Efficient Driving Corridor Extraction for Autonomous Vehicles Using Best-First Reachability Analysis

Florian Lercher and Matthias Althoff

***Abstract*— Planning traffic rule-compliant trajectories for autonomous vehicles is computationally expensive. It has been shown that restricting the search space of the trajectory planner to a driving corridor facilitates planning. Existing methods based on reachability analysis have to compute the entire reachable set before they can provide a single corridor. Since planners often require only one corridor, the computation of reachable states irrelevant to that corridor is superfluous. To address this issue, we propose treating reachability analysis for driving corridor extraction as a best-first search problem. Thus, we quickly identify a first corridor to start trajectory planning as early as possible; additional corridors are computed on demand. We experimentally compare different best-first search strategies and demonstrate the advantages of our approach.**

I. Introduction

Motion planning for autonomous vehicles is a complex task, particularly since we need to respect traffic rules in addition to the dynamical constraints of the vehicle [1]. Previous work has shown that using set-based reachability analysis to restrict the search space for trajectories facilitates motion planning, particularly in scenarios with narrow solution spaces [2]–[5]. These approaches determine the set of all states that the *ego vehicle*, i.e., the vehicle under our control, can reach from its initial state and subdivide it into several driving corridors. Intuitively, each driving corridor maps to one possible high-level maneuver. In Fig. 1a, $\mathcal{DC}_1$ corresponds to aborting the intended overtaking maneuver to let the oncoming vehicle pass, while $\mathcal{DC}_2$ represents accelerating to complete overtaking before passing the oncoming vehicle. One selected driving corridor is then used to determine a reduced sampling space for sampling-based planners [2] as shown in Fig. 1b, or convex constraints for optimization-based methods [3]. If the planner cannot find a trajectory in the selected corridor, it tries again with another corridor if time permits, or triggers a previously computed fail-safe trajectory, e.g., as described in [6].

The existing approaches perform exhaustive reachability analysis to determine all possible driving corridors before selecting one corridor to pass to the trajectory planner. This has two drawbacks: First, trajectory planning cannot begin until the entire reachable set has been computed. Second, computing resources are spent on determining corridors that are later discarded anyway, which is particularly inefficient if many corridors exist. We propose an alternative method to compute the reachable set based on best-first search so that we obtain a first driving corridor quickly. We can then determine additional corridors on demand or, if idle cores are available, in parallel to trajectory planning within the first driving corridor. Combining set-based methods with search is challenging, as one must decide when to split the continuous reachable set in order to start a new search branch instead of propagating the whole set.

This work was supported by the German Research Foundation (DFG) under grant AL 1185/20-1 and 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.

The authors are with the School of Computation, Information and Technology, Technical University of Munich, 85748 Garching, Germany. florian.lercher@tum.de, althoff@tum.de

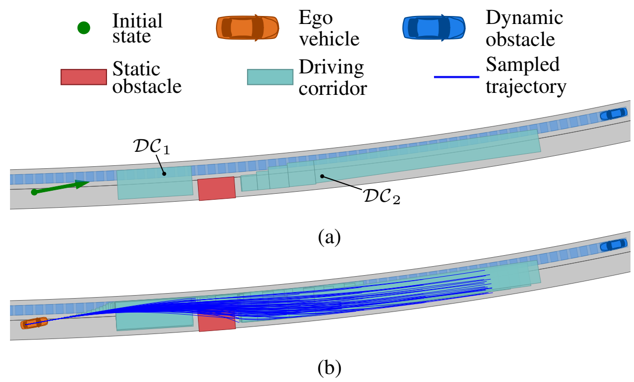


Fig. 1. Motivation: Sampling-based trajectory planning with driving corridors. (a) There are two driving corridors $\mathcal{DC}_1$ and $\mathcal{DC}_2$ for this scenario. For visual clarity, we only show the constraints provided by the corridors at the end of the planning horizon of 5 s. (b) Trajectories in the sampling space derived from the driving corridor $\mathcal{DC}_2$ (cf. [2, Fig. 1b]).

A. Related Work

In the literature, driving corridors are used to subdivide the search space for trajectories and distinguish between high-level maneuvers. In general, driving corridors expedite trajectory generation of all motion planners that accept the spatio-temporal constraints a corridor provides. Such constraints are particularly important for optimization-based planners: Typically, each maneuver induces its own locally optimal trajectory so that the search space becomes non-convex. By constraining the optimizer to stay within a chosen corridor, the search space is convexified, and the planner can consciously decide for a maneuver [7]. Sampling-based planners also benefit from these constraints, as they focus the sampling on regions that realize a selected maneuver [2]. Finally, driving corridors can be used to safeguard planners based on reinforcement learning [8]. Subsequently, we review existing methods for driving corridor extraction.

Topological approaches rely on systematically decomposing the search space based on the spatial relations to obstacles and lanes [9]–[12]. Many of these leverage the concept of homotopy [7], [13]–[17]. Intuitively, two trajectories belong to the same homotopy class if we can continuously deform one into the other; in particular, this means that homotopic trajectories must pass each obstacle on the same side. The aforementioned works employ a generalized version of homotopy that allows trajectories to end in different states, which [17] formalizes as *free-end* homotopy. Some topological approaches explicitly take traffic rules into account [11], [13].

None of the methods mentioned so far considers the dynamical constraints of the ego vehicle, which can result in many infeasible driving corridors. To address this issue, [18] extracts corridors as subsets of the reachable set of the ego vehicle that correspond to a sequence of high-level decisions. The authors of [3], [19] adapt this method for optimization-based trajectory planning. According to [19, Def. 6], a driving corridor must consist of connected and vertically convex sets. Therefore, the trajectories within a corridor are homotopic if we relax the definition of homotopy by allowing the trajectories to start and end in different states. The approach from [3] is extended to consider traffic rules in [4], [5]. By treating lane changes as discrete events, [20] only needs to compute the reachable set in the longitudinal direction. In [21], reachable sets are combined with a probabilistic measure of collision risk.

Other approaches create driving corridors by bloating trajectories obtained via sampling, e.g., [22], [23]. However, these methods inherit the challenges of sampling-based planners in scenarios with narrow solution spaces [2].

Finally, [24] identifies collision-free driving corridors using support vector machines. This requires choosing a passing side for each obstacle beforehand. Then, the support vector machine finds two surfaces that maximize the distance between left and right obstacles.

B. Contributions

For the first time, we approach reachability analysis for driving corridor extraction as a best-first search problem [25, Sec. 3.3.1]. This offers two key advantages over existing methods [3]–[5]:

- Trajectory planning starts earlier because we provide a first driving corridor quickly without necessarily computing the entire reachable set first.
- Additional driving corridors can be computed on demand, which avoids unnecessary consumption of computing resources.

Best-first search is very flexible, enabling us to use different search strategies like uniform-cost search and greedy best-first search just by changing the evaluation function in our algorithm. To facilitate informed search strategies, we develop a heuristic based on information from the previous planning cycle. Our experiments with recorded real-world scenarios show that we find a first driving corridor and thus a trajectory faster than existing approaches [3]–[5]. Moreover, we demonstrate that, similar to [3]–[5], the computation time of our method decreases in critical scenarios.

This paper is organized as follows: Sec. II discusses preliminaries and presents our problem statement. In Sec. III, we explain our best-first reachability analysis for driving corridor extraction. We present the results of our experiments in Sec. IV before coming to a conclusion in Sec. V.

II. Preliminaries and Problem Statement

Let us introduce the index $k \in \mathbb{N}_0$ to denote the discrete time step corresponding to the continuous time $t_k = k\Delta t$, where $\Delta t \in \mathbb{R}_{>0}$ is a fixed time increment. Without loss of generality, we choose the initial time step to be 0 and fix a final time step $k_\mathtt{h}$ as the *planning horizon*. We assume that the trajectory of the ego vehicle is replanned every $k_\mathtt{c}$ time steps, where $k_\mathtt{c} \ll k_\mathtt{h}$. Subsequently, we briefly explain set-based reachability analysis and driving corridors. Afterwards, we provide our problem statement.

A. Set-Based Reachability Analysis

Suppose the dynamics of the ego vehicle is described by the discrete-time model

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k),$$

where $\mathbf{x}_k \in \mathcal{X}_k$ is the state of the ego vehicle and $\mathbf{u}_k \in \mathcal{U}_k$ is the system input. The sets $\mathcal{X}_k \subseteq \mathbb{R}^{n_\mathtt{x}}$ and $\mathcal{U}_k \subseteq \mathbb{R}^{n_\mathtt{u}}$ denote the feasible states and inputs at step k; in particular, $\mathcal{X}_0$ is the set of initial states of the ego vehicle including measurement uncertainties. A *trajectory* $\mathbf{x}_{[0,k_\mathtt{h}]} := \langle \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{k_\mathtt{h}} \rangle$ is *dynamically feasible* if $\mathbf{x}_0 \in \mathcal{X}_0$ and

$$\forall k \in [0, k_\mathtt{h} - 1] : \exists \mathbf{u}_k \in \mathcal{U}_k : \mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k) \in \mathcal{X}_{k+1}.$$

In addition to the dynamical constraints on trajectories, we want to consider constraints imposed by traffic rules, e.g., keeping a safe following distance. We assume these constraints are modeled by a specification φ, e.g., in temporal logic [4], [5]. If a trajectory $\mathbf{x}_{[0,k_\mathtt{h}]}$ satisfies φ, we say that it is *specification-compliant* and write $\mathbf{x}_{[0,k_\mathtt{h}]} \models \varphi$.

We can now define the exact *specification-compliant reachable set* at time step $k \in [0, k_\mathtt{h}]$ as

$$\mathcal{R}^\mathtt{e}_k := \{\mathbf{x}_k \mid \mathbf{x}_{[0,k_\mathtt{h}]} \in \mathcal{X} \wedge \mathbf{x}_{[0,k_\mathtt{h}]} \models \varphi\},$$

where $\mathcal{X}$ denotes the set of all dynamically feasible trajectories. Intuitively, $\mathcal{R}^\mathtt{e}_k$ is the set of all states that the ego vehicle can reach by following a dynamically feasible and specification-compliant trajectory for k steps.

As it is impossible to compute the exact reachable set $\mathcal{R}^\mathtt{e}$ efficiently, we usually compute an overapproximation $\mathcal{R}$ so that $\mathcal{R}_k \supseteq \mathcal{R}^\mathtt{e}_k$ for all $k \in [0, k_\mathtt{h}]$ [3]–[5]. To this end, $\mathcal{R}_k$ is represented by several *partial reachable sets* $\mathcal{R}^{(i)}_k$ with $\mathcal{R}_k = \bigcup_i \mathcal{R}^{(i)}_k$. To simplify presentation, we will sometimes also treat $\mathcal{R}_k$ as the set of the partial reachable sets constituting it. The partial reachable sets are stored as nodes of a *reachability graph* $G_\mathtt{R}$, in which the edges indicate the reachability relationship between partial sets of successive time steps; Fig. 2 shows an example.

To abstract from the details of computing the overapproximation, we introduce two high-level operators: $\mathrm{init}(\cdot)$ determines the initial reachable set $\mathcal{R}_0$ from the initial states $\mathcal{X}_0$.

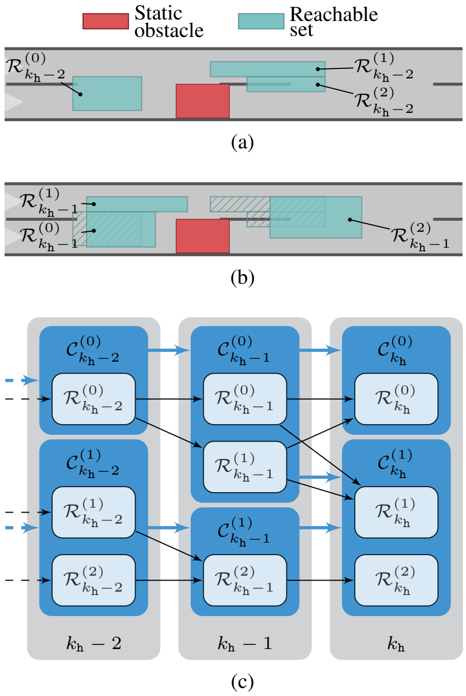


Fig. 2. Example of computing a reachability graph and its corresponding component graph. (a) The partial reachable sets at time step $k_{\mathrm{h}}-2$ (projected to the position domain). (b) Computing the successors of these sets using $\mathrm{succ}(\cdot)$ yields the partial reachable sets at $k_{\mathrm{h}}-1$. (c) The final reachability graph $G_{\mathtt{R}}$ and its corresponding component graph $G_{\mathtt{C}}$. The component graph is shown as an underlay with thick edges.

Given a set of partial reachable sets, $\mathrm{succ}(\cdot)$ computes their immediate successors in the reachability graph by performing reachability analysis for one time step (see Fig. 2b).

B. Driving Corridors

Because the reachable set $\mathcal{R}$ is an overapproximation, it contains all possible maneuvers of the ego vehicle. However, the set may be disconnected in the position domain, since we remove states that violate the specification φ, e.g., states where the ego vehicle collides with an obstacle. This makes it difficult to directly determine motion planning constraints from the reachable set. To resolve this issue, we identify driving corridors as connected subsets of the reachable set [3], [4]. Formally, a *connected component* $\mathcal{C}_k$ is a subset of $\mathcal{R}_k$ so that all contained partial reachable sets are connected (in the set-theoretic sense of [26, p. 2]) in the position domain. Intuitively, this means that their projection to the position domain consists of a single contiguous part. In Fig. 2a, there are two connected components: One consists only of $\mathcal{R}^{(0)}_{k_{\mathrm{h}}-2}$, while $\mathcal{R}^{(1)}_{k_{\mathrm{h}}-2}$ and $\mathcal{R}^{(2)}_{k_{\mathrm{h}}-2}$ form the other. If the partial reachable sets carry semantic annotations as in [4], [5], the sets in a connected component must also share the same annotation. A *driving corridor* is a sequence $\mathcal{C}_0, \dots, \mathcal{C}_{k_{\mathrm{h}}}$ of connected components so that $\mathcal{C}_{k+1}$ is reachable from $\mathcal{C}_k$.

To identify driving corridors, the works in [3]–[5] first incrementally compute the reachability graph $G_{\mathtt{R}}$ time step by time step. Then, $G_{\mathtt{R}}$ is converted into its *component graph* $G_{\mathtt{C}}$ by collapsing all partial reachable sets $\mathcal{R}^{(i)}_k$ belonging to the same connected component into a single node. Fig. 2c draws the reachability graph on top of its component graph, so that, e.g., $\mathcal{C}^{(1)}_{k_{\mathrm{h}}-2}$ comprises $\mathcal{R}^{(1)}_{k_{\mathrm{h}}-2}$ and $\mathcal{R}^{(2)}_{k_{\mathrm{h}}-2}$. Observe that there is an edge between two connected components in $G_{\mathtt{C}}$ if and only if they contain a pair of partial reachable sets linked in $G_{\mathtt{R}}$. Each path through the component graph from time step 0 to k_{h} constitutes a driving corridor. So, in Fig. 2c, $\mathcal{C}^{(0)}_{k_{\mathrm{h}}-2}, \mathcal{C}^{(0)}_{k_{\mathrm{h}}-1}, \mathcal{C}^{(1)}_{k_{\mathrm{h}}}$ is an example of a driving corridor assuming that $k_{\mathrm{h}} = 2$.

C. Problem Statement

To start motion planning early, our goal is to determine a first driving corridor without necessarily constructing the entire reachability graph. We want to find this corridor quickly, while still considering its quality, which may be defined as suitability for trajectory planning, but can also involve other metrics. We assume that a *cost function* $c(\cdot)$ measuring the quality of a connected component is given; the cost of a corridor $\mathcal{C}_0, \dots, \mathcal{C}_{k_{\mathrm{h}}}$ is defined as $\sum_{k\in[0,k_{\mathrm{h}}]} c(\mathcal{C}_k)$. Moreover, the search for corridors should be able to continue in order to identify additional corridors in case motion planning in the first corridor fails. This may be done in parallel to trajectory planning, if idle cores are available, or upon request by the planner.

We focus on developing a driving corridor extraction method that supports arbitrary cost functions. Selecting a specific cost function and analyzing the impact of this choice on the quality of the planned trajectories is beyond the scope of this paper. A discussion of different cost function components is provided in [4, Sec. VII-C].

III. Best-First Driving Corridor Extraction

Our algorithm, presented in Sec. III-A, addresses the problem statement by prioritizing the promising parts of the reachability graph $G_{\mathtt{R}}$ according to a given evaluation function. Once a first corridor is found, the algorithm can continue constructing the previously deferred parts of $G_{\mathtt{R}}$ to identify additional corridors. In Sec. III-B, we derive several evaluation functions from the cost function $c(\cdot)$.

A. Algorithm

Alg. 1 closely follows the standard skeleton for tree-like[1] best-first search [25, Fig. 3.7]. Given the set of initial states $\mathcal{X}_0$, we first compute the initial partial reachable sets $\mathcal{R}_0$. Then, we determine the connected components of $\mathcal{R}_0$ to initialize the *search frontier* $\mathcal{F}$. The frontier always contains those connected components that we have already computed but not yet added to the reachability graph. The goal states of the search are the connected components at the final time step k_{h}, because adding one of them to the reachability graph means that we can find new driving corridors.

As long as the frontier $\mathcal{F}$ is not empty, we select the best connected component $\mathcal{C}^*_k$ from $\mathcal{F}$ according to our *evaluation function* $e(\cdot)$. We remove $\mathcal{C}^*_k$ from $\mathcal{F}$ and add the contained partial reachable sets to the reachability graph. Next, we

[1]This means we do not use a reached set, which is unnecessary as the reachability graph is acyclic by construction and its depth is limited to k_{h}.

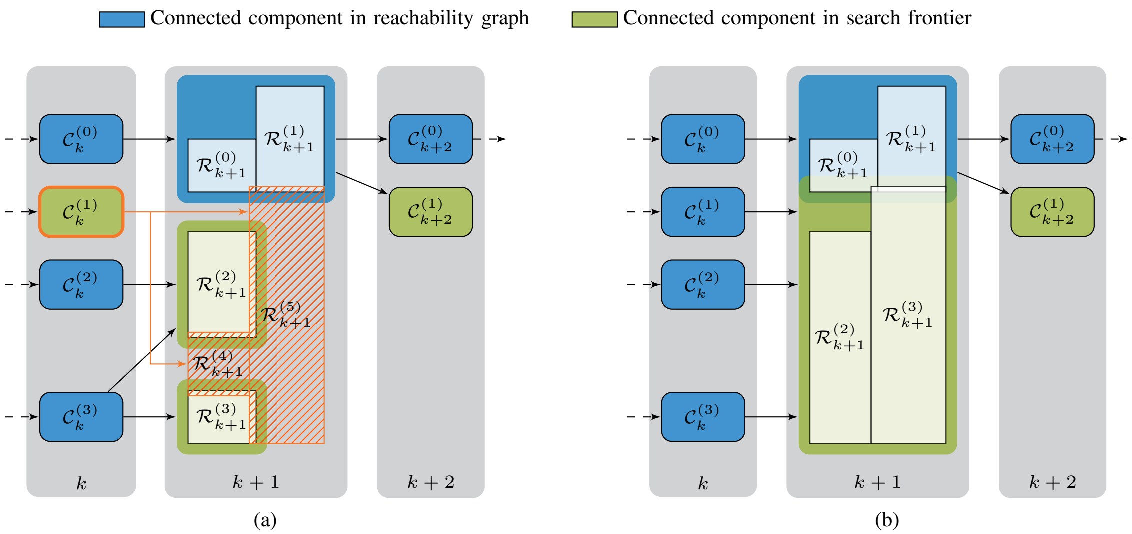


Fig. 3. Visualization of Alg. 1 for $\mathcal{C}_k^* = \mathcal{C}_k^{(1)}$. At time step $k+1$, we show the partial reachable sets belonging to each connected component (projected to the position domain). (a) Computing the successors of $\mathcal{C}_k^{(1)}$ yields the partial reachable sets $\mathcal{R}_{k+1}^{(4)}$ and $\mathcal{R}_{k+1}^{(5)}$ (l. 17). (b) After adding the new reachable sets, we repartition the frontier at time step $k+1$ (l. 18) and recompute the connected components (l. 19).

Algorithm 1 Best-First Driving Corridor Extraction

Input: Planning horizon $k_\mathtt{h}$, initial states $\mathcal{X}_0$
Output: A sequence of driving corridors

1: $\mathfrak{C} \leftarrow \emptyset$ ▷ Set of driving corridors returned so far
2: $\mathcal{R}_0 \leftarrow \mathrm{init}(\mathcal{X}_0)$
3: $\mathcal{F} \leftarrow$ ConnectedComponents($\mathcal{R}_0$)
4: **while** $\mathcal{F} \neq \emptyset$ **do**
5: ▷ Move the best connected component from $\mathcal{F}$ to $G_\mathtt{R}$
6: $\mathcal{C}_k^* \leftarrow \arg\min_{\mathcal{C}_{k'} \in \mathcal{F}} e(\mathcal{C}_{k'})$
7: $\mathcal{F} \leftarrow \mathcal{F} \setminus \mathcal{C}_k^*$
8: $G_\mathtt{R}$.Add($\mathcal{C}_k^*$)
9: **if** $k = k_\mathtt{h}$ **then** ▷ New terminal node in $G_\mathtt{R}$?
10: ▷ Determine $G_\mathtt{C}$ and yield new corridors
11: $G_\mathtt{C} \leftarrow$ ComponentGraph($G_\mathtt{R}$)
12: $\mathfrak{C}_\mathtt{new} \leftarrow$ IdentifyCorridors($G_\mathtt{C}$)
13: **yield** $\mathfrak{C}_\mathtt{new} \setminus \mathfrak{C}$
14: $\mathfrak{C} \leftarrow \mathfrak{C} \cup \mathfrak{C}_\mathtt{new}$
15: **else**
16: ▷ Add the successors of $\mathcal{C}_k^*$ to $\mathcal{F}$
17: $\mathcal{R}_{k+1} \leftarrow \mathrm{succ}(\mathcal{C}_k^*)$
18: $\mathcal{R}'_{k+1} \leftarrow$ Repartition($(\bigcup \mathcal{F}_{k+1}) \cup \mathcal{R}_{k+1}$)
19: $\mathcal{F}_{k+1} \leftarrow$ ConnectedComponents($\mathcal{R}'_{k+1}$)
20: **end if**
21: **end while**

check whether $\mathcal{C}_k^*$ belongs to the final time step $k_\mathtt{h}$. If it does, we can identify new driving corridors. Thus, we construct the component graph $G_\mathtt{C}$ of the current reachability graph $G_\mathtt{R}$ as described in Sec. II-B. We employ graph algorithms like Yen's algorithm [27] to find up to n driving corridors in $G_\mathtt{C}$ with the lowest cost. Choosing $n > 1$ is reasonable, as the planner may fail to find a trajectory in the first corridor due to the overapproximative nature of reachability analysis. We provide all corridors we did not see before to the trajectory planner without exiting the algorithm (keyword **yield** in l. 13). If idle cores are available, the algorithm continues the reachability analysis immediately in parallel to planning; otherwise, it waits for the planner to request additional corridors.

Let us now consider the **else**-branch: If $\mathcal{C}_k^*$ does not belong to the final time step, i.e., $k < k_\mathtt{h}$, we compute its successors and add them to the frontier. Fig. 3 visualizes this process, assuming that the connected component $\mathcal{C}_k^{(1)}$ (highlighted in the figure) was selected from the frontier in l. 6. Applying the $\mathrm{succ}(\cdot)$ operator in l. 17 yields the partial reachable sets represented by the hatched area in Fig. 3a. In ll. 18–19, we add the new partial reachable sets for time step $k+1$ to the search frontier $\mathcal{F}$. We first merge the new reachable sets with the reachable sets that are already in $\mathcal{F}_{k+1}$ and repartition them to remove redundancies in the set representation. Here, $\mathcal{F}_{k+1}$ denotes the subset of the frontier that holds the connected components belonging to time step $k+1$. Fig. 3b shows the result of the repartitioning. Note that $\mathcal{R}_{k+1}^{(0)}$ and $\mathcal{R}_{k+1}^{(1)}$, which are already in the reachability graph and not in the frontier, are not considered for repartitioning, although they overlap with the newly added sets. The details of the repartitioning procedure depend on the set representation used; we refer the reader to [5, Sec. III-A.3] for an example. As is the case in Fig. 3, the newly added sets might connect previously separate connected components. Thus, we need to recompute the connected components for time step $k+1$.

Remark 1. Several of the standard search methods are not useful for our problem: Breadth-first search [25, Sec. 3.4.1] leads to a procedure similar to existing methods [3]–[5] and thus has the same drawbacks. Iterative deepening techniques [25, Sec. 3.4.4] do not make sense here, since we know that all goals are at depth $k_\mathtt{h}$.

B. Evaluation Functions

By defining appropriate evaluation functions $e(\cdot)$ based on the cost function $c(\cdot)$, we can employ any search strategy that can be cast as a best-first search [25, Sec. 3.3.1]. This

TABLE I

EVALUATION FUNCTIONS

Search strategy	Evaluation function $e(\mathcal{C}_k)$
Uninformed	
Uniform-cost	$c_{\mathtt{acc}}(\mathcal{C}_k) + c(\mathcal{C}_k)$
Uninformed speedy	$(k_{\mathtt{h}} - k, c(\mathcal{C}_k))$
Informed	
A*	$c_{\mathtt{acc}}(\mathcal{C}_k) + c(\mathcal{C}_k) + h(\mathcal{C}_k)$
Greedy best-first	$h(\mathcal{C}_k)$
Informed speedy	$(k_{\mathtt{h}} - k, h(\mathcal{C}_k))$

includes, but is not limited to, the search strategies with their respective evaluation functions listed in Tab. I. Below, we discuss these evaluation functions in detail.

1) Uninformed Search: *Uninformed* search does not use any external information about the problem [25, Sec. 3.4]. Our first evaluation function results in *uniform-cost* search, also known as Dijkstra's algorithm [25, Sec. 3.4.2]. Here, $c_{\mathtt{acc}}(\mathcal{C}_k)$ is the *accumulated cost* along the cheapest path to $\mathcal{C}_k$, which is formally given by

$$c_{\mathtt{acc}}(\mathcal{C}_k) := \begin{cases} 0 & \text{if } k = 0 \\ \min\limits_{\mathcal{C}_{k-1} \in \mathcal{P}} (c_{\mathtt{acc}}(\mathcal{C}_{k-1}) + c(\mathcal{C}_{k-1})) & \text{otherwise} \end{cases},$$

where $\mathcal{P}$ denotes the parents of $\mathcal{C}_k$, i.e., the connected components of time step $k-1$ from which we can reach $\mathcal{C}_k$. To efficiently compute $c_{\mathtt{acc}}(\cdot)$, we cache the cost of the connected components and track their parent relationships in Alg. 1. Note that we need to update the parents after repartitioning and recomputing the connected components in ll. 18–19. Uniform-cost search is guaranteed to find the optimal corridor first.

Using the second uninformed evaluation function leads to *uninformed speedy* search in the spirit of [28], because we order tuples lexicographically. Speedy search prefers progressing towards the planning horizon and uses the cost only to break ties between connected components belonging to the same time step. Consequently, it quickly yields a first driving corridor, which may, however, be suboptimal regarding its cost. Our variant of speedy search differs from [28] in that it uses only the immediate cost to break ties instead of a heuristic and the accumulated cost.

2) Informed Search: In order to use *informed* search strategies, we require a *heuristic* $h(\cdot)$ that estimates the future costs of a connected component [25, Sec. 3.5]. We design a general purpose heuristic that is agnostic of the employed cost function $c(\cdot)$; for fixed $c(\cdot)$, one could define specialized heuristics or derive them from data using machine learning.

For our heuristic, we exploit that the trajectory is replanned periodically, and we thus know the reachable set of the ego vehicle from its state $k_{\mathtt{c}}$ time steps ago. We can estimate the future costs of a connected component by finding matching connected components in the component graph $\tilde{G}_{\mathtt{C}}$ of the previous planning cycle and considering their future costs.

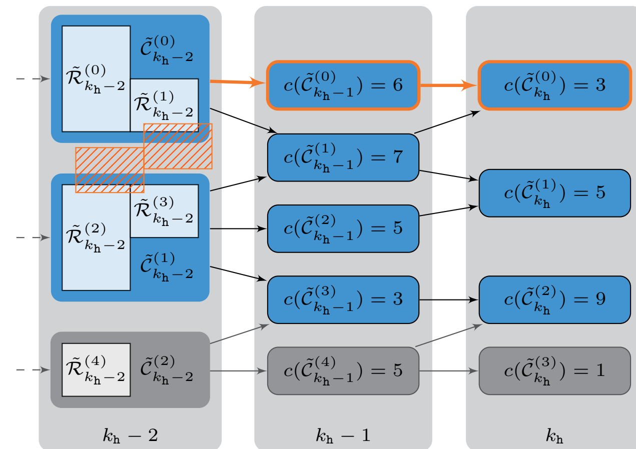


Fig. 4. Heuristic computation based on the component graph $\tilde{G}_{\mathtt{C}}$ of the previous planning cycle. At step $k_{\mathtt{h}} - 2$, we show the partial reachable sets belonging to each connected component (projected to the position domain).

A connected component $\tilde{\mathcal{C}}_{k+k_{\mathtt{c}}} \in \tilde{G}_{\mathtt{C}}$ *matches* $\mathcal{C}_k$ if

$$\exists \mathcal{R}^{(i)}_k \in \mathcal{C}_k, \tilde{\mathcal{R}}^{(j)}_{k+k_{\mathtt{c}}} \in \tilde{\mathcal{C}}_{k+k_{\mathtt{c}}} : \mathcal{R}^{(i)}_k \cap \tilde{\mathcal{R}}^{(j)}_{k+k_{\mathtt{c}}} \neq \emptyset.$$

Fig. 4 visualizes this for the connected component $\mathcal{C}_{k_{\mathtt{h}}-4}$ that is represented by the hatched area. Assuming that $k_{\mathtt{c}} = 2$, we check which connected components at time step $k_{\mathtt{h}} - 2$ in $\tilde{G}_{\mathtt{C}}$ intersect with $\mathcal{C}_{k_{\mathtt{h}}-4}$ and find that $\tilde{\mathcal{C}}^{(0)}_{k_{\mathtt{h}}-2}$ and $\tilde{\mathcal{C}}^{(1)}_{k_{\mathtt{h}}-2}$ match. To determine $h(\mathcal{C}_{k_{\mathtt{h}}-4})$, we use Dijkstra's algorithm to find the cost of the cheapest path from $\tilde{\mathcal{C}}^{(0)}_{k_{\mathtt{h}}-2}$ or $\tilde{\mathcal{C}}^{(1)}_{k_{\mathtt{h}}-2}$ to the final time step $k_{\mathtt{h}}$. Here, this path has the cost 9 and is highlighted in Fig. 4; the part of $\tilde{G}_{\mathtt{C}}$ that is disregarded when searching for the cheapest path is grayed out. Formally, we define our heuristic $h(\mathcal{C}_k)$ as the cost of the cheapest path in $\tilde{G}_{\mathtt{C}}$ from a connected component matching $\mathcal{C}_k$ to some connected component at the final time step $k_{\mathtt{h}}$. If no such path exists, e.g., because $k > k_{\mathtt{h}} - k_{\mathtt{c}}$, the heuristic conservatively returns 0.

Using our heuristic $h(\cdot)$, we can define evaluation functions that result in informed search strategies. Here, we consider the well-known *A** algorithm and *greedy best-first* search [25, Sec. 3.5]. Moreover, we define an informed variant of speedy search using $h(\cdot)$ in place of $c(\cdot)$. While greedy best-first search implicitly favors connected components with higher time steps as they tend to have lower heuristic values, the speedy strategy makes this preference explicit. Even though our heuristic $h(\cdot)$ strives to underestimate the actual cost, we cannot provide a guarantee since the behavior predictions for the other traffic participants might be inaccurate. Consequently, $h(\cdot)$ is generally not an *admissible* heuristic [25, Sec. 3.5.2], and thus A* will not necessarily find the optimal corridor first.

IV. EVALUATION

We compare *our approach* from Alg. 1 with the procedure from [5], which we refer to as the *baseline approach*. For our approach, we consider the search strategies listed in Tab. I.

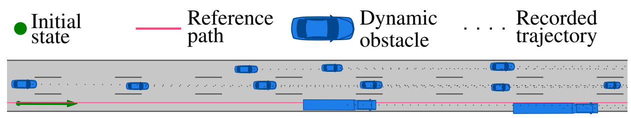


Fig. 5. Example scenario from the highD dataset.

A. Experiment Setup

For our experiments, we use $c(\mathcal{C}_k) := e^{-wA(\mathcal{C}_k)}$ as the cost function, where $A(\mathcal{C}_k)$ is the area of the road covered by the connected component $\mathcal{C}_k$ and w is empirically set to 0.001. In general, a larger area provides the trajectory planner with more flexible constraints [18], but the benefit of extra area diminishes as the overall area becomes larger. Moreover, a corridor with a larger area makes the planned trajectory more robust with respect to inaccurate behavior predictions of other traffic participants.

We use the extracted driving corridors to derive a sampling space for the popular trajectory planner in [29], following the method described in [2]. The planner uses a modified version of the Hydra monitor [30] to check whether the sampled trajectories are specification-compliant. We refer the reader to [31, Sec. V and VIII] for a high-level overview of how the sampling-based planner is combined with driving corridors and monitoring.

Our prototype implementation is developed using the CommonRoad[2] [32] framework. We performed our measurements on a laptop with an Intel Core i7-12700H 4.7 GHz processor. The reported computation times are average values from five runs.

B. Benchmarking in Real-World Scenarios

For our benchmark, we extracted 100 CommonRoad scenarios from the highD dataset [33] in which the ego vehicle starts on the rightmost lane of the interstate; an example scenario is shown in Fig. 5. We set $\Delta t = 0.08\,\mathrm{s}$ and $k_{\mathtt{h}} = 50$. We enforce a safe distance to vehicles ahead by applying the traffic rule R_G1 from [34, Tab. I]. Finally, we assume that there is a solid-dashed line marking at the left boundary of the right lane, so that the ego vehicle must not re-enter the right lane after changing to another lane. Using predicates from [34, Sec. IV] and V_{ego} to denote the ego vehicle, we formalize this in temporal logic as $\mathbf{G}\,\varphi$, where

$$\begin{aligned}\varphi := \neg\text{main\_carriageway\_right\_lane}(V_{\text{ego}}) \rightarrow \\ \mathbf{G}\,\neg\text{main\_carriageway\_right\_lane}(V_{\text{ego}})\end{aligned}$$

and the temporal operator $\mathbf{G}$ demands that a formula holds at all future time steps. Due to this rule, the ego vehicle has to decide at which time step to leave the right lane; this decision induces several driving corridors.

We first complete one planning cycle using the baseline approach; our evaluation then considers the next planning cycle $k_{\mathtt{c}} = 5$ time steps later so that our heuristic $h(\cdot)$ can access the component graph of the previous planning cycle. To simulate inaccurate predictions for other traffic participants, the first cycle uses a simple constant velocity prediction [35], while the second cycle uses the actual recorded trajectories.

For all 100 scenarios, the planner found a trajectory in the first corridor provided by the baseline approach. This demonstrates that it makes sense to quickly compute a first corridor and only determine additional corridors on demand.

Tab. II lists the average number of connected components in the component graph $G_{\mathtt{C}}$ and the frontier $\mathcal{F}$ with different search strategies after all corridors required by the planner have been computed. The numbers shown are relative to the size of the component graph computed with the baseline approach. Greedy best-first search and the two speedy strategies construct roughly one fourth of the component graph, while uniform-cost search and A* construct half of it. For all search strategies except uninformed speedy search, the frontier contains only a few connected components in the end. This is because uninformed speedy search first computes the corridor that stays in the right lane, since a lane change initially produces a connected component with a smaller area. Thus, it needs to consider changing the lane at each time step, which adds at least two connected components to the frontier. The other strategies decide to change lanes earlier and then no longer need to make this decision.

TABLE II

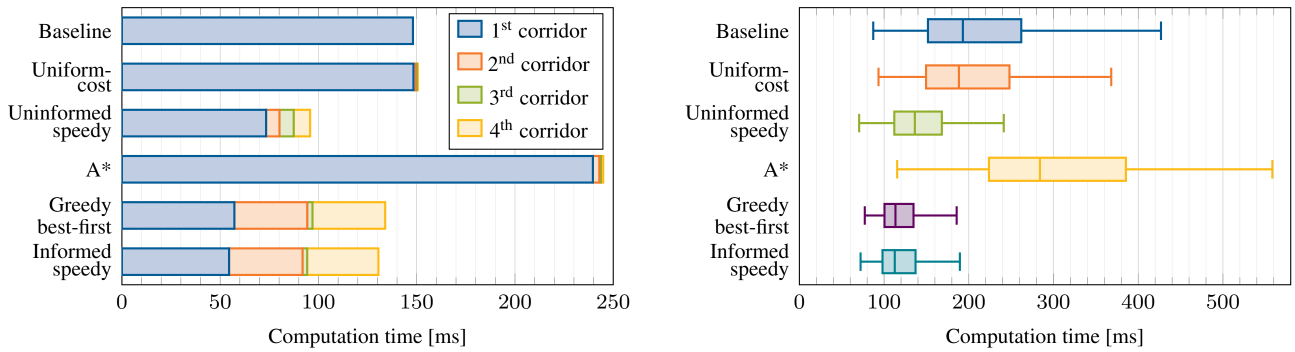
Number of Connected Components Relative to Baseline (Mean ± Standard Deviation)

Search strategy	Size $G_{\mathtt{C}}$	Size $\mathcal{F}$
Uniform-cost	$(56 \pm 18)\,\%$	$(2.3 \pm 0.8)\,\%$
Uninformed speedy	$(27 \pm 11)\,\%$	$(22 \pm 4)\,\%$
A*	$(56 \pm 21)\,\%$	$(2.8 \pm 1.2)\,\%$
Greedy best-first	$(27 \pm 11)\,\%$	$(5.4 \pm 6.6)\,\%$
Informed speedy	$(27 \pm 11)\,\%$	$(5.3 \pm 6.5)\,\%$

Fig. 6a visualizes the average time required to identify the first four driving corridors. With the baseline approach, all corridors become available at the same time. We observe a very similar behavior for uniform-cost search and A*, indicating that corridors two to four are already mostly computed before the first corridor is returned. While uniform-cost search performs similarly to the baseline, the performance of A* is poor: Compared to uniform-cost search, A* incurs the additional overhead of computing the heuristic $h(\cdot)$. Greedy best-first search and the two speedy variants identify four corridors before the baseline provides the first. It is at first surprising that uninformed speedy search is slower in determining the first corridor than the informed algorithms. However, this is also explained by its aforementioned preference to stay in the right lane and the associated higher number of decisions to be made. The boxplots of the total computation time until finding a dynamically feasible and specification-compliant trajectory in Fig. 6b show similar relationships between the search strategies.

The median total computation time of informed speedy

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



(a)

(b)

Fig. 6. Comparison of the computation times of different search strategies on scenarios from the highD dataset. (a) Average computation time of the first four corridors. (b) Total computation time for planning a feasible and specification-compliant trajectory (outliers are not shown).

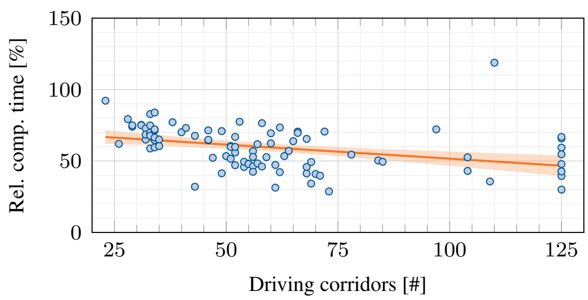


Fig. 7. Total computation time of informed speedy search relative to the baseline approach over the number of driving corridors in the scenario. The regression line includes a 95 % confidence interval. The number of driving corridors is clamped to 125. Note that the horizontal axis starts at 20.

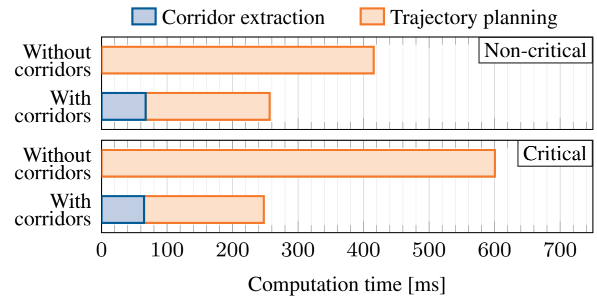


Fig. 8. Comparison of the computation time for planning with and without driving corridors in the critical and non-critical scenario variants.

search is 62 % of the baseline approach. In Fig. 7, we plot this relative computation time over the number of driving corridors in the scenario identified by the baseline. Expectedly, the benefit of our method increases with the number of corridors, as the linear regression illustrates. We can observe similar trends for uninformed speedy search and greedy best-first search with median relative computation times of 70 % and 63 %. However, there is no clear trend for uniform-cost search or A*, which have a median relative computation time of 99 % and 137 %, respectively. We measured an increase in total computation time by more than 50 % in four scenarios for informed speedy search; these are excluded from Fig. 7. Such scenarios also exist for some of the other search strategies: 34 for A*, four for greedy best-first search, and one for uninformed speedy search.

C. Scenarios with Narrow Solution Spaces

It has been shown that set-based methods have the desirable property of decreasing the computation time in critical scenarios with narrow solution spaces [2, Sec. IV-B], [3, Sec. VII-E], [4, Sec. VIII-D]. We briefly demonstrate that our approach retains this property. To this end, we consider two variants of the scenario shown in Fig. 1, which differ only in the velocity of the dynamic obstacle: In the critical variant shown in Fig. 1, the obstacle moves 1.4 m/s faster than in the non-critical variant. Thus, the critical variant is particularly challenging because the gap between the two obstacles that the ego vehicle must pass through in order to complete the overtaking maneuver is narrow. For our experiment, we use uninformed speedy search and set $\Delta t = 0.1\,\mathrm{s}$ and $k_{\mathrm{h}} = 50$.

As shown in Fig. 8, the computation time for trajectory planning without driving corridors increases by 45 % for the critical variant compared to the non-critical variant. In contrast, the total computation time with corridors in the critical variant even decreases slightly by 9 ms.

V. Conclusion

By treating reachability analysis for driving corridor extraction as a best-first search problem, our novel approach can quickly provide the trajectory planner with a first driving corridor. Additional corridors beyond the first one, which the planner rarely needs, are computed on demand or in parallel to planning if idle cores are available. Our method reduces the waiting time of the planner for a first corridor compared to existing approaches, which require the entire reachable set to be computed before any corridors become available. As our experiments show, the advantages of our approach are particularly evident in complex scenarios with many driving corridors and scenarios with narrow solution spaces. We will include our implementation in a future release of CommonRoad.

REFERENCES

[1] E. Plaku and S. Karaman, "Motion planning with temporal-logic specifications: Progress and challenges," *AI Communications*, vol. 29, no. 1, pp. 151–162, 2015.

[2] G. Würsching and M. Althoff, "Sampling-based optimal trajectory generation for autonomous vehicles using reachable sets," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, 2021, pp. 828–835.

[3] S. Manzinger, C. Pek, and M. Althoff, "Using reachable sets for trajectory planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 2, pp. 232–248, 2021.

[4] E. Irani Liu and M. Althoff, "Specification-compliant driving corridors for motion planning of automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 9, pp. 4180–4197, 2023.

[5] F. Lercher and M. Althoff, "Specification-compliant reachability analysis for autonomous vehicles using on-the-fly model checking," in *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, 2024, pp. 1484–1491.

[6] M. Althoff and J. M. Dolan, "Online verification of automated road vehicles using reachability analysis," *IEEE Transactions on Robotics*, vol. 30, no. 4, pp. 903–918, 2014.

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

[8] X. Wang and M. Althoff, "Safe reinforcement learning for automated vehicles via online reachability analysis," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 9, pp. 5490–5505, 2024.

[9] F. Altché and A. de La Fortelle, "Partitioning of the free space-time for on-road navigation of autonomous ground vehicles," in *Proc. of the IEEE Conf. on Decision and Control (CDC)*, 2017, pp. 2126–2133.

[10] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, 2018, pp. 1053–1060.

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

[12] M. Schmidt, C. Wissing, J. Braun, T. Nattermann, and T. Bertram, "Maneuver identification for interaction-aware highway lane change behavior planning based on polygon clipping and convex optimization," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, 2019, pp. 3948–3953.

[13] R. Kohlhaas, D. Hammann, T. Schamm, and J. M. Zöllner, "Planning of high-level maneuver sequences on semantic state spaces," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, 2015, pp. 2090–2096.

[14] T. Gu, J. M. Dolan, and J.-W. Lee, "Automated tactical maneuver discovery, reasoning and trajectory planning for autonomous driving," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016, pp. 5474–5480.

[15] B. Yi, P. Bender, F. Bonarens, and C. Stiller, "Model predictive trajectory planning for automated driving," *IEEE Transactions on Intelligent Vehicles*, vol. 4, no. 1, pp. 24–38, 2019.

[16] V. Z. Patterson, F. E. Lewis, and J. C. Gerdes, "Optimal decision making for automated vehicles using homotopy generation and nonlinear model predictive control," in *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, 2021, pp. 1045–1050.

[17] Y. Chen, S. Veer, P. Karkus, and M. Pavone, "Interactive joint planning for autonomous vehicles," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 987–994, 2024.

[18] S. Söntges and M. Althoff, "Computing possible driving corridors for automated vehicles," in *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, 2017, pp. 160–166.

[19] L. Schäfer, S. Manzinger, and M. Althoff, "Computation of solution spaces for optimization-based trajectory planning," *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 1, pp. 216–231, 2023.

[20] N. Kochdumper and S. Bak, "Real-time capable decision making for autonomous driving using reachable sets," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2024, pp. 14 169–14 176.

[21] Y. Liu, X. Pei, H. Zhou, and X. Guo, "Spatiotemporal trajectory planning for autonomous vehicle based on reachable set and iterative LQR," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 8, pp. 10 932–10 947, 2024.

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

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

[24] M. Morsali, E. Frisk, and J. Åslund, "Spatio-temporal planning in multi-vehicle scenarios for autonomous vehicle using support vector machines," *IEEE Transactions on Intelligent Vehicles*, vol. 6, no. 4, pp. 611–621, 2021.

[25] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 4th ed. Pearson, 2021.

[26] H. T. Croft, K. J. Falconer, and R. K. Guy, *Unsolved Problems in Geometry: Unsolved Problems in Intuitive Mathematics*. Springer, 1991, vol. 2.

[27] J. Y. Yen, "Finding the k shortest loopless paths in a network," *Management Science*, vol. 17, no. 11, pp. 712–716, 1971.

[28] W. Ruml and M. B. Do, "Best-first utility-guided search," in *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2007, pp. 2378–2384.

[29] 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 (ICRA)*, 2010, pp. 987–993.

[30] M. Raszyk, D. Basin, S. Krstić, and D. Traytel, "Multi-head monitoring of metric temporal logic," in *Automated Technology for Verification and Analysis (ATVA)*, 2019, pp. 151–170.

[31] M. Althoff, S. Maierhofer, G. Würsching, Y. Lin, F. Lercher, and R. Stolz, "No more traffic tickets: A tutorial to ensure traffic-rule compliance of automated vehicles," *Proceedings of the IEEE*, pp. 1–30, 2025, Early Access.

[32] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symp. (IV)*, 2017, pp. 719–726.

[33] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems," in *Proc. of the IEEE Int. Intelligent Transportation Systems Conf. (ITSC)*, 2018, pp. 2118–2125.

[34] 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 Symp. (IV)*, 2020, pp. 752–759.

[35] R. Schubert, E. Richter, and G. Wanielik, "Comparison and evaluation of advanced motion models for vehicle tracking," in *Proc. of the Int. Conf. on Information Fusion (FUSION)*, 2008, pp. 1–6.