Determining the Nonexistence of Evasive Trajectories for Collision Avoidance Systems

Sebastian Söntges and Matthias Althoff
Department of Computer Science
Technische Universität München
Boltzmannstraße 3, 85748 Garching, Germany
Email: soentges@in.tum.de

***Abstract*—It is of utmost importance for automatic collision avoidance systems to correctly evaluate the risk of a current situation and constantly decide, if and what kind of evasive maneuver must be initiated. Most motion planning algorithms find such maneuver by searching a deterministic or random subset of the state space or input space. These approaches can be designed to be complete in the sense that they converge to a feasible solution as sampling is made denser. However, they are not suitable to determine whether a solution exists. In this paper, we present an approach which overapproximates the reachable set of the host vehicle considering workspace obstacles. Thus, it provides an upper bound of the solution set and it can report if no solution exists. Furthermore, the calculated set can be used for guiding the search of an underlying planning algorithm to find a solution as each trajectory of the host vehicle is ensured to be contained within this set.**

I. Introduction

In recent years, the number and capabilities of advanced driver assistance systems and its implementation in production cars has steadily increased [1]. In addition to a gain in comfort by automatic distance or lane keeping, these systems may provide a tremendous increase in traffic safety beyond the conventional passive safety systems. However, since these collision avoidance or mitigation systems must take over control of the vehicle, they bear the risk to endanger passengers or other traffic participants. It must be guaranteed that the system intervenes only if neccessary and the consequences of an automatic maneuver is predictable. Based on an assessment of the current traffic situation, the decision to intervene the driver is made in case a collision is "almost inevitable". The notion of almost inevitable is ambiguous and a number of different criteria like time to collision or dynamical load have been proposed.

A common idea to assess a situation is to continuously determine if there is any collision-free trajectory or if there is none, and assess all the possible maneuvers found. Most motion planning algorithms applicable to cars cannot finally answer the question of the existence of such trajectories. Resolution complete lattice planners, which search a deterministic subset of the input space or the state space, find a feasible solution if the resolution is made sufficiently small [2]. Probabilistically complete randomized planners eventually converge to a solution as the number of samples is increased [2]. Both may find a solution to a planning problem in finite time, but if they do not, we cannot conclude about the existence or nonexistence of a solution. Complete algorithms like decomposition-based planners are difficult to apply since computational complexity restricts their use to low-dimensional problems with simple motion models.

In this paper, we address the problem of proving the nonexistence of any evasive trajectory using reachability analysis. The reachable set is defined as the set of all states that can be reached from an initial set of states at a given time and without a collision. Both underapproximation and overapproximation of this set are useful to a planner since they provide a lower and an upper bound of the existence and nonexistence of feasible trajectories. A nonempty underapproximated set indicates the existence of a feasible trajectory and may reduce false intervention. An empty overapproximated reachable set shows that a collision is inevitable. The system may intervene earlier to, at least, mitigate the collision. Since the overapproximation of the reachable set shows the upper bound of states that can be reached at a point in time, it can be used to identify possible goal states at the end of the planning horizon and guide the trajectory planner.

II. Related Work

To assess the severity of a traffic situation, a common approach is to evaluate a set of preselected or sampled evasive trajectories. A typical selection for evasion maneuvers are strong steering to the left and right, braking, or combined steering and braking maneuvers [3], [4]. By checking each maneuver for collisions, potential evasive trajectories are identified. They are evaluated by their dynamical properties like longitudinal and lateral acceleration, jerk, or steering angle rate. Based on one or more of these values, the decision to intervene is made if no maneuver or only maneuvers with uncommonly high dynamical load can avoid a collision. Other common quantities to assess the risk include e.g. variations of time to collision [5].

A low number of simple, but carefully selected maneuvers, have the advantage that they need little computational power and are readily deployed in current production cars. Due to their simplicity, they cannot deal with the diversity of more complex traffic situations [6]. More elaborate techniques must be applied to further increase the capabilities for evaluating situations with several dynamic objects and to plan appropiate trajectories considering follow-up collisions. A combinatorial search is used in [6], [7] which passes each obstacle either to the left or right. The resulting trajectories are built by concatenation of all combinations of passing maneuvers. In [6], [7] the problem is simplified by decoupling longitudinal and lateral motion. However, this can yield to non-drivable

trajectories. An advanced search on state lattices as presented in autonomous driving [8], [9] seems difficult to apply. In emergency situations we expect that an evasion trajectory is not necessarily aligned with a lane and will have to pass very narrow passages in the state space. This would require a prohibitively dense lattice. Also, model predictive control is proposed to plan trajectories and assess their risk [10]. Nevertheless, it needs a high-level planner which selects a homotopic path. This method is intrinsically unsuitable for choosing to pass an obstacle on one side or the other as both correspond to a local optimum. A comparison of the approaches can be found in [11].

If we only consider a subspace of all possible actions, the host vehicle might perform well in a wide range of scenarios. Yet, it does not provide any formal guarantuee that in some cases there is actually an evasive maneuver which just has not been found yet or a guarantee that definitely none exists. Furthermore, these methods are limited to a finite time horizon only. To formally verify safety in motion planning, the concept of Inevitable Collision State (ICS) was introducted in [12]. An ICS is defined as a state, which – whatever input is selected – eventually ends in a collision. Similar concepts under different names are known, e.g. [13]. Although the ICS concept is theoretically sound, it is difficult to determine the set of ICS efficiently, except for a number of simple scenarios and motion models. Even for a bicycle motion model in a moderately complex static scenario, it is hard to calculate the set of ICS in a timely manner [14]. Also, for all arbitrary time-dependent obstacles, technically, their future evolution must be known for an infinite time horizon, which is impossible for obvious reasons [15]. Approximations of ICS set may lead to similar problems as the evaluation of a subset of maneuvers as mentioned above.

In this work we present an algorithm which provides an upper bound of the set of all collision-free trajectories with bounded absolute acceleration. Rather than evaluating a number of single trajectories, we compute an overapproximation over the whole set of trajectories similar to the approach in [16]. Thus, we can provide a formal guarantee that no solution exists if the calculated set is empty. In contrast to the approach in [16], we use a four-dimensional state space, which considers both position and velocity, instead of only two-dimensional regions of position. Also, we do not use general polygons to describe the region, but a set composed of hyperrectangles. Each hyperrectangle has a simple rectangular shape in workspace, which can easily be checked for collision either in occupancy grids or with polygonal obstacles. Thus, we can seamlessly handle topological changes of the regions without the difficulties that may arise in polygonal representations.

III. Definitions and Problem Statement

The reachable set $\mathbf{reach}(\mathcal{R}_0, \Delta t)$ for an initial set of states $\mathcal{R}_0$ and time step Δt is defined as the set of all states that can be reached at time $t_0 + \Delta t$ for a feasible input $u \in \mathcal{U}$.

$$\mathbf{reach}(\mathcal{R}_0, \Delta t) = \left\{ s \;\middle|\; \exists u \in \mathcal{U}\ \exists s_0 \in \mathcal{R}_0, s = s_0 + \int_{t_0}^{t_0+\Delta t} f(s(t), u(t))\mathrm{d}t \right\}$$

This definition does not consider the restrictions imposed by the obstacles in the workspace. The obstacles are given by a time-varying set $\mathcal{O}(t)$. At all times t of the planning horizon, the occupied region of the vehicle $\mathcal{A}(s(t))$ in the workspace at state $s(t)$ must not intersect with $\mathcal{O}(t)$

$$\forall t: \ \mathcal{A}(s(t)) \ \cap \ \mathcal{O}(t) = \emptyset. \tag{1}$$

We denote by $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0, \Delta t)$ the subset of $\mathbf{reach}(\mathcal{R}_0, \Delta t)$ of states which can be reached without collision (i.e. its trajectory $s(t)$ fullfils (1) for all times $t \in [t_0,\ t_0 + \Delta t]$).

Even for rather simple motion models, the reachable set soon becomes too complicated to be calculated and represented efficiently. This complication arises from the obstacles, which forbid certain vehicle configurations. Therefore, a simplified motion model and overapproximations for the set representation are needed. Firstly, the set representation must be suitable for the chosen motion model. Secondly, it must be able to handle constraints of the plentiful shapes of obstacles in the workspace. In the proposed algorithm, we need efficient approximations for set union, intersection and complement operations. A common choice for the reachability of linear system are polyhedrons. We use more restricted hyperrectangles since they provide fast overapproximations for the set operations mentioned above.

A method to find useful approximations is to alter the motion model. Additional constraints can be added to the model to obtain an underapproximation or constraints can be relaxed to find an overapproximation. We describe the host vehicle dynamics by a two-dimensional double integrator in x- and the y-direction with bounded acceleration

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \tag{2}$$

$$\left\| \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} \right\|_\infty \leq a_0 \tag{3}$$

and state $s = [x, y, \dot{x}, \dot{y}]^T$. The occupied region of the car in the workspace is modeled by a disk $\mathcal{A}(s(t))$ with nonzero radius r_{car} of the inner circle of its footprint. The technical requirement of nonzero radius is pointed out in Sec. V-B. Clearly, this simple motion model does not account for many dynamical restrictions of vehicle motion. However, the limitation on acceleration is valid for all realistic vehicles. As mentioned above, relaxing constraints of a more accurate vehicle model yields an overapproximation of the reachable set.

In the following sections, we present an algorithm to calculate a conservative approximation of the reachable set $\mathbf{reach}_{\mathcal{O}}(s_0, t)$ for the simple motion model (2)-(3) under consideration of obstacles (1). If the calculated set is empty for a given planning horizon, it is guaranteed that no collision-free trajectory exists, neither for the double integrator model nor for any other more accurate but acceleration-bounded model. In the event that the set is nonempty, it cannot be said if there is a feasible trajectory, but if there exists any, it must be contained in $\mathbf{reach}_{\mathcal{O}}(s_0, t)$.

IV. Reachable Set with Constrained Positions

First, we consider the following one-dimensional problem. Given is the set of initial states at time t_0 which are contained

in the position interval $[x_0^{(L)},\ x_0^{(H)}]$ and the velocity interval $[v_0^{(L)},\ v_0^{(H)}]$. We determine the set of all possible velocities $[v_1^{(L)},\ v_1^{(H)}]$ that can be reached within a given position interval $[x_1^{(L)},\ x_1^{(H)}]$ at time $t_1 = t_0 + \Delta t$ for the system dynamics $\ddot{x} = u$ subject to the constraint $|u| \leq a_0$. We optimize over the inputs $u \in \mathcal{U}$ to obtain the maximum and minimum velocity at t_1.

The Pontryagin principle yields a bang-bang input candidate function with switching time γ:

$$u(t) = \begin{cases} -a, & t_0 \leq t < t_0 + \gamma \\ a & t_0 + \gamma \leq t \leq t_0 + \Delta t \end{cases}$$

for maximum velocity. To simplify the notation, we define $[d_L,\ d_H] = [x_1^{(L)} - x_0^{(H)},\ x_1^{(H)} - x_0^{(L)}]$, which is the maximum displacement to the left and to the right after Δt. The position $x_1 = x_0 + v_0\,\gamma - \frac{a\,\gamma^2}{2} + \frac{a\,(\Delta t - \gamma)^2}{2} + (v_0 - a\,\gamma)\ (\Delta t - \gamma)$ after time step Δt yields the set of possible switching times γ depending on v_0, x_0 and x_1. Using $v_1 = v_0 - a\,\gamma + a\ (\Delta t - \gamma)$ we obtain the optimization problem

$$\begin{aligned} \max_{d_x, v_0} \quad & -a\Delta t + v_0 + \sqrt{2a^2\Delta t^2 - 4v_0 a\Delta t + 4ad_x} \\ \text{subject to} \quad & v_l \leq v_0 \leq v_h, \\ & d_L \leq d_x \leq d_H, \\ & \frac{1}{2}a\Delta t^2 + v_0\Delta t \geq d_x, \\ & -\frac{1}{2}a\Delta t^2 + v_0\Delta t \leq d_x, \end{aligned}$$

where the latter two constraints follow from valid switching times. Similarly, to minimize the velocity we use the candidate function

$$u(t) = \begin{cases} a, & t_0 \leq t < t_0 + \gamma \\ -a & t_0 + \gamma \leq t \leq t_0 + \Delta t \end{cases}$$

and obtain

$$\begin{aligned} \min_{d_x, v_0} \quad & a\Delta t + v_0 - \sqrt{2a^2\Delta t^2 + 4v_0 a\Delta t - 4ad_x} \\ \text{subject to} \quad & v_l \leq v_0 \leq v_h, \\ & d_l \leq d_x \leq d_h, \\ & -\frac{1}{2}a\Delta t^2 + v_0\Delta t \leq d_x, \\ & \frac{1}{2}a\Delta t^2 + v_0\Delta t \geq d_x. \end{aligned}$$

The Hessian shows that the objective term in the maximization is convex in the interior of the feasible region, whereas the minimization term is concave in the interior of the feasible region. Thus, we have to search the boundary of the feasible region for the optimal value.

We use this result in Sec. V to find an overapproximation of all states that can be reached within a time step but are constrained by some obstacles to lie in a given position interval at t_1:

$$\begin{aligned} \mathbf{reach}([x_0^{(L)},\ x_0^{(H)}] \times [v_0^{(L)},\ v_0^{(H)}], \Delta t) \\ \cap\ [x_1^{(L)},\ x_1^{(H)}] \times [-\infty,\ \infty] \\ \subset [x_1^{(L)},\ x_1^{(H)}] \times [v_1^{(L)},\ v_1^{(H)}] \end{aligned}$$

1: **procedure** REACH($\mathcal{R}_0$)
2: **for** $i \leftarrow 1$ to steps **do**
3: $\mathcal{R}_i \leftarrow \emptyset$
4: **for all** $\text{box}_{(i-1)}$ in $\mathcal{R}_{i-1}$ **do**
5: $\text{box}_{(i)} \leftarrow$ PROPAGATE($\text{box}_{(i-1)}$)
6: $\mathcal{R}_i \leftarrow \{\mathcal{R}_i,$ SPLIT($\text{box}_{(i)}$)$\}$
7: **if** $|\mathcal{R}_i| >$ maximum number of elements **then**
8: $\mathcal{R}_i \leftarrow$ REPACK($\mathcal{R}_i$)
9: **end if**
10: **end for**
11: **end for**
12: **return** $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_{steps}$
13: **end procedure**

Fig. 1. Algorithm to determine $\mathbf{reach}_{\mathcal{O}}$

The one-dimensional problem can be readily extended to the two-dimensional case by considering the x- and y-axis separately. This is possible since by the infinity norm the acceleration constraint (3) applies to the x- and y-direction independently.

V. Algorithm

We now describe how we overapproximate the set $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0, \Delta t)$ introduced in Sec. III. It is assumed that a function, which provides collision checks of an axis-aligned rectangle with static and dynamic obstacles at any time step, is given. The function should report about any intersection with an obstacle and whether the rectangle is fully covered by the obstacle. In our implementation, we use summed area tables for checks with an occupancy grid [17], which facilitates constant time collision lookup, and the separated axis theorem for intersection checks with convex polygonial obstacles [18].

The main idea is to overapproximate the reachable set by a collection of sets of simple shapes. Each element of this collection is a four-dimensional hyperrectangle in state-space defined by an interval for each state variable $x, y, \dot{x}, \dot{y}$. In the following a hyperrectangular subspace at time t is interchangeably denoted as hyperrectangle, box or element r_t. The union of elements $\bigcup_i \mathsf{r}_t^{(i)}$ defines the approximated reachable set $\mathbf{reach}_{\mathcal{O}}$ at time t.

Fig. 1 shows the algorithm. Starting with an initial state s_0, the algorithm calculates at each time step the reachable set using the boxes from the previous time step (line 5). If the projection of a box into the position domain collides with an obstacle, it is replaced by a set of new noncolliding boxes (line 6). In most scenarios, this yields a rapidly increasing number of boxes due to frequent collisions. To keep the total number of elements tractable, the set of boxes is redistributed to a lower number of boxes covering the original set if a maximum number of elements is exceeded (lines 7-8). Each step is described in detail below.

Fig. 2. Before splitting (left) and after (right); green boxes denote the reachable region in the position domain at the current Δt; black rectangles denote obstacles

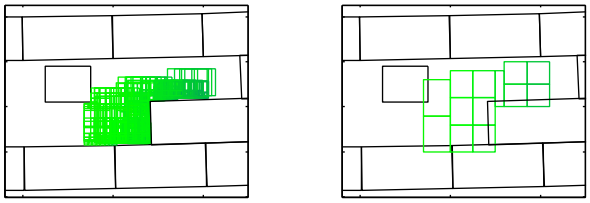


Fig. 3. Before repacking (left) and after (right); green boxes denote the reachable region in the position domain at the current Δt; black rectangles denote obstacles

A. Propagation

A hyperrectangle is propagated according to the integration of the transition equation (2):

$$
\begin{aligned}
x_{t+\Delta t}^{(L)} &= x_t^{(L)} + v_t^{(L)}\Delta t - \frac{1}{2}a\Delta t^2 \\
x_{t+\Delta t}^{(H)} &= x_t^{(H)} + v_t^{(H)}\Delta t + \frac{1}{2}a\Delta t^2 \\
v_{t+\Delta t}^{(L)} &= v_t^{(L)} - a\Delta t \\
v_{t+\Delta t}^{(H)} &= v_t^{(H)} + a\Delta t
\end{aligned}
$$

The y-direction is updated similarly. The superscript denotes the upper and lower limit of the hyperrectangle.

B. Splitting

Whenever the projection of a hyperrectangle intersects with an obstacle in the workspace, the box is replaced by a set of boxes with the following property: Any state in the original box is either contained in one of the new boxes or its distance to an obstacle is smaller than the radius of the inner circle of the robot footprint r_{car} (Fig. 2). Since, if the distance is smaller, the state collides and can be removed from the set. Using the inner circle of the robot footprint yields to an overapproximation of the reachable set.

We use a similar scheme as in quadtrees to split a box in the position domain [19]. The covered region is divided into four subregions until the subregion is collision free, fully covered by an obstacle, or colliding and its diagonal is less than r_{car}. The latter terminal condition guarantees that splitting is finite and each box has a minimum size. After the box is split in the position domain, we restrict the set of possible velocities using the result from Sec. IV.

C. Repacking

In common scenarios the number of elements increases in each step due to collisions. The purpose of the repacking step is to keep the number of elements tractable. A set of boxes is replaced by another set with a lower number of boxes which cover the original set. In general, this is at the expense of loosing accuracy. In our implementation we use a line-sweep algorithm and a segment tree to merge all boxes in the position domain to orthogonal polygons and split these polygons again to rectangles (Fig. 3) [20]. An orthogonal polygon is one whose edges are aligned to two orthogonal axes. From each of the newly created rectangles we create a new box and assign to it a velocity region by merging all velocities from intersecting original boxes.

Before repacking, the area in the spatial domain of each box is increased such that the boundary lies on a grid. This is needed as otherwise slightly misaligned (real-valued) region boundaries cannot be efficiently merged. As can been seen in Fig. 3, the new boxes may overlap with obstacles. This is admissible since we calculate an overapproximation.

VI. Evaluation

This section shows the output of the algorithm for a simulated scenario with two lanes. The host vehicle is driving behind vehicle A, which suddenly brakes. Vehicle B is driving on the second lane in the opposite direction and passes both vehicles at about the same time. The border of the road is modeled by a set of static obstacles (see Fig. 4).

We predict the future positions of other vehicles using a simplified approach from [21]. It is assumed that both vehicles stay on the lane but can arbitrarily accelerate or brake within given bounds. Using the currently perceived velocity, a worst case prediction is made by using a dynamical model with a maximum acceleration and braking capability of 10 m/s^2. These regions are marked in red. Note that the regions grow with time.

We choose a propagation time step of 150 ms, a maximum number of 200 elements before repacking, a maximum acceleration and braking capability of 9 m/s^2 for the host vehicle, a discretization grid before repacking of 1.0 m and the inner circle of the car of 1.0 m. The determination of $\mathbf{reach}_{\mathcal{O}}$ for a time horizon of 3 seconds (20 steps) took several seconds in our MATLAB implementation. The current implementation is not optimized. We expect that the runtime can be significantly reduced, particularly a large part of the algorithm can be easily parallelized.

Fig. 4 shows the resulting overapproximation of the reachable set for different Δt in position domain. The color represents the minimum velocity in x-direction (bright green for slower, dark green for faster). As can be seen, the algorithm implicitly covers all maneuvers such as braking or steering. The final set at $\Delta t = 3$ seconds shows that at most a braking maneuver exists. All possible goal states of an underlying trajectory planner must be within this region.

In Fig. 5, we compute $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t), 3\,\mathrm{s})$ for a three-second time horizon at different initial points in time $t = \{t_1, t_2, t_3\}$ of the scenario. By checking $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t), 3\,\mathrm{s})$ the algorithm shows that in Fig. 5b overtaking and braking

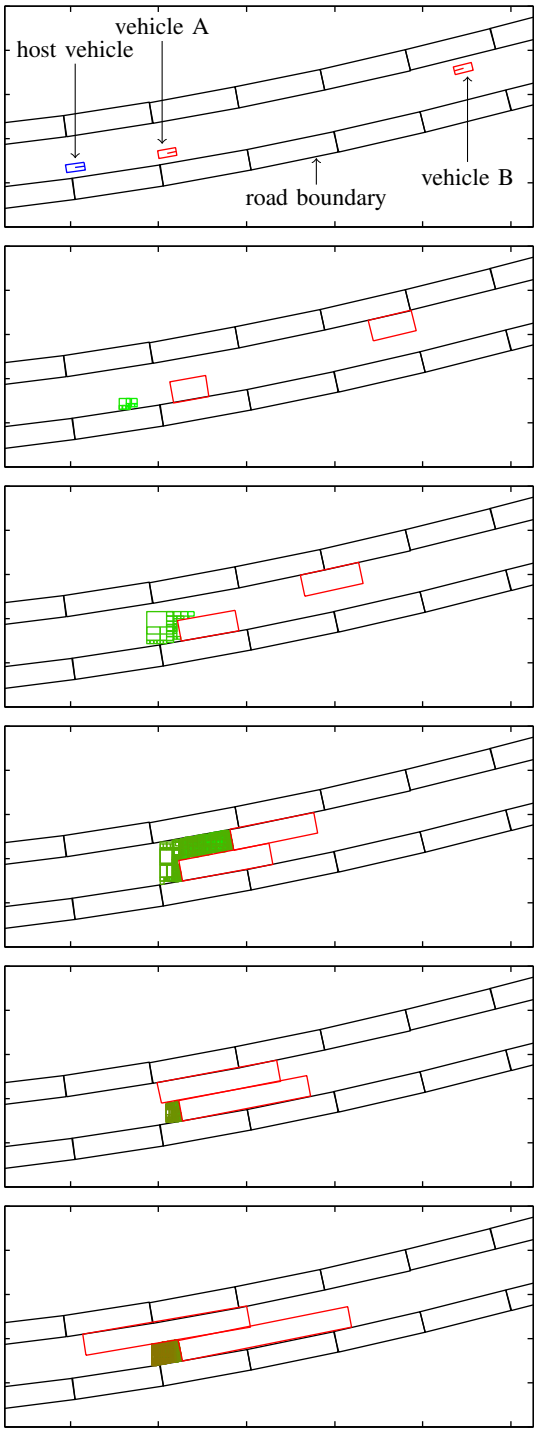
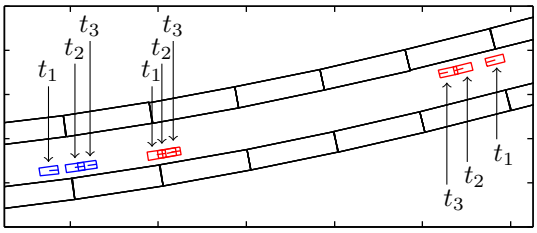
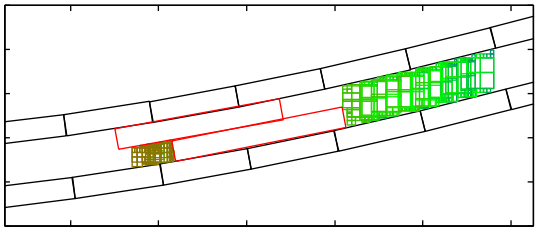


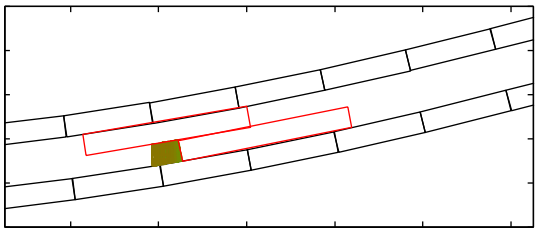
Fig. 4. Initial scenario and $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0, \Delta t)$ at different $\Delta t = 0.6$, 1.2, 1.8, 2.2, 2.6 and 3 seconds with two time-varying obstacles (red); the road boundary is modeled by a set of static obstacles (black)



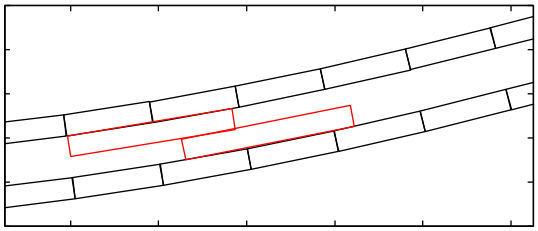
(a) initial scenario at t_1, t_2 and t_3



(b) t_1: $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t_1), 3\,\mathrm{s})$



(c) t_2: $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t_2), 3\,\mathrm{s})$



(d) t_3: $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t_3), 3\,\mathrm{s})$

Fig. 5. Scenario at three different initial times (a), and corresponding $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t), 3\,\mathrm{s})$. Either braking or overtaking possible (b); only braking possible (c); inevitable collision (d)

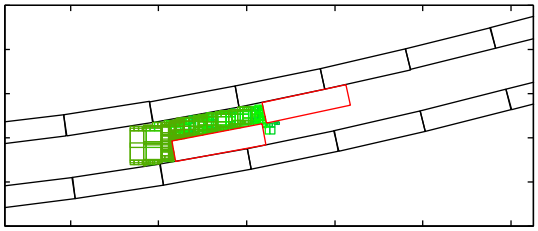


Fig. 6. t_1: $\mathbf{reach}_{\mathcal{O}}(\mathcal{R}_0(t_1), 1.8\,\mathrm{s})$; narrow passage that must be passed at $t_1 + 1.8\,\mathrm{s}$ for overtaking (cf. Fig. 5b)

may be possible; in Fig. 5c at most braking is possible; and in Fig. 5d it leads inevitably to a collision since the final set is empty. Accordingly, the system should not brake later than t_2 to avoid the collision, or at t_3 to at least mitigate the collision. Fig. 6 shows the narrow passage that must be passed for overtaking for initial time t_1. At $t_1 + 1.8\,\mathrm{s}$ the reachable region splits into two parts.

We make a short remark on the infinity norm in the acceleration constraint (3) of the motion model. The norm $\|\cdot\|_\infty$ simplifies the calculation of the velocities since the x- and y-direction can be considered independently. Also, the result depends on the chosen coordinate system as a rotation

of the x- and y-axes yields a slightly different result, which is nevertheless still a valid overapproximation. Restricting the acceleration by the $\|\cdot\|_2$ norm would be more realistic according to the friction limit given by Kamm's circle. In this norm, the reachable regions correspond to circles in the position domain instead of squares [16] and yield the same result independent of the chosen coordinate axis. However, first tests show that it is difficult to apply this norm in the splitting and repacking part of the presented algorithm (see Sec. V-C) where we need to efficiently replace a set of circles by another set of circles.

VII. Conclusion

An algorithm to overapproximate the set of all states that can be reached at a given time by an acceleration-bounded vehicle has been presented. The most common approach to assess the severity of a traffic situation and to find an emergency maneuver is by evaluating a set of possible evasive trajectories for a finite time horizon. By searching these candidates in the subspace of all trajectories, it may indicate the nonexistence of a collision-free trajectory but cannot finally guarantee it. As opposed to this, the proposed algorithm works on the whole set of trajectories. It cannot prove that there is a feasible trajectory, but it gives an upper bound and thus it can show if no feasible trajectory exists. Also, the calculated approximation of the reachable set may be used as a heuristic for an informed search or to find an initial solution for an optimization-based trajectory planner as it is guaranteed that the solution must be contained in this set.

The algorithm has been evaluated in a simulated scenario with two dynamic objects within a set of static obstacles. Different stages of the scenario have been shown. The algorithm can prove when overtaking, braking and finally avoiding a collision are no longer possible. The accuracy of the algorithm from coarse to fine can be balanced against the computational effort by adjusting a low number of parameters. The algorithm supports all main types of obstacle representation including time-varying occupancy grids or object lists of polygonal shape. It scales well with the number of dynamic obstacles. Also, it is not restricted to specific structure in environments like lanes.

As a future work, we wish to combine our approach with a trajectory planner. We want to investigate further the benefit from the reachable set information on different underlying planning algorithms.

Acknowledgment

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

References

[1] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 4, pp. 6–22, 2014.

[2] S. M. LaValle, *Planning algorithms*. Cambridge University Press, 2006.

[3] M. Brannstrom, E. Coelingh, and J. Sjoberg, "Model-based threat assessment for avoiding arbitrary vehicle collisions," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 658–669, 2010.

[4] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 678–687, 2009.

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

[6] A. Eidehall, "Multi-target threat assessment for automotive applications," in *IEEE Int. Conf. on Intelligent Transportation Systems*, 2011, pp. 433–438.

[7] C. Schmidt, *Fahrstrategien zur Unfallvermeidung im Straßenverkehr für Einzel-und Mehrobjektszenarien*. KIT Scientific Publishing, 2014, vol. 30.

[8] J. Ziegler and C. Stiller, "Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2009, pp. 1879–1884.

[9] M. McNaughton, C. Urmson, J. Dolan, and J.-W. Lee, "Motion planning for autonomous driving with a conformal spatiotemporal lattice," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 4889–4895.

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

[11] D. Madas, M. Nosratinia, M. Keshavarz, P. Sundstrom, R. Philippsen, A. Eidehall, and K.-M. Dahlen, "On path planning methods for automotive collision avoidance," in *IEEE Intelligent Vehicles Symposium*, 2013, pp. 931–937.

[12] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[13] I. M. Mitchell, "Comparing forward and backward reachability as tools for safety analysis," in *Hybrid Systems: Computation and Control*. Springer, 2007, pp. 428–443.

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

[15] R. Parthasarathi and T. Fraichard, "An inevitable collision state-checker for a car-like vehicle," in *IEEE Int. Conf. on Robotics and Automation*, 2007, pp. 3068–3073.

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

[17] F. C. Crow, "Summed-area tables for texture mapping," in *Proc. of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1984, pp. 207–212.

[18] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proc. of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996, pp. 171–180.

[19] R. A. Finkel and J. L. Bentley, "Quad trees: a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[20] W. Lipski and F. P. Preparata, "Finding the contour of a union of iso-oriented rectangles," *Journal of Algorithms*, vol. 1, no. 3, pp. 235 – 246, 1980.

[21] M. Althoff, D. Hess, and F. Gambert, "Road occupancy prediction of traffic participants," in *IEEE Int. Conf. on Intelligent Transportation Systems*, 2013, pp. 99–105.