A Multi-Step Approach to Accelerate the Computation of Reachable Sets for Road Vehicles

Moritz Klischat and Matthias Althoff

***Abstract*— We propose an approach for the fast computation of reachable sets of road vehicles while considering dynamic obstacles. The obtained reachable sets contain all possible behaviors of vehicles and can be used for motion planning, verification, and criticality assessment. The proposed approach precomputes computationally expensive parts of the reachability analysis. Further, we partition the reachable set into cells and construct a directed graph storing which cells are reachable from which cells at preceding time steps. Using this approach, considering obstacles reduces to deleting nodes from the directed graph. Although this simple idea ensures an efficient computation, the discretization can introduce considerable over-approximations. Thus, the main novelty of this paper is to reduce the over-approximations by intersecting reachable sets propagated from multiple points in time. We demonstrate our approach on a large range of scenarios for automated vehicles showing a faster computation time compared to previous approaches while providing the same level of accuracy.**

I. Introduction

Reachability analysis is considered a powerful tool to ensure safety for safety-critical applications such as self-driving vehicles. Although reachability analysis is a well-researched topic with continuous improvements in terms of scalability and/or tightness [1]–[3], most approaches do not consider time-varying forbidden regions originating from static or dynamic obstacles.

However, reachable sets excluding forbidden regions are especially useful for motion planning, e.g., for restricting the search space to safe regions [4]. Also, the size of reachable sets can be used to assess the criticality of traffic scenarios for generating safety-critical test cases for motion planners [5]. Another application of reachable sets is cooperative path planning for multiple agents [6], [7] or the computation of the time-to-react (TTR), i.e., the last point in time to avoid a collision [8]. Most of the above-mentioned applications are used in real time and require a fast computation. In this paper, we propose a novel method that computes reachable sets excluding forbidden regions more efficiently compared to previous work.

A. Related Work

General approaches which are based on Hamilton–Jacobi–Bellmann (HJB) equations are proposed in [9]–[11]. However, for real-time applications this method is computationally too expensive. An early work computing reachable sets for trajectory planning of vehicles is [12]. In [13], reachable sets are computed offline for parametrized trajectories with constant inputs. During run time, parameters of the collision-free reachable sets are selected to determine the set of safe inputs for the subsequent trajectory optimization. The method in [14] approximates the reachable set of automated vehicles using HJB equations. However, the method is restricted to rectangular obstacles and simple road configurations.

A related topic where obstacles are considered in reachability analysis is the computation of inevitable collision states (ICS), which eventually lead to a collision irrespective of the chosen input [15]. Works that compute ICS in dynamic environments using reachability analysis can be found in [9], [16], [17].

To emphasize that we are ultimately interested in reachable sets avoiding forbidden region projected to the road surface, we use the term *drivable area* henceforth. In our previous work [18], we also computed the drivable area, but that approach requires recomputing similar computations, which unnecessarily consumes computational resources.

B. Contributions

We propose a novel method for the graph-based computation of the drivable area. A similar graph-based method was used in [19]; however, the reachability was only approximated and not based on system dynamics. Although a spatio-temporal decomposition of the state space for reachability analysis was used, e.g., in [20]–[22], the novelty of our work is that

- it can limit the discretization error by considering multiple preceding time steps at every iteration;
- our algorithm can handle arbitrary obstacle shapes and road networks;
- ICS can be considered less conservatively compared with our previous work [18];
- we provide extensive testing and benchmarking on a large number of traffic scenarios.

The rest of the paper is organized as follows. First, we introduce the problem statement in Sec. II, before describing the proposed method in Sec. III, which is divided into offline and online computation. Finally, we evaluate our approach in Sec. IV using multiple numerical examples and compare it to related works.

II. Definitions and Problem Statement

Let the dynamics of a model M be given by $\dot{x}(t) = f(x(t), u(t)$ with inputs $u(t) \in \mathcal{U}$ bounded by the input set $\mathcal{U} \subset \mathbb{R}^m$. A solution originating from the initial state

All authors are with the Technische Universität München, Fakultät für Informatik, Lehrstuhl für Robotik und Echtzeitsysteme, Boltzmannstraße 3, 85748, Garching, Germany. {moritz.klischat, althoff}@tum.de

$x_0 \in \mathbb{R}^n$ is

$$x(t; u(\cdot), x_0) = x_0 + \int_{t_0}^{t} f(x(\tau), u(\tau)) d\tau \,, \tag{1}$$

where $x(t; u(\cdot), x_0) \in \mathbb{R}^n$ and $u(\cdot)$ denotes an input trajectory in contrast to points in time t. Because we often require the projection of states to the two-dimensional position domain, we define the projection operator $\mathrm{proj}(x) : \mathbb{R}^n \to \mathbb{R}^2$.

From the perspective of the vehicle, the possible future occupancy of obstacles and regions outside of the road surface are a set of forbidden states $\mathcal{F}(t) \subset \mathbb{R}^2$ dependent on time t. The anticipated reachable set is defined as the set of reachable states starting from an initial state x_0 while avoiding a set of forbidden states $\mathcal{F}(t)$ during time interval $t \in [t_0, t_h]$ [18]:

$$\begin{aligned} &\operatorname*{reach}_{t}(\mathcal{X}_0, \mathcal{U}, \mathcal{F}(t)) \coloneqq \{x(t; u(\cdot), x_0) \mid x_0 \in \mathcal{X}_0, \\ &\forall \tau \in [t_0, t_h]\colon\ u(\tau) \in \mathcal{U},\ \mathrm{proj}(x(\tau; u(\cdot), x_0)) \notin \mathcal{F}(\tau)\} \,. \end{aligned} \tag{2}$$

The time-dependent reachable set is computed iteratively for time increments $\Delta t \in \mathbb{R}^+$. Hence, we denote a set at time t_k by a subscript k. At each point in time $t_k = k \cdot \Delta t$ with $k \in \mathbb{N}$, we denote the reachable set by

$$\mathcal{R}_{k+1} \coloneqq \operatorname*{reach}_{t_{k+1}}(\mathcal{R}_k, \mathcal{U}, \mathcal{F}(t)), \quad \mathcal{R}_0 = \mathcal{X}_0. \tag{3}$$

When no exclusion of forbidden sets is considered ($\mathcal{F}(t) = \emptyset$) we write $\hat{\square}$, e.g., $\hat{\mathcal{R}}_k$. Since an exact solution of the drivable area $\mathrm{proj}(\mathcal{R}_k)$ cannot be computed for the general case [23], we compute it over-approximately. Moreover, we model road vehicles by a point-mass model M_a because it abstracts any high-fidelity model M for a road vehicle whose dynamics are bounded by the friction circle $\ddot{x}_\zeta^2 + \ddot{x}_\eta^2 \leq a_\mathrm{max}^2$ [12]:

$$\begin{pmatrix} \dot{x}_\zeta(t) \\ \ddot{x}_\zeta(t) \\ \dot{x}_\eta(t) \\ \ddot{x}_\eta(t) \end{pmatrix} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_\zeta(t) \\ \dot{x}_\zeta(t) \\ x_\eta(t) \\ \dot{x}_\eta(t) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_\zeta(t) \\ u_\eta(t) \end{pmatrix} \tag{4}$$

$$u(t) \in \mathcal{U},\ \mathcal{U} = \left\{u \mid u_\zeta^2 + u_\eta^2 \leq a_\mathrm{max}^2\right\}. \tag{5}$$

This abstraction guarantees that the drivable area of the high-fidelity model $\mathrm{proj}(\mathcal{R}_k^M)$ is always a subset of the abstracted model $\mathrm{proj}(\mathcal{R}_k^{M_a})$, i.e., $\mathrm{proj}(\mathcal{R}_k^M) \subseteq \mathrm{proj}(\mathcal{R}_k^{M_a})$.

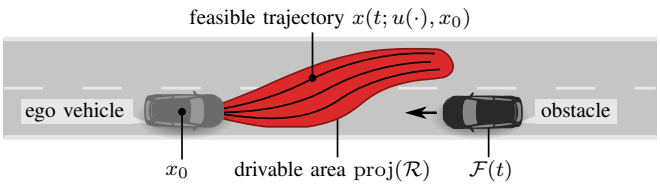


Fig. 1: The drivable area for a complete prediction horizon that represents the set of all feasible trajectories for the ego vehicle.

III. Concept of Offline and Online Computation

We divide the computation of the drivable area into two parts: offline and online. During the offline computation, the drivable area is computed without forbidden sets $\mathcal{F}(t)$, which are only known during online execution. The result is partitioned using a uniform grid of disjoint axis-aligned cells in the position domain

$$\mathcal{C}_k^{(i)} = \left[\underline{c}^{(i)}, \overline{c}^{(i)}\right] \subset \mathbb{R}^2, \quad \bigcup_{i=0}^{n_\mathrm{c}} \mathcal{C}_k^{(i)} \supseteq \mathrm{proj}(\hat{\mathcal{R}}_k), \tag{6}$$

where the index i refers to the i^th cell. We define a directed graph as a tuple $\mathcal{G} = (V, E)$ where nodes $v_{k,i} \in V$ correspond to the cells $\mathcal{C}_k^{(i)}$ of the drivable area and edges $(v_{k,i}, v_{k+1,j}) \in E$ express that a trajectory from one cell to another cell exists.

Thus, forbidden sets can be excluded during the online computation by deleting occupied nodes and their outgoing edges from the graph as illustrated in Fig. 2. Since only the position domain in $\mathbb{R}^2$ is discretized, the number of nodes only scales quadratically with the number of segments in each dimension. We compensate over-approximations from the discretization by adding edges between nodes spanning multiple time steps, as explained in Sec. III-C.

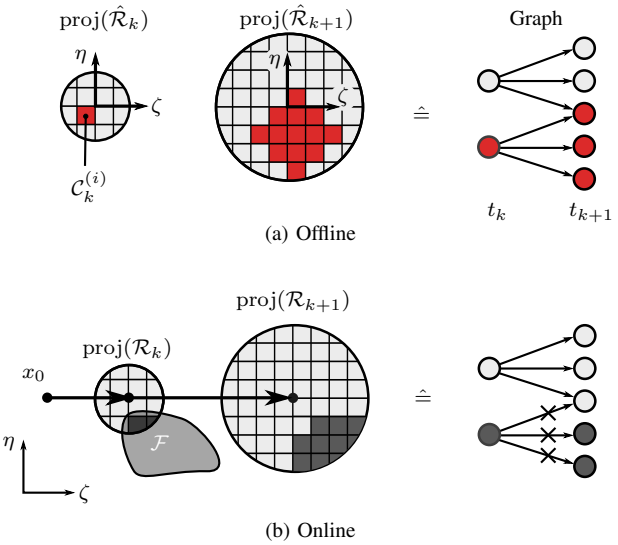


Fig. 2: (a) Offline computation of reachable cells $\mathcal{C}_{k+1}$ at t_{k+1} which is encoded in a graph. (b) Online: deleting nodes (represented in dark gray) that correspond to cells which intersect with forbidden sets $\mathcal{F}(t_k)$ or do not have a predecessor.

A. Offline Reachability Analysis

The offline computation consists of two steps:

1) *Propagation* of the reachable set $\hat{\mathcal{R}}_k$ to obtain $\hat{\mathcal{R}}_{k+1}$.
2) *Discretization* of the reachable set $\hat{\mathcal{R}}_{k+1}$ to $\hat{\mathfrak{D}}_{k+1}$ and construction of the graph.

1) Propagation: Since the vehicle model in (4) is linear, the superposition principle can be applied: The reachable set $\hat{\mathcal{R}}_k$ is obtained by adding the homogeneous solution $x(t_k; u{=}\mathbf{0}, x_0{\neq}\mathbf{0})$ resulting from the initial state x_0 and the inhomogeneous solution $x(t_k; u{\neq}\mathbf{0}, x_0{=}\mathbf{0})$. Since

$\mathcal{U}$ is a constant set, the set of inhomogeneous solutions $\mathrm{reach}_{t_k}(\mathbf{0}, \mathcal{U}, \emptyset)$ can be computed offline for an initial state located at the origin $\mathbf{0}$. For further derivation let us introduce the Minkowski sum of a singelton a and a set $\mathcal{Y}$ as $a \oplus \mathcal{Y} := \{a + y \mid y \in \mathcal{Y}\}$. During online execution, the reachable sets for an arbitrary initial state x_0 are obtained as

$$\underbrace{\underset{t_k}{\mathrm{reach}}(x_0, \mathcal{U}, \emptyset)}_{\hat{\mathcal{R}}_k} = \underbrace{\underset{t_k}{\mathrm{reach}}(x_0, \mathbf{0}, \emptyset)}_{x(t_k;0,x_0)\,:\,\text{online}} \oplus \underbrace{\underset{t_k}{\mathrm{reach}}(\mathbf{0}, \mathcal{U}, \emptyset)}_{\hat{\mathcal{R}}_k^{\mathrm{ori}}\,:\,\text{offline}}. \quad (7)$$

Since no forbidden set is excluded during the offline computation, the propagation step

$$\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}} = \underset{t_{k+1}}{\mathrm{reach}}(\hat{\mathcal{R}}_k^{\mathrm{ori}}, \mathcal{U}, \emptyset) \quad (8)$$

can be computed with standard tools for reachability analysis; a non-exhaustive list is given by Flow* [24], SpaceEx [25], C2E2 [26], JuliaReach [27], and CORA [28].

2) Discretization of $\hat{\mathcal{R}}_{k+1}$ and Construction of the Graph: For creating the graph $\mathcal{G}$, we first partition the reachable set $\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}}$ into disjoint subsets $\mathcal{B}_{k+1}^{(j)}$ using the cells in the position domain (6) to compute the Cartesian product

$$\forall i \in \mathcal{I}\colon \mathcal{B}_k^{(i)} = \mathcal{C}^{(i)} \times [\underline{\dot{x}}_{k,\zeta}^{(i)}, \overline{\dot{x}}_{k,\zeta}^{(i)}] \times [\underline{\dot{x}}_{k,\eta}^{(i)}, \overline{\dot{x}}_{k,\eta}^{(i)}],$$
$$\mathcal{I} = \left\{ i \mid \hat{\mathcal{R}}_{k+1}^{\mathrm{ori}} \cap \mathcal{C}^{(i)} \neq \emptyset \right\}. \quad (9)$$

The intervals $[\underline{\dot{x}}_{k,\zeta}^{(i)}, \overline{\dot{x}}_{k,\zeta}^{(i)}]$ and $[\underline{\dot{x}}_{k,\eta}^{(i)}, \overline{\dot{x}}_{k,\eta}^{(i)}]$ bound the velocities of the reachable set $\hat{\mathcal{R}}_{k+1}^{\mathrm{ori}}$ for the i^{th} cell. To obtain a discretized representation of a reachable set, we introduce the discretization operator:

$$\hat{\mathfrak{D}}_k^{\mathrm{ori}} = \mathrm{discr}(\hat{\mathcal{R}}_k^{ori}) = \bigcup_{i=0}^{n_b} \mathcal{B}_k^{(i)} \supseteq \hat{\mathcal{R}}_k, \quad (10)$$

which analogously yields $\mathfrak{D}_k = \mathrm{discr}(\mathcal{R}_k)$. An edge $(v_{k,i}, v_{k+1,j})$ is added to $\mathcal{G}$ if

$$\underset{t_{k+1}}{\mathrm{reach}}(\mathcal{B}_k^{(i)}, \mathcal{U}, \emptyset) \cap \mathcal{B}_{k+1}^{(j)} \neq \emptyset. \quad (11)$$

For efficient implementation, we represent the edges by adjacency matrices $P_k^{k+1} \in \mathbb{R}^{q_k \times q_{k+1}}$ with q_k being the number of cells at the respective time. Each element p_{ji} of P_k^{k+1} is a Boolean value

$$p_{ji} = \begin{cases} 1 & \text{if } \mathrm{reach}_{t_{k+1}}(\mathcal{B}_k^{(i)}, \mathcal{U}, \emptyset) \cap \mathcal{B}_{k+1}^{(j)} \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (12)$$

The adjacency matrices P_k^{k+1} are the main result of the offline computation and can be stored compactly as sparse matrices.

B. Online Computations

The objective of the online computation is to exclude forbidden sets $\mathcal{F}_k$ from the drivable area. Therefore, at every iteration step the drivable area is propagated and cells intersecting with $\mathcal{F}_k$ are excluded.

Algorithm 1 Offline Reachability Analysis

Require: input set $\mathcal{U}$

1: **for** $k = 0$ to n **do**
2: $\quad \hat{\mathcal{R}}_{k+1}^0 \leftarrow \mathrm{reach}_{t_{k+1}}(\hat{\mathcal{R}}_k^0, \mathcal{U})$
3: $\quad \hat{\mathfrak{D}}_{k+1} \leftarrow \mathrm{discr}(\hat{\mathcal{R}}_{k+1}^0)$ ▷ see (10)
4: $\quad \{v_i, v_j\} \leftarrow$ ReachableCells($\hat{\mathfrak{D}}_k, \hat{\mathfrak{D}}_{k+1}$) ▷ see (11)
5: $\quad P_k^{k+1} \leftarrow$ ConstructGraph($\{(\mathcal{B}_k^{(i)}, \mathcal{B}_{k+1}^{(j)})\}$) ▷ see (12)
6: **end for**
7: **return** $P_k^{k+1}, \mathfrak{D}_k$

1) Propagation: To propagate the drivable area using the adjacency matrices P_k^{k+1}, we introduce the Boolean vector $r_k \in \{0,1\}^q$ that denotes for each cell whether it is part of the drivable area:

$$r_k^{(i)} = \begin{cases} 1 & \text{if } \left(x(t_k;0,x_0) \oplus \mathcal{C}_k^{(i)}\right) \cap \mathrm{proj}(\mathfrak{D}_k) \neq \emptyset \\ 0 & \text{otherwise.} \end{cases} \quad (13)$$

Using (12), we write the graph-based propagation as

$$\hat{r}_{k+1} = P_k^{k+1} r_k\,. \quad (14)$$

2) Discretization of Obstacles: To exclude $\mathcal{F}_{k+1}$ from the Boolean representation $\hat{r}_{k+1}$, we discretize $\mathcal{F}_{k+1}$. As commonly done in motion planning, we consider the shape of the ego vehicle by dilating the occupied space of obstacles with a disk of radius ρ that under-approximates the shape of the ego vehicle [29]. The under-approximation is required to consistently over-approximate the drivable area when excluding $\mathcal{F}_{k+1}$. The resulting occupied space $\mathcal{F}_{k+1}$ is represented analogously to r_k by the occupancy vector o_{k+1} with elements

$$o_{k+1}^{(i)} = \begin{cases} 1 & \text{if } x(t_k;0,x_0) \oplus \mathcal{C}_{k+1}^{(i)} \subseteq \mathcal{F}_{k+1} \\ 0 & \text{otherwise} \end{cases}. \quad (15)$$

$\mathcal{F}_{k+1}$ is typically not connected and thus, occupancies from multiple obstacles need to be discretized individually. For efficiency, the discretization of an obstacle is only conducted if it intersects with the bounding box of the drivable area. From (13) and (15) follows that the exclusion of the occupied states is equivalent to

$$r_{k+1} = \hat{r}_{k+1} \wedge \neg o_{k+1}$$

with logical operators $\neg$ and $\wedge$ being performed elementwise. Thus, we can write the propagation of the reachable set and exclusion of forbidden sets as

$$\mathfrak{D}_{k+1} = \mathrm{discr}\left(\underset{t_{k+1}}{\mathrm{reach}}(\mathfrak{D}_k, \mathcal{U}, \emptyset)\right) \setminus \mathrm{discr}(\mathcal{F}(t_{k+1})),$$
$$\mathfrak{D}_0 = x_0, \quad (16)$$

which using (14) simplifies to

$$r_{k+1} = \left(P_k^{k+1} r_{t_k}\right) \wedge \neg o_{k+1}\,. \quad (17)$$

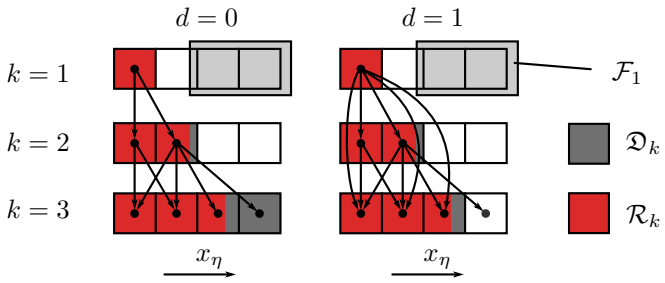


Fig. 3: One-dimensional example for the intersection of multiple propagated sets to reduce the discretization error of $\mathfrak{D}_3$. For $d = 0$, only cells at subsequent time steps are connected in the graph; for $d = 1$, edges from $k = 1$ to $k = 3$ are also considered and only those cells with an edge to $k = 1$ are reachable at $k = 3$. The undiscretized drivable area $\mathcal{R}$ is shown for comparison.

C. Compensating Discretization Errors with Multiple Propagations

When propagating the drivable area $\mathrm{proj}(\mathfrak{D}_k)$, the discretization accumulates as shown in Fig. 3. After multiple time steps, these errors quickly lead to an undesirably over-approximated drivable area. Furthermore, we over-approximate the velocities so that we only have to discretize the position domain.

To counteract the discretization errors, we add edges between cells of the drivable area that span multiple time steps. Using these edges, drivable areas from multiple preceding points in time are propagated up to time t_{k+1} and the resulting sets are intersected. Since the edges spanning multiple time steps are also computed offline using the same principle as in Sec. III-A, the discretization error is only added once, instead of aggregating the discretization errors from every intermediate time step as illustrated in Fig. 3.

Let us first formalize the multi-step approach using the set-based representation. We define the refined reachable set resulting from propagations from d points in time t_l, $l \in \{k-d, ..., k\}$ to time step t_{k+1} as

$$\overline{\mathfrak{D}}^d_{k+1} := \Big(\bigcap_{l=k-d}^{k} \mathrm{discr}\left(\underset{t_{k+1}}{\mathrm{reach}}(\overline{\mathfrak{D}}_l, \mathcal{U}, \emptyset) \right) \Big) \setminus \mathrm{discr}(\mathcal{F}(t_{k+1})),$$
$$\overline{\mathfrak{D}}_0 = x_0. \tag{18}$$

Since $\overline{\mathfrak{D}}^0_{k+1}$ is always among the intersected sets in (18), $\overline{\mathfrak{D}}^d_{k+1} \subseteq \mathfrak{D}_{k+1}$ is true for any $d > 0$ showing that multiple propagations obviously provide tighter results. The propagation steps $\mathrm{discr}(\mathrm{reach}_{t_{k+1}}(\overline{\mathfrak{D}}^0_l, \mathcal{U}, \emptyset)))$ can also be computed offline and represented by propagation matrices P^{k+1}_l using the same approach as in Sec. III-A. Thus, we can write the multi-step online propagation from (18) in matrix notation as

$$\overline{r}^d_{k+1} = \Big(\bigwedge_{l=k-d}^{k} P^{k+1}_l \overline{r}_l \Big) \wedge \neg o_{k+1}\,. \tag{19}$$

Since (19) can be implemented efficiently, the runtime is mainly dominated by the discretization of obstacles; even multiple propagations do not impact run time considerably, as shown in Sec. IV.

D. Exclusion of Inevitable Collision States

By utilizing the graph, we can efficiently exclude ICS from the previously computed drivable area. Even though the principle was formulated before in [18], it becomes especially effective when combined with our approach that uses a fine discretization of the whole drivable area. When no path in the graph $\mathcal{G}$ from a cell $\mathcal{C}^{(i)}_k$ to a cell at the final time step exists, all states in $\mathcal{C}^{(i)}_k$ eventually lead to a collision. Thus, we exclude these cells from the drivable area by iterating backward from the final set $\overline{r}^d_h$ and deleting nodes with no reachable set at a preceding time step. Using the propagation matrices and the multi-step propagation as in (19), this is computed at each time step as

$$\underline{r}^d_{k-1} = \Big(\bigwedge_{l=k}^{k+d} {P^l_{k-1}}^T \underline{r}_l \Big) \wedge \neg o_{k-1}, \quad \underline{r}_h = \overline{r}^d_h, \tag{20}$$

where the transpose of the propagation matrix follows from its definition in (12). The complete forward-backward algorithm of the online reachability analysis is summarized in Algorithm 2.

Algorithm 2 Online Reachability Analysis

Require: graph represented by propagation matrices P^{k+1}_k, forbidden set $\mathcal{F}(t)$, number of time steps h, and number d of considered time steps for propagation

1: **for** $k = 0$ to h **do**
2: $x(t_k;, 0, x_0) \leftarrow$ HOMOGENEOUSSOLUTION(x_0)
3: $o_{k+1} \leftarrow$ OCCUPANCYGRID$(\mathcal{F}_{k+1}, \mathcal{C}_{k+1}, x(t_k;, 0, x_0))$ ▷ see (15)
4: $\overline{r}^d_{k+1} \leftarrow$ PROPAGATE$(P^{k+1}_k, \{r_{k-d}, ..., r_k\}, o_{k+1})$ ▷ see (19)
5: **end for**
6: $\underline{r}_h \leftarrow \overline{r}^d_h$
7: **for** $k = h$ to 1 **do**
8: $\underline{r}^d_{k-1} \leftarrow$ EXCULDEICS$(P^k_{k-1}, \underline{r}^d_k)$ ▷ see (20)
9: **end for**
10: **return** $\{\underline{r}^d_k \mid k \in \{0, \ldots, h\}\}$

IV. Evaluation

We evaluate our approach using recorded traffic from the CommonRoad benchmark suite[1] [30] and use [31] to create obstacles representing road boundaries to detect leaving the road. Further, we compare our results with those of [18] using identical parameters for all scenarios as listed in Table I. Both methods are implemented in Python using C++ for computationally expensive operations. The computation times were measured on a laptop with an Intel i7-8650U 1.90 GHz processor and 16 GB of RAM.

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

TABLE I: Parameters used in the evaluation.

Parameter	Value
maximal acceleration $a_{\max}$	$5.0\ \mathrm{m/s^2}$
cell size dx_ζ, dx_η	$0.5\ \mathrm{m}$
number of time steps d	$\{0, 1, 7\}$
time step Δt	$0.1\mathrm{s}$
radius ρ	$1.25\mathrm{m}$

A. Computation Time

To compare the online computation time with the polytope-based approach in [18], we compute the drivable area for 339 scenarios from the CommonRoad benchmark suite. In this comparison, $d = 7$ time steps are propagated simultaneously. Fig. 4 shows the median and the ranges of the computation times over the number of computed time steps. The proposed algorithm requires $0.037\,\mathrm{s}$ for 34 time steps as the median computation time, compared with $0.170\,\mathrm{s}$ with the polytope-based approach. In particular, for longer time horizons and larger drivable areas, our method outperforms the polytope-based approach. The maximum computation time is also lower with $0.25\,\mathrm{s}$ compared with $0.34\,\mathrm{s}$.

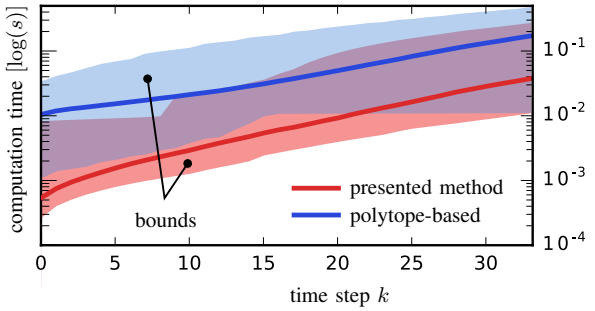
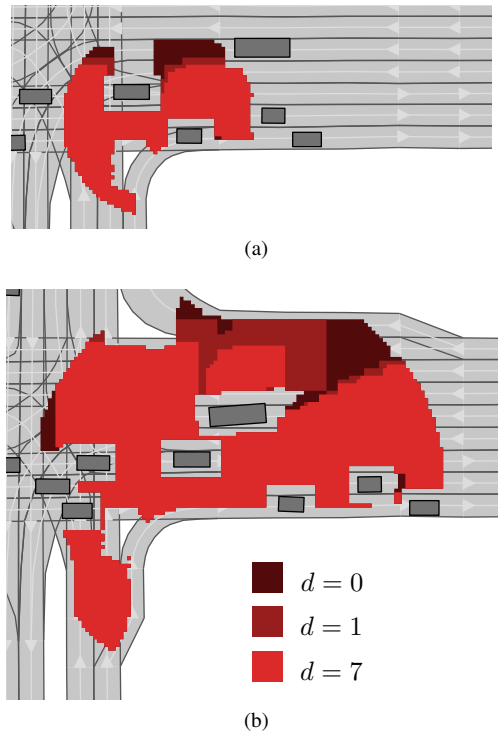


Fig. 4: Median computation times and min/max computation times depending on time steps k compared to polytope-based computation [18].

B. Scenario A

To illustrate the accuracy of our approach, we present results for two out of the considered 339 scenarios. The first scenario is an intersection from the NGSIM Lankershim Dataset[2] which can be found under ID USA_Lanker-1_1_T-1 in the CommonRoad benchmark suite. In Fig. 5, the evolution of the drivable area is depicted at different points in time and for a different number of additionally considered time steps d (see (19)). Thus, it shows that incorporating multiple time steps during the propagation has a noticeable effect even for $d = 1$, which is especially evident in the upper right region that is cut off by other vehicles. Comparing the results in Fig. 6 with those obtained by the algorithm from [18], it suggests that our computed area is slightly less over-approximative, which results from the box constraint for the input set $\mathcal{U}$ over-approximating the friction circle in [18]; in contrast, we directly use the friction circle in (5). Nevertheless, there are regions where our approach is more over-approximative, which indicates that our method over-approximates the velocity in these regions more.



(a)

(b)

Fig. 5: Scenario A: Effect of multiple propagations for different numbers of involved preceding time steps d at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

C. Scenario B

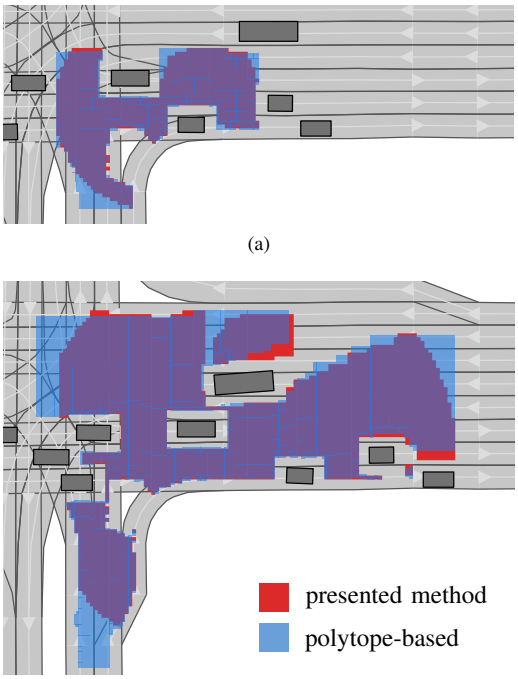
The second scenario is a highway scenario from the US 101 database[3], which can be found under ID USA_US101-27_1_T-1 in CommonRoad. The ego vehicle has an initial velocity of $13.88\,\mathrm{m/s}$. As in Scenario A, the effect of d is shown in Fig. 7 and the comparison to [18] is shown in Fig. 8. At $t = 2.4\,\mathrm{s}$ a smaller over-approximation can be observed due to the constraint of the friction circle that is only considered by our method. In contrast, the drivable area from our approach is larger at $t = 3.4\,\mathrm{s}$ in the upper right region since we tend to over-approximate the velocity.

D. Discussion

Online computation can be divided into discretization (15) and propagation (17). The discretization is the dominating part, which contributes 56 % of the overall run-time on average. In the worst case, when every obstacle needs to be discretized once every time step due to an intersection with the bounding box of the drivable area (see Sec. III-B.2), the complexity for the discretization is linear with respect to the number of obstacles n_o. In contrast, the computation time for the propagation in (19) does not depend on the number of obstacles, but only linearly on the number of nodes, the number of edges for every node and the number of considered time steps d.

[2]https://www.fhwa.dot.gov/publications/research/operations/07029/

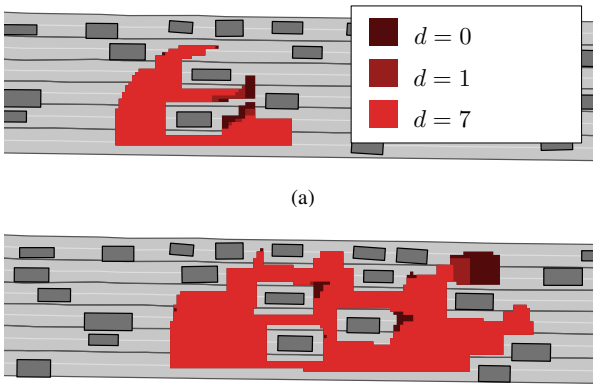
[3]https://www.fhwa.dot.gov/publications/research/operations/07030/



(a)

(b)

Fig. 6: Scenario A: Comparison of the approach with $d = 7$ to results from [18] at (a) $t = 2.4$ s and (b) $t = 3.4$ s.



(a)

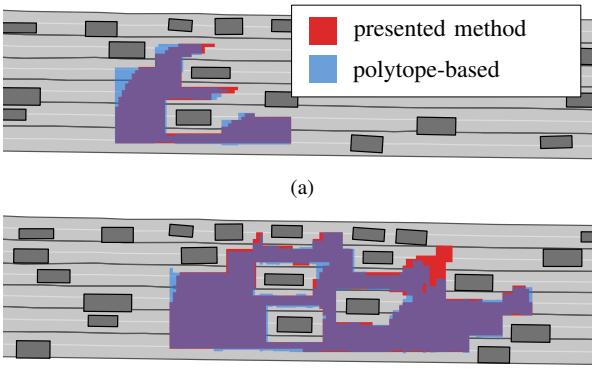
(b)

Fig. 7: Scenario B: Effect of multiple propagations for different numbers of involved preceding time steps d at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

The limiting factor for our approach is the required memory for storing the offline computed matrices P_k^{k+1} in Sec. III-A. However, for 34 time steps the resulting file still has a reasonable size of 173 MB and takes 134 minutes to compute using the tool CORA [28].

V. Conclusions

We present a method for the fast computation of drivable areas considering dynamic obstacles. In hundreds of scenarios, we show that our method results in a faster computation time compared with our previous approach while providing a comparable accuracy due to our multi-step approach. Compared to related work [12]–[14], our method can compute the drivable area more efficiently and is also able to handle more complex traffic situations and road layouts. The resulting graph from our method can be used subsequently by a trajectory planner for extracting a driving corridor efficiently or finding an initial solution with graph-based methods.



(a)

(b)

Fig. 8: Scenario B: Comparison of the approach with $d = 7$ to results from [18] at (a) $t = 2.4$ s and (b) $t = 3.4$ s.

Acknowledgments

The authors gratefully acknowledge financial support by the Central Innovation Programme of the German Federal Government under grant ZF4086007BZ8.

References

[1] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, 2005, pp. 291–305.

[2] Z. Han and B. H. Krogh, "Reachability analysis of nonlinear systems using trajectory piecewise linearized models," in *Proc. of the American Control Conference*, 2006, pp. 1505–1510.

[3] M. Althoff, O. Stursberg, and M. Buss, "Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization," in *Proc. of the 47th IEEE Conference on Decision and Control*, 2008, pp. 4042–4048.

[4] S. Dixit, U. Montanaro, S. Fallah, M. Dianati, D. Oxtoby, T. Mizutani, and A. Mouzakitis, "Trajectory planning for autonomous high-speed overtaking using MPC with terminal set constraints," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018, pp. 1061–1068.

[5] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2019, pp. 2352–2358.

[6] M. Chen, J. F. Fisac, S. Sastry, and C. J. Tomlin, "Safe sequential path planning of multi-vehicle systems via double-obstacle Hamilton-Jacobi-Isaacs variational inequality," in *Proc. of the European Control Conference*, 2015, pp. 3304–3309.

[7] S. Manzinger and M. Althoff, "Tactical decision making for cooperative vehicles using reachable sets," in *Proc. of the IEEE Int. Conf. on Intelligent Transportation Systems*, 2018.

[8] S. Söntges, 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.

[9] J. F. Fisac, M. Chen, C. J. Tomlin, and S. S. Sastry, "Reach-avoid problems with time-varying dynamics, targets and constraints," in *Proc. of the Int. Conf. on Hybrid Systems: Computation and Control*, 2015, pp. 11–20.

[10] K. Margellos and J. Lygeros, "Hamilton-Jacobi formulation for reach-avoid problems with an application to air traffic management," in *Proc. of the American Control Conference*, 2010, pp. 3045–3050.

[11] O. Bokanowski, N. Forcadel, and H. Zidani, "Reachability and minimal times for state constrained nonlinear problems without any controllability assumption," *SIAM Journal on Control and Optimization*, vol. 48, no. 7, pp. 4292–4316, Jan. 2010.

[12] C. Schmidt, F. Oechsle, and W. Branz, "Research on trajectory planning in emergency situations with multiple objects," in *Proc. of the IEEE Intelligent Transportation Systems Conference*, 2006, pp. 988–992.

[13] S. Kousik, S. Vaskov, M. Johnson-Roberson, and R. Vasudevan, "Safe trajectory synthesis for autonomous driving in unforeseen environments," in *Proc. of the ASME Dynamic Systems and Control Conf.*, Art. No. V001T44A005, 2017.

[14] I. Xausa, R. Baier, O. Bokanowski, and M. Gerdts, "Computation of avoidance regions for driver assistance systems by using a Hamilton-Jacobi approach," Art. No. OCA.2565, 2020.

[15] A. Lawitzky, A. Nicklas, D. Wollherr, and M. Buss, "Determining states of inevitable collision using reachability analysis," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 4142–4147.

[16] J. Nilsson, J. Fredriksson, and A. C. E. Ödblom, "Verification of collision avoidance systems using reachability analysis," *Proc. of the IFAC World Congress*, pp. 10 676–10 681, 2014.

[17] P. Falcone, M. Ali, and J. Sjoberg, "Predictive threat assessment via reachability analysis and set invariance theory," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1352–1361, 2011.

[18] S. Söntges and M. Althoff, "Computing the drivable area of autonomous road vehicles in dynamic road scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1855–1866, 2018.

[19] M. Ono, G. Droge, H. Grip, O. Toupet, C. Scrapper, and A. Rahmani, "Road-following formation control of autonomous ground vehicles," in *Proc. of the IEEE Conference on Decision and Control*, 2015, pp. 4714–4721.

[20] J. Lunze, "A timed discrete-event abstraction of continuous-variable systems," *International Journal of Control*, vol. 72, no. 13, pp. 1147–1164, 1999.

[21] M. Zamani, G. Pola, M. Mazo, and P. Tabuada, "Symbolic models for nonlinear control systems without stability assumptions," *IEEE Transactions on Automatic Control*, vol. 57, no. 7, pp. 1804–1809, 2012.

[22] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech." in *Hybrid Systems: Computation and Control*, 2005, pp. 258–273.

[23] A. Platzer and E. M. E. Clarke, "Formal verification of curved flight collision avoidance maneuvers: A case study," in *Proc. of the 16th International Symposium on Formal Methods*, 2009, pp. 547–562.

[24] X. Chen, S. Sankaranarayanan, and E. Abraham, "Flow* 1.2: More effective to play with hybrid systems," in *ARCH14-15. 1st and 2nd International Workshop on Applied veRification for Continuous and Hybrid Systems*, vol. 34, 2015, pp. 152–159.

[25] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proc. of the 23rd International Conference on Computer Aided Verification*, 2011, pp. 379–395.

[26] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, "C2E2: A verification tool for stateflow models," in *Tools and Algorithms for the Construction and Analysis of Systems*, 2015, pp. 68–82.

[27] S. Bogomolov, M. Forets, G. Frehse, F. Viry, A. Podelski, and C. Schilling, "Reach set approximation through decomposition with low-dimensional sets and high-dimensional matrices," in *Proc. of the 21st International Conference on Hybrid Systems: Computation and Control*, 2018, pp. 41–50.

[28] M. Althoff, "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015, p. 120151.

[29] J.-C. Latombe, *Robot Motion Planning*. Norwell: Kluwer Academic Publishers, 1991.

[30] 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.

[31] A. Zhu, S. Manzinger, and M. Althoff, "Evaluating Location Compliance Approaches for Automated Road Vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2018, pp. 642–649.