2021 IEEE Intelligent Transportation Systems Conference (ITSC)
Indianapolis, USA. September 19-21, 2021

Sampling-Based Trajectory Repairing for Autonomous Vehicles

Yuanfei Lin, Sebastian Maierhofer, and Matthias Althoff

***Abstract*— Ensuring the safety of autonomous vehicles is a challenging task, especially if the planned trajectories do not consider all traffic rules or they are physically infeasible. Since replanning the complete trajectory is often computationally expensive, efficient methods are necessary for resolving such situations. One solution is to deform or repair an initially-planned trajectory, which we call trajectory repairing. Our approach first detects the part of an invalid trajectory that can stay unchanged. Afterward, we use a hierarchical structure and our novel sampling-based algorithm *informed closed-loop rapidly-exploring random trees* (informed CL-RRTs) to efficiently repair the remaining part of the trajectory. We evaluate our approach with different traffic scenarios from the CommonRoad benchmark suite. The computational efficiency is demonstrated by comparing the computation times with those required when replanning the complete trajectory.**

I. Introduction

Autonomous vehicles have to solve the challenging task of finding feasible solutions in partially unknown, complex, and dynamic environments under severe real-time constraints. Recently, machine learning approaches have been widely used in motion planning problems of autonomous vehicles [1], [2], which aim to develop safe driving policies. However, these approaches often lack safety guarantees, and the obtained trajectories may disobey traffic rules. Additionally, it is possible that no safe action is found in the remaining time prior to a collision, which often occurs in highly dynamic environments. As replanning the complete trajectory would be computationally expensive, we present a novel approach for repairing invalid trajectories (cf. Fig. 1).

A. Literature Overview

Below, we concisely review related works on trajectory planning, criticality assessment, and trajectory repairing.

a) *Trajectory planning:* Overviews of trajectory planning algorithms for autonomous vehicles are presented in [3], [4]. Graph-search-based planners discretize the configuration space by a graph to find an optimal path using search algorithms, such as Dijkstra [5] or A* [6]. The authors of [7] present the concept of a state lattice, which is a discrete graph on a continuous state space respecting differential constraints. Edges in the graph can be precalculated, which are often called motion primitives [8]–[10]. However, for high-dimensional motion models, the number of possible state combinations is often exploding, which leads to high computational costs [11].

All authors are with the Department of Informatics, Technical University of Munich, 85748 Garching, Germany.
yuanfei.lin@tum.de, sebastian.maierhofer@tum.de, althoff@tum.de

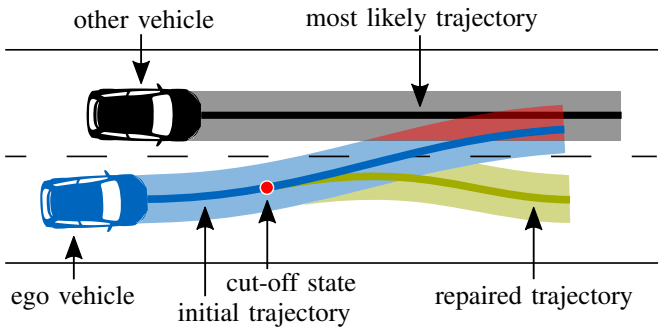


Fig. 1: Sketch of trajectory repairing: The initially-planned trajectory of the ego vehicle collides with the predicted obstacle trajectory. In our approach, the part of the ego vehicle's trajectory starting from the cut-off state can be repaired to avoid the collision.

Sampling-based algorithms randomly create samples and connect pairs of samples if a feasible solution exists between them. In particular, rapidly-exploring random trees (RRTs) have been shown to be effective for solving trajectory planning problems in complex configuration spaces [12]–[14]. To be more suitable for autonomous vehicles, an RRT is combined with closed-loop controllers in [15], [16], which is called closed-loop RRT (CL-RRT). However, the sampling strategy may still lead to high computational costs when searching the entire solution space. Improved sampling techniques exploiting heuristics are presented in [17]–[19], which are primarily designed for the shortest path computation without considering complex vehicle models.

Most interpolating curve-based planners, e.g., Bézier curves [20] and spline curves [21], are suitable for path smoothing in combination with a planner. However, dynamically feasible control commands are not directly obtained, and extra hyperparameters are introduced to adjust the position of control points. As an alternative, polynomial curves can effortlessly consider the position, angle, and curvature constraints of vehicles [22].

b) *Criticality assessment:* We use criticality assessment metrics to find the time when the repaired trajectory section begins. We call the corresponding state the *cut-off state*, see Fig. 1.

The following criticality metrics have been used in the literature: Time-to-collision (TTC) is the time that remains until a collision between two vehicles if their collision course and speed difference are maintained. A more precise measure is time-to-react (TTR), which returns the maximum remaining time for the last possible evasive maneuver since it does not require the assumptions made for TTC. The authors

of [23] approximate the TTR as the maximum of time-to-brake (TTB), time-to-steer (TTS), and time-to-kickdown (TTK), which are the maximum remaining times to execute evasive braking, steering, and accelerating, respectively. To determine the TTR, [24] lists an underapproximating method with high accuracy and flexibility, whilst an overapproximating approach by iteratively calculating the reachable set of the ego vehicle is proposed in [25].

c) *Trajectory repairing:* Compared to motion planning approaches, there exist much few studies on trajectory repairing. For instance, Pham and Nakamura introduce a fast trajectory correction algorithm in [26], which deforms the initially-planned trajectory by the so-called affine transformations. Similarly, a repairing algorithm for RRTs called Dynamic RRTs is introduced in [27], which only removes the invalid parts of the tree and maintains the rest. However, these approaches either barely include the constraints of the surrounding environments or do not consider the nonholonomic properties of vehicles, so that they cannot be directly applied to autonomous vehicles.

Another possible way to repair trajectories is to modify the invalid sections of the planned trajectory with timed elastic bands [28]. The initial trajectory is modeled via a chain of points connected by ideal springs. The repaired trajectory is then derived from a multi-objective least-squares optimization by adding extra constraints. Nonetheless, it is impossible to guarantee that a physically feasible solution can be obtained within a short time since deformations may result in trajectories that are not drivable. The authors of [29] mention a method for handling infeasible trajectories in the optimizer, where only parts of the trajectories are replanned. However, there exists no metric for determining the time intervals of the remaining parts.

B. Contributions

This paper presents a computationally efficient sampling-based trajectory repairing approach for arbitrary traffic scenarios. Unlike existing works, our approach simultaneously realizes the following features:

1) automatic selection of the invalid trajectory section that must be repaired on the basis of criticality measures;
2) efficient computation through separating the repairing problem into a high-level and a low-level planning problem; and
3) integration of quintic polynomials as a reference path into the CL-RRT algorithm to improve trajectory smoothness and computational efficiency.

The remainder of this paper is structured as follows: In Sec. II, some preliminaries are introduced. We provide an overview of our repairing approach in Sec. III. The detection of the cut-off state, the procedure to generate the reference path, and the modified CL-RRT algorithm are described in Sec. IV and V. We evaluate our approach in Sec. VI and conclude it in Sec. VII.

II. Preliminaries

A. Problem Statement

We define the trajectory repairing problem for autonomous vehicles based on [13], [30]. Let $\mathcal{X} \subset \mathbb{R}^n$ be the state space as the possible set of states x, $\mathcal{X}_{obs} \subset \mathcal{X}$ be the set of collision states, $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obs}$ be the resulting set of permissible states, and $\mathcal{U} \subset \mathbb{R}^m$ be the set of admissible control inputs u. The motion of the vehicle can be modeled by a dynamical system

$$\dot{x}(t) = f(x(t), u(t)). \tag{1}$$

We further introduce the initial state x_0 and the set of desired states $\mathcal{X}_g \subset \mathcal{X}_{free}$ in the goal region of the planning problem. The solution of (1) for an input trajectory $u(\cdot)$ within a time interval $[t_0, t_h]$ is then denoted by the state trajectory $x(\cdot)$, satisfying the following constraints:

$$\begin{aligned} x(t_0) = x_0,\ x(t_h) \in \mathcal{X}_g, \\ x([t_0,\, t_h]) \in \mathcal{X}_{free},\ u([t_0, t_h]) \in \mathcal{U}. \end{aligned} \tag{2}$$

Note that $x([t_0, t_h]) \in \mathcal{X}_{free}$ is a short form of $\forall t \in [t_0, t_h] : x(t) \in \mathcal{X}_{free}$, which holds for the entire state trajectory if no time interval is provided. Also, we are interested in repaired trajectories that minimize the following cost function [11]:

$$\min_u \int_0^T \|u(t)\|^2 dt + \rho T, \quad \text{s.t. (1), (2),} \tag{3}$$

where the parameter $\rho \geq 0$ determines the relative importance of the planning horizon $T := t_h - t_0$ versus the smoothness of the trajectory.

B. Vehicle Model

A kinematic single-track model [31] is used since it captures the relevant vehicle dynamics (cf. Fig. 2). The five-dimensional state vector $x = [s_x, s_y, \delta, v, \Psi]^T$ consists of the two-dimensional position at the center of the rear axle $[s_x, s_y]^T$, the steering angle δ, the longitudinal velocity v, and the orientation Ψ. The control input vector $u = [v_\delta, a_{long}]^T$ contains the steering velocity v_δ and the longitudinal acceleration a_{long}. Overall, the kinematics can be written in state-space form as follows:

$$\begin{aligned} \dot{x}_1 = x_4 \cos(x_5),\ \ \dot{x}_2 = x_4 \sin(x_5), \\ \dot{x}_3 = u_1,\ \ \dot{x}_4 = u_2,\ \ \dot{x}_5 = \frac{x_4}{l_{wb}} \tan(x_3), \end{aligned} \tag{4}$$

where l_{wb} is the wheelbase.

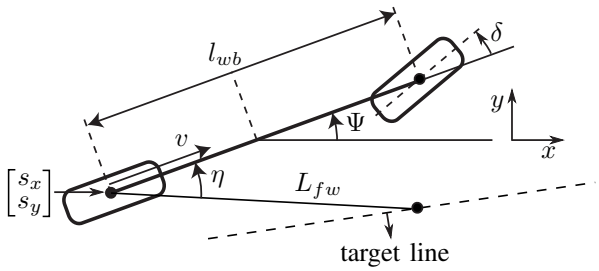


Fig. 2: Kinematic single-track model with the pure-pursuit strategy for tracking the current target line.

C. Definitions

For the remainder of this paper, we require the following definitions:

Definition 1 (Homotopy Class):
Two paths are homotopic if one can be deformed into the other without passing on or looping around obstacles [32].

Definition 2 (Frenét Frame):
A Frenét frame is a moving coordinate system, where the kinematic properties of a particle are described along a continuous and differentiable curve in three-dimensional Euclidean space [33].

Definition 3 (Dubins Path):
Dubins path describes the shortest curve between two configurations $[s_x, s_y, \Psi]^T$ of forward-only vehicles with a constrained turning radius [34]. For the detailed calculation of Dubins paths' length, we refer the readers to [35].

III. Overall Algorithm

Fig. 3 provides an overview of our trajectory repairing algorithm. We first detect whether the initially-planned trajectory disobeys traffic rules, which is performed by our previously-published traffic rule monitor [36]. For collision checking, the CommonRoad Drivability Checker [37] is utilized. Afterward, the TTR is computed to determine the cut-off state, and quintic polynomials are generated as a high-level reference path starting from the cut-off state to reach the goal region (cf. Sec. IV). Based on the optimal polynomial, the informed CL-RRT algorithm repairs the section of the initial trajectory starting from the cut-off state (cf. Sec. V). We consider only scenarios with collisions in this work, but our approach can be easily extended to other types of traffic rule violations.

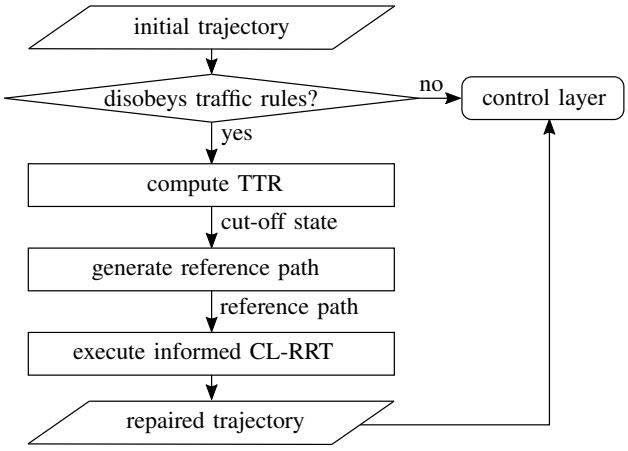


Fig. 3: Flowchart of the sampling-based trajectory repairing approach.

IV. Reference Path Generation

After obtaining predicted trajectories of other traffic participants and planned motions of the ego vehicle, we utilize the TTR to compute the cut-off time t_{cut} and generate the reference path via quintic polynomials to first explore the state space and to guide the sampling process.

A. Cut-off State Detection

To retain sufficient space for possible driving maneuvers starting from the cut-off state, we underapproximate the TTR using the evasive maneuvers in Tab. I, which are visualized in Fig. 4.

TABLE I: Description of evasive maneuvers.

Metric	Description
TTB	Full braking with maximum deceleration
TTK	Full acceleration until reaching the maximum velocity
TTS	Full steering to the left or right with maximum steering angle until the relative orientation change is equal to $\pm\ \pi/4$

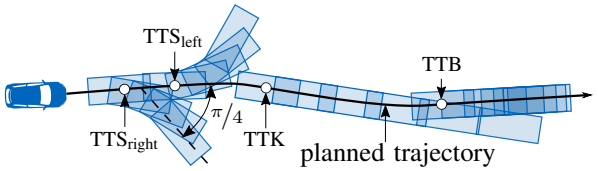


Fig. 4: Illustration of different evasive maneuvers. The white points indicate the start of the corresponding maneuvers.

An anytime-capable TTR algorithm modified from [24] is presented in Alg. 1. Given the initial trajectories of all traffic participants, the algorithm collects possible evasive maneuvers of the ego vehicle (line 1). Afterward, the function DETECTCOLLISION(·) calculates the TTC (line 2). The TTR is equal to 0 if a collision has already happened (line 4). If no collision is detected, we set the TTC and TTR to infinity (line 6). In all other cases, the function SEARCHTTM(·) uses the binary search algorithm described in [24] to detect the maximum remaining time for executing a maneuver TTM_m, $m \in \mathcal{M}$ (line 9). After iterating through all maneuvers, the TTR is determined (line 11).

Algorithm 1 Compute TTR.

Require: $\mathcal{P}_0$: Set of planned and predicted trajectories
1: $\mathcal{M} \leftarrow$ SETEVASIVEMANEUVERS($\mathcal{P}_0$)
2: TTC $\leftarrow$ DETECTCOLLISION($\mathcal{P}_0$)
3: **if** TTC $== 0$ **then**
4: TTR $\leftarrow 0$
5: **else if** TTC $== \infty$ **then**
6: TTR $\leftarrow \infty$
7: **else**
8: **for all** $m \in \mathcal{M}$ **do**
9: $\text{TTM}_m \leftarrow$ SEARCHTTM(m, TTC, $\mathcal{P}_0$)
10: **end for**
11: TTR $\leftarrow \max\{\text{TTM}_m \mid m \in \mathcal{M}\}$
12: **end if**
13: **return** TTR

B. Path Generation

Quintic polynomials minimize the jerk when connecting two vehicle states [38], which can be generated in the Frenét frame analogously as in [9]. The movement of vehicles is decoupled into the longitudinal motion $s(t)$ along a Frenét

curve and the lateral motion $d(t)$ perpendicular to the Frenét curve. The two coordinates and their derivatives form the Frenét state vector $\mathtt{f} = [s, \dot{s}, \ddot{s}; d, \dot{d}, \ddot{d}]^T$.

We first use the Dijkstra search algorithm to create a route without considering other obstacles, which determines the shortest path through the road network from the cut-off state to the goal region. This path is chosen as the longitudinal axis of the Frenét frame. Next, we use a series of avoidance points [39] to generate polynomials covering all possible homotopy classes. The avoidance points $[s_a, d_a]^T$ are shifted laterally from the point $[s_v, d_v]^T$, where a traffic rule is violated for the first time, e.g., the first collision point with an obstacle (cf. Fig. 5a):

$$s_a = s_v,\ d_a \in \{d_v + d_{min} + \frac{i}{n_d}(d_{max} - d_{min}) \mid 0 \leq i \leq n_d,\ i \in \mathbb{N}\},$$

where d_{min} and d_{max} are the minimum and maximum permissible lateral offsets, respectively, and n_d is the number of path candidates, which can be specified according to the vehicle width w, e.g., $n_d = 2(d_{max} - d_{min})/w$. To connect the cut-off state to the goal region by quintic polynomials, there exist 12 unknown coefficients to be determined:

$$s(t) = \alpha_5 t^5 + \alpha_4 t^4 + \alpha_3 t^3 + \alpha_2 t^2 + \alpha_1 t + \alpha_0,$$
$$d(t) = \beta_5 t^5 + \beta_4 t^4 + \beta_3 t^3 + \beta_2 t^2 + \beta_1 t + \beta_0,\ t \in [0, \tilde{T}],$$

where $\tilde{T} := t_h - t_{cut}$ is the remaining time horizon. As boundary conditions, the cut-off Frenét state $\mathtt{f}_{cut}$ and the goal Frenét state $\mathtt{f}_g$ are derived via the coordinate transformation introduced in [9]. According to [38], some coefficients are determined directly:

$$\alpha_0 = s_{cut},\ \ \alpha_1 = \dot{s}_{cut},\ \ \alpha_2 = \frac{1}{2}\ddot{s}_{cut},$$
$$\beta_0 = d_{cut},\ \ \beta_1 = \dot{d}_{cut},\ \ \beta_2 = \frac{1}{2}\ddot{d}_{cut},$$

and other coefficients are determined using matrices as

$$\begin{bmatrix} \tilde{T}^3 & \tilde{T}^4 & \tilde{T}^5 \\ 3\tilde{T}^2 & 4\tilde{T}^3 & 5\tilde{T}^4 \\ T_a^3 & T_a^4 & T_a^5 \end{bmatrix} \begin{bmatrix} \alpha_3 \\ \alpha_4 \\ \alpha_5 \end{bmatrix} = \begin{bmatrix} s_g - s_{cut} - \dot{s}_{cut}\tilde{T} - \frac{1}{2}\ddot{s}_{cut}\tilde{T}^2 \\ \dot{s}_g - \dot{s}_{cut} - \ddot{s}_{cut}\tilde{T} \\ s_a - s_{cut} - \dot{s}_{cut}T_a - \frac{1}{2}\ddot{s}_{cut}T_a^2 \end{bmatrix},$$
$$\begin{bmatrix} \tilde{T}^3 & \tilde{T}^4 & \tilde{T}^5 \\ 3\tilde{T}^2 & 4\tilde{T}^3 & 5\tilde{T}^4 \\ T_a^3 & T_a^4 & T_a^5 \end{bmatrix} \begin{bmatrix} \beta_3 \\ \beta_4 \\ \beta_5 \end{bmatrix} = \begin{bmatrix} d_g - d_{cut} - \dot{d}_{cut}\tilde{T} - \frac{1}{2}\ddot{d}_{cut}\tilde{T}^2 \\ \dot{d}_g - \dot{d}_{cut} - \ddot{d}_{cut}\tilde{T} \\ d_a - d_{cut} - \dot{d}_{cut}T_a - \frac{1}{2}\ddot{d}_{cut}T_a^2 \end{bmatrix},$$

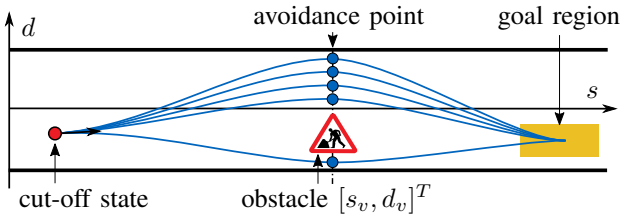
where $T_a = \frac{s_a - s_{cut}}{s_g - s_{cut}}\tilde{T}$ is the intermediate time value of the avoidance points.

C. Reference Path

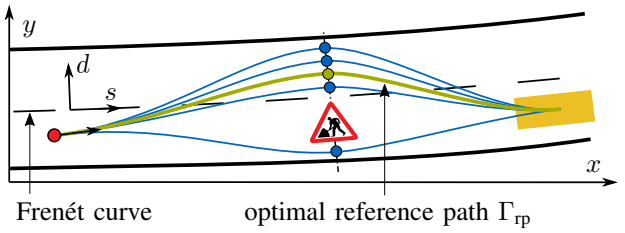
To provide smooth trajectories using the Frenét states, the cost for each polynomial π is modified from [22, (3)] as

$$J(\pi) = \omega_1 \int_0^{\tilde{T}} \ddot{s}(t)dt + \omega_2 \int_0^{\tilde{T}} \ddot{d}(t)dt + \omega_3 p(\pi), \qquad (5)$$

where $p(\pi)$ represents the penalty function, which could be chosen as the inverse of the distance to the colliding obstacle or the degree of traffic rule violations, and ω_1 to ω_3 are weights that need to be manually tuned.



(a) Generated path set in the Frenét frame.



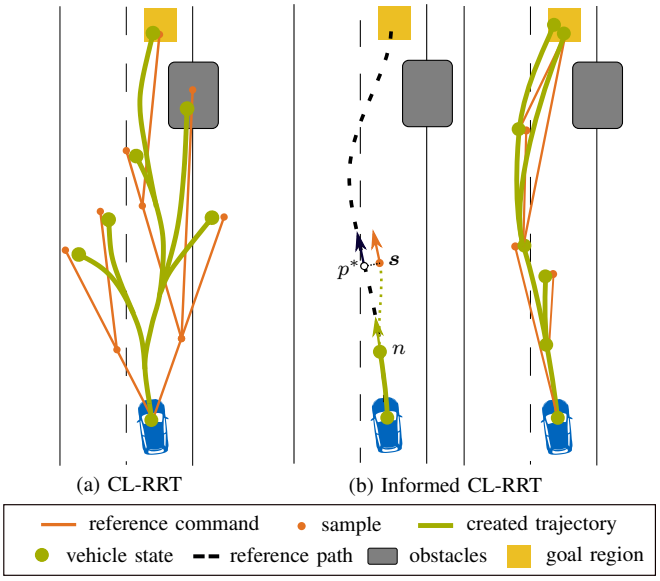
(b) Transformed paths in the Cartesian frame.

Fig. 5: Illustration of path generation and coordinate transformation.

After sorting the costs in ascending order, we transform the corresponding paths from the Frenét frame into the Cartesian frame (cf. Fig. 5b). Then, we traverse the list of paths in the Cartesian frame and choose the first collision-free path as the reference path Γ_{rp}.

V. Repairing using informed CL-RRT

After obtaining the reference path, we repair the section of the invalid trajectory starting from the cut-off state. To shrink the planning problem to subsets of the original domain similarly as in [17], [18], an informed CL-RRT is presented based on the reference path (cf. Fig. 6 and Alg. 2), which extends the CL-RRT with respect to the sampling strategy, the controller design, and the tree expansion method. Each expanded node contains information about the current vehicle state, the cost from the root to the node, and the reference to its predecessor.



(a) CL-RRT (b) Informed CL-RRT

Fig. 6: Comparison between the CL-RRT and the informed CL-RRT.

A. Sampling Strategies

It is inefficient to generate samples within the entire state space in a purely random manner [16]. To make better use of the reference path, we adopt a sampling method that generates samples following a Gaussian mixture model distribution analogously to [40] (lines 1 and 2 in Alg. 2). The equidistant points on the reference path are chosen as centers of the Gaussians (cf. Fig. 7). As a result, the samples are located in the vicinity of the reference path. Each sample $\boldsymbol{s}$ is a quadruple $(s_{x,\boldsymbol{s}}, s_{y,\boldsymbol{s}}, \Psi_{\boldsymbol{s}}, v_{cmd})$, where $\Psi_{\boldsymbol{s}}$ denotes the orientation of the sample and v_{cmd} denotes the desired velocity (line 6). To provide the capability of vehicles moving from the current sample to the goal region, $\Psi_{\boldsymbol{s}}$ is chosen to be equal to the heading of its closest point p^* on the reference path Γ_{rp} (cf. Fig. 6b):

$$\Psi_{\boldsymbol{s}} = \Psi_{p^*} \text{ with } p^* = \underset{p \in \Gamma_{\text{rp}}}{\arg\min} \|[s_{x,\boldsymbol{s}}\, s_{y,\boldsymbol{s}}]^T - p\|. \quad (6)$$

Afterward, we use the same evaluation function for node selection as the original CL-RRT algorithm, which attempts to connect a sample to a node with the shortest Dubins path's length (line 7).

B. Trajectory Creation

After obtaining the target line, i.e., a straight line connecting a node to a sample, the function CREATETRAJECTORY(·) runs a forward simulation of the vehicle to create a state trajectory starting from the state in the node to reach the vicinity of the sample (lines 8 and 14 in Alg. 2). During the trajectory creation, a closed-loop controller is used to move and stabilize the vehicle along the given target line. Since we only need to simulate feasible trajectories, which are not actually executed, robust trajectory tracking controllers [41] are unnecessary. In our implementation, steering and velocity controllers are used similarly as in [16].

Algorithm 2 Informed CL-RRT.

Require: x_{cut}: Cut-off state, $\mathcal{X}_g$, $\mathcal{X}_{free}$
1: $\Gamma_{rp} \leftarrow$ REFERENCEPATH(x_{cut}, $\mathcal{X}_g$, $\mathcal{X}_{free}$)
2: $\mathcal{G} \leftarrow$ GAUSSIANMIXTUREMODEL(Γ_{rp})
3: $\mathcal{C}_{traj} \leftarrow \infty$
4: $\mathcal{T}$, $\boldsymbol{s}_g \leftarrow$ INITIALIZETREE(x_{cut}, $\mathcal{X}_g$)
5: **while** not REACHTIMELIMIT( ) **do**
6: $\quad \boldsymbol{s} \leftarrow$ SAMPLE($\mathcal{G}$, $\mathcal{X}_{free}$)
7: $\quad n_{min} \leftarrow$ FINDMINUMUMEVALUATEDNODE($\boldsymbol{s}$, $\mathcal{T}$)
8: $\quad \tilde{x}(\cdot) \leftarrow$ CREATETRAJECTORY(n_{min}, $\boldsymbol{s}$)
9: $\quad$ **if** $\tilde{x}(\cdot) \in \mathcal{X}_{free}$ **then**
10: $\quad\quad \tilde{\boldsymbol{n}} \leftarrow$ CONVERTSTATESTONODES($\tilde{x}(\cdot)$)
11: $\quad\quad \mathcal{T} \leftarrow$ ADDNODESTOTREE($\mathcal{T}$, $\tilde{\boldsymbol{n}}$)
12: $\quad\quad$ **for all** $n \in \tilde{\boldsymbol{n}}$ **do**
13: $\quad\quad\quad$ **if** WITHINGOALORIENTATION(n, $\boldsymbol{s}_g$) **then**
14: $\quad\quad\quad\quad \hat{x}(\cdot) \leftarrow$ CREATETRAJECTORY(n, $\boldsymbol{s}_g$)
15: $\quad\quad\quad\quad$ **if** $\hat{x}(\cdot) \in \mathcal{X}_{free}$ and REACHGOAL($\hat{x}(\cdot)$) and COST($\hat{x}(\cdot)$) $< \mathcal{C}_{traj}$ **then**
16: $\quad\quad\quad\quad\quad \mathcal{C}_{traj} \leftarrow$ COST($\hat{x}(\cdot)$)
17: $\quad\quad\quad\quad\quad x_{sel}(\cdot) \leftarrow \hat{x}(\cdot)$
18: $\quad\quad\quad\quad$ **end if**
19: $\quad\quad\quad$ **end if**
20: $\quad\quad$ **end for**
21: $\quad$ **end if**
22: **end while**
23: **return** $x_{sel}(\cdot)$

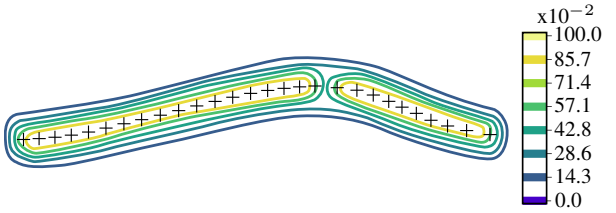


Fig. 7: Contour of the Gaussian mixture sampling. The black crosses denote the centers of the Gaussians. The right colorbar shows the contour lines of the probability distribution.

1) Steering Controller: The pure-pursuit controller is simple to implement and is flexible with respect to path representations. It returns the desired steering angle

$$\delta_d = \tan^{-1}\left(\frac{2l_{wb}\sin(\eta)}{L_{fw}}\right), \quad (7)$$

where L_{fw} is the forward drive look-ahead distance and η is the heading of the look-ahead points on the target line from the rear axle based on the vehicle orientation (cf. Fig. 2). Different from the original CL-RRT, we use the steering velocity instead of the steering angle as the lateral input to avoid jerky motions. As a result, the steering controller combines the pure-pursuit controller with a PI controller (cf. Fig. 8). The PI controller can be written as

$$v_\delta = K_{P,\delta}(\delta_d - \delta) + K_{I,\delta}\int_0^t (\delta_d - \delta)d\tau, \quad (8)$$

where $K_{P,\delta}$ and $K_{I,\delta}$ are the proportional and integral gain, respectively. The linear stability analysis of the steering controller is similar to that of [15].

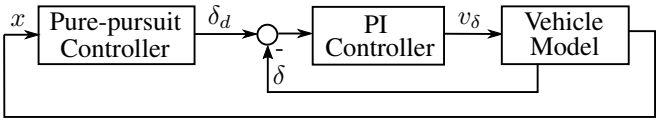


Fig. 8: Structure of the steering controller.

2) Velocity Controller: For velocity tracking of the desired value v_{cmd}, we adopt the same PI controller as in [16]. Additionally, the acceleration variation needs to satisfy the jerk limit to enhance comfort by including the constraint $\frac{da}{dt} \in [j_{min}, j_{max}]$, where j_{min} and j_{max} denote the minimum and maximum allowable jerk values, respectively.

C. Tree Expansion

The initial cost of the repaired trajectory $\mathcal{C}_{traj}$ is set to infinity (line 3 in Alg. 2). After initializing the RRT with the cut-off state and choosing the center of the goal region as the goal sample $\boldsymbol{s}_g$ (line 4), we sample until the user-defined

time limit is reached. When the collision-free trajectory $\tilde{x}(\cdot)$ is generated using the closed-loop simulation, each state of the trajectory is converted into a node and added to the tree (lines 10 and 11). At the same time, if the orientation of the connecting line between the node and the goal sample is located in the desired goal orientation interval (line 13), i.e.,

$$\Psi_{n,\boldsymbol{s}_g} = \tan^{-1}\left(\frac{s_{y,\boldsymbol{s}_g} - s_{y,n}}{s_{x,\boldsymbol{s}_g} - s_{x,n}}\right) \in [\Psi_{g,min}, \Psi_{g,max}],$$

a new target line connecting the node to the goal sample is forwarded to the controller (line 14). We return the collision-free trajectory with the lowest cost. (lines 15–18).

VI. Evaluation

We evaluate our approach with traffic scenarios from the CommonRoad benchmark suite[1] [31]. Our approach is implemented in *Python* and executed on a computer with an Intel i7 1.90 GHz processor and 24 GB of DDR4 2400 MHz memory. We use the polynomial sampling-based motion planner introduced in [9] to compute an initial trajectory and then repair it using our approach if needed. To predict the movement of dynamic obstacles, the set-based prediction tool *SPOT* [42] is used. The parameters of the ego vehicle are listed in Tab. II. In addition, the maximum execution time of all algorithms is limited to 1.0 s.

TABLE II: Parameters of the ego vehicle.

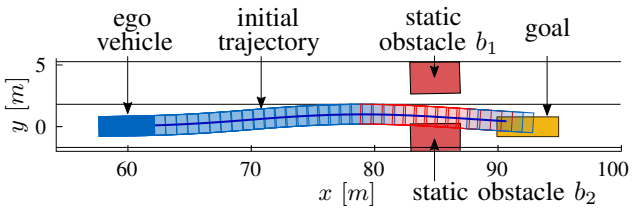
Parameter	Symbol	Value
Vehicle size	l, w	4.508 m, 1.610 m
Wheelbase	l_{wb}	2.578 m
Velocity interval	$[v_{min}, v_{max}]$	[0.0, 50.8] m/s
Maximum acceleration	a_{max}	11.5 m/s^2
Jerk interval	$[j_{min}, j_{max}]$	$[-10, 10]$ m/s^3
Steering angle interval	$[\delta_{min}, \delta_{max}]$	$[-1.066, 1.066]$ rad
Steering velocity interval	$[v_{\delta,min}, v_{\delta,max}]$	$[-0.4, 0.4]$ rad/s
Time step size	Δt	0.1 s

A. Static Obstacles on the Driving Lane

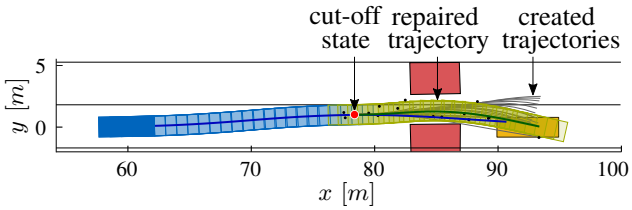
The first evaluated scenario is a rural two-lane road[2]. In this scenario, two static obstacles block the ego vehicle's driving lane, creating a narrow passage close to the goal region (parameters listed in Tab. III). The obstacles represent, e.g., construction sites or parked vehicles. The initial planner shrinks the size of the obstacles to first explore the state space with low computational effort, which relaxes the planning problem but generates a trajectory that collides with the obstacle b_2 at the 24th time step (cf. Fig. 9a). Figs. 9b–9c show the results using our approach, which modifies the initial trajectory. We mark the occupancy of the ego vehicle with a collision in red and without in blue and the trajectory starting from the cut-off state in green.

[1] commonroad.in.tum.de

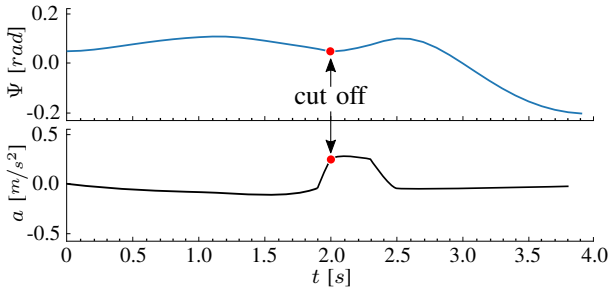
[2] CommonRoad ID: ZAM_Urban-3_3



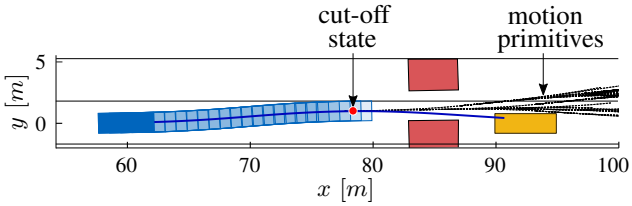
(a) Initial configuration.



(b) Repaired trajectory using informed CL-RRT.



(c) The motion profiles (Ψ, a) of the repaired trajectory (cf. Fig. 9b).



(d) Repaired trajectory using A*.

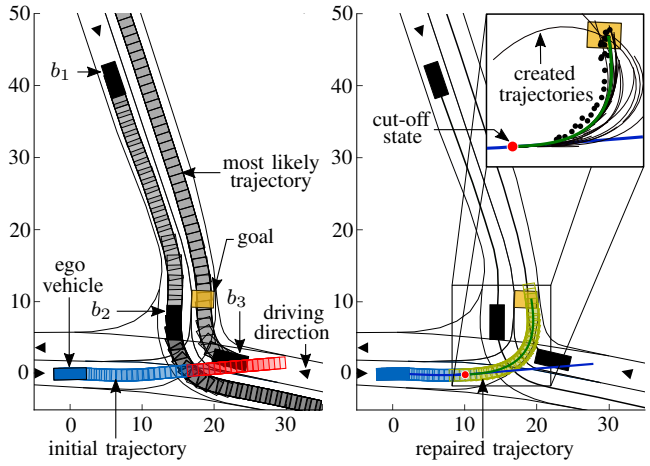
Fig. 9: Scenario with static obstacle in driveway.

TABLE III: Parameters of the rural two-lane scenario (cf. Fig. 9).

Parameter	Description
Ego vehicle	$x_{ego} = [60\,m, 0.06\,m, 0\,rad, 9\,m/s, 0.02\,rad]^T$
Static obstacle	$[s_x, s_y, \Psi]_{b_1} = [85\,m, 4\,m, 0.02\,rad]^T$
	$[s_x, s_y, \Psi]_{b_2} = [85\,m, -1\,m, 0.02\,rad]^T$
	$[l, w]_{b_1,b_2} = [4\,m, 2.5\,m]^T$
Goal region	$[s_x, s_y, l, w]_{goal} = [92.5\,m, 0\,m, 5\,m, 1.6\,m]^T$
	$[\Psi_{g,min}, \Psi_{g,max}] = [-0.2,\ 0.2]$ rad
Time horizon	$t_0 = 0.0\,s$, $t_h = 3.0$–$5.0\,s$
Criticality	TTC = 2.4 s, t_{cut} = TTR = 2.0 s

B. Comparison with A* Search Algorithm

We compare our approach with the A* search algorithm described in [10]. Motion primitives are generated by a so-called constraint graph, which consists of all possible trajectories starting from the same initial state under the same physical boundaries (cf. Tab. II). A vast amount of motion primitives in the search graph results in high computational costs although they can be generated offline. Determining the



(a) Initial configuration. (b) Repaired trajectory.

Fig. 10: Urban T-intersection scenario.

TABLE IV: Parameters of the urban T-intersection scenario (cf. Fig. 10).

Parameter	**Description**
Ego vehicle	$x_{ego} = [0\,m, 0\,m, 0\,rad, 7\,m/s, 0.02\,rad]^T$
Other vehicles	$[s_x, s_y, \Psi, v]_{b_1} = [5.9\,m, 40.8\,m, -1.3\,rad, 14\,m/s]^T$
	$[s_x, s_y, \Psi, v]_{b_2} = [14.5\,m, 7.3\,m, -1.6\,rad, 14\,m/s]^T$
	$[s_x, s_y, \Psi, v]_{b_3} = [22.2\,m, 1.9\,m, 2.9\,rad, 12\,m/s]^T$
	$[l, w]_{b_1,b_2,b_3} = [4.8\,m, 2\,m]^T$
Goal region	$[s_x, s_y, l, w]_{goal} = [18.4\,m, 10.4\,m, 2.3\,m, 3.1\,m]^T$
	$[\Psi_{g,min}, \Psi_{g,max}] = [1.3,\ 1.7]\ rad$
Time horizon	$t_0 = 0.0\,s$, $t_h = 3.0$–$5.0\,s$
Criticality	TTC = 2.8 s, t_{cut} = TTR = 1.5 s

resolution of the search graph may be challenging when balancing the tradeoff between time complexity and optimality of the search process. Additionally, possible solutions may be missed due to the inappropriate choice of motion primitives from the constraint graph (cf. Fig. 9d). Compared to the A* search algorithm, our approach does not need to consider the discretization of the state space. Furthermore, we generate feasible trajectories without selecting them from a large set of candidates.

C. Urban T-Intersection

We also evaluate our approach on an urban T-intersection[3] (cf. Fig. 10a), where the ego vehicle passes through the intersection along with three other vehicles $b_i, i \in \{1, 2, 3\}$ (parameters listed in Tab. IV). The most likely trajectories of the vehicles are marked in black. However, due to the time limit, the initially-used planner fails to provide a safe trajectory to reach the goal and the intermediate planned result enters the opposite lane. After detecting the cut-off state, our repairing algorithm generates a collision-free and physically feasible trajectory respecting the right-of-way rule (cf. Fig. 10b).

[3]CommonRoad ID: DEU_Ffb_2_1_T-1

D. Computation Costs

Tab. V compares the two scenarios in terms of their computation costs using our approach, which consists of the computation times for the cut-off state, the reference path, and the first solution of the informed CL-RRT algorithm. In addition, the computational costs of repairing a trajectory from the initial state using the informed CL-RRT algorithm and replanning the complete trajectory using the polynomial sampling-based planner are listed, where the time horizon of each sampled polynomial is chosen as 1.4 s. The results show that our approach has a significant advantage in computational efficiency, particularly when generating the first safe and physically feasible trajectory. Since the first evaluated scenario has a shorter remaining time horizon after cutting off, its computational costs are smaller.

VII. Conclusions

We have presented a sampling-based approach for autonomous vehicles to repair invalid trajectories in arbitrary traffic scenarios and demonstrated its benefits using real traffic data. In contrast to existing planning and repairing approaches, our approach reasonably uses the initially-planned trajectories based on criticality measures. This leads to significantly less computational effort than replanning the complete trajectory from scratch, especially when traffic rule violations are minimal close to the goal. With our approach, the smoothness of the repaired trajectory is first ensured by interpolating the reference path, which is resolution optimal among all possible homotopy classes. In addition, our approach generates comfortable, stable, and directly executable motions using a closed-loop controller, which enables autonomous vehicles to safely and efficiently handle complex traffic situations.

Acknowledgments

The authors gratefully acknowledge partial financial support by the BMW Group within the CAR@TUM project and the German Research Foundation (DFG) grant AL 1185/3-2.

References

[1] H. Krasowski, X. Wang, and M. Althoff, "Safe reinforcement learning for autonomous lane changing using set-based prediction," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2020, pp. 1–7.

[2] S. Aradi, "Survey of deep reinforcement learning for motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2020. [Online]. Available: https://doi.org/10.1109/tits.2020.3024655

[3] D. Gonzalez Bautista, J. Pérez, V. Milanes, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 4, pp. 1135–1145, 2016.

[4] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[5] S. J. Anderson, S. B. Karumanchi, and K. Iagnemma, "Constraint-based planning and control for safe, semi-autonomous operation of vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 383–388.

[6] 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.

TABLE V: Average number of samples and computation times of 100 simulation runs for each scenario. Their mean and standard deviation (cf. numbers in brackets) are listed. To repair from the initial state, we use the informed CL-RRT algorithm without integrating the cut-off state. Replanning denotes that the complete trajectory is recomputed with the polynomial sampling-based planner.

Scenario	**Repairing from the cut-off state**				**Repairing from the initial state**	**Replanning**
	Cut-off state	**Reference path**	**Informed CL-RRT**	**Total**		
Rural static obstacle	9.8 (0.1) ms	8.0 (0.1) ms	3.6 (3.0) samples, 39.0 (31.6) ms	**56.8 (36.8)** ms	**438.5 (425.5)** ms	**511.4 (5.5)** ms
Urban T-intersection	14.5 (0.2) ms	7.6 (0.1) ms	4.2 (3.8) samples, 44.3 (34.9) ms	**66.4 (35.2)** ms	**timeout**	**timeout**

[7] M. Pivtoraiko and A. Kelly, "Efficient constrained path planning via search in state lattices," in *Proc. of the Int. Symposium on Artificial Intelligence, Robotics and Automation in Space*, 2005, pp. 308–333.

[8] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

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

[10] J. Salvado, L. M. Custódio, and D. Hess, "Contingency planning for automated vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2016, pp. 2853–2858.

[11] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2017, pp. 2872–2879.

[12] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *Int. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[13] 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.

[14] J. H. Jeon, S. Karaman, and E. Frazzoli, "Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*," in *Proc. of the IEEE Conf. on Decision and Control and European Control Conference*, 2011, pp. 3276–3282.

[15] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," in *Proc. of the AIAA Guidance, Navigation and Control Conference and Exhibit*, 2008, pp. 7166:1–22.

[16] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Transactions on Control Systems Technology*, vol. 17, no. 5, pp. 1105–1118, 2009.

[17] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2014, pp. 2997–3004.

[18] D. Kim, J. Lee, and S. Yoon, "Cloud RRT*: Sampling cloud based RRT*," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2014, pp. 2519–2526.

[19] J. Gammell, S. Srinivasa, and T. Barfoot, "Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2015, pp. 3067–3074.

[20] D. Gonzalez Bautista, J. Pérez, R. Lattarulo, V. Milanes, and F. Nashashibi, "Continuous curvature planning with obstacle avoidance capabilities in urban scenarios," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2014, pp. 1430–1435.

[21] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola *et al.*, "Odin: Team VictorTango's entry in the DARPA urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[22] L. Wang, Z. Wu, J. Li, and C. Stiller, "Real-time safe stop trajectory planning via multidimensional hybrid A*-algorithm," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2020, pp. 1–7.

[23] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approach – its situation assessment, decision making, and performance tradeoffs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 528–540, 2006.

[24] A. Tamke, T. Dang, and G. Breuel, "A flexible method for criticality assessment in driver assistance systems," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2011, pp. 697–702.

[25] S. Soentges, M. Koschi, and M. Althoff, "Worst-case analysis of the time-to-react using reachable sets," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 1891–1897.

[26] Q. Pham and Y. Nakamura, "A new trajectory deformation algorithm based on affine transformations," *IEEE Transactions on Robotics*, vol. 31, no. 4, pp. 1054–1063, 2015.

[27] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 2006, pp. 1243–1248.

[28] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, "Trajectory modification considering dynamic constraints of autonomous robots," in *Proc. of the German Conf. on Robotics*, 2012, pp. 1–6.

[29] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for Bertha — a local, continuous method," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 450–457.

[30] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 798–814, 2021.

[31] 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.

[32] D. Yi, M. A. Goodrich, and K. D. Seppi, "Homotopy-aware RRT*: Toward human-robot topological path-planning," in *Proc. of the ACM/IEEE Int. Conf. on Human-Robot Interaction*, 2016, pp. 279–286.

[33] M. G. Wagner and B. Ravani, "Curves with rational Frenet-Serret motion," *Computer Aided Geometric Design*, vol. 15, no. 1, pp. 79–101, 1997.

[34] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[35] J. Enright, E. Frazzoli, K. Savla, and F. Bullo, "On multiple UAV routing with stochastic targets: Performance bounds and algorithms," in *Proc. of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2005, pp. 1–15.

[36] S. Maierhofer, A.-K. Rettinger, E. C. Mayer, and M. Althoff, "Formalization of interstate traffic rules in temporal logic," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 752–759.

[37] C. Pek, V. Rusinov, S. Manzinger, M. C. Üste, and M. Althoff, "CommonRoad Drivability Checker: Simplifying the development and validation of motion planning algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2020, pp. 1013–1020.

[38] J. R. Movellan, "Minimum jerk trajectories," 2011. [Online]. Available: https://inc.ucsd.edu/mplab/tutorials/minimumJerk.pdf

[39] U. Lee, S. Yoon, H. Shim, P. Vasseur, and C. Demonceaux, "Local path planning in a complex environment for self-driving car," in *Proc. of the IEEE Int. Conf. on Cyber Technology in Automation, Control and Intelligent*, 2014, pp. 445–450.

[40] Y. Chen, H. Ye, and M. Liu, "Hierarchical trajectory planning for autonomous driving in low-speed driving scenarios based on RRT and optimization," *arXiv preprint arXiv:1904.02606*, 2019.

[41] D. Calzolari, B. Schürmann, and M. Althoff, "Comparison of trajectory tracking controllers for autonomous vehicles," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2017, pp. 1–8.

[42] M. Koschi and M. Althoff, "Set-based prediction of traffic participants considering occlusions and traffic rules," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 249–265, 2021.