2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Computing Possible Driving Corridors for Automated Vehicles

Sebastian Söntges and Matthias Althoff[1]

Abstract— **Motion planning in dynamic traffic scenes is a challenging problem. In particular, since it is unknown during planning whether a certain decision, such as passing another traffic participant on the left or right, will result in a safe and comfortable motion. Exhaustive exploration of all principle driving paths is computationally expensive, so that one typically reverts to heuristics—this, however can be unsatisfactory in situations when the heuristics fail to find a solution although it exists. We address this problem by computing the union of all possible motions for a sequence of high-level decisions (e.g. overtake vehicle on the left and then another one on the right), which we refer to as a driving corridor. Our proposed algorithm is over-approximative, i.e. the union of driving corridors provably encloses all possible motions. Thus, if the set of reachable positions within a driving corridor becomes empty, the corresponding sequence of high-level decisions is infeasible and can be discarded by the motion planner. Driving corridors also facilitate selecting high-level plans: Large driving corridors should be preferred since they provide more opportunities for optimizing motions and are more robust towards unpredicted changes. Numerical examples demonstrate the usefulness of our approach.**

I. Introduction

There exist two major techniques for motion planning of automated vehicles: Graph-based search techniques and variational-based optimization techniques [1], [2]. While graph-based techniques explore the search space by discretizing the action space or the state space, they can implicitly explore high-level decisions, such as passing another vehicle on the left or right [3]. The situation is different for variational-based optimization techniques. Continuous optimization can get stuck in local minima [4], [5] and requires a high-level strategy upfront to set up constraints for collision avoidance [6], [7]. These constraints define the driving corridor of the vehicle and how obstacles are passed. Even though a graph-based planner does not necessarily require a set of promising high-level plans, it certainly speeds up the search process.

Due to the aforementioned necessity for variational-based optimization and the benefits for graph-based search, there is a high interest in finding feasible high-level plans. High-level decision making is often implemented by state-machines and traffic situation-specific rules [2], [8], [9]. More general situations can be handled by planners which search different combinations of high-level actions [10]. One difficulty in these approaches is to guarantee physical feasibility of the planned action sequence. This issue is addressed in [3] by discovering possible maneuvers through sampling of feasible trajectories. The sampled trajectories are grouped by topological properties in the spatial-time domain into different high-level maneuvers. Topology-based approaches like path homotopy or homology are used in a range of applications in robotics for high-level classification of paths [11]. However, these techniques cannot be applied directly to automated driving, since path homotopy and homology requires the paths to share the same goal configurations [5]. A combined approach which integrates topological analysis in a trajectory optimization formulation with mixed-integer programming is presented in [12]. However, the proposed method examines the topology by a cell-decomposition of a two-dimensional workspace, which makes it only applicable to quasi-static environments.

In this work, we propose a method for computing the reachable set of all possible motions of the ego vehicle and for dividing the reachable set into subsets, where each subset corresponds to a different sequence of high-level decisions. We refer to these subsets as driving corridors. The numerical computation of the reachable set is over-approximative, i.e. no feasible trajectory is missed. One advantage of using the reachable set instead of sampled trajectories is that we do not rely on a particular sampling strategy, which may not perform equally well in different scenarios. The driving corridor of a high-level plan might be very small, meaning that small variations in the traffic prediction could render a chosen motion unsafe. If, at a certain point in time, the driving corridor vanishes, no solution for the considered high-level plan exists. In contrast to sampling-based works, we can prove the nonexistence of solutions by relying on our over-approximative computation of feasible motions. The presented work is an extension of our previous work [13][14], which only computes drivable areas without grouping them into driving corridors.

Our paper is organized as follows: Section II introduces the basic definitions and states the problem. Section III describes our model assumptions and presents the proposed algorithm to compute the reachable set and its division into driving corridors. Section IV demonstrates the proposed approach on two examples. Finally, Section V presents our conclusions.

II. Definitions and Problem Statement

Definition 1 (Trajectory planning problem): Let us assume we have:

1) A dynamical system $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$ of the vehicle with state $\mathbf{x}(t) \in \mathcal{X}$ and input $\mathbf{u}(t) \in \mathcal{U}(\mathbf{x}(t))$,
2) a set of (time-dependent) obstacles $\mathcal{O}(t) \subseteq \mathbb{R}^2$,
3) the covered region of the ego vehicle $\mathcal{A}(\mathbf{x}(t)) \subseteq \mathbb{R}^2$,

[1]Sebastian Söntges and Matthias Althoff are with the Department of Informatics, Technical University of Munich (TUM), Boltzmannstraße 3, 85748 Garching, Germany. Corresponding email: {soentges, althoff}@in.tum.de

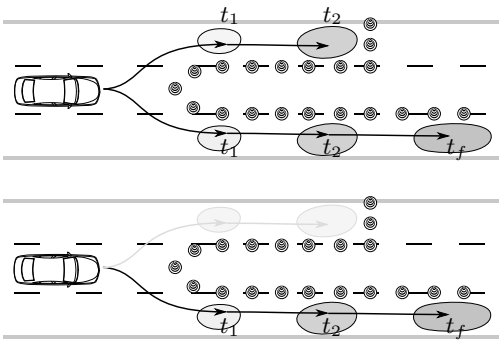


Fig. 1. Reachable set considering collision free trajectories until the given time step (top); anticipated reachable set considering only collision free trajectories for the whole planning horizon $[t_0, t_f]$ (bottom).

4) an initial state $\mathbf{x}_0$ at time t_0,
5) and a goal region $\mathcal{X}_G$ at time t_f.

Given an input signal $\mathbf{u}(t)$ and initial state $\mathbf{x}_0$, the trajectory of the system $f(\mathbf{x}(t), \mathbf{u}(t))$ is:

$$\mathbf{x}(t) = \mathbf{x}_0 + \int_{t_0}^{t} f(\mathbf{x}(t'), \mathbf{u}(t'))\mathrm{d}t' \tag{1}$$

which we denote by $\tau(t; \mathbf{u}, \mathbf{x}_0) := \mathbf{x}(t)$.

A solution τ of the trajectory planning problem connects the initial state $\tau(t_0; \mathbf{u}, \mathbf{x}_0) = \mathbf{x}_0$ to a state in the goal region $\tau(t_f; \mathbf{u}, \mathbf{x}_0) \in \mathcal{X}_G$ and does not collide with any obstacle $\mathcal{A}(\tau(t'; \mathbf{u}, \mathbf{x}_0)) \cap \mathcal{O}(t') = \emptyset,\ \forall t' \in [t_0, t_f]$.

The states attained by the set of all trajectories at some time t are related to the reachable set of the dynamical system.

Definition 2 (Anticipated reachable set): The reachable set is defined as the set of all states which can be reached by the system from a given initial set $\mathcal{X}_0$ at a given time t. We extend the common definition to the *anticipated reachable set* through two restrictions. First, we only consider states that can be reached at time t without colliding with the obstacle set $\mathcal{O}(t)$. Second, we require that the state can be continued by a collision-free trajectory until the end of the planning horizon t_f. To simplify the notation, we introduce the set of forbidden states $\mathcal{F}(t) := \{\mathbf{x}(t) | \mathcal{A}(\mathbf{x}(t)) \cap \mathcal{O}(t) \neq \emptyset\}$. Thus,

$$\begin{aligned}\mathbf{reach}(\mathcal{X}_0, t, t_f) := \{\tau(t; \mathbf{u}, \mathbf{x}_0) \,|\exists \mathbf{u}(t') \in \mathcal{U},\ \exists \mathbf{x}_0 \in \mathcal{X}_0,\\ \tau(t'; \mathbf{u}, \mathbf{x}_0)) \notin \mathcal{F}(t') \text{ for } t' \in [t_0, t_f]\Big\},\end{aligned} \tag{2}$$

where $\mathcal{U}$ is the set of all possible input signals.

Fig. 1 shows the motivation behind this definition. For a given scenario, the ego vehicle must merge either into the left or right lane in order to avoid a collision. If the vehicle merges into the left lane, it can reach some region in the left lane for several time steps. However, it will eventually collide with some obstacle within the planning horizon t_f. Planning trajectories in these regions must be avoided. The anticipated reachable set excludes these regions.

Often, not the full state $\mathbf{x}(t)$, but only the position of the vehicle $(x(t), y(t))^T$ is of interest.

Definition 3 (Projection): Given the state $\mathbf{x}(t)$, the projection

$$\mathrm{proj}_{xy}(\mathbf{x}(t)) := (x(t), y(t))^T \tag{3}$$

is defined as the mapping from the state to the position of the vehicle.

We use the same notation to project a set of states $\mathcal{X}$:

$$\mathrm{proj}_{xy}(\mathcal{X}) := \{\mathrm{proj}_{xy}(\mathbf{x}) | \mathbf{x} \in \mathcal{X}\}. \tag{4}$$

A frequent decision for a driver is to choose between passing an obstacle on the left or right. In order to group feasible trajectories into different high-level decisions, a common method is to apply topological concepts of path homotopy and homology to motion planning [11]. Two paths with the same starting point and the same endpoint are said to be homotopic, if and only if there exists a continuous mapping, which deforms one path to the other. In particular this means that the path must not "jump" over any obstacle.

Fig. 2 shows an example of the ego vehicle and two moving vehicles in the spatial-time domain: Vehicle A drives in the same lane as the ego vehicle, vehicle B drives in the opposite lane and passes both other vehicles. Three possible trajectories $\tau_{1,2,3}$ of the ego vehicle are shown. This example is a modified version of the one presented in [5] with the difference that we assume that there is free space between vehicle A and B when both pass each other. In this example, τ_2 and τ_3 first pass vehicle B and then overtake vehicle A. However, since both do not share the same endpoint, they are not homotopic. In contrast, τ_1 and τ_3 are homotopic since they share both the same starting point and endpoint and can continuously be deformed to each other (under the assumption that vehicle A and B do not touch and the ego vehicle can drive between both). However, this grouping seems counterintuitive. It seems more natural to group trajectories τ_2 and τ_3 as the maneuver "first pass vehicle B, then overtake vehicle A". In this work, we propose a method for grouping trajectories which is inspired by a method for identifying homotopy between paths using two-dimensional sub-manifolds in the three dimensional position-time domain of the vehicle [15]. However, in contrast to [15], our goal is not to group paths into homotopy classes in its strict mathematical definition due to the aforementioned problems, but to find a grouping which fits the semantic meaning in terms of automated driving (e.g. passing another vehicle on the left).

The idea is to construct for each trajectory a "word" which describes the trajectory and assigns it to a unique driving corridor. All trajectories with the same word belong to the same driving corridor.

Definition 4 (Word construction): Given a set of two-dimensional oriented surfaces U_i in the three-dimensional spatial-time domain with an associated "letter" r_i and a trajectory τ. We assign a word to τ by concatenating the letters in the order of all intersections of τ with U_i. If U_i is crossed in the direction of its orientation we use the letter r_i

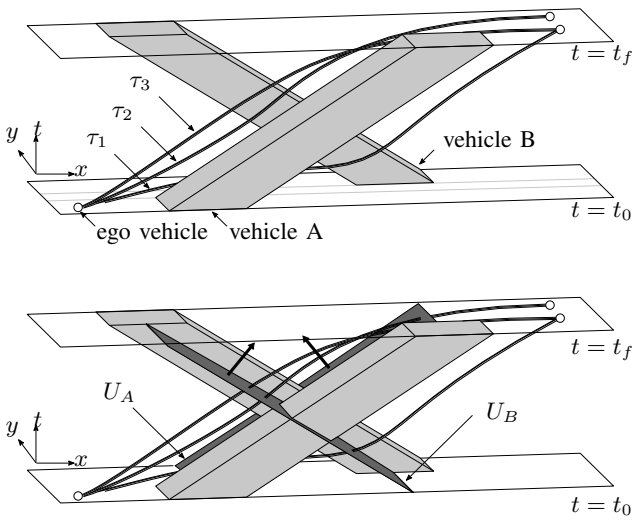


Fig. 2. Three different trajectories to overtake vehicle A and pass vehicle B (top); surfaces U_A and U_B with their orientation are added (bottom).

and if it is crossed opposing its orientation, we use the inverse letter r_i^{-1}. If a letter r_i and its inverse r_i^{-1} follow directly in a word, they cancel (e.g.: $r_j r_i r_i^{-1} = r_j$ and $r_j r_i^{-1} r_i = r_j$) [15].

For example in the scenario in Fig. 2, τ_3 first intersects U_B along its orientation and then crosses U_A against its orientation. The assigned word is therefore $r_B r_A^{-1}$. The same word is assigned to τ_2. In contrast, τ_1 intersects U_A first and then U_B, which produces the word $r_A^{-1} r_B$. τ_2 and τ_3 are assigned the same word and therefore belong to the same driving corridor.

The selection of the surfaces U_i depends on the high-level actions which should be considered in the scenario. An obvious choice is to create surfaces perpendicular to each relevant obstacle for passing it either on the left or right. In case of moving obstacles the letter cancellation rule in Def. 4 allows one to handle back-overtaking properly. For example, if the ego vehicle drives parallel to another vehicle with similar speed and it moves on or near to the associated surface of the other vehicles, it may intersect this surface back and forth and each time add a letter to the assigned word. The letter cancellation rule removes these letters.

We follow the approach in [5], [3] to relax the assumption that the endpoints of two trajectories must necessarily match, but instead must lie only within the same goal region. This assumption is more applicable to automated driving since usually there is a goal region given rather than a single goal state.

Definition 5 (Driving corridor): Given:

1) A dynamical system $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$ of the vehicle with state $\mathbf{x}(t) \in \mathcal{X}$ and input $\mathbf{u}(t) \in \mathcal{U}(\mathbf{x}(t))$,
2) a set of (time-dependent) obstacles $\mathcal{O}(t) \subseteq \mathbb{R}^2$,
3) the covered region of the ego vehicle $\mathcal{A}(\mathbf{x}(t)) \subseteq \mathbb{R}^2$,
4) a set of initial states $\mathcal{X}_0$ at time t_0,
5) a set of surfaces U_j with corresponding letters r_j,

we define the driving corridor $\mathcal{M}$ for a given $word$ and a given goal region $\mathcal{X}_G$ as:

$$\begin{aligned}\mathcal{M}(t; word, \mathcal{X}_G) := \{\tau(t; \mathbf{u}, \mathbf{x}_0) \,|\exists \mathbf{u} \in \mathcal{U},\ \exists \mathbf{x}_0 \in \mathcal{X}_0,\\ \tau(t_f; \mathbf{u}, \mathbf{x}_0) \in \mathcal{X}_G,\\ \tau(t'; \mathbf{u}, \mathbf{x}_0)) \notin \mathcal{F}(t') \text{ for } t' \in [t_0, t_f],\\ \tau \text{ is assigned the } word\Big\}.\end{aligned}$$

The aim of this paper is to compute all driving corridors for given $words$ and/or given goal regions, respectively discover all possible $words$ and/or reachable goal regions of non-empty driving corridors.

III. System model and drivable region computation

For general vehicle models $f(\mathbf{x}(t), \mathbf{u}(t))$ and obstacles $\mathcal{O}(t)$ the reachable set cannot be computed efficiently. Therefore, we resort to a simplified vehicle model and a conservative approximation of the reachable set as presented in [13].

A. Reachable set approximation

We model the vehicle dynamics with two double integrators in longitudinal and lateral direction. The state $\mathbf{x} = \begin{pmatrix} x & \dot{x} & y & \dot{y} \end{pmatrix}^T$ is given by position and velocity in the two-dimensional plane and the input $\mathbf{u} = \begin{pmatrix} u_x & u_y \end{pmatrix}^T$ by the acceleration in both directions. We assume bounded acceleration and speed.

$$f(\mathbf{x}, \mathbf{u}) = \begin{pmatrix} \dot{x} \\ \ddot{x} \\ \dot{y} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_x \\ u_y \end{pmatrix} \tag{5}$$

$$\begin{aligned} |u_x| &\leq a_{max,x} \\ |u_y| &\leq a_{max,y} \\ v_{min,x} &\leq \dot{x} \leq v_{max,x} \\ v_{min,y} &\leq \dot{y} \leq v_{max,y} \end{aligned}$$

The reachable set for the given model is computed at discrete points in time in iterative steps. The representation of the set and the algorithm is summarized subsequently.

The reachable set at time step t_i is strictly over-approximated by the union of sets $\mathcal{B}_i$

$$\mathbf{reach}(\mathcal{X}_0, t_i, t_f) \subseteq \bigcup_q \mathcal{B}_i^{(q)}, \tag{6}$$

where each $\mathcal{B}_i$ is the Cartesian product of two convex polytopes $\mathcal{B}_i^{(q)} = \mathcal{P}_{i,x}^{(q)} \times P_{i,y}^{(q)}$. The polytopes represent a set of position/velocity pairs, which can be reached in the x- and y-direction. Polytopes are chosen because they provide an efficient computation of linear dynamical systems (5).

The algorithm to compute the over-approximated reachable set is shown in Alg. 1. First, the set of states of the previous step is propagated in time according to the system model (5) (Alg. 1, ll. 3-5). Second, the forbidden states $\mathcal{F}(t_i)$

Algorithm 1

Input: Initial set: $\mathcal{X}_0 = \cup_q \mathcal{B}_0^{(q)}$; forbidden region: $\mathcal{F}(t)$
Output: Graph $\mathcal{G}$ with nodes storing the sets $\mathcal{B}_i^{(q)}$ for $i = 1, \ldots, n$

1: **for** $i = 1$ to number of time steps **do**
2: **for all** $\mathcal{B}_{i-1}^{(q)}$ in time step $i-1$ **do**
3: $\hat{\mathcal{P}}_{i,x}^{(q)} \leftarrow$ PROPAGATE($\mathcal{P}_{i-1,x}^{(q)}$)
4: $\hat{\mathcal{P}}_{i,y}^{(q)} \leftarrow$ PROPAGATE($\mathcal{P}_{i-1,y}^{(q)}$)
5: $\hat{\mathcal{B}}_i^{(q)} \leftarrow \hat{\mathcal{P}}_{i,x}^{(q)} \times \hat{\mathcal{P}}_{i,y}^{(q)}$
6: **end for**
7: $\cup_r \mathcal{B}_i^{(r)} \leftarrow$ OVERAPPROXIMATE($\cup_q \hat{\mathcal{B}}_i^{(q)} \setminus \mathcal{F}(t_i)$)
8: **for all** r in $\cup_r \mathcal{B}_i^{(r)}$ **do**
9: $\mathcal{G}$.ADD_NODE(n($\mathcal{B}_i^{(r)}$))
10: **for all** q in $\cup_q \hat{\mathcal{B}}_i^{(q)}$ **do**
11: **if** OVERLAP($\hat{\mathcal{B}}_i^{(q)}$, $\mathcal{B}_i^{(r))}$) **then**
12: $\mathcal{G}$.ADD_EDGE(n($\mathcal{B}_{i-1}^{(q)}$),n($\mathcal{B}_i^{(r))}$))
13: **end if**
14: **end for**
15: **end for**
16: **end for**

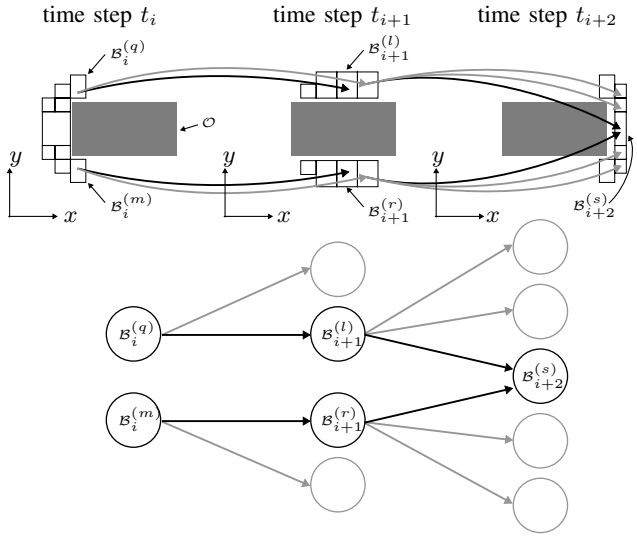


Fig. 3. Two time steps of Alg. 1. Shown are the projections of $\mathcal{B}$ into the position domain and a constant obstacle $\mathcal{O}$. $\mathcal{B}_i^{(q)}$ reaches in the time step t_{i+1} several sets. These again reach several sets in time step t_{i+2}. The relationships between the sets are represented in the directed acyclic graph $\mathcal{G}$.

are removed from this set (Alg. 1, l. 7). In order to get the same set representation (6) in the next time step, the resulting set is over-approximated [13]. The over-approximation step (Alg. 1, l. 7) is necessary since $\mathcal{F}(t)$ can be of arbitrary shape and thus $\cup_q \hat{\mathcal{B}}_i^{(q)} \setminus \mathcal{F}(t_i)$ can generally not be cast in the representation as defined in (6).

Fig. 3 illustrates the algorithm. The projection of the sets $\mathcal{B}_i$, $\mathcal{B}_{i+1}$, $\mathcal{B}_{i+2}$ into the position domain and a constant obstacle $\mathcal{O}$ are both shown for three successive time steps. A set $\mathcal{B}_i$ in step i may reach several sets $\mathcal{B}_{i+1}$ in the next time step. This relations is stored in a graph $\mathcal{G}$. A node n($\mathcal{B}$) is created in $\mathcal{G}$ for each $\mathcal{B}$ (Alg. 1, l. 9). If, like in the example in Fig. 3, $\mathcal{B}_i^{(q)}$ reaches $\mathcal{B}_{i+1}^{(l)}$, a directed edge (n($\mathcal{B}_i^{(q)}$), n($\mathcal{B}_{i+1}^{(l)}$)) between the pair of corresponding nodes

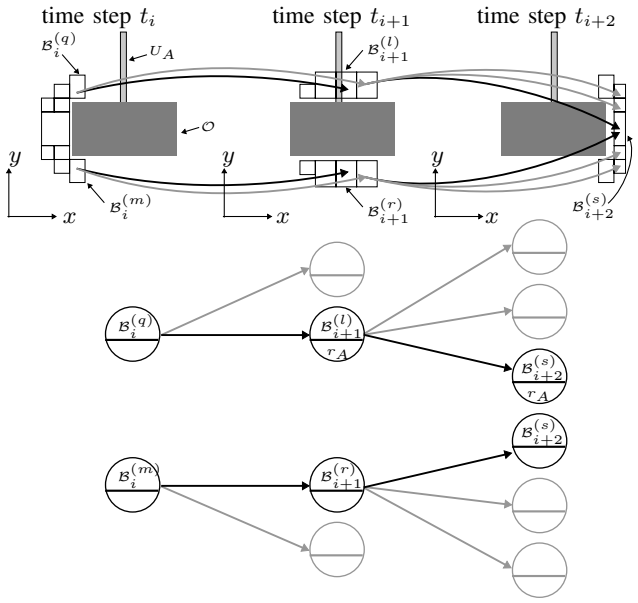


Fig. 4. Example from Fig. 3 with a surface U_A and the extended graph $\mathcal{G}_w$. $\mathcal{B}_i^{(q)}$ reaches $\mathcal{B}_{i+1}^{(l)}$ by crossing U_A, therefore the letter r_A is added in the node. $\mathcal{B}_{i+2}^{(s)}$ is assigned to two different nodes, one with the empty word and the other with the word r_A.

is added to the graph (Alg. 1, l. 12). Only nodes of one successive time step are connected.

Thus, by starting from a node n($\mathcal{B}$) with region $\mathcal{B}$ and going backward in the graph, all regions which can reach this set in the future can be identified. Accordingly, by going forward in the graph, one can identify all regions which can be reached from the past set $\mathcal{B}$.

B. Reachable set approximation with words

In Def. 4 the concept of words is introduced to characterize and group trajectories given a set of surfaces U_j. In order to apply the same concept to reachable sets, a natural extension is to define the reachable set as the set of (word, state) pairs, which can be reached at a particular time. The difficulty is that in the reachable set computation, a whole set of states moves over time (see Fig. 5). Therefore, in principle all the contained states must be tracked for intersection with all U_j. This is in contrast to a trajectory, where just a single state moves over time. Therefore, we detect the intersection of the reachable set with U_j approximatively as shown in Fig. 5: We connect the midpoints of two sets $\mathcal{B}_i^{(a)}$, $\mathcal{B}_{i+1}^{(b)}$ in the spatial-time domain with a straight line and only check this line for intersection with all U_j to assign the corresponding letters r_j.

We incorporate this word information by an extension of the previously introduced graph $\mathcal{G}$ to a graph $\mathcal{G}_w$ with word information in each node (see Fig. 4). In $\mathcal{G}$, each $\mathcal{B}$ is assigned to exactly one node n($\mathcal{B}$). In $\mathcal{G}_w$, each *pair* of $(word, \mathcal{B})$ is assigned to exactly one node n$_w(word, \mathcal{B})$. The same set $\mathcal{B}$ may be assigned to several nodes with different words. For example in Fig. 4, $\mathcal{B}_{i+2}^{(s)}$ can be reached from

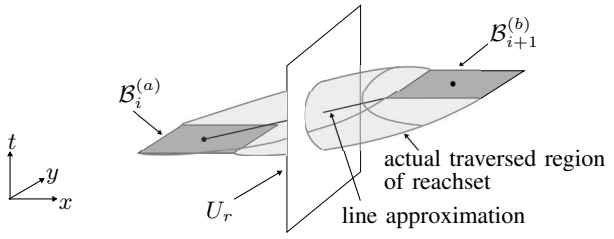


Fig. 5. Actual traversed region in continuous time and its intersection with a surface U_r; implemented in the *intersections* function in Alg. 2.

$\mathcal{B}_{i+1}^{(l)}$ and $\mathcal{B}_{i+1}^{(r)}$. However, both nodes have a different word and therefore correspond to different driving corridors.

The construction of the extended graph $\mathcal{G}_w$ from the graph $\mathcal{G}$ is shown in Alg. 2. A new initial graph $\mathcal{G}_w$ is created by copying all root nodes from $\mathcal{G}$ (Alg. 2, l. 10). Starting from a newly created node $\mathrm{n}_w(word, \mathcal{B})$ in $\mathcal{G}_w$ (Alg. 2, ll. 11-15) the algorithm searches in $\mathcal{G}$ for all sets $\mathcal{B}_{child}$, which are stored as children of $\mathcal{B}$ (Alg. 2, l. 2). These sets are added as child nodes to $\mathrm{n}_w(word, \mathcal{B})$ (Alg. 2, ll. 4-7). Additionally, a word is appended which considers the crossings of the motion from $\mathcal{B}$ to $\mathcal{B}_{child}$ with any U_i (Alg. 2, l. 3). The *intersections* function is implemented using the approximation in Fig. 5. If several surfaces are crossed, the order of intersection is considered.

Fig. 3 and Fig. 4 show $\mathcal{G}$ and the constructed $\mathcal{G}_w$ of the same example scenario. $\mathcal{B}_{i+1}^{(l)}$ is the child of $\mathcal{B}_i^{(q)}$ in both graphs. However, in $\mathcal{G}_w$ the node also contains the word r_A because of the crossing of U_A from $\mathcal{B}_i^{(q)}$ to $\mathcal{B}_{i+1}^{(l)}$. In the next time step $\mathcal{B}_{i+2}^{(s)}$ is contained in two nodes of $\mathcal{G}_w$, since it can be reached with two different words.

Algorithm 2

Input: Input graph: $\mathcal{G}$; surfaces and letters: U_i, r_i
Output: Extended graph with word information: $\mathcal{G}_w$

1: **procedure** ADD_CHILDREN($\mathrm{n}_w(word, \mathcal{B})$)
2: **for all** $\mathrm{n}(\mathcal{B}_{child})$ in CHILDREN($\mathrm{n}(\mathcal{B})$) **do**
3: $new\_word \leftarrow word$ + INTERSECTIONS($\mathcal{B}$, $\mathcal{B}_{child}$)
4: **if** $\mathrm{n}_w(new\_word, \mathcal{B}_{child})$ not in $\mathcal{G}_w$ **then**
5: $\mathcal{G}_w$.ADD_NODE($\mathrm{n}_w(new\_word, \mathcal{B}_{child})$)
6: **end if**
7: $\mathcal{G}_w$.ADD_EDGE($\mathrm{n}_w(word, \mathcal{B})$, $\mathrm{n}_w(new\_word, \mathcal{B}_{child})$)
8: **end for**
9: **end procedure**
10: $\mathcal{G}_w$.ADD_NODES($\mathrm{n}_w(empty\_word, \mathcal{B}_0^{(q)})$)
11: **for** $i = 1$ to number of time steps **do**
12: **for all** $\mathrm{n}_w(word, \mathcal{B})$ in time step i **do**
13: $\mathcal{G}_w$.ADD_CHILDREN($\mathrm{n}_w(word, \mathcal{B})$)
14: **end for**
15: **end for**

IV. Results

We demonstrate our method with two examples. In the first experiment, different driving corridors in an overtaking maneuver are identified by introducing two word-inducing surfaces. In the second experiment, different driving corridors are identified by specifying different goal regions.

A. Combinatorial aspects of motion planning

In the first experiment we apply our algorithm to the example discussed in [5]. The scenario is similar to the one shown in Fig. 2, but this time we assume that the obstacle regions covered by vehicle A and vehicle B touch each other. The ego vehicle drives with an initial speed of 20 m/s. In this example vehicle A is predicted to drive with a constant speed of 14 m/s and vehicle B with a constant speed of 17 m/s. However, we emphasize that any set-based prediction of the traffic participants can be used [16]. In particular, uncertainty in the prediction can be considered by enlarging the predicted sets appropriately.

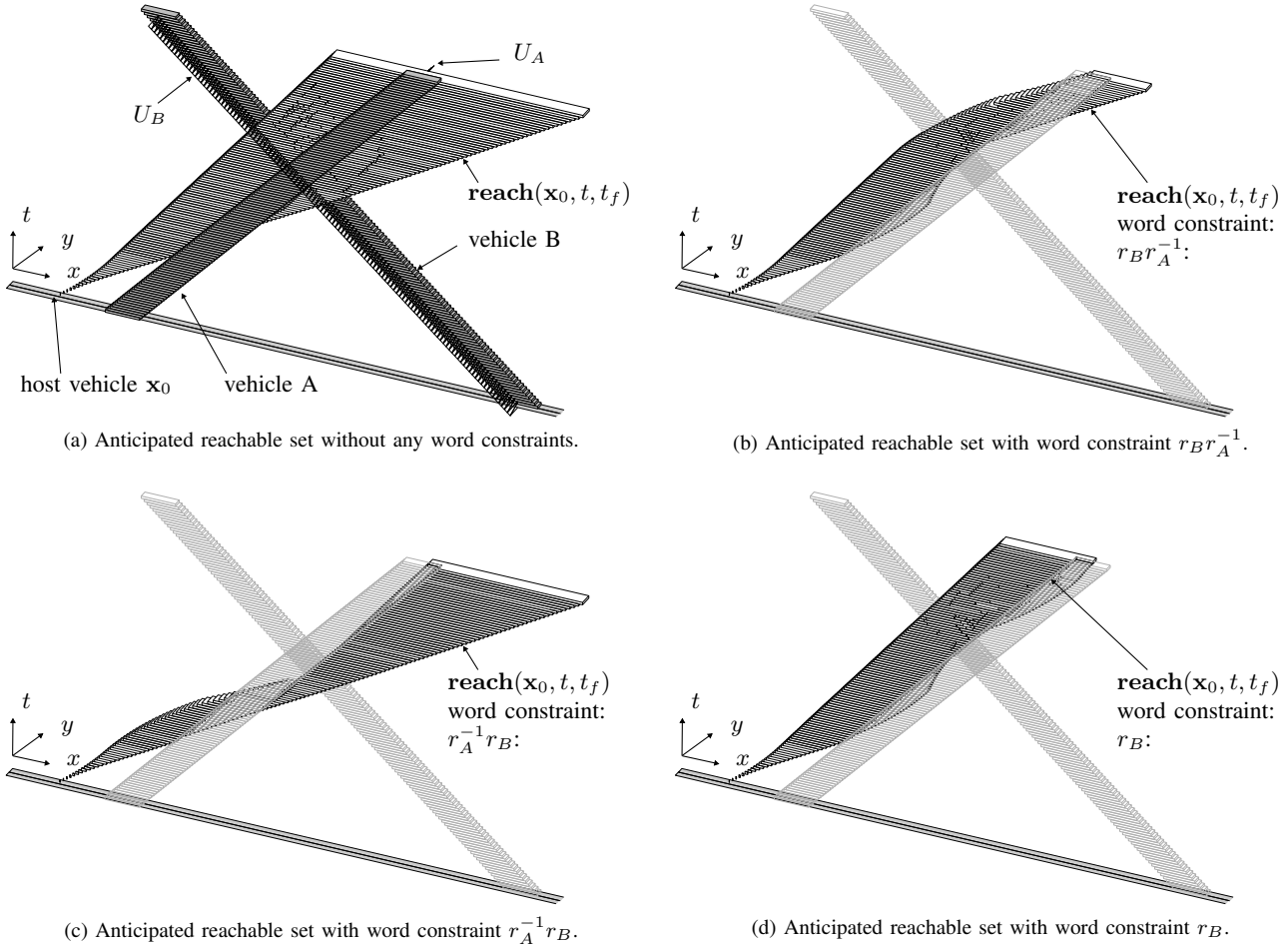
In this scenario, the high-level planner must decide to either follow vehicle A or to overtake it and determine if and how the vehicle B can be passed. The two properties "overtaking vehicle A" and "passing vehicle B" of a trajectory are specified by two surfaces U_A and U_B with two associated letters r_A and r_B. U_A is a surface connecting the region between vehicle A and the road boundary over time. U_B is a surface connecting the region between vehicle B and the boundary of the other side of the road over time. Fig. 6a shows the scenario in the spatial-time domain. In this space the prediction of vehicle A and vehicle B are "tubes" with a constant slope. The surfaces U_A and U_B are represented by two triangle meshes.

The anticipated reachable set is computed using the parameters given in Table I. Fig. 6a shows the resulting reachable set. Three different words $r_B r_A^{-1}$, $r_A^{-1} r_B$ and r_B are identified. These words correspond to the maneuvers "first pass B, then overtake A", "first overtake A, then pass B" and "only pass B". For each distinguished maneuver, all nodes in $\mathcal{G}_w$ from the final time step t_f with the corresponding word are traced back to the initial time step and the associated driving corridor is shown in Fig. 6.

Although the plots in Fig. 6 are three-dimensional, we emphasize, that the actual reachable set is five-dimensional and also contains explicit velocity information. A rough estimate of the reachable velocities can also be obtained by the slope of driving corridors in t-direction. For example, in Fig. 6c at the beginning the slope is comparatively flat which corresponds to higher speeds. The vehicle must accelerate to become fast enough to overtake A before it passes B. In contrast, in Fig. 6d the slope of the driving corridor is steep, since the ego vehicle must lower its speed to follow vehicle A.

B. Identification of gaps for lane change maneuvers

In this section we distinguish driving corridors through different goal regions. This example is motivated by trajectory planning for lane change maneuvers [6]. A lane change maneuver is a combination of a lateral motion and a longitudinal motion along the road. The high-level planner must decide whether a lane change is possible and select a sufficiently large gap between traffic on the goal lane, which is reachable by a collision free trajectory. In [6], an approach is presented to formulate the lane change trajectory as an optimization problem, which relies on a given driving



(a) Anticipated reachable set without any word constraints.

(b) Anticipated reachable set with word constraint $r_B r_A^{-1}$.

(c) Anticipated reachable set with word constraint $r_A^{-1} r_B$.

(d) Anticipated reachable set with word constraint r_B.

Fig. 6. Overtaking scenario.

TABLE I

PARAMETERS FOR DRIVABLE AREA COMPUTATION IN OVERTAKING SCENARIO

Parameter	Value
time step Δt	0.10s
number of time steps	125
absolute maximum longitudinal acceleration $a_{max,x}$	$4.0 \frac{\text{m}}{\text{s}^2}$
minimum longitudinal speed $v_{min,x}$	$12.5 \frac{\text{m}}{\text{s}}$
maximum longitudinal speed $v_{max,x}$	$25.0 \frac{\text{m}}{\text{s}}$
absolute maximum lateral acceleration $a_{max,y}$	$1.0 \frac{\text{m}}{\text{s}^2}$
minimum lateral speed $v_{min,y}$	$-2.0 \frac{\text{m}}{\text{s}}$
maximum lateral speed $v_{max,y}$	$2.0 \frac{\text{m}}{\text{s}}$

TABLE II

PARAMETERS FOR DRIVABLE AREA COMPUTATION IN LANE CHANGE SCENARIO

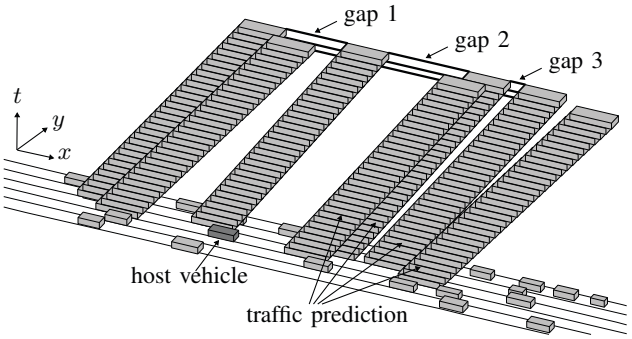
Parameter	Value
time step Δt	0.10s
number of time steps	30
absolute maximum longitudinal acceleration $a_{max,x}$	$5.0 \frac{\text{m}}{\text{s}^2}$
minimum longitudinal speed $v_{min,x}$	$0.0 \frac{\text{m}}{\text{s}}$
maximum longitudinal speed $v_{max,x}$	$45.0 \frac{\text{m}}{\text{s}}$
absolute maximum lateral acceleration $a_{max,y}$	$2.0 \frac{\text{m}}{\text{s}^2}$
minimum lateral speed $v_{min,y}$	$-2.0 \frac{\text{m}}{\text{s}}$
maximum lateral speed $v_{max,y}$	$2.0 \frac{\text{m}}{\text{s}}$

corridor. In this example, we present a reachable set analysis for the coupled lateral and longitudinal motion to compute such a driving corridor.

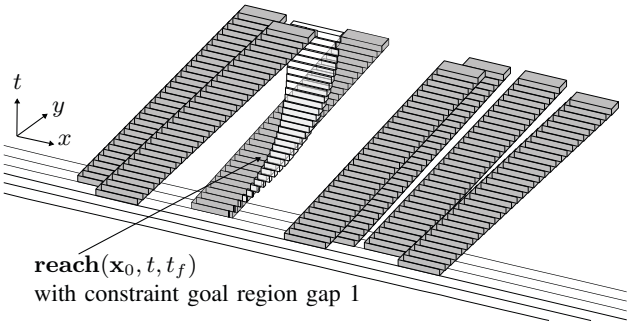
We assume that a traffic prediction is provided. Again, for the sake of the example we use a constant speed prediction with all vehicles staying in their road. However, as mentioned before, any kind of set-based prediction can be used [16]. The gaps between traffic participants in the adjacent lanes at the predicted position at the end of the planning horizon are chosen as possible goal regions. We test our approach on a scenario taken from the NGSIM US 101 Highway Dataset[1]. Fig. 7 (a) shows the initial scenario with the traffic prediction and possible goal regions. Only the prediction of vehicles in the start lane and the goal lane are shown. Double lane changes are not considered.

The anticipated reachable sets with different goal constraints are computed using the parameters given in Table II. The computation shows that only gap 1 and gap 2, but not gap 3, are reachable.

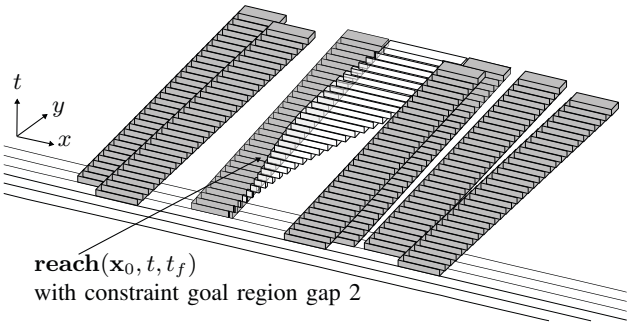
[1]http://www.fhwa.dot.gov/publications/research/operations/07030/



(a) Initial scenario with traffic prediction and possible gaps between the traffic on the goal lane. The prediction of irrelevant vehicles is not shown.



(b) Anticipated reachable set for gap 1.



(c) Anticipated reachable set for gap 2.

Fig. 7. Lane change scenario with different goal regions.

V. Conclusion

In this work, we present a method for identifying driving corridors in dynamic road scenarios. Contrary to existing methods, we do not sample trajectories but instead use a set representation of all reachable states. This has two benefits: First, it is independent of any particular sampling strategy. Second, it does not miss any feasible trajectories and makes it possible to prove that certain high-level plans are infeasible. Also, unlike combinatorial planning methods which decompose the configuration space to find high-level plans and rely on a specific obstacle representation, our method only requires a simple collision detection with obstacles.

In the future, we plan to integrate the driving corridor computation with a variational-based trajectory planner.

Acknowledgment

The authors gratefully acknowledge financial support by the German Research Foundation (DFG) AL 1185/3-1.

References

[1] B. Paden, M. p, 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, Mar. 2016.

[2] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*. Springer, 2009.

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

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

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

[6] J. Nilsson, M. Brännström, E. Coelingh, and J. Fredriksson, "Lane change maneuvers for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, no. 99, pp. 1–10, 2016.

[7] S. J. Anderson, S. C. Peters, T. E. Pilutti, and K. Iagnemma, "An optimal-control-based framework for trajectory planning, threat assessment, and semi-autonomous control of passenger vehicles in hazard avoidance scenarios," *International Journal of Vehicle Autonomous Systems*, vol. 8, no. 2, pp. 190–216, 2010.

[8] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making bertha drive —an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[9] M. Aeberhard, S. Rauch, M. Bahram, G. Tanzmeister, J. Thomas, Y. Pilat, F. Homm, W. Huber, and N. Kaempchen, "Experience, results and lessons learned from automated driving on Germany's highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 42–57, 2015.

[10] R. Kohlhaas, D. Hammann, T. Schamm, and J. M. Zöllner, "Planning of high-level maneuver sequences on semantic state spaces," in *IEEE International Conference on Intelligent Transportation Systems*, Sep. 2015, pp. 2090–2096.

[11] S. Bhattacharya, M. Likhachev, and V. Kumar, "Topological constraints in search-based robot path planning," *Autonomous Robots*, vol. 33, no. 3, p. 273, 2012.

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

[13] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, [under review].

[14] ——, "Determining the nonexistence of evasive trajectories for collision avoidance systems," in *IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 956–961.

[15] S. Bhattacharya and R. Ghrist, "Path homotopy invariants and their application to optimal trajectory planning," *IMA Conference on Mathematics of Robotics (IMAMR)*, 2015.

[16] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, 2016.