priori. The work in [22] addresses the issue of singularities and presents an offline approach by formulating a nonlinear programm (NLP) that explicitly pushes the evloute of the reference curve off the borders of a race track. The approach ensures that the singular region is outside of the track boundary, however, NLPs are computationally expensive and are often intractable for online use.

B. Contributions

Although curvilinear coordinate frames are well established for motion planning, previous works cannot guarantee that the used reference path ensures uniqueness of the coordinate representation. Approaches using NLPs are often intractable for online applications, which can impede planning algorithms that rely on a swift generation of an appropriate reference path. Therefore, we propose an efficient, iterative reference path adaptation scheme to ensure robust and correct transformations for all points within a given subset of the map. More specifically, our work provides the following contributions:

- we ensure for the first time that the coordinate transformation is unique for all points within a given area of the road network;
- we ensure that the reference path is sufficiently smooth for motion planning;
- we present a sweep-line-based method to efficiently compute the unique projection domain;
- we evaluate our approach on real-world maps from the CommonRoad [23] scenario database.

The remainder of this paper is structured as follows: Sec. II introduces necessary preliminaries and Secs. III, IV describe our approach. In Sec. V, we demonstrate the effectiveness of our approach using numerical experiments. Finally, we draw conclusions in Sec. VI.

II. Preliminaries

A. Notations and Definitions

In this study, vectors and scalars $x \in \mathbb{R}^n$ are denoted by lowercase letters, sets $\mathcal{S} \subset \mathbb{R}^n$ by caligraphic letters, and ordered lists/sequences $\mathbf{L} = (L_1, \dots, L_n)$ by bold uppercase letters. Empty lists and sets are denoted by $\emptyset$. We use $\exists!$ to refer to a unique existential quantification. We adhere to the notations $\underline{\square}$ and $\overline{\square}$ to represent the lower and upper bounds of the possible values of a variable $\square$, respectively.

Definition 1 (Polyline):
A polyline $\mathbf{P} = (p_1, \dots, p_n)$ is a piecewise linear curve which is defined by a sequence of points $p_i = (x_i, y_i)^T, i \in \{1, \dots, n\}$.

We denote the length of a segment of a polyline by Δs_i, the unit tangent vector by $t_i \in \mathbb{R}^2$ and the unit normal vector by $n_i \perp t_i$. The signed curvature of a parametrized curve

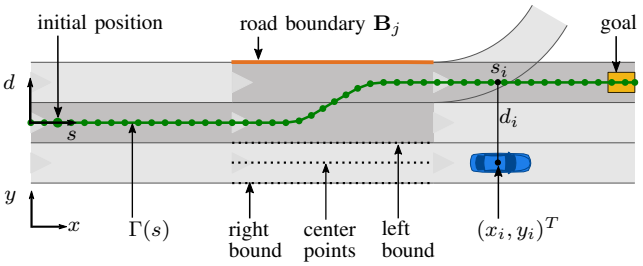


Fig. 2: Environment representation: The road network is described using lanelets. A reference path $\Gamma(s)$ is obtained using the center points of lanelets. The sequence of lanelets $\mathbf{R}$ containing $\Gamma(s)$ are referred to as route lanelets $L_i \in \mathbf{R}$.

$h(t) = (x(t), y(t))^T$ can be computed via [24, p. 245]

$$\kappa = \frac{x(t)'y(t)'' - y(t)'x(t)''}{(x(t)'^2 + y(t)'^2)^{3/2}}. \tag{1}$$

For a (non-uniformly) sampled polyline, the required derivatives can be approximated by means of first central and second finite difference [25]. Without loss of generality, $\kappa > 0$ represents a left bending curve and $\kappa < 0$ represents a right bending curve. We refer to the side towards which the curve bends as the *inner* side of the curve.

Definition 2 (B-Spline):
A B-spline is a piecewise polynomial curve of degree p_B. Given a list of knots $\mathbf{U} = (u_0, \dots, u_m)$ and a list of control points $\mathbf{P} = (p_0, \dots, p_n)$, the spline curve is defined as

$$C(u) = \sum_{i=0}^{n} N_{i,p_B}(u)\, p_i, \quad u \in \mathbb{R},\ p_i \in \mathbb{R}^2, \tag{2}$$

where each basis function $N_{i,p_B}(u)$ is non-zero over the knot interval $u \in [u_i, u_{i+p_B+1})$ and zero otherwise. Thus, each $N_{i,p_B}(u)$ only affects the spline curve $C(u)$ on this interval. Basis functions of degree $p_B > 0$ are recursively defined via Cox-de-Boor's formula (cf. [26, pp. 51-52]).

B. Environment Representation

We consider motion planning problems in structured environments, where a map provides necessary information about the drivable space. Maps are specified by the CommonRoad format [23] [27], which models the road network as a set of lanelets $\mathcal{L} = \{L_i | i \in \{1 \dots n\}\}$ [28] (see Fig. 2). A lanelet L_i is defined by its left and right bound, where each bound is represented as a polyline. The center points can be retrieved from both bounds. Addtionally, each lanelet contains attributes about its spatial relations to surrounding lanelets, e.g, successors and lateral adjacencies. We assume that laterally adjacent lanelets share their adjacency over their entire length, as specified in [27, Sec. III-E.6-7]. Moreover, we define a lanelet bound as a road boundary $\mathbf{B}_j$ if it is not shared by two laterally adjacent lanelets (cf. Fig. 2). A reference path $\Gamma(s) : \mathbb{R} \to \mathbb{R}^2$ from an initial position to a goal position is obtained using the center points of the lanelets (cf. Fig. 2). For lanelets with successor relationships, their center points are concatenated and for lane changes, the center points of two lanelets can be connected via a transition

function (e.g., using splines or polynomials), as shown in Fig. 2. We introduce the sequence of lanelets which contain $\Gamma(s)$ as a route $\mathbf{R} = (L_1, \dots, L_n)$, where we reorder the list indices such that L_1 is the lanelet containing the start of $\Gamma(s)$. We refer to a lanelet $L_i \in \mathbf{R}$ as a route lanelet.

Definition 3 (Curvilinear Coordinate Frame [29, p. 2]): *A curvilinear coordinate frame is aligned with a reference path Γ. Therein, a position $(x, y)^T$ in the global Cartesian frame is expressed in terms of the arclength s along Γ and the orthogonal deviation d to $\Gamma(s)$. The representation of a Cartesian point in the curvilinear coordinate frame is obtained by a function $T_\Gamma : (x, y) \in \mathbb{R}^2 \to (s, d) \in \mathbb{R}^2$.*

The function T_Γ is not bijective for all Cartesian points $(x, y)^T$, i.e., there exist points for which the representation is not unique, as discussed in Sec. I. To formally define the set of points for which uniqueness is guaranteed, we introduce the projection domain (see Fig. 1).

Definition 4 (Projection domain $\mathcal{P}_\Gamma$ [30, p. 3047]): *The unique projection domain $\mathcal{P}_\Gamma \subset \mathbb{R}^2$ is the set of Cartesian points for which the projection function is bijective, i.e.,* $\mathcal{P}_\Gamma = \{(s, d) \in \mathbb{R}^2 | \exists! (x, y) \in \mathcal{P}_\Gamma : T_\Gamma(x, y) = (s, d)\}$

C. Problem Statement

Let $\mathbf{P}_0$ be the polyline of the initial reference path with corresponding route lanelets $\mathbf{R}$. Moreover, we have a set of m provided road boundaries $\mathbf{B}_j$. We want to adapt $\mathbf{P}_0$ to generate an alternative reference path $\mathbf{P}_{\mathrm{ref}}$ such that the projection domain $\mathcal{P}_\Gamma$ of the associated curvilinear coordinate system encloses all provided boundaries, i.e.,

$$\forall j \in \{1, \dots, m\} : \mathbf{B}_j \subset \mathcal{P}_\Gamma \tag{3}$$

III. Reference Path Adaptation

A. Overview

Our procedure is based on viewing the polyline $\mathbf{P}_0$ as the control polyline of a cubic B-spline, which is recursively refined via a subdivision scheme (cf. Lemma 1). We combine this idea with a resampling step to iteratively reduce the curvature (cf. Lemma 3) and produce a smooth C^2 path. The overall concept is illustrated in Figs. 4-5 and explained in Sec. III-B. We motivate our choice of approximating cubic B-splines due to their desirable properties [31]:

1) Cubic B-splines are curvature-continuous by nature, i.e., we do not need to enforce C^2-continuity explicitly.
2) B-splines can be modified locally; changing a control point p_i only affects the knot interval $[u_i, u_{i+p_B+1})$.
3) Strong convex hull property: B-spline segments are bounded by the convex hull of $p_B + 1$ control points.

If the original polyline $\mathbf{P}_0$ is extracted from a map, it may contain unnecessarily many points, which result in noisy curvatures. For numerical stability one can simplify $\mathbf{P}_0$ beforehand, e.g., by using the Ramer-Douglas-Peucker algorithm [32].

Algorithm 1 Iterative reference path adaptation

Input: Polyline $\mathbf{P}_0$, Route $\mathbf{R}$, Lanelets $\mathcal{L}$, global abs. curvature limit $\overline{\kappa}$, max. iterations $\overline{n}_{\mathrm{iter}}$, num. refinements k
Output: Adapted polyline $\mathbf{P}_{\mathrm{ref}}$

1: Set of partitions $\mathcal{G} \leftarrow$ GetPartitions($\mathbf{P}_0$), ▷ Fig. 3
2: Boundaries $\mathcal{B} \leftarrow$ GetBoundaries($\mathbf{G}$, $\mathbf{R}$, $\mathcal{L}$), ▷ Fig. 3
3: iter $= 0$
4: **for each** partition $\mathbf{G}_m$ in $\mathcal{G}$ **do**
5: dist$_{p_i} \leftarrow$ DistToBoundary(p_i, $\mathbf{B}_j$), $\forall p_i \in \mathbf{G}_m$
6: $\tilde{p}_i \leftarrow$ point with the maximum ratio $\overline{\varrho}$, ▷ cf. (4)
7: $\overline{\kappa}_{\mathbf{G}_m} \leftarrow$ absolute curvature limit for $\mathbf{G}_m$, ▷ cf. (5)
8: **while** iter $\leq \overline{n}_{\mathrm{iter}}$ **do**
9: $\mathbf{G}_m \leftarrow$ CurveSubdivision($\mathbf{G}_m$, k), ▷ Fig. 4
10: **if** $|\kappa_{p_i}| < \overline{\kappa}_{\mathbf{G}_m}, \quad \forall p_i \in \mathbf{G}_m$ **then**
11: dist$_{p_i} \leftarrow$ DistToBoundary(p_i, $\mathbf{B}_j$), $\forall p_i \in \mathbf{G}_m$
12: **if** $\varrho_{p_i} < 1, \quad \forall p_i \in \mathbf{G}_m$ **then**
13: **break**
14: **end if**
15: **end if**
16: $\widehat{\mathbf{G}}_m \leftarrow$ Resampling($\mathbf{G}_m$, Δs), ▷ Fig. 5
17: intersect $\leftarrow$ CheckConvexHulls($\widehat{\mathbf{G}}_m$, $\mathbf{B}_j$), ▷ Fig. 6
18: **if** intersect **then break**
19: **else** $\mathbf{G}_m = \widehat{\mathbf{G}}_m$
20: **end if**
21: iter $=$ iter $+ 1$
22: **end while**
23: **end for**

B. Iterative Reference Path Adaptation

Subsequently, we explain the individual steps of our approach, which is summarized in Alg. 1. We are interested in retrieving the road boundaries on the *inner* side of the curve to determine the minimum required extent of the projection domain for all parts of the curve. We first partition $\mathbf{P}_0$ at its inflection points, i.e., where the sign of the curvature changes. Additionally, we partition $\mathbf{P}_0$ at the transition between two succeeding lanelets. Thus, each partition is associated to one road section of laterally adjacent lanelets (cf. Fig. 3). After determining the set $\mathcal{G} = \{\mathbf{G}_1, \dots, \mathbf{G}_m\}$ of partitions (line 1), we determine a boundary $\mathbf{B}_j$ (line 2) for each $\mathbf{G}_m$, which we use to limit the maximum allowed curvature per partition. The boundary $\mathbf{B}_j$ is determined by traversing the adjacent lanelets of $\mathbf{R}_{\mathbf{G}_m}$ until the inner boundary is reached, where $\mathbf{R}_{\mathbf{G}_m}$ is the subset of route lanelets which enclose $\mathbf{G}_m$. For example, for a "left bending" partition ($\kappa_{p_i} > 0, \ \forall p_i \in \mathbf{G}_m$), we iterate over all left adjacancies of $\mathbf{R}_{\mathbf{G}_m}$. Thus, we obtain for each partition $\mathbf{G}_m$ a boundary $\mathbf{B}_j$ which is located on the *inner* side (cf. Fig. 3). Since each $\mathbf{G}_m$ lies within one road section of laterally adjacent lanelets, $\mathbf{B}_j$ is the only corresponding boundary.

Next, we determine an absolute curvature limit $\overline{\kappa}_{\mathbf{G}_m}$ for each $\mathbf{G}_m$, which is determined from the distances of each point $p_i \in \mathbf{G}_m$ to the boundary (lines 5-7). The function DistToBoundary computes the distance dist_{p_i} along the normal n_i to the boundary $\mathbf{B}_j$ (cf. Fig. 3). To account for numerical imprecisions, we over-approximate the distance by a small user-defined value ε. Then, we determine the

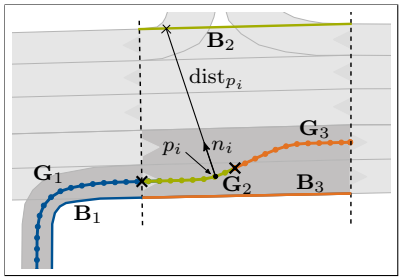


Fig. 3: Exemplary polyline with three partitions $\mathbf{G}_1, \mathbf{G}_2, \mathbf{G}_3$. The route lanelets $\mathbf{R}$ are highlighted in dark gray. For each partition we determine the corresponding road boundary polylines $\mathbf{B}_1, \mathbf{B}_2, \mathbf{B}_3$, which are colored according to the associated partition. The road section of laterally adjacent lanelets containing $\mathbf{G}_2$ and $\mathbf{G}_3$ is marked by the dashed lines.

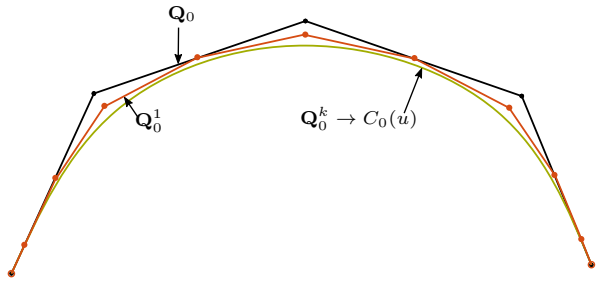


Fig. 4: Refinement step using recursive subdivision for an original polyline $\mathbf{Q}_0$. The polyline $\mathbf{Q}_0^1$ (orange) is the results after one refinement. For $k \to \infty$ the refined polyline converges to the cubic B-spline of the original polyline, i.e., $\mathbf{Q}_0^k \to C_0(u)$ (green).

curvature ratio which we define as

$$\varrho_{p_i} = |\kappa_{p_i}| \ \text{dist}_{p_i}, \quad p_i \in \mathbf{G}_m, \tag{4}$$

and retrieve the point $\tilde{p}_i$ with the maximum ratio $\overline{\varrho}$. We compute the absolute curvature limit for the partition $\mathbf{G}_m$ as

$$\overline{\kappa}_{\mathbf{G}_m} = \min(\text{dist}_{\tilde{p}_i}^{-1}, \ \overline{\kappa}), \tag{5}$$

where $\text{dist}_{\tilde{p}_i}$ is the distance of the point $\tilde{p}_i$ to the boundary and $\overline{\kappa}$ is a user-defined global absolute curvature limit.

We now describe our iterative procedure starting in line 8 of Alg. 1. Each iteration consists of two steps: a refinement step which smooths the polyline $\mathbf{G}_m$ and a resampling step which reduces its curvature. We begin with the refinement step (line 9), for which we use the Lane-Riesenfeld subdivision procedure [33]. For illustration purposes, we consider the exemplary polyline $\mathbf{Q}_0$ shown in Fig. 4, where we recursively apply the cubic B-spline curve subdivision algorithm of [33]. In Fig. 4 we show the resulting polyline $\mathbf{Q}_0^1$ after one recursion and the refined polyline $\mathbf{Q}_0^k$ after k recursions.

Lemma 1 (Cubic B-spline curve subdivision):
The refinement converges to the limit curve $C_0(u) = \lim_{k\to\infty} \mathbf{Q}_0^k$*, which is the cubic B-Spline that uses* $\mathbf{Q}_0$ *as its control polyline. Thus, for* $k \to \infty$*, we obtain a refined polyline* $\mathbf{Q}_0^k$ *which approximates the* C^2*-continuous curve* $C_0(u)$. □

Proof: The proof follows directly from [33, Thm. 3.1] ■

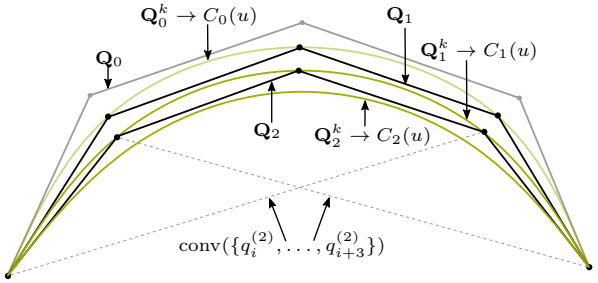


Fig. 5: Illustration showing two iterations of our procedure: The original polyline $\mathbf{Q}_0$ is refined to obtain its B-spline approximation $\mathbf{Q}_0^k$ (see Fig. 4). Then, we resample $\mathbf{Q}_0^k$ equidistantly with a step size Δs while keeping the end points fixed and obtain the polyline $\mathbf{Q}_1$. In the next iteration, this polyline again serves as a control polyline for its B-Spline approximation $\mathbf{Q}_1^k$ after refinement. After two iterations, we obtain $\mathbf{Q}_2^k$, which is enclosed by the convex hulls $\text{conv}(\{q_i^{(2)}, \ldots, q_{i+3}^{(2)}\})$ of its control polyline $\mathbf{Q}_2$.

The function CURVESUBDIVISION($\mathbf{G}_m, k$) performs the recursion k times and produces the refined polyline $\mathbf{G}_m$. Before proceeding with the resampling step, we first check if the curvature of the refined polyline fulfils our desired criteria for the projection domain $\mathcal{P}_\Gamma$ and if the iteration can thus be terminated (lines 10-15). To this end, we first check whether the absolute curvature $|\kappa_{p_i}|$ for all points $p_i \in \mathbf{G}_m$ is lower than the curvature limit $\overline{\kappa}_{\mathbf{G}_m}$ (cf. (5)). If the condition is fulfilled, we additionally check whether the ratio ϱ_{p_i} satisfies

$$\varrho_{p_i} < 1 \ , \ \forall p_i \in \mathbf{G}_m. \tag{6}$$

Lemma 2 (Termination criterion):
If condition (6) *is met for all points* p_i *in* $\mathbf{G}_m$*, the projection domain extends beyond* $\mathbf{B}_j$ *and the iteration can be terminated.* □

Proof: We know that $\mathbf{B}_j$ *is the only road boundary on the inner side of* $\mathbf{G}_m$ *(cf. Fig. 3) and thus determines the lateral limit of the road network for the lanelets in* $\mathbf{R}_{\mathbf{G}_m}$*. By enforcing* (6)*, the curvature radius at any point* p_i *is greater than its distance* dist_{p_i} *to the boundary* $\mathbf{B}_j$*. In this case, the radius of the local oscullating circle extends beyond* $\mathbf{B}_j$ *for all points* $p_i \in \mathbf{G}_m$*, which means that the projection domain encloses* $\mathbf{B}_j$ *and we can terminate the iteration.* ■

We now explain the resampling step of our iteration, which reduces the curvature of the refined polyline. For illustration purposes, we continue the previous example from Fig. 4 and show two steps of our iteration in Fig. 5. $\mathbf{Q}_0^k$ is the refined polyline after recursively subdividing the original polyline $\mathbf{Q}_0$ k times. Now, we resample $\mathbf{Q}_0^k$ equidistantly with a segment length of Δs while keeping the end points fixed and obtain $\mathbf{Q}_1$, which we use in the next iteration. Refining $\mathbf{Q}_1$ yields $\mathbf{Q}_1^k$, i.e., an approximation of its B-spline $C_1(u)$ (cf. Lemma 1). Similary, a second iteration yields $\mathbf{Q}_2^k$.

Lemma 3 (Iterative curvature reduction):
After each iteration iter *consisting of a refinement and a resampling step, the curvature of the resulting curve* $\mathbf{Q}_{\text{iter}}^k$ *is reduced.* □

Proof: The proof follows from Lemma 1 and the strong convex hull property of B-splines (see Sec. III-A). Let us

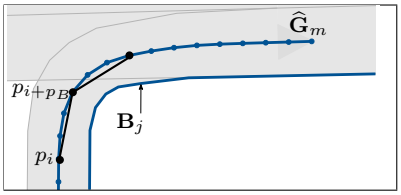


Fig. 6: For $\widehat{\mathbf{G}}_m$, we check if the line $\overline{p_i p_{i+p_B}}$ intersects with the road boundary $\mathbf{B}_j$ to ensure that the resulting curve after subdivision will not cross the boundaries of the road network.

consider the polyline $\mathbf{Q}_1 = (q_1^{(1)}, \dots, q_n^{(1)})$ *in Fig. 5. After subdiving* $\mathbf{Q}_1$ k *times, we obtain the refined polyline* $\mathbf{Q}_1^k$ *approximating the cubic B-spline* $C_1(u)$*. From the convex hull property for cubic B-Splines we know that every point of* $C_1(u)$ *must lie within the union of the convex hulls of four control points, i.e.,*

$$\forall q_j^{(1)} \in \mathbf{Q}_1^k : \; q_j^{(1)} \in \bigcup_{\substack{q_i^{(1)} \in \mathbf{Q}_1 \\ i \in \{0,\dots,n-3\}}} \operatorname{conv}(\{q_i^{(1)}, \dots, q_{i+3}^{(1)}\}). \tag{7}$$

For the next iteration, $\mathbf{Q}_2$ *is obtained by resampling* $\mathbf{Q}_1^k$*. From* (7) *we know that the refined polyline* $\mathbf{Q}_2^k$ *must lie within the union of convex hulls of* $\mathbf{Q}_2$*. Since the convex hulls of* $\mathbf{Q}_2$ *are strictly placed on the inner side of* $C_1(u)$*, the resulting polyline* $\mathbf{Q}_2^k$ *also lies on the inner side of* $C_1(u)$ *and thus its maximum curvature* $|\overline{\kappa}|$ *is reduced.* ■

Figuratively speaking, we gradually push the control points of a B-spline towards the *inner* side of the curve. The chosen resampling step size Δs determines the rate of the curvature reduction. Before starting the next iteration with the resampled polyline $\widehat{\mathbf{G}}_m$ (line 16), we want to ensure sufficient clearance to the boundary $\mathbf{B}_j$. We use the convex hull property and check whether the line segments $\overline{p_i p_{i+p_B}}$ intersect with the boundary (line 17 and Fig. 6). Only if the line segments $\overline{p_i p_{i+p_B}}$ do not intersect with $\mathbf{B}_j$ we set $\mathbf{G}_m = \widehat{\mathbf{G}}_m$ (lines 18-21).

Theorem 1 (Convergence):
Alg. 1 convergences and terminates when all boundaries $\mathbf{B}_j$ *are enclosed in* $\mathcal{P}_\Gamma$*.* □

Proof: The proof follows from Lemma 2 and Lemma 3. We know that with each iteration, the maximum curvature $|\overline{\kappa}_{\mathbf{G}_m}|$ *of each partition* $\mathbf{G}_m$ *is reduced (cf. Lemma 3). Moreover, the resulting polyline after each iteration lies on the inner side of the previous polyline (cf. Lemma 3), thus* $\operatorname{dist}_{p_i}$ *monotonically decresases. Additionally, we know that the radius of the local oscullating circle, which limits* $\mathcal{P}_\Gamma$*, extends beyond the corresponding boundary* $\mathbf{B}_j$ *for all points* $p_i \in \mathbf{G}_m$ *if condition* (6) *is fulfilled (cf. Lemma 2). Since each iteration step reduces* $|\kappa_{p_i}|$ *and* $\operatorname{dist}_{p_i}$*, condition* (6) *will be reached for all points* p_i *and thus Alg. 1 converges.* ■

IV. Transformation Routine

We now describe how we use the adapted reference path within the transformation routine of the curvilinear coordinate system. To project a point onto $\mathbf{P}_{\mathrm{ref}}$, we use the *Lanelet transformation routine* [28]. This routine matches a Cartesian point to the closest segment of $\mathbf{P}_{\mathrm{ref}}$ and interpolates the curvilinear abscissa s to avoid discontinuities near the discrete vertices (cf. [34, Eq. 6-9]). Finding the closest segment for a point becomes increasingly expensive the more segments $\mathbf{P}_{\mathrm{ref}}$ has. Since the subdivision algorithm of [33] increases the number of points in each recursion, $\mathbf{P}_{\mathrm{ref}}$ can be sampled very densely. Therfore we resample $\mathbf{P}_{\mathrm{ref}}$ adaptively for efficiency during the coordinate transformation: We adjust the segment length for resampling $\mathbf{P}_{\mathrm{ref}}$ inversely proportional to the curvature with a user-defined minimum $\Delta\underline{s}$ and maximum $\Delta\overline{s}$ step size, i.e.,

$$\Delta s_i = 1/(\alpha\, \kappa_i), \qquad \Delta\underline{s} \le \Delta s_i \le \Delta\overline{s}.$$

We set the scaling factor α such that the minimum step size $\Delta\underline{s}$ is applied to the point with the maximum curvature $\overline{\kappa}_{\mathrm{ref}}$, i.e., $\alpha = 1/(\Delta\underline{s}\, \overline{\kappa}_{\mathrm{ref}})$.

Proposition 1 (Adaptive resampling of $\mathbf{P}_{\mathrm{ref}}$):
By adjusting $\Delta\underline{s}$ *we can make the approximation error* e *of the resampled reference path arbitrarily small to ensure that the properties of* $\mathbf{P}_{\mathrm{ref}}$ *are preserved.*

Proof: Let us consider a circular arc segment with a curvature $\overline{\kappa}_{\mathrm{ref}}$*. By approximating the arc with line segments of length* $\Delta\underline{s}$ *the chord error can be computed as*

$$e = \frac{1}{\overline{\kappa}_{\mathrm{ref}}} \cdot \left[1 - \cos\left(\frac{\overline{\kappa}_{\mathrm{ref}} \Delta\underline{s}}{2}\right)\right],$$

which follows from the trigonometry of chord lengths for circular arcs [35, pp. 57-59.]. Thus, we can set $\Delta\underline{s}$ *to bound the chord error to a desired maximum value* $e < e_{\max}$*. Since we adjust the error bound for an arc segment with the maximum curvature* $\overline{\kappa}_{\mathrm{ref}}$*, we can guarantee that the error* e *is bounded for all points in the resampled reference path.* ■

For the interested reader, we also provide an efficient algorithm for computing the polygon of the unique projection domain $\mathcal{P}_\Gamma$ (e.g., as visualized in Fig. 1) in the Appendix.

V. Numerical Experiments

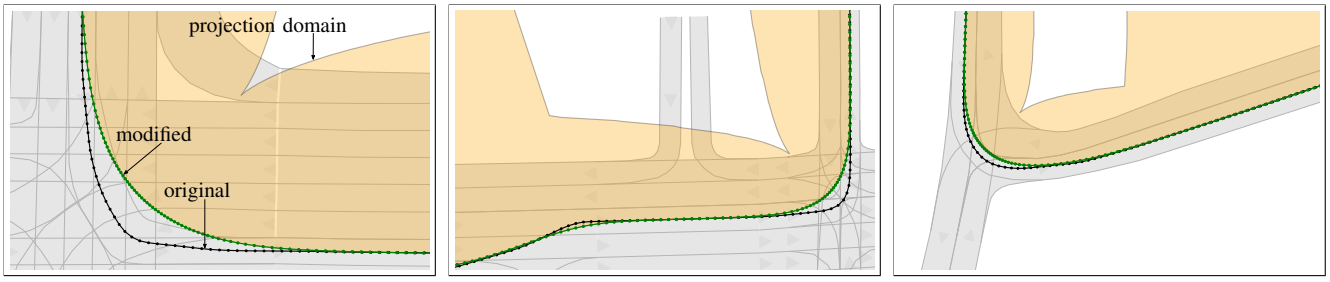
We evaluate our approach numerically using real-world maps from the CommonRoad [23] benchmark suite. To generate the initial reference path, we utilize the CommonRoad Route Planner[1]. The algorithm for the projection domain computation is implemented in C++ and the algorithm for adapting the reference path is implemented in Python[1]. All experiments are performed on a single thread of a 1.80 GHz Intel Core™ i7-10510U CPU with 32 GB of memory.

A. Reference Path Adaptation

We demonstrate our method using the maps from the following CommonRoad scenarios.

I : `USA_Lanker-2_5_T-1`
II : `USA_Peach-2_1_T-1`

[1]Code provided at: `https://commonroad.in.tum.de/`



(a) Scenario I (b) Scenario II (c) Scenario III

Fig. 7: Our approach modifies the reference path such that projection domain covers the relevant parts of the road network.

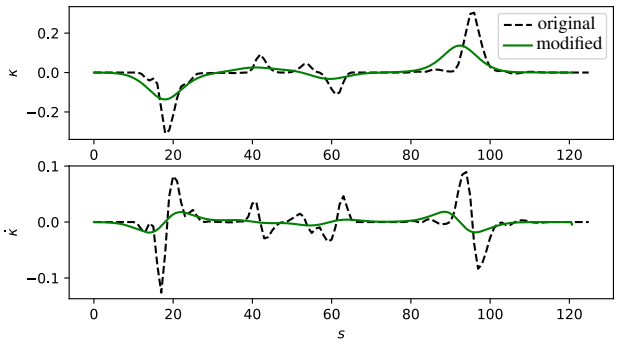


Fig. 8: Scenario II: Curvature κ and curvature rate $\dot{\kappa}$ of the original and modified reference path.

TABLE I: QUANTITATIVE EVALUATION

Scenario	I	II	III
(a) Curvature			
$\Delta\lvert\overline{\kappa}\rvert$	0.164	0.172	0.050
$\Delta\lvert\overline{\dot{\kappa}}\rvert$	0.079	0.107	0.019
(b) Deviations			
Δs [m]	4.078	4.22	1.535
Δd [m]	1.107	0.332	0.067
$\Delta\theta$ [rad]	0.118	0.287	0.011
(c) Runtime [ms]			
reference path	73.44	87.69	167.59
projection domain	0.873	1.326	1.005

III : `ZAM_Tjunction-1_42_T-1`

Each of the chosen scenarios showcases a reference path for a tight turning maneuver in an intersection, which is particularly challenging for curvilinear representations (see Fig. 7). We set the number of recursions for each refinement step to $k = 5$ (see line 9, Alg. 1) and the step size for the resampling step to $\Delta s = 2$ (see line 16, Alg. 1). For all three scenarios, we observe that our approach modifies the original reference path extracted from the center points of the lanelets such that the unique projection domain covers the relevant parts of the road network along the route. Moreover, we see how the reference path is sampled more densely in parts where the curvature is high, thus preserving the smoothness of the path and simultaneously reducing the number of segments for an efficient coordinate transformation (cf. Sec. IV).

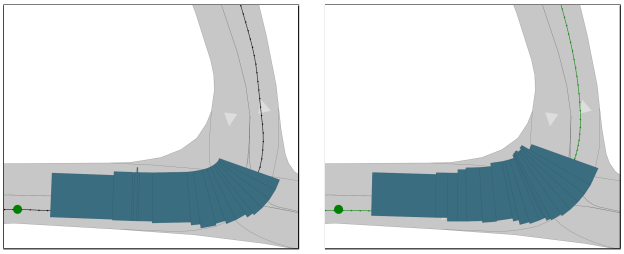
To investigate the smoothness of the resulting reference path, we compare the profiles of the curvature κ and curvature change $\dot{\kappa}$ for scenario II (see Fig. 8). The modified reference path is smooth and both the curvature and curvature change are continuous (C^2). In comparison, the profiles of the original reference path are very noisy and especially the curvature change shows high peaks. The improvements for κ and $\dot{\kappa}$ are evaluated in Tab. I (a) for all three scenarios, which shows that in all cases our approach reduces the maximum absolute values for κ and $\dot{\kappa}$. Low curvature rates of the reference path are required to ensure curvature-continuity of trajectories which are transformed from the curvilinear coordinate frame to the Cartesian frame and vice versa [1].

Our procedure modifes the path only locally, while the overall shape of the initial reference path is mostly preserved. We evaluate the deviation from the original path in terms of the changed path length Δs, as well as the average lateral deviation Δd and average deviation of the orientation $\Delta\theta$ in Tab. I. Our approach reduces the overall path length, due to the reduced curvature. The average lateral deviation as well as the deviation in the orientation of the path are minor, i.e., the overall shape of the original reference path is retained. One might argue that the modified reference path cuts corners at intersections, which may result in undesirable maneuvers when following this path with a local trajectory planner. However, keeping a desired lane can easily be enforced in a trajectory planner, for example by adding a high cost term for lane deviations. The runtimes for both the reference path adaptation and the projection domain computation in Tab. I (c) show that our method is efficient and suitable for online applications.

B. Combination with Motion Planning

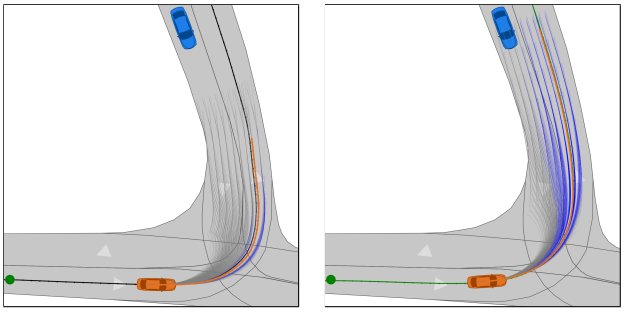
Next, we demonstrate the efficacy of our approach for two downstream planning tasks: computing the reachable set of the ego vehicle and planning feasible trajectories.

1) Reachable Set Computation: We compute the collision-free reachable set of the ego vehicle using the polytopic set propagation method [36] implemented in CommonRoad-Reach [37]. The over-approximative reachable sets are computed in the curvilinear frame for a horizon of 30 time steps. Fig. 9 shows the reachable sets in the position domain at time step $k = 27$ after transforming the sets to Cartesian coordinates. We observe that the reachable set in Fig. 9a does not cover the entire



(a) Original reference path. (b) Modified reference path.

Fig. 9: Scenario III: Computed reachable sets of the ego vehicle at time step $k = 27$.



(a) Original reference path. (b) Modified reference path.

Fig. 10: Scenario III: Trajectory samples at $k = 30$, with feasible trajectories (blue), infeasible trajectories (gray) and the optimal trajectory (orange).

width of the road in the apex of the curve, in contrast to Fig. 9b. Hence, our approach is necessary to ensure that the reachable set can be represented correctly for the entire feasible space of the road.

2) Sampling-based planner: We further demonstrate our approach with a planner that samples a set of trajectories in the curvilinear frame [1]. Fig. 10 shows the set of 360 trajectory samples at planning step $k = 30$ for a planning horizon of 5 s. With the original reference path (see Fig. 10a), 305 trajectories are infeasible, as they either can not be transformed between the coordinate frames or exceed the kinematic limits for the curvature and curvature rate of the ego vehicle. In contrast, our modified reference path ensures that all samples can be transformed correctly and the number of infeasible trajectories is reduced to 170 (see Fig. 10b). The remaining infeasible samples stem from the high lateral accelerations and yaw rates required for the tight turning maneuver in this scenario. Both the enhanced projection domain as well as the improved smoothness and continuity of the reference path contribute to a higher sample efficiency and feasibility rate.

VI. Conclusions

We present for the first time an optimization-free approach for adapting a reference path for motion planning to ensure that the representation of points in the curvilinear coordinate frame is unique for a given subset of the map. Our iterative procedure is based on a curve subdivision scheme which gradually reduces the curvature and simultaneously produces smooth (C^2) paths. The approach is model-free, in contrast to using optimization techniques, and thus well suited as a robust method for generating a suitable reference path for curvilinear coordinate frames. Our numerical experiments using CommonRoad scenarios demonstrate the efficacy of the method when combined with downstream planning tasks, e.g., for computing the reachable set or improving the feasibility rate of trajectory planners.

Algorithm 2 Computation of projection domain $\mathcal{P}_\Gamma$

Input: Polyline $\mathbf{P}_{\text{ref}}$, discrete curvature vector κ_{ref}, maximum lateral distance $\overline{d}_\mathcal{P}$;
Output: Projection domain polygon $\mathcal{P}_\Gamma$
1: Left normal line segments $\mathbf{N}_\text{L} \leftarrow \emptyset$;
2: Right normal line segments $\mathbf{N}_\text{R} \leftarrow \emptyset$;
3: **for each** point p_i in $\mathbf{P}_{\text{ref}}$ **do**
4: $\quad \mathbf{N}_\text{L} \leftarrow$ add $n_i \cdot \overline{d}_\mathcal{P}$
5: $\quad \mathbf{N}_\text{R} \leftarrow$ add $n_i \cdot (-\overline{d}_\mathcal{P})$
6: **end for**
7: Left border $\mathbf{P}_{\mathcal{P},\text{L}} \leftarrow$ SweepLineIntersect ($\mathbf{N}_\text{L}$)
8: Right border $\mathbf{P}_{\mathcal{P},\text{R}} \leftarrow$ SweepLineIntersect ($\mathbf{N}_\text{R}$)
9: $\mathcal{P}_\Gamma \leftarrow$ CreatePolygon($\mathbf{P}_{\mathcal{P},\text{L}}, \mathbf{P}_{\mathcal{P},\text{R}}$)

Appendix

Algorithm for computing the projection domain $\mathcal{P}_\Gamma$

The vertices of the polygon $\mathcal{P}_\Gamma$ are obtained by the concatenation of two lists $\mathbf{P}_{\mathcal{P},\text{L}}$ and $\mathbf{P}_{\mathcal{P},\text{R}}$, which we denote as the left border and right border, respectively. The borders are computed based on the intersections of the normal vectors n_i of the polyline $\mathbf{P}_{\text{ref}}$, as shown in Fig. 11. For each n_i, we compute the set of intersection points

$$\mathcal{A}_i = \{n_i \cap n_j \mid j \neq i\} \tag{8}$$

with the other vectors n_j. Afterwards, we determine for each n_i the point

$$\underline{a}_i = \underset{a \in \mathcal{A}_i}{\text{argmin}} \|a - p_i\|_2, \quad p_i \in \mathbf{P}_{\text{ref}}, \tag{9}$$

i.e., we retrieve the intersection point $a \in \mathcal{A}_i$ closest to $p_i \in \mathbf{P}_{\text{ref}}$. The points $\underline{a}_i$ are added to the corresponding border $\mathbf{P}_\mathcal{P}$ of the projection domain $\mathcal{P}_\Gamma$. The procedure is summarized in Alg. 2. To efficiently compute the intersections we use the Bentley-Ottmann sweep-line algorithm [38], which has a complexity of $\mathcal{O}(n + k)log(n)$ for n lines with k intersections. We scale the normals by a user-defined value $\overline{d}_\mathcal{P}$ (line 4–5). This value determines the maximum lateral distance of the vertices of $\mathcal{P}_\Gamma$ to $\mathbf{P}_{\text{ref}}$ and is required for cases where there are no intersections between the normals, e.g., if $\mathbf{P}_{\text{ref}}$ is a straight line. The function `SweepLineIntersect` implements (9) and returns the borders $\mathbf{P}_{\mathcal{P},\text{L}}$ and $\mathbf{P}_{\mathcal{P},\text{R}}$ which are used to construct $\mathcal{P}_\Gamma$ (line 7–9).

Acknowledgments

This work was funded by the German Federal Ministry of Education and Research (BMBF) within the *Munich Cluster for the Future of Mobility in Metropolitan Regions (MCube)* under grant 03ZU1105AA. We thank Evald Nexhipi for assisting with implementing the sweep-line algorithm.

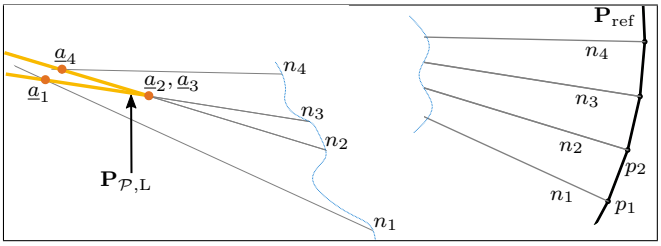


Fig. 11: Example for constructing the left border $\mathbf{P}_{\mathcal{P},\mathrm{L}}$ (orange line) for a polyline $\mathbf{P}_{\mathrm{ref}}$ (bold black line) with normals $n_1 \dots n_4$. The intersection points $\underline{a}_i \in \mathcal{A}_i$ (orange points) of the normals n_i are determined via the sweep-line algorithm. They are used to construct $\mathbf{P}_{\mathcal{P},\mathrm{L}}$ and form the vertices of the polygon of the projection domain $\mathcal{P}_\Gamma$.

REFERENCES

[1] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét frame," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2010, pp. 987–993.

[2] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *Proc. of the IEEE Intell. Veh. Symp.*, 2012, pp. 1162–1167.

[3] C. Samson, "Control of Chained Systems Application to Path Following and Time-Varying Point-Stabilization of Mobile Robots," *IEEE Trans. Autom. Control*, vol. 40, no. 1, pp. 64–77, 1995.

[4] B. Gutjahr, L. Gröll, and M. Werling, "Lateral Vehicle Trajectory Optimization Using Constrained Linear Time-Varying MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 6, pp. 1586–1595, 2017.

[5] J. Pegna and F. E. Wolter, "Surface curve design by orthogonal projection of space curves onto free- form surfaces," *Journal of Mech. Design, Transactions of the ASME*, vol. 118, no. 1, pp. 45–52, 1996.

[6] H. Wang, J. Kearney, and K. Atkinson, "Robust and efficient computation of the closest point on a spline curve," in *Proc. of the 5th Int. Conf. on Curves and Surfaces*, 2002, pp. 397–406.

[7] F. Bayer and J. Hauser, "Trajectory optimization for vehicles in a constrained environment," in *Proc. of the IEEE Conf. on Decision and Control*, 2012, pp. 5625–5630.

[8] B. Li, Y. Ouyang, L. Li, and Y. Zhang, "Autonomous Driving on Curvy Roads Without Reliance on Frenet Frame: A Cartesian-Based Trajectory Planning Method," *IEEE Trans. Intell. Transp. Syst.*, pp. 1–13, 2022.

[9] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha - A local, continuous method," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 450–457.

[10] C. Pek and M. Althoff, "Computationally Efficient Fail-safe Trajectory Planning for Self-driving Vehicles Using Convex Optimization," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Syst.*, 2018, pp. 1447–1454.

[11] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Trans. Intell. Veh.*, vol. 6, no. 2, pp. 232–248, 2020.

[12] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *Proc. of the IEEE Int. Conf. on Intell. Robots and Systems,*, 2009, pp. 1879–1884.

[13] T. Gu, J. Snider, J. M. Dolan, and J. W. Lee, "Focused trajectory planning for autonomous on-road driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2013, pp. 547–552.

[14] G. Würsching and M. Althoff, "Sampling-Based Optimal Trajectory Generation for Autonomous Vehicles Using Reachable Sets," in *Proc. of the IEEE Conf. on Intell. Transp. Syst.*, 2021, pp. 828–835.

[15] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 4, pp. 1599–1616, 2012.

[16] Y. Sun, D. Ren, S. Lian, S. Fu, X. Teng, and M. Fan, "Robust Path Planner for Autonomous Vehicles on Roads with Large Curvature," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 2503–2510, 2022.

[17] 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.

[18] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, "Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications," *IEEE/ASME Trans. on Mechatronics*, vol. 21, no. 2, pp. 740–753, 2015.

[19] X. Li, Z. Sun, A. Kurt, and Q. Zhu, "A sampling-based local trajectory planner for autonomous driving along a reference path," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 376–381.

[20] J. Zubaca, M. Stolz, and D. Watzenig, "Smooth Reference Line Generation for a Race Track with Gates based on Defined Borders," in *Proc. of the IEEE Intell. Veh. Symp.*, 2020, pp. 604–609.

[21] A. Heilmeier, A. Wischnewski, L. Hermansdorfer, J. Betz, M. Lienkamp, and B. Lohmann, "Minimum curvature trajectory planning and control for an autonomous race car," *Vehicle System Dynamics*, pp. 1–31, 2019.

[22] R. Reiter and M. Diehl, "Parameterization Approach of the Frenet Transformation for Model Predictive Control of Autonomous Vehicles," in *Proc. of the Europ. Control Conf.*, 2021, pp. 2414–2419.

[23] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intell. Veh. Symp.*, 2017, pp. 719–726.

[24] K. Königsberger, *Analysis 2*. Springer-Verlag, 2013.

[25] A. K. Singh and B. S. Bhadauria, "Finite Difference Formulae for Unequal Sub-Intervals Using Lagrange's Interpolation Formula," *Int. Journal of Mathematical Analysis*, vol. 3, no. 17, pp. 815–827, 2009.

[26] C. De Boor, "On calculating with B-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[27] S. Maierhofer, Y. Ballnath, and M. Althoff, "Map verification and repairing using formalized map specifications," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Syst.*, 2023, pp. 1277–1284.

[28] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intell. Veh. Symp.*, 2014, pp. 420–425.

[29] J. F. Thompson, "General curvilinear coordinate systems," *Appl. Math. and Comp.*, vol. 10, pp. 1–30, 1982.

[30] A. Konyukhov and K. Schweizerhof, "On the solvability of closest point projection procedures in contact analysis: Analysis and solution strategy for surfaces of arbitrary geometry," *Computer Methods in Appl. Mech. and Eng.*, vol. 197, no. 33-40, pp. 3045–3056, 2008.

[31] C. De Boor, *A practical guide to splines*. Springer-Verlag New York, 1978, vol. 27.

[32] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The Int. Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[33] J. M. Lane and R. F. Riesenfeld, "A theoretical development for the computer generation and display of piecewise polynomial surfaces," *IEEE Trans. Patt. Anal. and Mach. Intell.*, no. 1, pp. 35–46, 1980.

[34] E. Héry, S. Masi, P. Xu, and P. Bonnifait, "Map-based curvilinear coordinates for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Syst.*, 2017, pp. 1–7.

[35] I. M. Gelfand and M. Saul, *Trigonometry*. Springer, 2001.

[36] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 6, pp. 1855–1866, 2017.

[37] E. I. Liu, G. Würsching, M. Klischat, and M. Althoff, "CommonRoad-Reach: A toolbox for reachability analysis of automated vehicles," in *Proc. of the IEEE Int. Conf. on Intell. Transp. Syst.*, 2022, pp. 2313–2320.

[38] Bentley and Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Trans. Computers*, vol. 100, no. 9, pp. 643–647, 1979.