Set-Based Prediction of Traffic Participants on Arbitrary Road Networks

Matthias Althoff and Silvia Magdici

***Abstract*—Safety is of paramount importance in automated driving. One of the main challenges ensuring safety is the unknown future behavior of surrounding traffic participants. Previous works ignore this uncertainty or often address it by computing probability distributions of other traffic participants over time. Probabilistic approaches make it possible to predict the collision probability with other traffic participants, but cannot formally guarantee (i.e. cannot mathematically prove for given assumptions) whether a planned maneuver is collision-free. Our approach addresses exactly this problem: instead of computing probability distributions, we compute an over-approximation of all possible occupancies of surrounding traffic participants over time. This makes it possible to prove whether an automated vehicle can possibly collide with other traffic participants. The presented algorithm for occupancy prediction works on arbitrary road networks and produces results within a fraction of the prediction horizon. Experiments based on real-world data validate our approach and show that we could not find a behavior of a traffic participant that is not enclosed in our prediction.**

***Index Terms*—Occupancy prediction, reachability analysis, formal verification, autonomous vehicles, arbitrary road networks.**

I. Introduction

Autonomous vehicles are expected to provide a broad range of benefits compared to human-driven vehicles. Among them are increased road safety, mobility for disabled people, and better traffic throughput. Today, more than 80% of vehicle crashes are caused by human error[1], which could be prevented by fully or partially automated vehicles. In order to ensure that future autonomous vehicles are safe, it is desirable to develop provably correct decision algorithms using formal methods [1]–[3]. A further important aspect in autonomous driving is the ability to cope with uncertainties in several respects: measurements of the ego vehicle (i.e. the host vehicle performing the computations), measurements of the environment (road borders, static obstacles, other traffic participants), and the future behavior of other traffic participants. We present an approach that enables us to compute formally correct maneuvers despite the aforementioned uncertainties by computing the set of future occupancies of other traffic participants. If none of these computed occupancies intersects with the occupancy of the ego vehicle for all points in time, one can guarantee that the ego vehicle does not cause a collision [3].

In this paper, we solely focus on the occupancy prediction and exclude the following aspects: $1)$ maneuver planning [4], which is required to avoid positions possibly occupied by other traffic participants in the future; $2)$ intention predictions for particular actions, such as a lane change or a turn, as performed in e.g. [5], [6]; $3)$ path predictions of the ego vehicle [7]–[10]; and $4)$ predictions of traffic participants for fixed road sections, such as a particular intersection, which are typically obtained using machine learning techniques [11]–[13]. Recently, an overview article on behavior prediction has been published [14]. The literature in that work is structured into physics-based, maneuver-based, and interaction-aware prediction. This is a somewhat orthogonal categorization compared to ours, which is grouped into four main categories: approaches computing $a)$ a single future behavior, $b)$ a countable set of future behaviors, $c)$ probability distributions of future behaviors, and $d)$ uncountable sets of future behaviors.

a) Single future behavior: Previous work that predicts only a single behavior of other traffic participants can be found in [15]–[19]. In [15], the future position of surrounding vehicles is estimated by assuming constant acceleration and yaw rate, which are tracked by an extended Kalman filter. In [16], only a single behavior of each surrounding traffic participant is predicted to compute possible evasive trajectories of the ego vehicle. Single trajectories of other road users are considered in [17] in order to determine whether a collision should be avoided by steering and/or braking. Also, the threat assessment of traffic situations is performed in [18] by considering multiple possible maneuvers of the ego vehicle, but only a single future trajectory of surrounding vehicles. Since the maneuver planner in [19] targets comfortable driving under consideration of social behavior, only a single future behavior of surrounding traffic participants is considered.

In driving assistant systems, where the formal correctness of warnings is not highly prioritized and a computationally cheap assessment is desired, a number of heuristic methods based on a single behavior have been developed, which are often referred to as *surrogate measures*. The most known measure is time to collision (TTC) [20], but many extensions, such as time-exposed TTC and time-integrated TTC [21], as well as combinations of several surrogate measures [22], also exist.

b) Countable set of future behaviors: To mitigate the fact that infinitely many possible future trajectories exist, many works consider a finite number of future behaviors rather than a single future behavior. These methods are widely used for collision assessment. Multiple physically possible trajectories of the ego vehicle and one other vehicle are generated offline in [23] to compute the time to trigger an emergency brake. Multiple simulations are often weighted by probabilities, which is also known as Monte Carlo simulation.

Silvia Magdici and Matthias Althoff are with the Department of Computer Science, Technische Universität München, Boltzmannstraße 3, 85748 Garching, Germany e-mail: {silvia.magdici, matthias.althoff}@tum.de.

Manuscript received month day, year; revised month day, year.

[1]http://www.nsc.org/learn/pages/nsc-on-the-road.aspx

This technique is often used for online threat assessment of vehicles [24]–[26] or to create a threat database [27]. Motion clusters in combination with a particle filter are used in [28] to determine the most likely future motions. However, since there exist infinitely many traffic scenarios, techniques based on a finite number of predictions cannot guarantee safety.

c) Probability distribution of future behaviors: The fact that infinitely many possible future behaviors exist is addressed by a group of works predicting the probability distribution of possible future behaviors. The behavior of other traffic participants is predicted using Dynamic Bayesian Networks in [29]. Another probabilistic approach has been suggested in [30], where Gaussian distributions are used to represent the future occupancy with a special focus on an efficient computation of the collision probability. The aforementioned approaches cannot formally guarantee safety since a collision probability of zero does not imply that no collision is possible. This, however, is guaranteed in [31] by formally abstracting traffic participants to Markov chains using reachability analysis. A comparison of this approach with Monte Carlo simulation can be found in [32].

d) Uncountable set of future behaviors: In the architecture proposed in [33], the importance of set-based prediction of other vehicles is highlighted, but no algorithm to formally compute the occupancy prediction is proposed. In our previous work [3], we utilize techniques from reachability analysis to rigorously compute an over-approximation of the occupancy of surrounding vehicles. The novelty compared to the aforementioned papers is later presented. Due to the over-approximative computation, all possible behaviors are considered so that the approach is predestined for certification, see e.g. [34]. Set-based prediction of traffic participants is challenging since their dynamics are typically nonlinear and subject to constraints, e.g. one has to exclude behaviors, such as leaving the road boundary. Since those restrictions cannot be formulated as bounds of inputs that do not depend on the current state of the system, traditional techniques for reachability analysis cannot be applied. We group previous work by the applied set representation: polytopes [35], zonotopes [36], zonotope bundles [37], rectangular grids [38], ellipsoids [39], support functions [40], oriented rectangular hulls [41], and axis-aligned boxes [42]. Standardized set representations can only be used when unrealistic behavior does not have to be excluded; otherwise, the obtained reachable sets can even become non-convex or disjoint [43], [44].

A combination of set-based prediction and stochastic prediction is presented in [45]. However, the set-based computations in [45] are heuristic and thus not applicable to a formal analysis. Reachability analysis is also applied to driving assistant systems [2], [46] rather than using it for occupancy prediction as done in this work. Reachable sets have also been applied to mobile robots, typically assuming simple models that are overly simplified for road traffic applications: [47] assumes intervals on possible velocities in all directions; more advanced models assume intervals on the velocity and acceleration [48]. More complicated models are based on Dubin's car [49] or a tricycle model [50].

The review of the existing work shows that a formal approach for computing the occupancy prediction for other traffic participants is needed in order to guarantee safe maneuvers. This paper is an extension to the authors' previous work [3], [51], where it was shown that the occupancy prediction of other traffic participants can be computed in a formal manner. This work significantly extends previous work:

- For the first time, we present a new framework that rigorously predicts the occupancy region of surrounding traffic participants on an arbitrary road network. Our previous work has only considered single lanes without forks or joints and did not consider multi-lane roads.
- A further extension to our previous work is that we consider arbitrary lane changes of other vehicles.
- In our previous work, we assumed that a shortest path through a road network exists, without providing any algorithm (we consider a shortest path as defined in Problem 1). In this work, we provide an efficient algorithm that provides an over-approximation for the fastest way through a lane.
- For the first time, we validate our approach against real traffic data and check if the prediction always encloses the actual recorded behavior of other traffic participants.

The remainder of this paper is organized as follows: Sec. II presents the basic idea of how our occupancy prediction can be integrated in a collision avoidance concept and which constraints we make on the behavior of other traffic participants. The representation of the road network and the mathematical modeling of other traffic participants are presented in Sec. III. The algorithm for computing the occupancy prediction is introduced in Sec. IV. Sec. V presents the experimental results that are compared with a high-fidelity vehicle model and real traffic data; we also show how our approach can be integrated into a motion planner.

II. Motivation and Objective

The main idea of our proposed approach is quite simple: A planned trajectory of an automated vehicle can be verified as safe if the occupancy of the ego vehicle does not intersect with the occupancy of any other traffic participant at any time. Since even for a bounded time interval, infinitely many points in time exist, we check whether there exists a possible intersection of occupancies for consecutive time intervals as shown in Fig. 1. This only requires a finite number of collision checks while still ensuring that the reference trajectory of the ego vehicle is safe when there is no intersection for consecutive time intervals. When too few time intervals are chosen, the result may become too conservative, i.e. there exists an intersection of occupancies although a reference trajectory would be free of collision. This, however, can be easily addressed by recursively splitting time intervals $[t_k, t_{k+1}]$ into $[t_k, \tilde{t}]$ and $[\tilde{t}, t_{k+1}]$ with $\tilde{t} = 0.5(t_k + t_{k+1})$ for which occupancy sets are intersecting – the other time intervals remain unchanged. A recommended initial time step $t_{k+1} - t_k$ is 0.5 s.

The procedure for obtaining safe reference trajectories of the ego vehicle is illustrated in Fig. 2. To better guide the reader through the figure, we first introduce a few different types of trajectories:

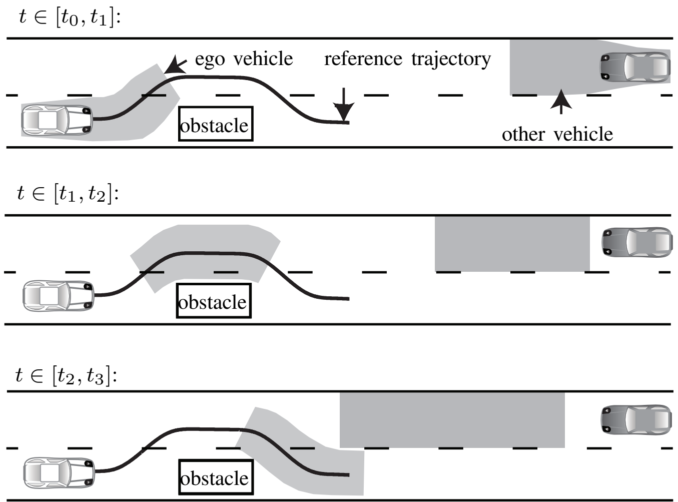


Fig. 1: Snapshots of the occupancy of traffic participants for selected consecutive time intervals.

- *Long-term reference trajectory*: A trajectory received from a trajectory planner (not the subject of this work) for a time horizon of typically several seconds that the ego vehicle should follow based on a non-formal prediction of other traffic participants.
- *Intended trajectory*: First part of a long-term reference trajectory subject to formal verification.
- *Fail-safe trajectory*: A trajectory that brings the vehicle to a safe state, such as a standstill on a shoulder or a safe distance behind another vehicle.
- *Potential trajectory*: Concatenation of an intended trajectory and a fail-safe trajectory, which has not yet been verified.
- *Safe trajectory*: Potential trajectory, which is verified as safe. A safe trajectory is verified for all times, since the ego vehicle is not accountable if it is hit in a safe state. If the reference trajectory would not result in a safe state, a collision might be inevitable when the reference trajectory is replanned [52].

The procedure in Fig. 2 is performed online and is continuously repeated to consider the constantly changing traffic situation. We start with a verified safe trajectory from the previous time step t_k (Fig. 2, ①). Next, the predicted occupancy of other traffic participants is computed (Fig. 2, ②). Please note that in order to keep the illustration compact, we do not show the occupancy for consecutive time intervals separately, but only the union of partial occupancy sets. Based on the occupied regions over time, new potential trajectories are planned (Fig. 2, ③). All potential trajectories are designed to branch off the previous one at a point B (see Fig. 2, ③), which is chosen based on the observation that the computation time of the verification t_{comp} is roughly linear in the prediction time horizon t_h: $t_{comp} \approx \lambda t_h$ ($\lambda \in \mathbb{R}, \lambda > 0$), where λ depends on the computational power. In order to obtain in most cases the verified result of the new potential trajectory upon arriving at B, we choose B as the point at which the vehicle will arrive at time $\lambda t_h + \epsilon$ ($\epsilon \in \mathbb{R}, \epsilon > 0$); larger values of ϵ provide more conservative results but limit the risk of starting the fail-safe trajectory in case the verification result is not obtained on time. A potential trajectory is safe if the ego vehicle following it would never intersect with the occupancy of another traffic participant. Out of those safe trajectories, the one that minimizes a user-defined cost function is selected (Fig. 2, ④ b). If no new safe trajectory is found, the previous trajectory is continued (Fig. 2, ④ a), which may result in executing the fail safe trajectory; this depends on whether there is enough time to replan before the fail safe trajectory is reached. However, in most cases, new safe trajectories are found so that the fail-safe trajectory is only executed in rare situations as demonstrated in Sec. V-D.

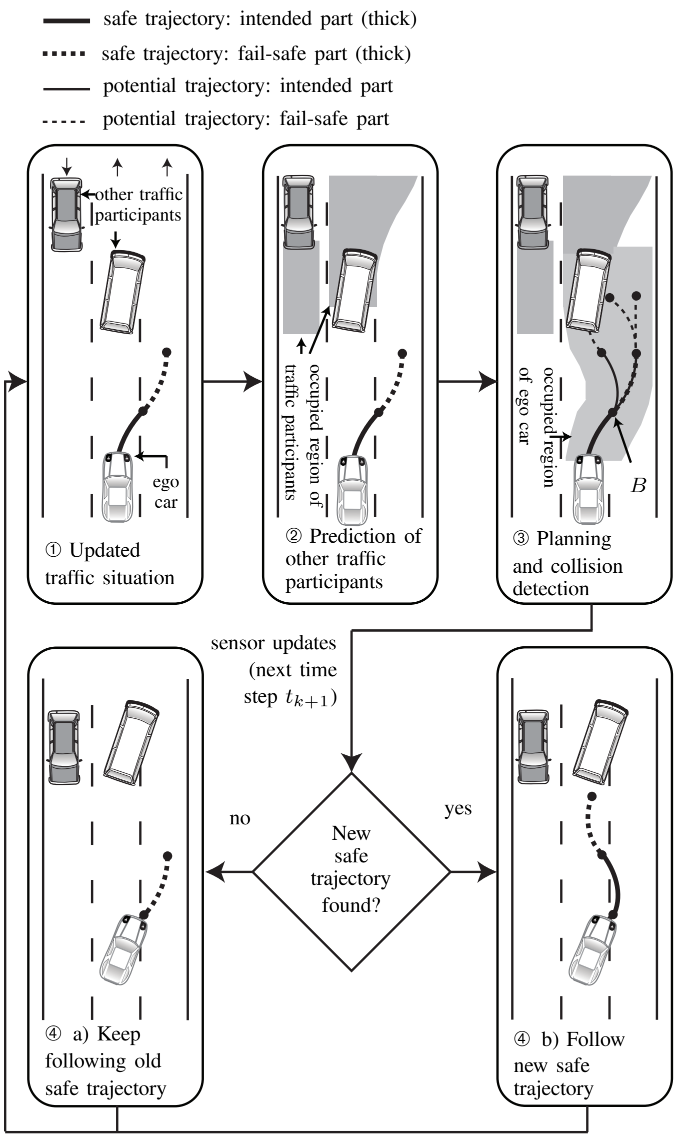

safe trajectory: intended part (thick)
safe trajectory: fail-safe part (thick)
potential trajectory: intended part
potential trajectory: fail-safe part
other traffic participants
ego car
① Updated traffic situation
occupied region of traffic participants
② Prediction of other traffic participants
occupied region of ego car
B
③ Planning and collision detection
sensor updates (next time step t_{k+1})
no
yes
New safe trajectory found?
④ a) Keep following old safe trajectory
④ b) Follow new safe trajectory

Fig. 2: Overview of online verification of automated vehicles. Gray regions show the occupancy of traffic participants including the ego vehicle. Black lines indicate different kinds of reference trajectories as illustrated in the legend at the top of the figure.

The set-based occupancy prediction results in increasingly larger regions when the prediction horizon is increased. Larger occupancies of other vehicles block space for the trajectory planning of the ego vehicle, as depicted in Fig. 2, ③. Thus, trajectory planning should be performed for two time horizons in parallel: long-term trajectories should be generated based on non-formal occupancy prediction techniques, such as single behaviors or probabilistic methods. A single behavior for a long-term maneuver (overtaking) is shown in the upper illustration of Fig. 3. Since the uncertainty of behaviors of other traffic participants grows over time, we apply our verification concept only to short potential trajectories (first part of a long-term reference trajectory plus a fail-safe trajectory) as depicted in the bottom illustration of Fig. 3. Due to the short time horizon, the proposed set-based techniques do not block overly large regions for trajectory planning. If the maneuver is safe, the next part of the long-term plan is executed; otherwise, the fail-safe trajectory is initiated. As a result, the proposed set-based occupancy prediction guarantees safe maneuvers, while non-formal techniques provide long-term plans based on likely behaviors of other traffic participants.

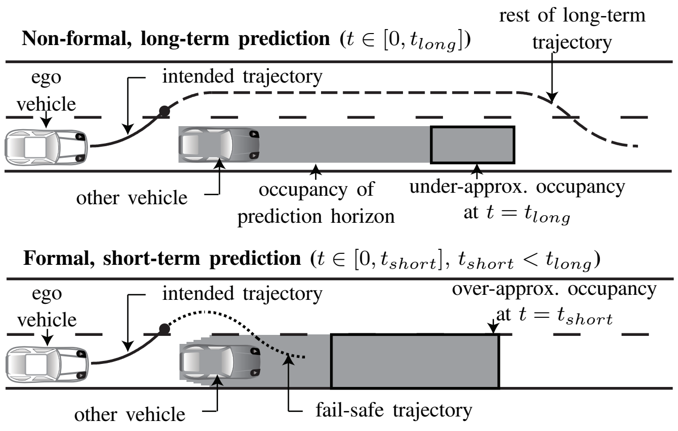

Non-formal, long-term prediction ($t \in [0, t_{long}]$)
rest of long-term trajectory
ego vehicle
intended trajectory
other vehicle
occupancy of prediction horizon
under-approx. occupancy at $t = t_{long}$
Formal, short-term prediction ($t \in [0, t_{short}]$, $t_{short} < t_{long}$)
ego vehicle
intended trajectory
over-approx. occupancy at $t = t_{short}$
other vehicle
fail-safe trajectory

Fig. 3: Comparing non-formal, long-term prediction with formal, short-term prediction.

The objective of this work is to create a framework for automatically computing the occupancy sets of other traffic participants on arbitrary road networks. A fast and robust algorithm for computing the occupancy prediction is required in order to meet real-time constraints. We further require computing an over-approximation of occupancy sets since it is provably impossible to compute exact ones [53]. Due to the over-approximation, our approach can be used for formal analysis – if the ego vehicle never enters the over-approximative prediction, then it also does not enter the exact prediction. A special feature of our approach is that we explicitly consider uncertainties in both the future behavior of traffic participants and their measurement of position, orientation, length, width, velocity, and acceleration. The occupancy is computed based on a model of other traffic participants and a model of the road network, which are introduced next.

III. Mathematical Modeling

We introduce a formal representation of road networks and the vehicle model used for predicting occupancies.

A. Road Network Representation

A formal and robust representation of road networks is required in order to compute the occupancy prediction. To this end, *lanelets* [54] are used, which are atomic, interconnected, and drivable road segments.

Definition 1 (Lanelet [54]): A lanelet is defined by its *left* and *right bound*, where each bound is represented by an array of points (a polyline), as shown in Fig. 4. □

It can be expected that lane information is provided in future autonomous vehicles, see e.g. [55]. The lane boundary information for lanelets can be directly obtained from an open-source map (e.g. OpenSteetMap[2]) by e.g. importing a raw map into the free editor *JavaOpenStreetMap* (JOSM)[3]. We additionally annotate the raw data consisting of *left border* and *right border* by the *speed limit* (maximum allowed speed on that specific lanelet) and a unique *ID*.

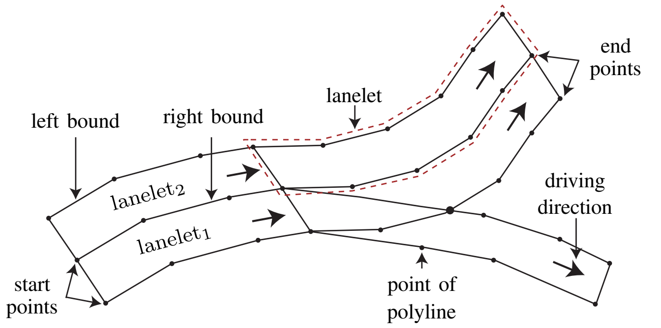


Fig. 4: Lanelets.

To consider all possible routes through the provided road network, an *adjacent lanelet matrix* is computed upfront. For this purpose, we represent the road network as a *directed graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the set of vertices $\mathcal{V}$ represents the lanelets and the set of directed edges $\mathcal{E}$ represents the possible transitions between two adjacent lanelets, where each node has four types of outgoing edges: longitudinal, left, right, empty. The adjacent lanelet matrix $A_\mathcal{G}$ is defined as follows: $A_\mathcal{G} : \mathcal{V} \times \mathcal{V} \to \{\mathrm{long}, \mathrm{right}, \mathrm{left}, \emptyset\}$, where $\times$ denotes the Cartesian product. Without loss of generality, it is assumed that all laterally adjacent lanes have the same length. This is exemplarily shown for the road network in Fig. 4, where $\mathrm{lanelet}_1$ has the same length as $\mathrm{lanelet}_2$. This assumption reduces the number of lateral adjacencies for multi-lane roads.

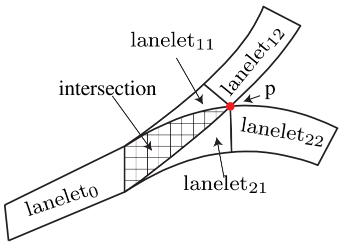
We define *start points* and *end points* of a lanelet as the first and the final points of the left and right border in driving direction, as shown in Fig. 4. Two lanelets are called *longitudinally adjacent*, i.e., $A_\mathcal{G}(\mathrm{lanelet}_1, \mathrm{lanelet}_2) = \mathrm{long}$, if the left and right start point of one lanelet are identical with the corresponding final points of the other lanelet. We say that $\mathrm{lanelet}_2$ is left-adjacent to $\mathrm{lanelet}_1$, i.e., $A_\mathcal{G}(\mathrm{lanelet}_1, \mathrm{lanelet}_2) = \mathrm{left}$, if the points of the left border of $\mathrm{lanelet}_1$ are identical to the ones of the right border of $\mathrm{lanelet}_2$. This is analogously defined for right-adjacent lanes. For implementation reasons, one might accept small deviations of connection points of lanelets rather than demanding that the values are identical.

For road networks which contain only longitudinally and laterally adjacent lanes, the construction of the adjacent road

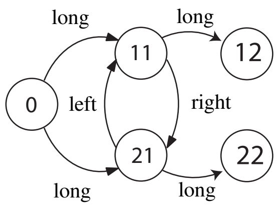
[2]www.openstreetmap.org
[3]https://josm.openstreetmap.de

matrix is straightforward. However, most road networks also contain road forks, see Fig. 4, which have to be constructed in a special way to ensure that the computed occupancies are over-approximative. This is ensured by considering possible lane changes as long as there exists an intersection of lanes as shown in Fig. 5(a). Since we can only model that lane crossings are either possible or not along the entire length of a lanelet, we have to partition the lanelets accordingly. Therefore, we introduce the point p as the intersection of the corresponding lane bounds of the bifurcating lanes, see Fig. 5(a). If the final points of the outer bounds of $\mathrm{lanelet}_{11}$ and $\mathrm{lanelet}_{21}$ correspond with the point p, and $\mathrm{lanelet}_{21}$ and $\mathrm{lanelet}_{22}$ continue the corresponding lanes as shown in Fig. 5(a), all lanelets fulfill the constraint that they are either adjacent along their full length or not at all. The resulting adjacency matrix is presented in Fig. 5(b). The adjacency makes it possible to define lanes:

Definition 2 (Lane): A lane is defined as the union of lanelets, which are longitudinally adjacent. □



(a) Lanelets.



(b) Adjacency graph.

Fig. 5: Road fork.

Next, we describe how we model other traffic participants.

B. Model of Other Traffic Participants

Inspired by [3], the mathematical model of other traffic participants is derived assuming the following constraints:

$C1$: positive longitudinal acceleration is stopped when a parameterized speed $v_{\max}$ is reached ($v_{\max}$ could be set to a certain percentage above the official speed limit);

$C2$: positive longitudinal acceleration is inversely proportional to speed above a parameterized speed v_S (modeling a maximum engine power);

$C3$: driving backwards in a lane is not allowed;

$C4$: maximum absolute acceleration is limited by $a_{\max}$;

$C5$: actions that cause leaving the road/lane/sidewalk/crosswalk boundary are forbidden. Crossing lanes is allowed if not forbidden by lane markings or traffic regulations.

While $C2$ and $C4$ are physical constraints, the other constraints are derived from the traffic rules described in the Vienna Convention on Road Traffic [56]. Note that $C1$ can also be a physical constraint if no speed limit exists, as can be found on the German Autobahn. Formalization of traffic rules is a new research area [57], but out of the scope of this paper. A further extension is to integrate further constraints when surrounding vehicles are automated and communicate their future plans. In that case, one only has to consider the uncertainty of following planned trajectories due to sensor noise as demonstrated in [58]. If only more high-level information is broadcast, such as the information that the vehicle will not perform a lane change, we can simply adjust the adjacent lanelet matrix $A_{\mathcal{G}}$ to consider such a constraint. Of course, further constraints which reflect other traffic rules can be considered. However, removing some of the constraints does not affect the soundness of the verification procedure, but it increases the behavior uncertainty of other traffic participants, which only leads to more conservative behavior of the ego vehicle. We model the dynamics of other traffic participants by a point mass (while a rectangle is considered to enclose their size):

$$\ddot{s}_x(t) = a_x(t), \quad \ddot{s}_y(t) = a_y(t), \tag{1}$$

where $s_x(t), s_y(t)$ denotes the position and $a_x(t), a_y(t)$ denotes the acceleration in x and y coordinates. In order to restrict $a_x(t)$ and $a_y(t)$ according to the constraints $C1$-$C4$, unit vectors that point towards the longitudinal and lateral direction of the vehicle are introduced: $\Phi_{\mathrm{long}}(t) = \frac{1}{v}[v_x(t),\, v_y(t)]^T$, $\Phi_{\mathrm{lat}}(t) = \frac{1}{v}[-v_y(t),\, v_x(t)]^T$, where $v = \|[v_x, v_y]^T\|_2$ and $\|.\|_2$ denotes the Euclidean norm. Thus, we can define a_x, a_y as a function of the longitudinal acceleration $a_{\mathrm{long}}(t)$ and the lateral acceleration $a_{\mathrm{lat}}(t)$:

$$\begin{bmatrix} a_x \\ a_y \end{bmatrix} = \Phi_{\mathrm{long}} a_{\mathrm{long}} + \Phi_{\mathrm{lat}} a_{\mathrm{lat}}.$$

Let us define a normalized steering input u_1, where $u_1 = 1$ refers to full steering to the left and $u_1 = -1$ refers to full steering to the right considering the maximum tire friction potential, from which results the lateral acceleration

$$a_{\mathrm{lat}} = a_{\max}\, u_1.$$

In order to consider constraint $C4$, the remaining acceleration potential in longitudinal direction is limited to

$$a_{\mathrm{c1,long}} = \sqrt{(a_{\max})^2 - (a_{\mathrm{lat}})^2}$$

according to Kamm's circle, which is a good approximation since the peak forces of tires are almost identical for the longitudinal and the lateral direction, see e.g. [59, Fig. 14-16]. The maximum longitudinal acceleration for the engine power P and the vehicle mass m is $\frac{P}{m\,v} = a_{\max}\frac{v_S}{v}$, where $v_S = \frac{P}{a_{\max} m}$ is the speed above which the acceleration is limited by the engine power and no longer by the tire friction. However, since this parameter may not be easily estimated, another option is to set $v_S = \infty$ as a fallback solution, which provides an over-approximation of the occupancy set. Similarly to the lateral acceleration, a normalized control input u_2 for the longitudinal acceleration is introduced, where $u_2 = \pm 1$ represents full braking and full acceleration within the acceleration potential. Limited engine power, the restriction to forward driving, and the maximum speed (constraints $C1$-$C3$) are considered by limiting the longitudinal acceleration to

$$a_{\mathrm{c2,long}} = \begin{cases} a_{\max}\frac{v_S}{v}, & v_S < v < v_{\max} \wedge u_2 > 0, \\ a_{\max}, & (0 < v \le v_S \vee (v > v_S \wedge u_2 \le 0)), \\ 0, & v \le 0 \vee (v \ge v_{\max} \wedge u_2 \ge 0). \end{cases}$$

Combining $a_{\mathrm{c1,long}}$ and $a_{\mathrm{c2,long}}$ results in the longitudinal acceleration which satisfies the proposed constraints $C1$-$C4$ ($C5$ for not leaving the road is considered later):

$$a_{\mathrm{long}} = \begin{cases} a_{\mathrm{c2,long}}\, u_2, & a_{\mathrm{c2,long}}\, |u_2| \le a_{\mathrm{c1,long}}, \\ a_{\mathrm{c1,long}}\, \mathrm{sgn}(u_2), & a_{\mathrm{c2,long}}\, |u_2| > a_{\mathrm{c1,long}}. \end{cases}$$

Of course, *static obstacles*, e.g. stationary vehicles or just other objects, can simply be modeled as other traffic participants with zero velocity.

IV. Occupancy Prediction

Based on the models of the road network and other traffic participants, we predict the occupancy of other traffic participants in this section. Since the occupancy prediction is periodically computed as described in Sec. II, it is desirable to efficiently compute the occupancy to realize frequent updates and to be able to quickly react to changing traffic situations. Let us first introduce what we refer to as a model in this paper.

Definition 3 (Model): Given is a dynamical system $\dot{x} = f(x(t), u(t))$, where x is the state, u is the input, and t is the time. The possible initial states and the inputs are bounded by sets: $x(0) \in \mathcal{X}_0$, $\forall t: u(t) \in \mathcal{U}$. The model of this system is defined as the tuple (ordered set) $M = (f, \mathcal{X}_0, \mathcal{U})$. □

Given a model M, we define a reachable set as follows.

Definition 4 (Reachable set): The reachable set of a model M (see Def. 3) at time $t = r$ is

$$\mathcal{R}(M, r) = \left\{ \int_0^r f(x(t), u(t))dt \middle| x(0) \in \mathcal{X}_0, \forall t: u(t) \in \mathcal{U} \right\}$$

and the reachable set of a time interval $t \in [0, r]$ is

$$\mathcal{R}(M, [0, r]) = \bigcup_{t \in [0,r]} \mathcal{R}(M, t). \qquad \square$$

However, due to the consideration of constraints $C1$-$C5$ of the vehicle dynamics as described in Sec. III-B, we would have to model the dynamics of the vehicle using a mixture of discrete and continuous dynamics, which are also referred to as a hybrid system [60]. For instance, if the vehicle reaches the maximum velocity $v_{\max}$, the dynamics has to change to constant velocity. Since the computation of reachable sets of hybrid systems is too time consuming for our application, we instead use abstractions of the original model M.

Definition 5 (Abstraction): Given is a model M of a dynamical system. The model M_i is an abstraction of M if the reachable set of the abstraction contains the reachable set of the model, i.e. $\forall t > 0 : \mathcal{R}(M, t) \subseteq \mathcal{R}(M_i, t)$. □

The abstract model should be chosen such that it preserves the essential behavior of the original model and permits computationally more efficient methods for reachability analysis. Since ultimately, we are only interested in the occupancy, which is determined by the position and orientation of other traffic participants, we aim to find abstractions that are tight with respect to these variables. Given a state vector $x \in \mathbb{R}^n$, where the first two elements x_1, x_2 are the x-position and the y-position, and the third element x_3 is the orientation, we denote by $\mathrm{proj}(x) = [x_1, x_2, x_3]^T$ the operator that projects the state vector onto position and orientation. Further, $\mathrm{proj}\big(\mathcal{R}(M, t)\big) := \{\mathrm{proj}(x) | x \in \mathcal{R}(M, t)\}$ returns the set of possible positions and orientations. We exploit the fact that in the end we are only interested in the position and orientation using the following proposition taken from [3, Prop. 5.1]:

Proposition 1 (Over-approximative occupancy): Given are models M_i, $i = 1, \dots, m$ which are abstractions of model M_0, i.e., $\forall t > 0 : \mathcal{R}(M_0, t) \subseteq \mathcal{R}(M_i, t)$. The occupancy of the model M_0 can be over-approximated by

$$\forall t > 0 : \mathrm{proj}\big(\mathcal{R}(M_0, t)\big) \subseteq \bigcap_{i=1}^{m} \mathrm{proj}\big(\mathcal{R}(M_i, t)\big).$$

□

The result remains an over-approximation, but the reachable set computation of the abstractions chosen in this paper is significantly faster, while still providing accurate results. The abstractions are significantly easier to compute since 1) they do not require one to consider the complete set of states, which is a big advantage considering that the computational complexity of reachability analysis is superlinear in the number of state variables, and 2) we do not have to consider constraints $C1$-$C5$ by a hybrid dynamics, but by an intersection of results of abstractions that are purely continuous. The larger the number of computed abstract models is, the more accurate the occupancy prediction becomes, but as a drawback, the computing time may increase. This allows us to tune the ratio between accuracy and computational time. However, one can initialize a parallel thread for each abstract model considered when hardware with multiple cores is used.

We compute the occupancy for consecutive time intervals $\tau_k := [t_k, t_{k+1}]$ for a user-specified step size $r = t_{k+1} - t_k$ and user-specified time horizon t_h. Consecutive time intervals instead of points in time are used to ensure that no collision detection is missed between points in time. In this paper, we consider two abstract models for computing the occupancy of each traffic participant. The first abstraction is denoted by M_1, and it considers constraint $C3$ and $C4$ (see Sec. III-B). The predicted occupancy using abstraction M_1 is denoted by $\mathcal{O}_1(t) \supseteq \mathrm{proj}\big(\mathcal{R}(M_1, t)\big)$, and we call it *acceleration-based occupancy*. The second abstraction M_2 considers the constraints $C1, C2$, and $C4$ in longitudinal direction. Its occupancy is denoted by $\mathcal{O}_2(t)$, and we refer to it as *lane-following occupancy*. Finally, the constraint $C5$ is considered by intersecting the drivable road area $\mathcal{O}_{\mathrm{road}}$ with the other occupancies. Thus, the overall occupancy is

$$\mathcal{O}(t) = \mathcal{O}_1(t) \cap \mathcal{O}_2(t) \cap \mathcal{O}_{\mathrm{road}}. \qquad (2)$$

It remains to decide on a set representation for $\mathcal{O}(t)$. To efficiently perform collision checks of the occupied regions with the ego vehicle and to exactly compute the intersections in (2), we require a set representation that a) is closed under intersection, b) can represent non-convex sets, and c) for which efficient algorithms for collision detection exist. We choose polygons since other common set representations (ellipsoids, boxes, zonotopes, polyhedra, etc.) are all convex.

Definition 6 (Polygon): Given is a tuple $(v_1, \ldots, v_p)$ of p vertices $v_i \in \mathbb{R}^2$. A polygon $P(v_1, \ldots, v_p)$ is a set in the plane bounded by a border that consists of straight lines between neighboring vertices v_i and v_{i+1}, except that the last vertex v_p is connected with the first one v_1, see e.g. Fig. 7(b). □

For intersections of polygons as required in (2), a fast algorithm is required. A lot of algorithms already exist for computing the intersection of two polygons [61]. Considering the strict time requirements, we choose the *Greiner Hormann Polygon Clipping Algorithm* since it can efficiently deal with non-convex shapes [62].

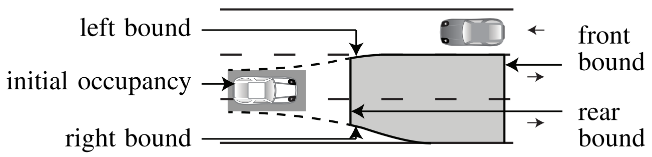


Fig. 6: Initial occupancy and boundaries of the occupancy set.

To obtain $\mathcal{O}_1(t)$ and $\mathcal{O}_2(t)$ in (2), each vehicle is characterized by a set of parameters, listed in Table I. Since the prediction of the occupancy sets is done in an over-approximative manner, all possible lanelets a vehicle can follow have to be considered. To this end, we search in the adjacency graph of the road network for all reachable lanelets (see e.g. road fork in Fig. 5(b)). Furthermore, the algorithm is parallelized for every vehicle, in order to improve the computational performance. In the following subsections, the occupancy prediction for each considered abstract model is presented, where abstraction M_1 is considered in Sec. IV-A and abstraction M_2 in Sec. IV-B.

TABLE I: Vehicle parameters.

parameter	variable [unit]	parameter	variable [unit]
max. acceleration	a_{max} [m/s^2]	vehicle width	$\tilde{w}$ [m]
max. velocity	v_{max} [m/s]	vehicle length	$\tilde{l}$ [m]
switching velocity	v_S [m/s]		

A. Acceleration-Based Occupancy (Abstraction M_1)

The first abstraction of the vehicle dynamics considers that driving backwards is not allowed ($C3$) and that absolute acceleration is limited ($C4$). The latter implies that the over-approximated occupancy at time t can be described by a circle with center $c(t)$ and radius $r(t)$ (see [43]) as shown in Fig. 8 when $C3$ is not yet enforced:

$$c(t) = \begin{bmatrix} s_x(0) \\ s_y(0) \end{bmatrix} + \begin{bmatrix} v_x(0) \\ v_y(0) \end{bmatrix} t, \quad r(t) = \frac{1}{2} a_{max} t^2. \tag{3}$$

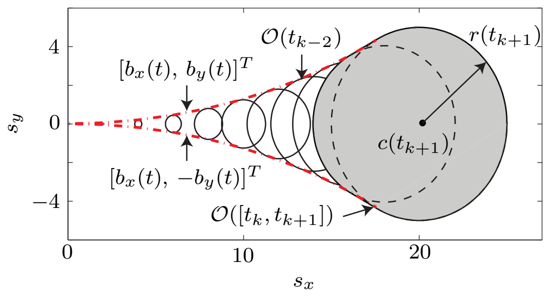


Fig. 8: Occupancy sets.

Assuming that $s_x = 0, s_y = 0, v_x = v$ and $v_y = 0$ (without loss of generality), the boundary of the occupancy is described by a two-dimensional function $[b_x(t), b_y(t)]^T$, where

$$b_x(t) = v_0 t - \frac{a_{max}^2 t^3}{2 v_0}, b_y(t) = \sqrt{\frac{1}{4} a_{max}^2 t^4 - \Big(\frac{a_{max}^2 t^3}{2 v_0}\Big)^2}, \tag{4}$$

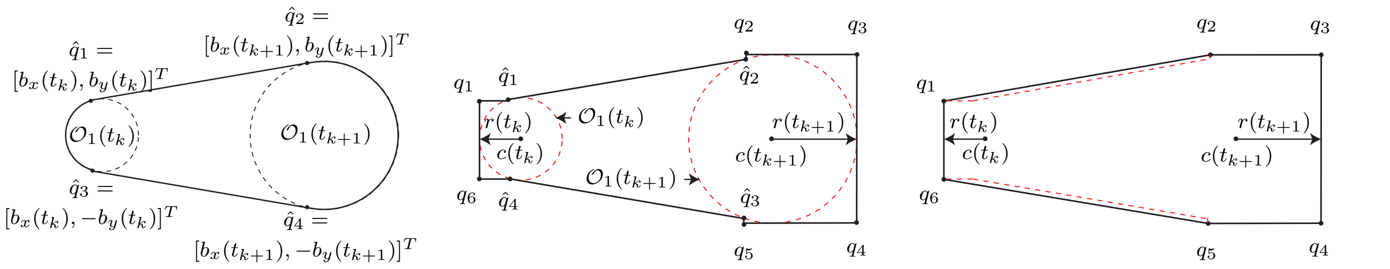
as shown in [3]. In order to prevent driving backward, the solution in x-direction is to set the maximum value of $b_{x_{max}} := b_x(t_{max})$ for $t \geq t_{max}$. The value $t_{max} = \sqrt{2/3} \frac{v_0}{a_{max}}$ is found by solving for $\dot{b}_x(t) = 0$. The outer bound of the occupancy $\mathcal{O}_1(\tau_k)$ for a given time interval $\tau_k = [t_k, t_{k+1}]$ is obtained by the next lemma.

Lemma 1 ($\mathcal{O}_1(\tau_k)$ without vehicle dimensions): The occupancy $\mathcal{O}_1(\tau_k)$ without considering vehicle dimensions is over-approximated by a polygon $P(q_1, \ldots, q_6)$, where

$$\begin{aligned}
q_1 &= [c_x(t_k) - r(t_k),\, c_y(t_k) + r(t_k)]^T, \\
q_2 &= [b_x(t_k),\, c_y(t_{k+1}) + r(t_{k+1})]^T, \\
q_3 &= [c_x(t_{k+1}) + r(t_{k+1}),\, c_y(t_{k+1}) + r(t_{k+1})]^T, \\
q_4 &= [c_x(t_{k+1}) + r(t_{k+1}),\, c_y(t_{k+1}) - r(t_{k+1})]^T, \\
q_5 &= [b_x(t_k),\, c_y(t_{k+1}) - r(t_{k+1})]^T, \\
q_6 &= [c_x(t_k) - r(t_k),\, c_y(t_k) - r(t_k)]^T.
\end{aligned}$$
□

Proof: We over-approximate the occupancy by the convex hull of the occupancy $\mathcal{O}(t_k)$ and $\mathcal{O}(t_{k+1})$, as shown in Fig. 7(a). This is an over-approximation since the boundary $[b_x(t), b_y(t)]^T$ form (4) is concave. The points where $[b_x(t), b_y(t)]^T$ and $[b_x(t), -b_y(t)]^T$ intersect with $\mathcal{O}_1(t_k)$ and $\mathcal{O}_1(t_{k+1})$ are denoted by $\hat{q}_1$ - $\hat{q}_4$, see Fig. 7(a). The exact boundary $\mathcal{O}_1(\tau_k)$ is over-approximated by the vertices q_1-q_6 and $\hat{q}_1$ - $\hat{q}_4$, see Fig. 7(b). Finally, we over-approximate the result in Fig. 7(b) by its convex hull, such that $\hat{q}_1$ - $\hat{q}_4$ are removed, see Fig. 7(c), resulting in the vertices of the lemma.■

The motivation for removing $\hat{q}_1$ - $\hat{q}_4$ is to obtain fewer vertices in order to reduce memory consumption and computation time for collision detection, at the cost of a negligible over-approximation. So far, we have assumed that each vehicle is a point mass without considering their dimensions. We enclose the shape of a vehicle including its uncertain initial position in a rectangle with length $\tilde{l}$ and width $\tilde{w}$ and whose reference point is the centroid. After adding measurement uncertainties, we enclose the occupancy of the vehicle in a larger rectangle of length l and width w as depicted in Fig. 9. Assuming that



$\hat{q}_1 = [b_x(t_k), b_y(t_k)]^T$
$\hat{q}_2 = [b_x(t_{k+1}), b_y(t_{k+1})]^T$
$\mathcal{O}_1(t_k)$
$\mathcal{O}_1(t_{k+1})$
$\hat{q}_3 = [b_x(t_k), -b_y(t_k)]^T$
$\hat{q}_4 = [b_x(t_{k+1}), -b_y(t_{k+1})]^T$
q_1 q_2 q_3 q_4 q_5 q_6
$r(t_k)$ $c(t_k)$ $r(t_{k+1})$ $c(t_{k+1})$

(a) Convex hull of occupancy. (b) Enclosing, non-convex polygon. (c) Enclosing, convex polygon.

Fig. 7: Computation steps to obtain the over-approximative occupancy.

each point of the vehicle has the acceleration bound a_{max} and after introducing the notation $M(\mathcal{X}^*)$ for a model whose set of initial states is $\mathcal{X}_0 = \mathcal{X}^*$, we can formulate that

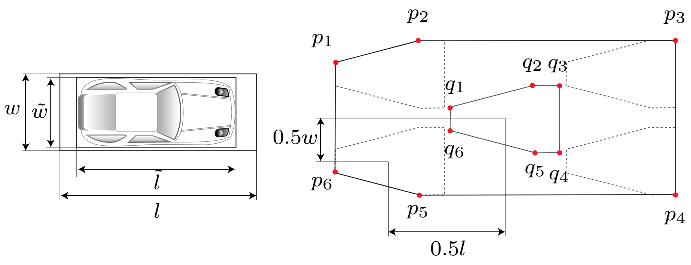
$$\mathcal{R}(M(\mathcal{X}_0), t) = \bigcup_{x_0 \in \mathcal{X}_0} \mathcal{R}(M(x_0), t), \tag{5}$$

which directly follows from the definition of reachable sets in Def. 4. Using (5), we can over-approximate the occupancy for a vehicle within an initial occupancy set:

Theorem 1 ($\mathcal{O}_1(\tau_k)$ with vehicle dimensions): The exact occupancy $\mathcal{O}_1(\tau_k)$ when $\mathcal{X}_0$ is a rectangle of length l and width w, is over-approximative by a polygon $P(p_1, \ldots, p_6)$ using the vertices q_1-q_6 from Lemma 1, where

$$\begin{aligned} p_1 &= q_1 + [-0.5l,\ 0.5w]^T, & p_4 &= q_4 + [0.5l,\ -0.5w]^T, \\ p_2 &= q_2 + [-0.5l,\ 0.5w]^T, & p_5 &= q_5 + [-0.5l,\ -0.5w]^T, \\ p_3 &= q_3 + [0.5l,\ 0.5w]^T, & p_6 &= q_6 + [-0.5l,\ -0.5w]^T. \square \end{aligned}$$

Proof: The proof follows directly from Lemma 1 and (5). Fig. 9 illustrates the resulting points of the theorem. ■



w $\tilde{w}$ $\tilde{l}$ l
p_1 p_2 p_3 p_4 p_5 p_6
q_1 q_2 q_3 q_4 q_5 q_6
$0.5w$
$0.5l$

Fig. 9: Occupancy polygon of a vehicle. The points q_1-q_6 are taken from Fig. 7

We have assumed without loss of generality, due to position and orientation invariance, a special relative position and orientation of the initial set. To obtain the occupancy for an arbitrary situation, the occupancy is finally translated and rotated according to the initial position and orientation.

B. Lane-Following Occupancy (Abstraction M_2)

In the previous subsection, we considered limited absolute acceleration of other traffic participants. However, as constraints $C1 - C2$ describe, a vehicle cannot always accelerate with a_{max} in longitudinal direction (i.e. driving direction) to consider maximum velocity ($C1$) and maximum engine power ($C2$). In order to efficiently consider $C1 - C2$, we enforce these constraints along partial paths along inner lane bounds. Once the maximum position along these partial paths is determined, we use the abstraction that the vehicle can be anywhere in lateral direction (i.e. perpendicular to the path) within the lane boundaries. Please note that in abstraction M_2, $C4$ is still enforced in longitudinal direction, but not in lateral direction, which we denote by $C4^*$. Thus, the traveled distance along a path is irrespective of its shape when considering $C1$, $C2$, and $C4^*$:

Proposition 2 (Traveled distance along path for M_2): Since we do not consider lateral acceleration in constraints $C1$, $C2$, and $C4^*$, the maximum velocity is not affected by the shape of the followed path. □

Lateral acceleration is considered, but only in model M_1. The computation of the front position along a path, which is denoted by $\xi_f(t)$ (f: front), is computed for model M_2 as presented in [3]: After introducing the interval of initial path coordinates $[\underline{\xi}_0, \overline{\xi}_0]$ and velocities $[\underline{v}_0, \overline{v}_0]$, $\xi_f(t)$ is obtained by simulating the vehicle model in Sec. III-B with maximum acceleration $u_2 = 1$ from $\xi_f(0) = \overline{\xi}_0$ and $v(0) = \overline{v}_0$. Please note that a single simulation suffices due to the monotonicity property of longitudinal dynamics. This raises the question of how to choose the shortest path since infinitely many paths exist, even if the vehicle just follows a single lanelet. In our previous work [3], we did not address this problem and assumed the shortest path to be the center path along a lane, which does not result in an over-approximation since one can cut corners. In this work, we provide an abstraction that encloses all behaviors possible under constraints $C1$, $C2$, and $C4^*$, no matter which path through a sequence of lanelets is chosen. For this purpose, we first present an efficient algorithm to compute a lower bound of the shortest path through a sequence of lanelets when following the road network.

Let us first formalize the problem of finding the shortest path through a lane. Without loss of generality, we can assume that the vehicle is only a point when reducing the distances of lane boundaries to account for the dimension of vehicles. If this step is not performed to save computation time, the result would still be conservative, i.e. the shortest path would be shorter than the exact solution.

Problem 1 (Shortest path through a lane): Given is an arbitrary lane as depicted in Fig. 11. Let us denote the start border by $b_{start}(s_x, s_y)$ and the end border by $b_{end}(s_x, s_y)$.

A point $[s_x, s_y]^T$ is on the start line for $b_{start}(s_x, s_y) = 0$ and $b_{start}(s_x, s_y) < 0$ when a point is inside the lane segment. This is analogous for $b_{end}(s_x, s_y)$. The lane boundaries are formulated as inequality constraints, where one is on the right hand side from the left boundary in driving direction if $b_{left}(s_x, s_y) < 0$ and analogously for $b_{right}(s_x, s_y) < 0$. This makes it possible to formulate the area of the lane segment as the set

$$\begin{aligned} \mathcal{O}_{segment} =& \{[s_x, s_y]^T | b_{start}(s_x, s_y) < 0, b_{end}(s_x, s_y) < 0, \\ & b_{left}(s_x, s_y) < 0, b_{right}(s_x, s_y) < 0\}. \end{aligned} \tag{6}$$

We also introduce a path variable ξ so that the x-position and y-position are functions of it: $s_x(\xi)$, $s_y(\xi)$. The path variable ξ can be obtained from $s_x(\xi)$, $s_y(\xi)$ as

$$\begin{aligned} \xi &= \int_0^{\xi} \sqrt{s_x'^2(\hat{\xi}) + s_y'^2(\hat{\xi})}\, \mathrm{d}\,\hat{\xi}, \\ s_x'(\hat{\xi}) &= \frac{\mathrm{d}\, s_x(\xi)}{\mathrm{d}\,\xi}\Big|_{\xi=\hat{\xi}}, \quad s_y'(\hat{\xi}) = \frac{\mathrm{d}\, s_y(\xi)}{\mathrm{d}\,\xi}\Big|_{\xi=\hat{\xi}}. \end{aligned} \tag{7}$$

The problem of finding the shortest path from b_{start} to b_{end} while respecting the road boundaries can be formulated as

$$\min_{s_x(\xi), s_y(\xi)} J(s_x(\xi), s_y(\xi)) = \int_0^{\xi_f} \sqrt{s_x'^2(\hat{\xi}) + s_y'^2(\hat{\xi})} \mathrm{d}\,\hat{\xi} \quad \square$$

under the constraints

$$\begin{aligned} b_{start}(s_x(0), s_y(0)) &= 0, \\ b_{end}(s_x(\xi_f), s_y(\xi_f)) &= 0, \\ b_{left}(s_x(\xi), s_y(\xi)) &< 0, \\ b_{right}(s_x(\xi), s_y(\xi)) &< 0. \end{aligned} \tag{8}$$

Problem 1 is too time-consuming to solve for our time-critical application, even when only polygonal obstacles are considered [63]–[65]. Instead of exactly solving Problem 1, we compute an under-approximation of the shortest path. For this reason, we introduce the notion of corresponding paths.

Definition 7 (Corresponding path): We define the corresponding path $h(\xi)$ with path variable ξ of the border b as

$$b(s_x(\xi), s_y(\xi)) = 0 \quad \Rightarrow \quad h(\xi) = [s_x, s_y]^T.$$

The other direction $h(\xi) = [s_x, s_y]^T \Rightarrow b(s_x(\xi), s_y(\xi)) = 0$ is not unique, and we only demand that a possible solution is provided such that the set $\{[s_x, s_y]^T | b(s_x, s_y) < 0, s_x, s_y \in \mathbb{R}\}$ refers to the inner part behind the border. $\square$

The path $h(\xi)$ either refers to the left or right border of a lane, depending on context. When using $h_{right}(\xi)$, we explicitly refer to the right border and analogously for $h_{left}(\xi)$. We denote the well-known signed curvature by

$$\kappa(\xi) = f_\kappa(h(\xi)) = \frac{s_x'(\xi) s_y''(\xi) - s_x''(\xi) s_y'(\xi)}{(s_x'(\xi)^2 + s_y'(\xi)^2)^{3/2}}.$$

Let us first under-approximate the length of the shortest path when the inner border has no inflection point.

Lemma 2 (Shortest path without inner inflection point): Given is a lane segment as described in Problem 1, where the inner lane bound does not change the sign of the curvature, i.e., $\nexists \xi : \kappa_{right}(\xi) := f_\kappa(h_{right}(\xi)) = 0$. Without loss of generality, we consider a right curve so that the inner bound is h_{right}. We further require that h_{start}, h_{end} are straight lines perpendicular to h_{right}. This can be formalized by introducing the normal vectors $n_{start}, n_{end} \in \mathbb{R}^2$ and the distance values $d_{start}, d_{end} \in \mathbb{R}$:

$$\begin{aligned} & \forall \xi \in [0, \xi_f] : \\ & \kappa_{right}(\xi) < 0 \text{ (right turn)}, \\ & \text{constraints in (8) (lane borders)}, \\ & b_{start}(s_x, s_y) = n_{start}^T [s_x, s_y]^T - d_{start} \text{ (start line)}, \\ & b_{end}(s_x, s_y) = n_{end}^T [s_x, s_y]^T - d_{end} \text{ (finish line)}, \\ & \frac{|n_{start}^T h_{right}'(0)|}{\|n_{start}\|_2 \|h_{right}'(0)\|_2} = 1 \text{ (b_{start} perpendicular to h_{right})}, \\ & \frac{|n_{end}^T h_{right}'(0)|}{\|n_{end}\|_2 \|h_{right}'(0)\|_2} = 1 \text{ (b_{end} perpendicular to h_{right})}. \end{aligned} \tag{9}$$

The shortest path according to Problem 1 is $[s_x^*(\xi), s_y^*(\xi)]^T = h_{right}(\xi)$ for $\xi \in [0, \xi_f]$. $\square$

Proof: Let us divide the problem of the lemma into sub-problems with auxiliary line segments $h_{aux,i}(\alpha) = p_i + \alpha l_i$, $\alpha \in [0, 1]$, $p_i, l_i \in \mathbb{R}^2$, and the corresponding border is denoted by $b_{aux,i}$, see Def. 7. Then, we obtain the same sub-problems, except that b_{start} is replaced by $b_{aux,i}$ and b_{end} is replaced by $b_{aux,i+1}$. We are now interested in the shortest distance between two auxiliary line segments. Let us denote the start and end points of the shortest line segment connecting both line segments $h_{aux,i}$ and $h_{aux,i+1}$ by $p_{d,i}$ and $p_{d,i+1}$ (see Fig. 10). There exist four cases for the shortest distance between two line segments [66]: A) $p_{d,i}$ and $p_{d,i+1}$ are within $h_{aux,i}$ and $h_{aux,i+1}$, B) $p_{d,i}$ is within $h_{aux,i}$ and $p_{d,i+1}$ is one of the end points of $h_{aux,i+1}$, C) opposite of case B, D) $p_{d,i}$ and $p_{d,i+1}$ are end points of $h_{aux,i}$ and $h_{aux,i+1}$. In the two-dimensional setting, case A only occurs when the line segments intersect, which can be ruled out, see Fig. 10. Case B and C result in longer solutions compared to case D; this can be easily seen, since either the movement of $p_{d,i}$ along $h_{aux,i}$ or the movement of $p_{d,i+1}$ along $h_{aux,i+1}$ increases the distance.

When we let the number N of auxiliary straight lines $h_{aux,i}$ approach infinity, the union $\bigcup_i^N \{p_{d,i} + \gamma(p_{d,i+1} - p_{d,i}) | \gamma \in [0, 1]\}$ of the shortest line segments approaches the set of points on $h_{right}(\xi)$. ■

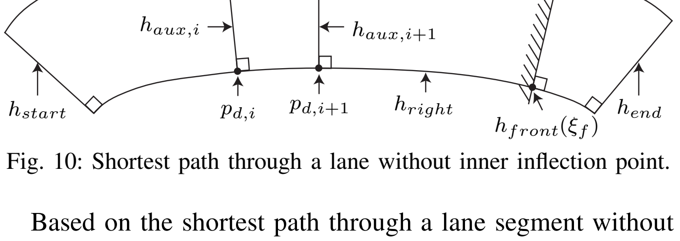


Fig. 10: Shortest path through a lane without inner inflection point.

Based on the shortest path through a lane segment without inner inflection point, we can compute an over-approximative occupancy through such a lane segment.

Lemma 3 (Occupancy without inner inflection point): Given is a lane segment as formalized in (9), where the inner lane bound does not have an inflection point. Let us introduce the front position $\xi_f(t)$ along the inner path under constraints $C1$, $C2$, and $C4^*$, and the front border $b_{front}(\xi_f(t), s_x, s_y) = n_f^T(\xi_f(t))\big[[s_x, s_y]^T - h_{right}(\xi_f(t))\big]$ (see Fig. 10), which is perpendicular to the inner bound (here: h_{right}) so that n_f is aligned with the inner bound: $|n_f^T(\xi)h'_{right}(\xi)|/(\|n_f(\xi)\|_2\|h_{right}(\xi)\|_2) = 1$. Then, by using $\mathcal{O}_{segment}$ from (6), the over-approximative occupancy under constraints $C1$, $C2$, and $C4^*$ is

$$\mathcal{O}_2(t) = \{[s_x, s_y]^T | b_{front}(\xi_f(t), s_x, s_y) < 0\} \cap \mathcal{O}_{segment}$$

under the assumption that the segment describes a turn of less than 90°. If this is not the case, one can split the segment into smaller segments without loss of generality. □

Proof: Due to the invariance of the velocity with respect to the shape of a followed path (see Prop. 2), the fastest trajectory for getting from one border of a lane to another one is determined by the shortest path. From Lemma 2 we already know that the shortest path of lanes with no inner inflection point is the inner border itself. Hence, it suffices to compute the front bound along the path of the inner border and perpendicular to it. According to Lemma 2 no shorter path exists between b_{start} and b_{front}, which concludes the proof. ■

To over-approximate the occupancy along an arbitrary lane, we introduce our inflection-point segmentation.

Definition 8 (Inflection-point segmentation): We define our inflection-point segmentation of a lane algorithmically, whose result is illustrated in Fig. 11:

1) If $\kappa_{left} > 0$ (inner bound), start at $[\tilde{s}_x, \tilde{s}_y]^T$, where $b_{start}(\tilde{s}_x, \tilde{s}_y) = 0$, $b_{left}(\tilde{s}_x, \tilde{s}_y) = 0$, and follow h_{left}. Otherwise, start at $[\hat{s}_x, \hat{s}_y]^T$, where $b_{start}(\hat{s}_x, \hat{s}_y) = 0$, $b_{right}(\hat{s}_x, \hat{s}_y) = 0$, and follow h_{right}. Note that it is possible that the left and right bound of the lane are concave initially, but then the left side is the default according to this algorithm.
2) Follow the current bound until an inflection point γ_i is reached, which is defined by a sign change of κ_{left} or κ_{right}, depending on the border that is followed (see Fig. 11). Construct a line segment $h_{cross,i}(\alpha) = \gamma_i + \alpha g_i$, $\gamma_i, g_i \in \mathbb{R}^2$, $\alpha \in [0, 1]$, from one bound to the other, which is perpendicular to the bound belonging to γ_i. For the change from the left to the right bound, we have $g_i^T h'_{left} = 0$, $h_{cross,i}(0) = h_{left}(\gamma_{i,x}, \gamma_{i,y})$, $h_{cross,i}(1) = h_{right}(\mu_{i,x}, \mu_{i,y})$, and μ_i is the point from where one continues on the other bound (see Fig. 11). The procedure is analogous for the change from the right to the left bound.
3) To obtain a proper segmentation, we need the additional line segments $\tilde{h}_{cross,i}(\alpha) = \mu_i + \alpha\tilde{g}_i$, $\alpha \in [0, 1]$ back to the previously followed border (see Fig. 11). In contrast to $h_{cross,i}(\alpha)$, $\tilde{h}_{cross,i}(\alpha)$ is perpendicular to the opposite lane bound ($\tilde{g}_i^T h'_{right} = 0$).
4) Given the corresponding bounds $b_{cross,i}$ and $\tilde{b}_{cross,i}$ of the line segments $h_{cross,i}$ and $\tilde{h}_{cross,i}$, we define two types of regions: the regions $R_i = \{[s_x, s_y]^T | b_{cross,i}(s_x, s_y) < 0, \tilde{b}_{cross,i-1}(s_x, s_y) < 0\} \cap \mathcal{O}_{segment}$ without inner inflection point and the intermediate regions $IR_i = \{[s_x, s_y]^T | \tilde{b}_{cross,i}(s_x, s_y) < 0, b_{cross,i}(s_x, s_y) < 0\} \cap \mathcal{O}_{segment}$, where one border only consists of the single point μ_i.
5) The bound to which μ_i belongs is followed until another inflection point is reached and the same procedure is repeated until the final border is reached ($b_{end}(s_x, s_y) = 0$). □

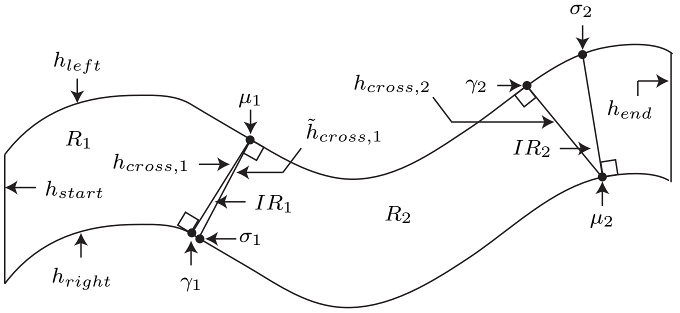


Fig. 11: Inflection point segmentation of a lane.

Let us introduce the j^{th} inner paths $h_{in,j}(\xi)$ of an inflection-point segmentation, where $\xi \in [0, \xi_{max,j}[$, $h_{in,j}(0) = \mu_{i-1}$, and $\xi_{max,j}$ is defined as the value that ensures $h_{in,j}(\xi_{max,j}) = \gamma_i$. The inner path $h_{in,j}(\xi)$ is perpendicular to $\tilde{h}_{cross,j-1}$ at $\xi = 0$ and perpendicular to $h_{cross,j}$ at $\xi = \xi_{max,j}$. This is not necessarily ensured for the first and last partial inner path $h_{in,1}$ and $h_{in,e}$, touching h_{start} and h_{end}, respectively. To also ensure orthogonality in those cases, we construct the auxiliary first and last inner bound $h_{in,init}(\xi)$ and $h_{in,final}(\xi)$ as shown in Fig. 12, which are perpendicular to h_{start}, h_{end} and touch $h_{in,1}(\xi)$, $h_{in,e}(\xi)$.

Finally, we define the partially-connected compound path from partial inner paths $h_{in,j}(\xi)$ as

$$h(\xi) = \begin{cases} h_{in,init}(\xi), & \xi \in [0, \tilde{\xi}_0[, \\ h_{in,1}(\xi), & \xi \in [\tilde{\xi}_0, \tilde{\xi}_1[, \\ h_{in,2}(\xi - \tilde{\xi}_1), & \xi \in [\tilde{\xi}_1, \tilde{\xi}_2[, \\ \vdots \\ h_{in,final}(\xi - \tilde{\xi}_{e-1}), & \xi \in [\tilde{\xi}_{e-1}, \tilde{\xi}_e]. \end{cases} \tag{10}$$

Given the front path variable ξ_f, one has to evaluate the compound path (10) to obtain $[s_{x,f}, s_{y,f}]^T = h(\xi_f)$. From this point, the front bound is constructed as described in Lemma 3.

Theorem 2 (Occupancy along lane): Let us introduce by $j_\xi(t)$ the index of the region corresponding to $\xi(t)$, i.e. $\xi(t) \in [\tilde{\xi}_{j-1}, \tilde{\xi}_j[$ in (10). We further introduce the occupancy within the j_ξ^{th} inflection-point region obtained from Lemma 3 as $\mathcal{O}_{2,j_\xi}(t)$. Thus,

$$\mathcal{O}_2(t) = \Big(\bigcup_{j=0}^{j_\xi(t)-1} R_j\Big) \cup \Big(\bigcup_{j=0}^{j_\xi(t)-1} IR_j\Big) \cup \mathcal{O}_{2,j_\xi}(t). \quad \square$$

Proof: The result follows directly from Fig. 11, Fig. 12, and Lemma 3. ■

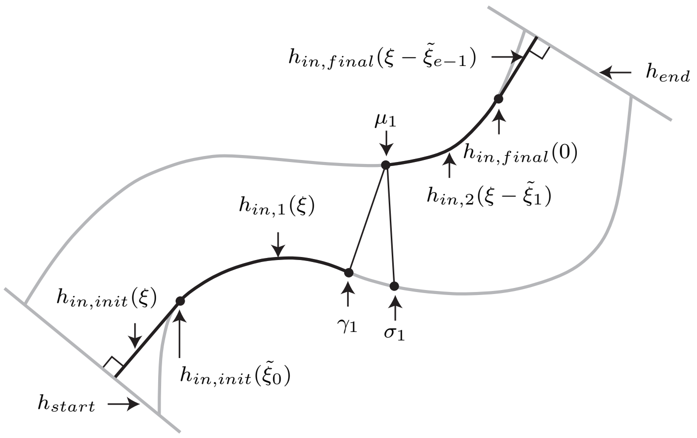

$h_{in,final}(\xi - \tilde{\xi}_{e-1})$
h_{end}
μ_1
$h_{in,final}(0)$
$h_{in,2}(\xi - \tilde{\xi}_1)$
$h_{in,1}(\xi)$
$h_{in,init}(\xi)$
γ_1
σ_1
$h_{in,init}(\tilde{\xi}_0)$
h_{start}

Fig. 12: Inner paths of a lane segment.

The occupancy is not only computed for the current lane, but also for neighboring lanes if a lane change is possible according to the adjacency graph of the lanelets.

V. Numerical Experiments

We demonstrate our proposed approach for different multi-lane road networks involving road forks. Additionally, we compare our results to results obtained from a high-fidelity model. The behaviors of the high-fidelity model are generated by using rapidly exploring random trees (RRTs) [67]. To validate our approach, we use real-world measurements from US highway 101 and check whether all the recorded data is enclosed by our set-based occupancy prediction. Finally, we demonstrate how our set-based occupancy prediction can be integrated into a trajectory planning algorithm.

All results are obtained using MATLAB 2015a on a machine with a 2.2 GHz Intel Core i7 processor with 16 GB 1600 MHz DDR3 memory. For simplicity we use the same values of the required vehicle parameters listed in Tab. I for all predictions: $a_{max} = 10$ m/s^2 (obtained from friction coefficient $\mu = 1.02$ and $g = 9.81$ m/s^2; see [68, Fig. 3.3]), $v_{max} = 30$ m/s (assuming US highway speed limit of 65 mph plus overspeeding), $v_S = 10$ m/s, $w = 1.8$ m, $l = 4.2$ m.

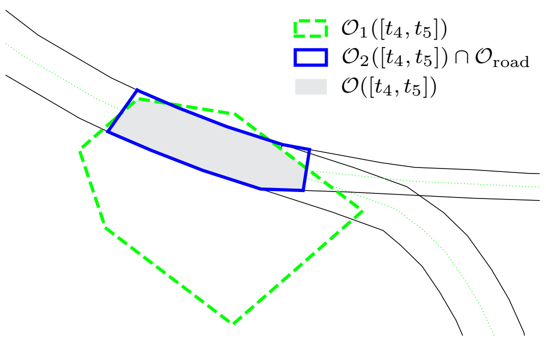
A. Multi-Lane Road Networks Involving Road Forks

We consider two different scenarios involving road forks. Both scenarios are created from real roads modeled in *OpenStreetMap*, which we have processed with *JavaOpenStreetMap*(JOSM)[4]. For both scenarios we use a time step size of $\Delta t = t_{i+1} - t_i = 0.5$ s, a time horizon of $t_h = 3$ s, and an initial velocity of $v(0) = 25$ m/s. Please note that we do not miss any occupancy, since we compute the occupancy for consecutive time intervals as explained in Sec. IV.

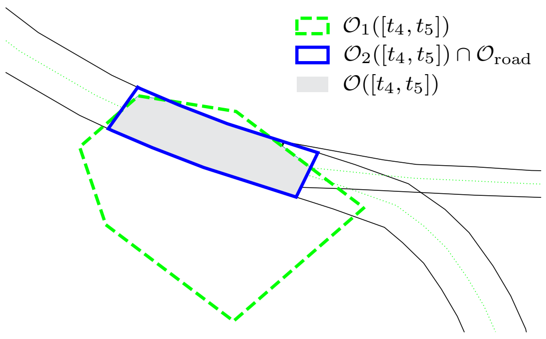
The first scenario is shown in Fig. 13, where the considered vehicle can follow two possible lanes. To illustrate the underlying computations, we separately plot the occupancy for going left (Fig. 13(a)) and right (Fig. 13(b)) for the time interval $[t_4, t_5]$. The occupancy $\mathcal{O}([t_0, t_f])$ of the entire time horizon for both lane options is presented in Fig. 13(c).

The second scenario consists of three lanes in one direction and one lane in the opposite direction as shown in Fig. 14, where the considered vehicle is initially in the center lane; hence, a lane change towards both neighboring lanes is possible. Fig. 14 shows the union of all occupied regions for all time intervals until the prediction horizon of $t_h = 3$ s.

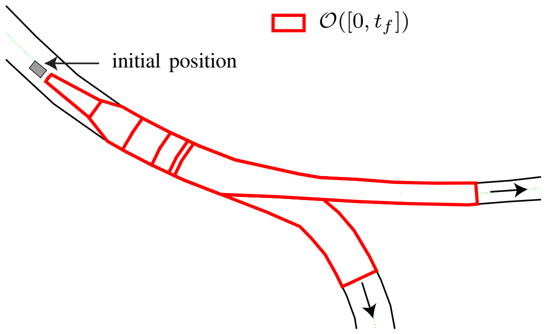
The computation times of both scenarios are shown in Tab. II. It can be seen that the computations only require a fraction of the prediction horizon (less than 10 %) and thus make it possible to use our prediction online for planning and verifying maneuvers as illustrated in Fig. 2. Since the prediction of each vehicle can be performed independently, one can easily parallelize the prediction and thus compute in similar time the future occupancy of many surrounding vehicles.


$\mathcal{O}_1([t_4, t_5])$
$\mathcal{O}_2([t_4, t_5]) \cap \mathcal{O}_{\text{road}}$
$\mathcal{O}([t_4, t_5])$

(a) Occupancy for left turn and $t \in [t_4, t_5]$.


$\mathcal{O}_1([t_4, t_5])$
$\mathcal{O}_2([t_4, t_5]) \cap \mathcal{O}_{\text{road}}$
$\mathcal{O}([t_4, t_5])$

(b) Occupancy for right turn and $t \in [t_4, t_5]$.



(c) Overall occupancy of the entire prediction horizon

Fig. 13: Occupancy prediction of scenario I.

[4]https://josm.openstreetmap.de

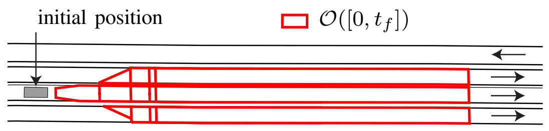


Fig. 14: Occupancy prediction of scenario II.

TABLE II: Computation Time for Scenario I and II.

	Computation time	**Prediction horizon**	**Fraction of prediction horizon**
Scenario I	0.19 s	3 s	6.3 %
Scenario II	0.17 s	3 s	5.7 %

TABLE III: Initial values of the high-order model (see [72]).

sprung mass		**unsprung mass**		**other**	
name	**init. val.**	**name**	**init. val.**	**name**	**init. val.**
yaw ang.	Ψ_0	roll ang. (f)	0	wheel speed (lf)	ω_0
yaw rate	$\dot{\Psi}_0$	roll rate (f)	0	wheel speed (rf)	ω_0
roll angle	0	roll ang. (r)	0	wheel speed (lr)	ω_0
roll rate	0	roll rate (r)	0	wheel speed (rr)	ω_0
pitch ang.	0	y-vel. (f)	$v_{yf,0}$	pin joint diff. (f)	0
pitch rate	0	y-vel. (r)	$v_{yr,0}$	pin joint diff. (r)	0
x-velocity	$v_{x,0}$	z-pos. (f)	$z_{f,0}$	x-position	$s_{x,0}$
y-velocity	$v_{y,0}$	z-vel. (f)	0	y-position	$s_{y,0}$
z-position	0	z-pos. (r)	$z_{r,0}$		
z-velocity	0	z-vel. (r)	0		

B. Comparison with a High-Fidelity Vehicle Model

For the computation of our occupancy sets, the constraints $C1-C5$ in Sec. III-B as well as Prop. 1, Lemma 1, 2, 3 and Theorem 1, 2, ensure that the predicted occupancies are over-approximative. The remaining question is how conservative our results are. To investigate this issue, we introduce a high-fidelity vehicle model to compute possible behaviors using rapidly exploring random trees (RRTs) [67]. RRTs have proven useful for computing subsets of reachable sets [69] and are more effective compared to Monte Carlo simulation when diversity of solutions is important, as has been demonstrated in e.g. [70]. We would like to stress that RRTs are no formal method so that the comparison with our set-based prediction does not provide a proof for the over-approximative computation.

Our high-order model considers the vertical load of all 4 wheels due to roll, pitch, and yaw, their individual spin and slip, and nonlinear tire dynamics. The multi-body dynamics is described by 3 masses: The unsprung mass and the sprung mass of the front and rear axles. The forces between these masses are described by the dynamics of the suspension and the tire model. Our model is taken from [71, Appendix A] and has been used in a previous publication of the authors [72]. We use the vehicle parameters from vehicle 14 in [71, Appendix E], which is a BMW 320i. For the tire dynamics we use the PAC2002 Magic-Formula tire model whose parameters are taken from the example of a PAC2002 tire property file in [73]. We initialize the high-order model as summarized in Tab. III, where f/r indicates front/rear. We further introduce the initial heading Ψ_0, the initial yaw rate $\dot{\Psi}_0$, the initial rotational speed of the wheels $\omega_0 = v_0/R$ (v_0: initial velocity, R radius of the wheel), the initial velocity in x/y-direction of the vehicle coordinate system $v_{x,0}/v_{y,0}$, the initial height over ground z_0, and the initial x/y-position $s_{x,0}/s_{y,0}$.

In contrast to standard RRT approaches, such as [69], we slightly modify the procedure to account for the fact that we are interested in the results for consecutive time intervals and thus aim at a constant sampling density for each time interval, see Fig. 15. The used approach is taken from [72]:

1) Initialize the discrete set of states for the next time interval as $\mathcal{X}(\tau_{k+1}) = \emptyset$.
2) Generate a sample x_s from the state space.
3) Find the nearest state x_n according to a distance measure ρ so that $x_n = \arg\min(\rho(x_s, x_i))$, where $x_i \in \mathcal{X}(\tau_k)$.
4) Obtain the input u which drives x_n to the new state x_{add} closest to x_s.
5) Add x_{add} to the set of states for the next time interval $\mathcal{X}(\tau_{k+1})$.
6) Repeat steps 2-5 for a predefined number of samples, then go to the next time interval and start with step 1.

When initializing $\mathcal{X}(\tau_{k+1}) = \mathcal{X}(\tau_k)$, one obtains the approach in [69]. The distance measure is simply chosen as the Euclidean distance, and we only consider x-position and y-position for the sampling space $\mathcal{X}$. The optimal input u consisting of the steering angle and the brake/acceleration pedal angle, which drives x_n to x_s, i.e., minimizes $\rho(x_{add}, x_s)$, is chosen by testing 12 combinations of steering and acceleration so that the resulting acceleration lies on Kamm's circle. Inputs are changed every 0.25 s, and each time interval $\mathcal{X}(\tau_{k+1})$ contains 100 samples. We use different initial states compared to the previous subsection (Sec. V-A) to show how it affects the prediction. The initial velocity is 10 m/s instead of 25 m/s, and the initial position is closer to the road fork.

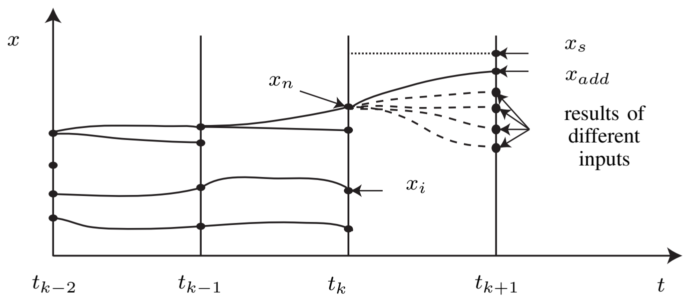
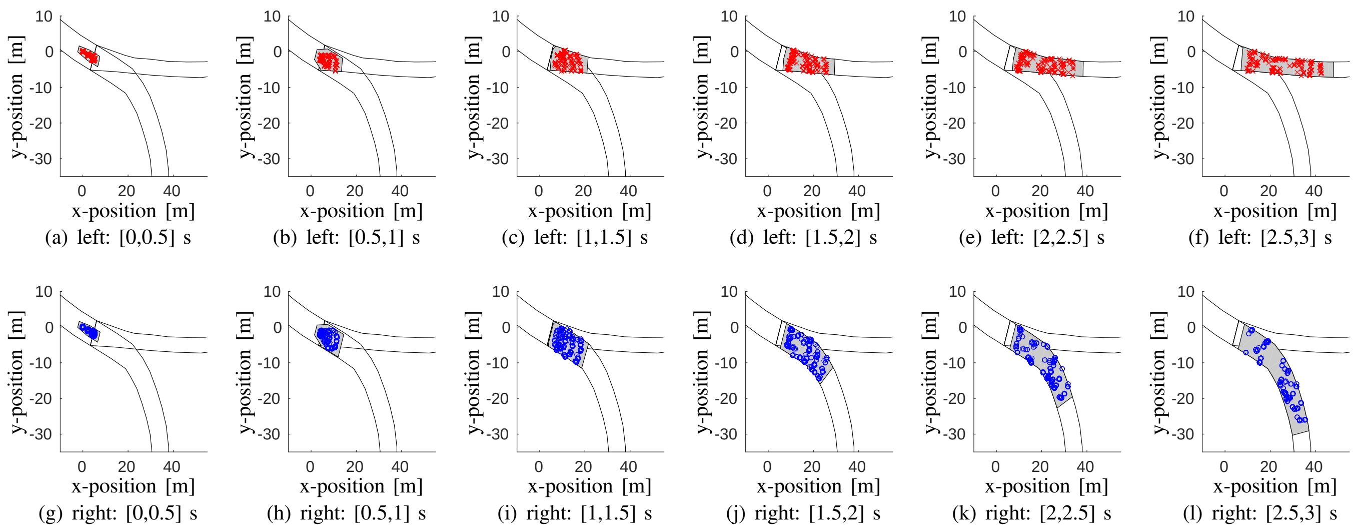


Fig. 15: Sampling procedure of our RRT approach.

The results are illustrated in Fig. 16. It can be seen that for each time interval, a high-fidelity vehicle model can drive close to the borders of the predicted occupancy. The case for turning left and right is shown separately in Fig. 16. Please note that computing the RRT of high-fidelity models is not an alternative for online computations since the prediction using our model takes around 10 hours in MATLAB on the same machine used for the occupancy prediction, which in contrast produces results only within a fraction of a second.



(a) left: [0,0.5] s (b) left: [0.5,1] s (c) left: [1,1.5] s (d) left: [1.5,2] s (e) left: [2,2.5] s (f) left: [2.5,3] s

(g) right: [0,0.5] s (h) right: [0.5,1] s (i) right: [1,1.5] s (j) right: [1.5,2] s (k) right: [2,2.5] s (l) right: [2.5,3] s

Fig. 16: Comparison of set-based occupancy prediction with results from the RRT computation for each time interval. Crosses show possible positions of the high-fidelity model for the left turn and dots show possible positions for the right turn. Gray regions indicate the predicted occupancy.

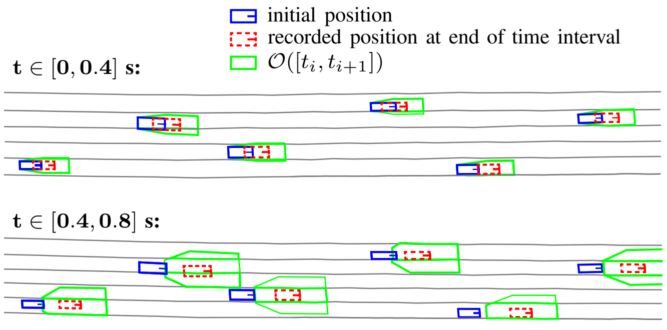
C. Comparison with Real Traffic Data

We evaluate the presented approach on data recorded from a real highway by showing that the tracked occupancy of each vehicle lies inside our predicted over-approximative occupancy sets. For the validation we use a time step size of $\Delta t = t_{i+1} - t_i = 0.4$ s and a time horizon of $t_h = 2$ s. Let us denote the time of the i^{th} recorded set of occupancies as $t_{rec,i}$. Starting from the occupancies at $t_{rec,i}$, we predict the occupied sets for the time interval $[t_{rec,i}, t_{rec,i} + t_h]$ and check if the recorded occupancies are enclosed. If the enclosure is successful, we proceed with the next time interval $[t_{rec,i+1}, t_{rec,i+1} + t_h]$ until all times $t_{rec,i}$ have been checked.

The dataset is provided by the Next Generation Simulation (NGSIM) program[5]. The traffic data was collected on a 0.6 km segment of US highway 101 (Hollywood Freeway) located in Los Angeles, California, on June 15th, 2005, see Fig. 17. The dataset contains detailed trajectory data for each vehicle for the entire time period from 07:50 am to 8:35 am. The vehicles were tracked using eight video cameras on different buildings.

For illustration purposes, we only present the results of a sector of the recorded highway section in Fig. 18. Otherwise, the prediction of individual cars would not be visible due to the length of the recorded highway section. However, the occupancy prediction has been performed for all vehicles. The initial occupancy of each vehicle is shown in Fig. 18 by solid rectangles with an inner line to indicate the driving direction. The recorded occupancy of each vehicle at the end of the time horizon ($t = t_{rec,i} + t_h$) is represented by dashed rectangles (see Fig. 18). The predicted occupancy is shown as polygons with a solid border.

The average time for computing the occupancy prediction is 0.2 s per vehicle for 5 instead of 6 time intervals as in scenarios I and II, and thus is only slightly slower than the results of Tab. II. In total we have considered 1074 vehicles and none of the recorded vehicles has breached the predicted occupancy.


initial position
recorded position at end of time interval
$\mathcal{O}([t_i, t_{i+1}])$
$\mathbf{t} \in [\mathbf{0}, \mathbf{0.4}]$ s:
$\mathbf{t} \in [\mathbf{0.4}, \mathbf{0.8}]$ s:
Fig. 18: Occupancy prediction using recorded vehicles from US highway 101.

D. Integration of Occupancy Prediction in Motion Planning

In this subsection we demonstrate how our occupancy prediction sets can be embedded into a motion planning problem. As described in Sec. II, we continuously plan a fail-safe motion accounting for all possible behaviors of other traffic participants (captured by our set-based prediction) to guarantee collision avoidance. In this demonstration, we use standstill as a safe state and thus as the final state of the fail-safe trajectory.

The basis of our scenario is from the previously used US 101 dataset with two surrounding vehicles (vehicle #1 and vehicle #2) and the ego vehicle as shown in Fig. 19. The initial velocity of vehicle #1 is 12.0 m/s, that of vehicle #2 is 15.9 m/s, and that of the ego vehicle is 20 m/s. The time horizons considered for computing the occupancy prediction and the long-term trajectory as illustrated in Fig. 3 are $t_{short} = 0.5$ s, $t_{long} = 5$ s. The simulation of the whole situation with continuous updates of the occupancy prediction is performed for 10 s. Our intended trajectories consist of the first 0.2 s of the long-term trajectory followed by the planned

[5]http://www.fhwa.dot.gov/

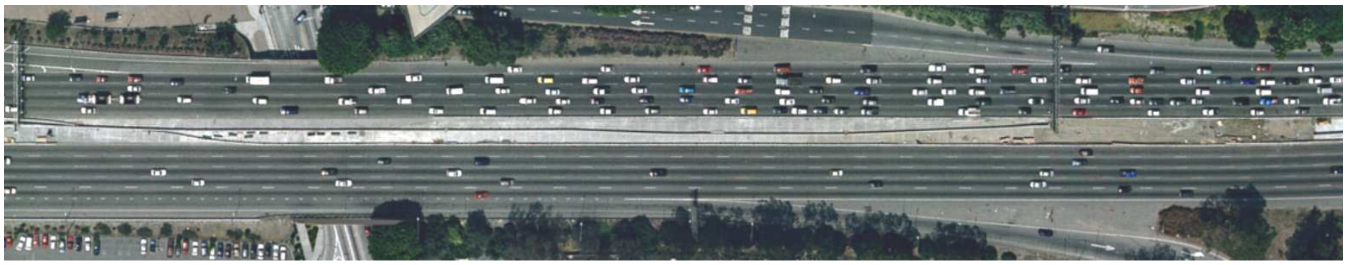


Fig. 17: Aerial photo of US highway 101 section.

fail-safe trajectory. Both the long-term trajectory and fail-safe trajectory are computed utilizing optimal control in this work, while any other choice of trajectory planner would also work. Our objective function for the long-term plan penalizes the distance to the center of the lane and the variation of the yaw angle, whereas the objective function for the fail-safe trajectory penalizes the velocity.

In order to create a more challenging scenario, we partially modified the recorded motion of vehicle #1, while the recorded motion of vehicle #2 is unchanged. The first modification occurs at time step 2 when vehicle #1 suddenly steers towards the left lane and quickly returns to its original lane (see Fig. 19). However, the verification procedure of the ego vehicle determines that the long-term trajectory continues to be safe so that it is continued. At time step 8 the second modification occurs: vehicle #1 again steers towards the left lane and quickly returns to its original lane, causing a dangerous situation. This time, the ego vehicle has to execute its fail-safe trajectory, which avoids a potential collision. The fail-safe trajectory is executed until a new safe trajectory is generated for the ego vehicle. Since Fig. 19. only illustrates the position over time, we also provide the velocity profile in Fig. 20. This demonstration shows the benefit of keeping a fail-safe trajectory available that is based on a set-based prediction of other traffic participants.

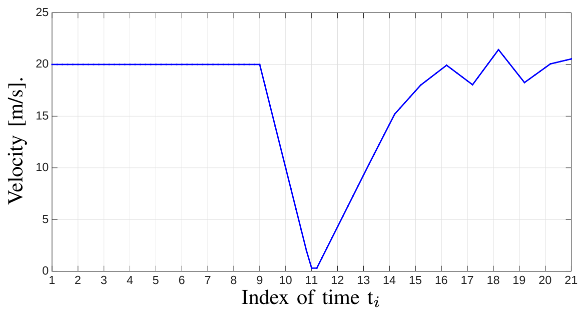


Fig. 20: Velocity of the ego vehicle for the planned motion in Fig. 19.

VI. Conclusions

We present a novel framework for automatically computing the occupancy prediction of other traffic participants on arbitrary road networks. In contrast to existing work in this area, our approach formally guarantees the bounds of occupied regions using techniques from reachability analysis. If we detect that a certain constraint in the model of traffic participants is violated by a traffic participant, we can remove the corresponding constraint from the model. As a consequence, the predicted occupancy grows faster and thus the ego vehicle has to plan more conservative trajectories to avoid the enlarged occupancy of that traffic participant. This reveals that our approach is inherently safe in the sense that forgetting constraints of the model results in safer behavior of the ego vehicle and not in less safe behavior. Please note that this property can only be achieved by set-based techniques, since the occupied region obtained with other techniques, such as multiple simulations, is not monotonically growing with respect to removing constraints.

The feasibility of our approach is demonstrated by real traffic data, collected from a segment of US highway 101. The validation results show the over-approximative nature of our approach and confirm that predictions are computed in less than 10 % of the considered time horizon, which makes the proposed approach applicable for online planning and verification. We have further demonstrated that our results are tight by comparing results with a high-fidelity model and that our approach can be easily integrated into motion planning algorithms.

Acknowledgment

The authors would like to gratefully acknowledge financial support by the German Research Foundation (DFG) grants *Graduiertenkolleg 1480 (PUMA)* and *AL 1185/3-1*.

References

[1] S. Mitra, T. Wongpiromsarn, and R. M. Murray, “Verifying cyber-physical interactions in safety-critical systems,” *IEEE Security and Privacy*, vol. 11, no. 4, pp. 28–37, 2013.
[2] R. Kianfar, P. Falcone, and J. Fredriksson, “Safety verification of automated driving systems,” *IEEE Intelligent Transportation Systems Magazine*, vol. 5, pp. 73–86, 2013.
[3] 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.
[4] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[5] I. Dagli and D. Reichardt, “Motivation-based approach to behavior prediction,” in *Proc. of the Intelligent Vehicles Symposium*, 2002, pp. 227–233.
[6] M. Liebner, C. Ruhhammer, F. Klanner, and C. Stiller, “Generic driver intent inference based on parametric models,” in *Proc. of the 16th International IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 268–275.
[7] C.-F. Lin, A. G. Ulsoy, and D. J. LeBlanc, “Vehicle dynamics and external disturbance estimation for vehicle path prediction,” *IEEE Transactions on Control Systems Technology*, vol. 8, pp. 508–518, 2000.
[8] Y. U. Yim and S.-Y. Oh, “Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction,” *IEEE Transactions on Vehicular Technology*, vol. 53, pp. 1076–1084, 2004.

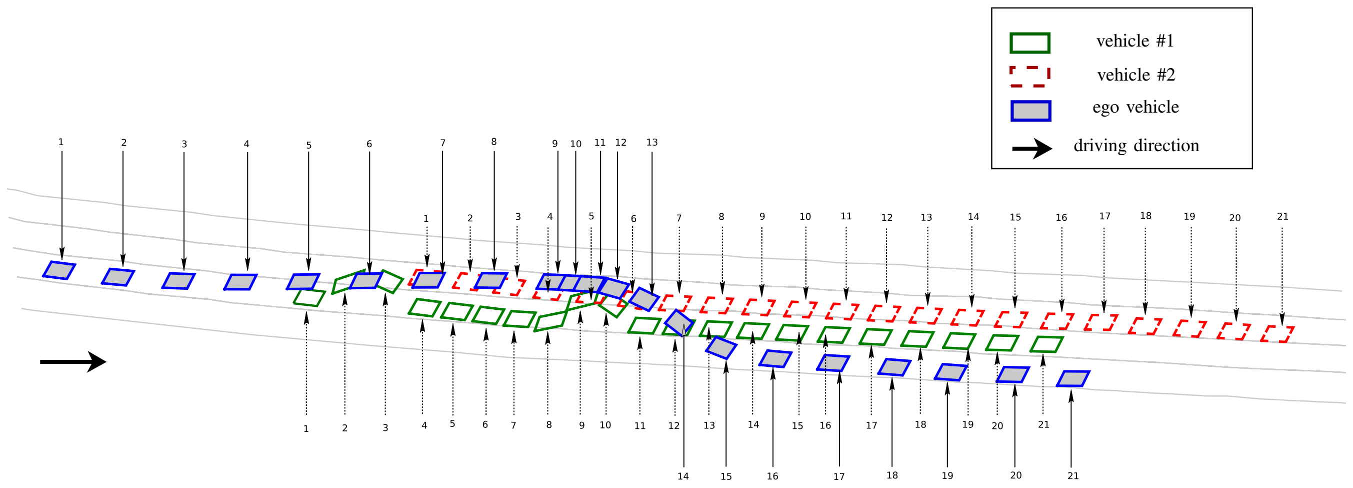


Fig. 19: Fail-safe motion planning using predicted occupancy sets; numbers represent the time steps t_i.

[9] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 549–562, 2007.

[10] B. Kim and K. Yi, "Probabilistic and holistic prediction of vehicle states using sensor fusion for application to integrated vehicle safety systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2178–2190, 2014.

[11] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1450–1464, 2006.

[12] N. Sumpter and A. J. Bulpitt, "Learning spatio-temporal patterns for predicting object behaviour," *Image and Vision Computing*, vol. 18, pp. 679–704, 2000.

[13] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden Markov models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 403–416, 2009.

[14] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH journal*, vol. 1, no. 1, pp. 1–14, 2014.

[15] A. Barth and U. Franke, "Where will the oncoming vehicle be the next second?" in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2008, pp. 1068–1073.

[16] A. Eidehall, "Multi-target threat assessment for automotive applications," in *Proc. of the 14th Int. IEEE Conference on Intelligent Transportation Systems*, 2011, pp. 433–438.

[17] M. Brännström, E. Coelingh, and J. Sjöberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 3, pp. 658–669, 2010.

[18] J.-H. Kim and D.-S. Kum, "Threat prediction algorithm based on local path candidates and surrounding vehicle trajectory predictions for automated driving vehicles," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2015, pp. 1220 – 1225.

[19] J. Wei, J. M. Snider, T. Gu, J. M. Dolan, and B. Litkouhi, "A behavioral planning framework for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, June 2014.

[20] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis & Prevention*, vol. 35, pp. 427–433, 2003.

[21] M. M. Minderhoud and P. H. L. Bovy, "Extended time-to-collision measures for road traffic safety assessment," *Accident Analysis & Prevention*, vol. 33, pp. 89–97, 2001.

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

[23] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 678–687, 2009.

[24] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, pp. 137–147, 2008.

[25] S. Danielsson, L. Petersson, and A. Eidehall, "Monte Carlo based threat assessment: Analysis and improvements," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2007, pp. 233–238.

[26] A. E. Broadhurst, S. Baker, and T. Kanade, "Monte Carlo road safety reasoning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2005, pp. 319–324.

[27] T. Kim and H.-Y. Jeong, "A novel algorithm for crash detection under general road scenes using crash probabilities and an interactive multiple model particle filter," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 6, pp. 2480–2490, 2014.

[28] C. Hermes, C. Wöhler, K. Schenk, and F. Kummert, "Long-term vehicle motion prediction," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 652–657.

[29] T. Gindele, S. Brechtel, and R. Dillmann, "Learning driver behavior models from traffic observations for decision making and planning," *IEEE Intelligent Transportation Systems Magazine*, vol. 7, no. 1, pp. 69–79, 2015.

[30] A. Lambert, D. Gruyer, G. S. Pierre, and A. N. Ndjeng, "Collision probability assessment for speed control," in *Proc. of the 11th International IEEE Conference on Intelligent Transportation Systems*, 2008, pp. 1043–1048.

[31] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299 – 310, 2009.

[32] M. Althoff and A. Mergel, "Comparison of Markov chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 1237–1247, 2011.

[33] B. Vanholme, D. Gruyer, B. Lusetti, S. Glaser, and S. Mammar, "Highly automated driving on highways based on legal safety," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 333–347, 2013.

[34] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," *Autonomous Robots*, vol. 32, no. 3, pp. 303–331, 2012.

[35] A. Chutinan and B. H. Krogh, "Computational techniques for hybrid system verification," *IEEE Transactions on Automatic Control*, vol. 48, no. 1, pp. 64–75, 2003.

[36] 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.

[37] M. Althoff and B. H. Krogh, "Zonotope bundles for the efficient computation of reachable sets," in *Proc. of the 50th IEEE Conference on Decision and Control*, 2011, pp. 6814–6821.

[38] T. Dang and O. Maler, "Reachability analysis via face lifting," in *Hybrid Systems: Computation and Control*, 1998, pp. 96–109.

[39] A. B. Kurzhanski and P. Varaiya, "Ellipsoidal techniques for reachability analysis," in *Hybrid Systems: Computation and Control*, ser. LNCS 1790. Springer, 2000, pp. 202–214.

[40] A. Girard and C. Le Guernic, "Efficient reachability analysis for linear systems using support functions," in *Proc. of the 17th IFAC World Congress*, 2008, pp. 8966–8971.

[41] O. Stursberg and B. H. Krogh, "Efficient representation and computation of reachable sets for hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS 2623. Springer, 2003, pp. 482–497.

[42] T. A. Henzinger, B. Horowitz, R. Majumdar, and H. Wong-Toi, "Beyond HyTech: Hybrid systems analysis using interval numerical methods," in *Hybrid Systems: Computation and Control*, ser. LNCS 1790. Springer, 2000, pp. 130–144.

[43] 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.

[44] S. Söntges and M. Althoff, "Determining the nonexistence of evasive trajectories for collision avoidance systems," in *Proc. of the 18th IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 956 – 961.

[45] D. Greene, J. Liu, J. Reich, Y. Hirokawa, A. Shinagawa, H. Ito, and T. Mikami, "An efficient computational architecture for a collision early-warning system for vehicles, pedestrians, and bicyclists," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 4, pp. 942–953, 2011.

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

[47] J. van den Berg, D. Ferguson, and J. Kuffner, "Anytime path planning and replanning in dynamic environments," in *Proc. of the International Conference on Robotics and Automation*, 2006, pp. 2366–2371.

[48] S. Bouraine, T. Fraichard, and H. Salhi, "Provably safe navigation for mobile robots with limited field-of-views in dynamic environments," *Autonomous Robots*, vol. 32, no. 3, pp. 267–283, 2012.

[49] A. Wu and J. P. How, "Guaranteed infinite horizon avoidance of unpredictable, dynamically constrained obstacles," *Autonomous Robots*, vol. 32, no. 3, pp. 227–242, 2012.

[50] C. F. Chung, T. Furukawa, and A. H. Göktogan, "Coordinated control for capturing a highly maneuverable evader using forward reachable sets," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2006, pp. 1336–1341.

[51] M. Althoff, D. Heß, and F. Gambert, "Road occupancy prediction of traffic participants," in *Proc. of the 16th International IEEE Conference on Intelligent Transportation Systems*, 2013, pp. 99–105.

[52] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2007, pp. 3068–3073.

[53] A. Platzer and E. M. Clarke, "The image computation problem in hybrid systems model checking," in *Hybrid Systems: Computation and Control*, ser. LNCS 4416. Springer, 2007, p. 473486.

[54] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2014, pp. 420–425.

[55] 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. Knöppel, 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.

[56] L. M. Surhone, M. T. Timpledon, and S. F. Marseken, Eds., *Vienna Convention on Road Traffic*. VDM Publishing, 2010.

[57] A. Rizaldi and M. Althoff, "Formalising traffic rules for accountability of autonomous vehicles," in *Proc. of the 18th IEEE International Conference on Intelligent Transportation Systems*, 2015, pp. 1658 – 1665.

[58] M. Althoff, D. Althoff, D. Wollherr, and M. Buss, "Safety verification of autonomous vehicles for coordinated evasive maneuvers," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 1078–1083.

[59] L. Segel, *The Physics of Tire Traction*. Springer, 1974, ch. Tire Traction on Dry, Uncontaminated Surfaces, pp. 65–98.

[60] R. Alur, C. Coucoubetis, N. Halbwachs, T. A. Henzinger, P. H. Ho, X. Nicolin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[61] M. K. Agoston, *Computer Graphics and Geometric Modeling: Implementation and Algorithms*. Springer, 2005.

[62] G. Greiner and K. Hormann, "Efficient clipping of arbitrary polygons," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 71–83, 1998.

[63] S. K. Ghosh and D. M. Mount, "An output-sensitive algorithm for computing visibility graphs," *SIAM Journal on Computing*, vol. 20, no. 5, pp. 888–910, 1991.

[64] J. Hershberger and S. Suri, "An optimal algorithm for Euclidean shortest paths in the plane," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 2215–2256, 1999.

[65] H.-P. Huang and S.-Y. Chung, "Dynamic visibility graph for path planning," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004, pp. 2813–2818.

[66] V. J. Lumelsky, "On fast computation of distance between line segments," *Information Processing Letters*, vol. 21, no. 2, pp. 55–61, 1985.

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

[68] C.-G. Wallman and H. Åström, "Friction measurement methods and the correlation between road friction and traffic safety," in *VTI meddelande*. Swedish National Road and Transport Research Institute, 2001.

[69] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, ser. LNCS 2993. Springer, 2004, pp. 142–156.

[70] D. Heß, M. Althoff, and T. Sattel, "Comparison of trajectory tracking controllers for emergency situations," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2013, pp. 163–170.

[71] R. W. Allen, H. T. Szostak, D. H. Klyde, T. J. Rosenthal, and K. J. Owens, "Vehicle dynamic stability and rollover," U.S. Department of Transportation, Final Report DOT HS 807 956, 1992.

[72] M. Althoff and J. M. Dolan, "Reachability computation of low-order models for the safety verification of high-order road vehicle models," in *Proc. of the American Control Conference*, 2012, pp. 3559–3566.

[73] *Adams/Tire help*, MSC Software, 2 MacArthur Place, Santa Ana, CA 92707, April 2011, documentation ID: DOC9805. [Online]. Available: http://simcompanion.mscsoftware.com/infocenter

Matthias Althoff is assistant professor in computer science at Technische Universität München, Germany. He received the diploma engineering degree in Mechanical Engineering in 2005, and the Ph.D. degree in Electrical Engineering in 2010, both from Technische Universität München, Germany. From 2010 to 2012 he was a postdoctoral researcher at Carnegie Mellon University, Pittsburgh, USA, and from 2012 to 2013 an assistant professor at Technische Universität Ilmenau, Germany. His research interests include formal verification of continuous and hybrid systems, reachability analysis, planning algorithms, nonlinear control, automated vehicles, and power systems.

Silvia Magdici is currently a Ph.D. candidate at Technische Universität München, Germany. She received her B.Sc. degree in Computer Engineering in 2012 and her M.Sc. degree in Systems and Control in 2014, both from Technical University of Iasi, Romania. She joined the Department of Informatics at Technische Universität München in 2015. Her research interests include autonomous vehicles, formal methods, reachability analysis, and optimal control.