Set-Based Computation of Vehicle Behaviors for the Online Verification of Autonomous Vehicles

Matthias Althoff and John M. Dolan

***Abstract*— We compute the set of all possible behaviors of an autonomous vehicle using reachability analysis. A reachable set is the set of states a system can possibly reach for a given set of initial states, disturbances, and sensor noise values. We consider autonomous vehicles which plan trajectories for a certain look-ahead horizon which are followed using feedback control. While a perfectly followed trajectory might not violate specified safety properties (e.g. lane departures or vehicle collisions), there might exist a violating deviation from the planned trajectory. Given the mathematical model of the controlled vehicle and bounds on uncertainty, our approach detects any possible violation. In addition, the approach provides results faster than real time such that maneuvers of vehicles can be checked before they are fully executed.**

I. Introduction

One of the main motivations for the development of fully or partially autonomous vehicles is to prevent accidents caused by human error. Unlike humans, a computer-controlled vehicle can exactly compute its future behavior when a mathematical description of the vehicle and its maneuver is provided. Based on these predictions, one can compute if the vehicle stays within lane boundaries and if static as well as dynamic obstacles are avoided. However, in reality, the exact behavior cannot be predicted due to uncertainties in the initial state, sensor measurements, and vehicle models. A commonly used technique to cope with those uncertainties is to compute many simulations. The drawback of this approach is that the number of required simulations typically scales exponentially with the number of uncertain variables and that not all behaviors can be captured since there exist infinitely many. This dilemma can be overcome by computing reachable sets which enclose all possible simulations of a system. In this work, an approach is presented to compute the reachable set of an autonomous vehicle faster than real time. This makes it possible to compute if a maneuver is safe under all considered uncertainties before it is fully executed.

The literature on reachability analysis applied to autonomous vehicles and car-like robots is rather limited. Unlike this work, most previous work considers simple dynamic models. A frequently used model is to bound the acceleration in the 2-dimensional plane such that the set of positions is obtained from a double integration resulting in reachable circles [1], [2]. Reachability analysis of slightly more complex models has been performed in coverage and pursuer problems: non-holonomic Dubins vehicles have been considered in [3], a tricycle model has been considered in [4], and vehicles in environments dominated by external drift are considered in [5].

There is more work on verifying maneuvers using reachability analysis for aircrafts. For this application domain, much work has been done on reachable set computation by solving Hamilton-Jacobi partial differential equations, see e.g. [6], [7]. Unlike in this work, the considered flight maneuvers are verified offline.

Besides reachability analysis, there is also work on verification of road traffic using theorem proving [8].

Online verification of road traffic scenes using reachability analysis has been presented in an earlier work for vehicles tracking arc segments at constant velocity [9]. These restrictions were introduced to obtain linear system dynamics. This work is an extension in many respects. First, the planned trajectory can be arbitrary instead of being restricted by connected arc segments. Second, the velocity of the maneuvers varies over time instead of being constant. Due to these two generalizations the differential equations describing the vehicle dynamics are no longer linear, but nonlinear, which makes reachability analysis much harder. Third, we consider measurement uncertainties which have not been considered in the previous work. Despite this more complex setting, we provide an approach which computes reachable sets faster than real time.

II. Verification Concept

In this work, we present a technique to compute all possible behaviors of an autonomous vehicle under uncertain measurements and initial states. One of the main applications of this method is a safety verification module for autonomous vehicles which decides if a maneuver can be safely executed, see Fig. 1. The maneuvers are mathematically described by reference trajectories[1] which are functions over time describing the goal position of the vehicle on the road.

The safety verification module requires information on the road network, as well as on static and dynamic obstacles for collision checks. The trajectory to be checked is provided from a trajectory planner. We assume that the suggested trajectories already passed a collision check under the assumption that the vehicle perfectly follows the trajectory.

Matthias Althoff is with Faculty of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: malthoff@ece.cmu.edu

John Dolan is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA, email: jmd@cs.cmu.edu

[1]Note that trajectory following differs from path following since in trajectory following the vehicle positions are specified by time, whereas in path following the timing along the path is irrelevant.

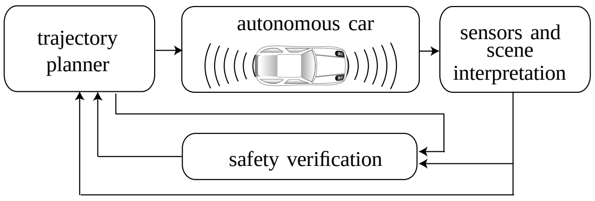


Fig. 1. Concept of the safety verification.

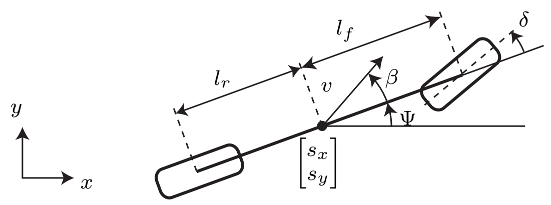


Fig. 2. Bicycle model.

This reduces the number of trajectories that have to be checked by more costly reachable set computations.

Depending on the reference trajectory to be checked, the verification is for finite or infinite time. The infinite verification can be achieved by additionally planning a braking maneuver which brings the vehicle to a safe stop – a condition in which the vehicle can stay forever without causing a crash. A stop is not considered safe if the vehicle stops in an intersection, a railroad crossing, or other unsafe locations. Note that the vehicle only executes the beginning of the planned trajectory while the reference trajectory is continuously replanned and verified such that the stops are not necessarily executed unless no safe alternative maneuver is to be found.

III. Model of the Controlled Vehicle

In order to compute the reachable set of the autonomous car, a mathematical model is required. We first derive the dynamics of the vehicle and secondly introduce a controller for trajectory tracking. Note that the presented reachability analysis can also be applied the same way to a modified dynamic model.

A. Vehicle Model

The vehicle model is an extended bicycle model which consists of 6 states: the slip angle at the center of mass $x_1 = \beta$, the heading angle $x_2 = \Psi$, the yaw rate $x_3 = \dot{\Psi}$, the velocity $x_4 = v$, the x-position $x_5 = s_x$, and the y-position $x_6 = s_y$, see Fig. 2. The bicycle model is widely used for control designs involving lateral vehicle dynamics and its name refers to the fact that both front and rear wheels are each lumped into one wheel since the roll dynamics is not considered [10, Chap. 2.6]. The bicycle model is accurate for small longitudinal accelerations and a tire model which linearly relates lateral force and slip angle. However, tires saturate at large slip angles, which is considered unsafe so that reference trajectories causing tire saturation are returned as unsafe and are not further verified. For large longitudinal accelerations, the vertical force shifts between the front and rear axle, which is not yet considered. Although this effect is not dominant, we plan to consider it in future work, see Sec. VII.

The differential equations of the vehicle dynamics are given in (1), where the equations for $\dot{x}_1$ and $\dot{x}_3$ describe the bicycle model for constant velocity. The orientation angle x_2 is obtained by integration of the yaw rate x_3 and the velocity x_4 by integration of the longitudinal acceleration a_x, see (1). Finally, the velocity and the direction of the center of mass $(x_1 + x_2)$ are used to geometrically obtain the position coordinates x_5 and x_6.

$$\begin{aligned}
\dot{x}_1 &= \Big(\frac{C_r l_r - C_f l_f}{m x_4^2} - 1\Big) x_3 + \frac{1}{m x_4}\Big(C_f \delta - (C_f + C_r) x_1\Big) \\
\dot{x}_2 &= x_3 \\
\dot{x}_3 &= \frac{1}{I_z}\Big((l_r C_r - l_f C_f) x_1 - (l_f^2 C_f + l_r^2 C_r)\frac{x_3}{x_4} + l_f C_f \delta\Big) \\
\dot{x}_4 &= a_x \\
\dot{x}_5 &= x_4 \cos(x_1 + x_2) \\
\dot{x}_6 &= x_4 \sin(x_1 + x_2)
\end{aligned} \tag{1}$$

B. Tracking Controller

In this subsection, the controllers for the steering angle δ and the acceleration command a_x are designed. It is assumed that the vehicle has internal controls which make it possible to realize commanded steering angles and acceleration commands at high accuracy. However, uncertainties due to unmodeled dynamics of internal controllers can be considered by enlarging the set of uncertain inputs.

The task of the tracking controller is to follow a reference trajectory which is specified at discrete points in time $t_k = k\, r$, where $k \in \mathbb{N}$ is the time step and $r \in \mathbb{R}^+$ is the step size. The values are held constant in between, i.e. during the time intervals $[t_k, t_{k+1}]$. For a short notation we introduce the time interval $\tau_k := [t_k, t_{k+1}]$. The reference trajectory consists of the desired values of the x- and y-position $s_{x,d}$, $s_{y,d}$ in a global coordinate system, from which the desired yaw angle Ψ_d, yaw rate $\dot{\Psi}_d$ and velocity v_d can be derived.

For the lateral and longitudinal control we use the position deviations in the local coordinates of the reference trajectory, see Fig. 3. The longitudinal deviation ϵ_x and the lateral deviation ϵ_y are:

$$\begin{aligned}
\epsilon_x &= \cos(\Psi_d)(s_{x,d} - s_x) + \sin(\Psi_d)(s_{y,d} - s_y), \\
\epsilon_y &= -\sin(\Psi_d)(s_{x,d} - s_x) + \cos(\Psi_d)(s_{y,d} - s_y).
\end{aligned}$$

For the lateral control we use the lateral deviation ϵ_y, as well as the deviations form the yaw angle and yaw rate to stabilize ϵ_y around zero:

$$\delta = k_1 \epsilon_y + k_2(\Psi_d - \Psi) + k_3(\dot{\Psi}_d - \dot{\Psi}).$$

For the longitudinal control we use the longitudinal deviation ϵ_x and the velocity deviation $\epsilon_v(t) = v_d(t) - v(t)$. The acceleration is determined by a weighted sum of ϵ_x and ϵ_v, where the velocity error can be seen as the D-part of a PD position control:

$$a_x = k_4 \epsilon_x + k_5 \epsilon_v.$$

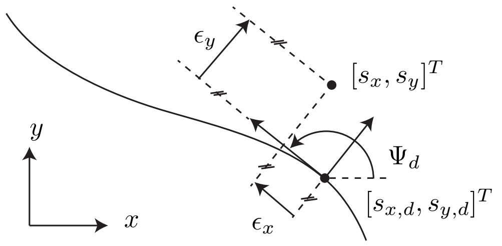


Fig. 3. Trajectory tracking: auxiliary variables.

It remains to introduce sensor noise in order to study the controller performance in realistic conditions. We use a positioning system that combines GPS data with inertial measurements to accurately measure the positions s_x, s_y, the yaw angle Ψ, the yaw rate $\dot{\Psi}$ and the velocity v. The corresponding sensor noise is denoted by u_x, u_y, u_Ψ, $u_{\dot{\Psi}}$, and u_v. After introducing the sensor noise vector $u = [u_x,\, u_y,\, u_\Psi,\, u_{\dot{\Psi}},\, u_v]^T$ and the reference vector $w = [s_{x,d},\, s_{y,d},\, \Psi_d,\, \dot{\Psi}_d,\, v_d]$, the final control equations are

$$\begin{aligned}\delta =& k_1\Big(\cos(w_3)(w_2 - x_6 - u_2) - \sin(w_3)(w_1 - x_5 - u_1)\Big)\\ &+ k_2(w_3 - x_2 - u_3) + k_3(w_4 - x_3 - u_4),\\ a_x =& k_4\Big(\cos(w_3)(w_1 - x_5 - u_1) + \sin(w_3)(w_2 - x_6 - u_2)\Big)\\ &+ k_5(w_5 - x_4 - u_5).\end{aligned}$$

Inserting the above equations $\delta = f_\delta(x, w, u)$ and $a_x = f_{a_x}(x, w, u)$ into (1) results in the final differential equation of the controlled vehicle $\dot{x} = f(x, w, u)$.

Typically, the next step is to show that the controlled system $\dot{x} = f(x, w, u)$ is globally stable for all possible reference trajectories $w(\cdot)$. Finding a Lyapunov function for a fixed reference trajectory is already a difficult problem, but inventing a Lyapunov function that can handle arbitrary reference trajectories might be a very challenging task. Note that in this work, the construction of a Lyapunov function is not required since we already formally prove safety using reachability analysis.

IV. Reachability Analysis

This section summarizes the steps that have to be undertaken to compute the set of states that the controlled vehicle with the dynamics $\dot{x} = f(x, w, u)$ can reach. Besides uncertain initial states $x(0) \in \mathcal{R}(0)$, we will also allow uncertain sensor noise values $u(t) \in \mathcal{U}$, where the only requirement for $u(t)$ is that it is piecewise continuous. Thus, we capture arbitrary noise frequencies of $u(t)$. A formal definition of a reachable set is

Definition 1 (Reachable Set): We denote the solution to $\dot{x} = f(x, w, u)$ for $x(0) = x_0$, $t \in [0, t_f]$, and trajectories $w(\cdot)$, $u(\cdot)$ by $\chi(t, x_0, w(\cdot), u(\cdot))$. Note that $w(\cdot)$ refers to a trajectory, where $w(t)$ refers to the value of the trajectory at time t. The exact reachable set for a given reference trajectory $w^*(\cdot)$ and a set of sensor noise values $\mathcal{U}$ is

$$\mathcal{R}^e([0, t_f]) = \Big\{\chi(t, x_0, w(\cdot), u(\cdot))\Big| x_0 \in \mathcal{R}(0), t \in [0, t_f],\\ w(t) = w^*(t), u(t) \in \mathcal{U}\Big\}.$$

In general, the set of reachable states cannot be computed exactly [11], so that one has to compute overapproximations denoted by $\mathcal{R}([0, t_f]) \supseteq \mathcal{R}^e([0, t_f])$. In this work, the reachable set of the time interval $[0, t_f]$ is obtained by computing reachable sets of smaller time intervals $[t_k, t_{k+1}]$, where t_k equals the times at which the reference vector $w(t_k)$ is updated, see Sec. III-B. It would also be possible to choose fractions of t_k as time intervals for the reachable set computation. The final reachable set is represented by a list of sets for all time intervals.

The overapproximations in this work are obtained by linearizing the nonlinear dynamics $\dot{x} = f(x, w, u)$ so that techniques for linear systems can be applied. In order to guarantee an overapproximative result, the linearization error is considered as an additional uncertain input, as presented in the next subsection.

A. Conservative Linearization

For a concise notation of the linearization procedure, the state vector x and the input vector u are combined in a new vector $z = [x^T,\, u^T]^T$. The reference trajectory is not included, since it is certain, and thus a linearization with respect to that vector is not required. Using a first-order Taylor expansion around the linearization point $[{z^*}^T,\, {w^*}^T]^T$, the original differential equation of the i^{th} coordinate is enclosed by the differential inclusion

$$\begin{aligned}&\forall t \in \tau_k :\\ &\dot{x}_i \in f_i(z^*,\, w^*) + \underbrace{\frac{\partial f_i(z, w)}{\partial z}\Big|_{z=z^*}(z - z^*)}_{=A(x-x^*)+B(u-u^*)} \oplus \mathcal{L}(\tau_k),\end{aligned}$$

where $\oplus$ denotes a Minkowski addition[2] and $\mathcal{L}$ is the set of Lagrange remainders

$$\mathcal{L}(\tau_k) = \Big\{\frac{1}{2}(z - z^*)^T \frac{\partial^2 f_i(\xi, w)}{\partial z^2}(z - z^*)\Big| \xi \in \mathcal{R}(\tau_k) \times \mathcal{U}\Big\}.$$

The Lagrange remainder covers all possible linearization errors when ξ may vary arbitrarily in the set of possible values of x and u given by the Cartesian product $\mathcal{R}(\tau_k) \times \mathcal{U}$, see [12].

The linearization point is updated for each time interval τ_k. A good linearization point is $z^*(\tau_k) = \texttt{center}(\mathcal{R}(\tau_k) \times \mathcal{U})$ (see [13]), where `center()` returns the the volumetric center of a set. Since the reachable sets $\mathcal{R}(\tau_k)$ are not known in advance, the vector $x^*(\tau_k)$ of z^* is obtained heuristically by state values $x(t_k)$ of a simulation starting in the center of $\mathcal{R}(0)$ under the input $u(t) = \texttt{center}(\mathcal{U})$.

Given the linearization points for each time interval, the conservative linearization is obtained as follows:

[2]$\mathcal{A} \oplus \mathcal{B} = \{a + b | a \in \mathcal{A}, b \in \mathcal{B}\}$

1) Define a set of allowed linearization errors $\overline{\mathcal{L}}(\tau_k)$ which should be a superset of the exact set of linearization errors.
2) Compute the reachable set $\overline{\mathcal{R}}(\tau_k)$ of the linearized system $\dot{x} \in f_i(z^*, w^*) + A(x - x^*) + B(u - u^*) \oplus \overline{\mathcal{L}}(\tau_k)$ as shown in the next subsection.
3) Compute the linearization errors $\mathcal{L}(\tau_k)$ based on $\overline{\mathcal{R}}(\tau_k)$ according to [13].
4) Check if $\mathcal{L}(\tau_k) \subseteq \overline{\mathcal{L}}(\tau_k)$, otherwise abort and return `unsafe`.
5) Compute the refined set $\mathcal{R}(\tau_k)$ as in step 2 using $\mathcal{L}$ instead of $\overline{\mathcal{L}}$.
6) Repeat this procedure for further time intervals.

The procedure for computing reachable sets of linear systems is described in the next subsection.

B. Reachable Set Computation of Linear Systems

The reachable set computation for linear systems takes advantage of the superposition principle, allowing one to separately obtain the reachable set due to the initial state solution and the input solution, denoted by $\mathcal{R}^h$ and $\mathcal{R}^i$, respectively. The step-by-step computation is illustrated in Fig. 4:

1) Compute the exact set of initial state solutions as $\mathcal{R}^h(t_{k+1}) = e^{Ar}\mathcal{R}(t_k)$.
2) Compute the convex hull $R^h_{\mathtt{CH}}(\tau_k)$ of $\mathcal{R}(t_k)$ and $\mathcal{R}^h(t_{k+1})$ which encloses all trajectories for $t \in \tau_k = [t_k, t_{k+1}]$ under the assumption that trajectories within the time interval are straight lines.
3) Enlarge the convex hull by $\mathcal{F}(t_k) \otimes \mathcal{R}(t_k)$ ($\mathcal{A} \otimes \mathcal{B} = \{ab | a \in \mathcal{A}, b \in \mathcal{B}\}$) to consider the curvature of trajectories, and by $\mathcal{R}^i(\tau_k)$ to consider uncertain inputs. A detailed description for computing $\mathcal{F}$ and $\mathcal{R}^i$ can be found in [14], where $\mathcal{F}$ is an interval matrix. Thus, the reachable set is obtained as $\mathcal{R}(\tau_k) = \mathcal{R}^h(\tau_k) \oplus (\mathcal{F}(t_k) \otimes \mathcal{R}(t_k)) \oplus \mathcal{R}^i(\tau_k)$.

Besides the time interval solution $\mathcal{R}(\tau_k)$, the set $\mathcal{R}(t_{k+1}) = \mathcal{R}^h(t_{k+1}) \oplus \mathcal{R}^i(t_{k+1})$ has to be computed, since the algorithm starts with the set at fixed times t_k, see step 1.

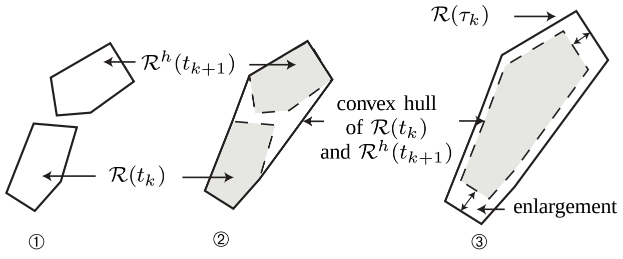


Fig. 4. Computation of the reachable set for a time interval.

The Minkowski addition of $\mathcal{R}^h(t_{k+1})$ and $\mathcal{R}^i(t_{k+1})$ increases the representation order of the reachable set, which requires reduction methods causing the so-called wrapping-effect due to the propagation of overapproximations through successive time steps. In this work, zonotopes are used to represent reachable sets for which efficient reduction techniques exist [15], [16].

C. Parallelization

Since the presented algorithms have to be faster than real time, we discuss which parts of the algorithm can be parallelized. All linearization points $z^*(\tau_k)$ are computed based on a single simulation as described in Sec. IV-A, so that they are known before computing reachable sets. From this follows that the linearized systems for each time interval τ_k can be obtained independently, i.e., the system matrices $A(\tau_k)$, the input matrices $B(\tau_k)$, and the constant inputs $f(z^*(\tau_k), w(\tau_k))$.

For each linearized system, the following expressions can be obtained in parallel: the matrix exponentials $e^{A(\tau_k)r}$, the interval matrices $\mathcal{F}(t_k)$ for the enlargement, and the input sets $\mathcal{R}^i(\tau_k)$ for the allowed linearization errors $\overline{\mathcal{L}}(\tau_k)$.

The remaining computation steps require less computation time than the ones that can be parallelized, as later demonstrated in the numerical example. A crucial step which cannot be parallelized is the computation of the linearization errors $\mathcal{L}$, since they depend on the reachable set $\overline{\mathcal{R}}(\tau_k)$. Usually, time-consuming interval arithmetic is used to overapproximate the set of linearization errors. In this work, we were able to take advantage of monotonicity properties of the Lagrange remainder, which make it possible to compute variable ranges of functions by simply evaluating the corner cases (see [17]). The detailed discussion is skipped since it would require deriving all second-order derivatives $\frac{\partial^2 f_i(\xi, w)}{\partial z^2}$.

V. Set of Occupied Positions

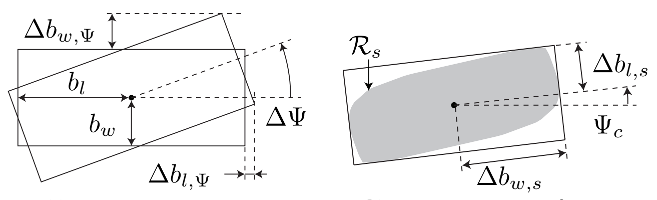
The ultimate goal of the reachability analysis is to check if the vehicle stays on the road and if a crash is possible, which requires knowing the occupancy of the vehicle body on the road. The relevant variables from the reachability computation to determine the occupancy on the road are the x- and y-position (s_x, s_y) and the orientation Ψ. We denote the projections of the reachable set onto the position coordinates as $\mathcal{R}_s$ (2-dimensional) and onto the orientation as $\mathcal{R}_\Psi$ (1-dimensional).

We model the vehicle body as a rectangle with width b_w and length b_l. For each time interval τ_k, the rectangle is oriented according to the center of possible orientations $\Psi_c = \mathtt{center}(\mathcal{R}_\Psi)$. The deviation around this center $\Delta\Psi = \max_{\Psi^* \in \mathcal{R}_\Psi}(|\Psi^* - \Psi_c|) \geq 0$ is considered by enlarging the width and the length of the vehicle body as illustrated in Fig. 5. The length and width increase is

$$\Delta b_{l,\Psi} = |(1 - \cos(\Delta\Psi))b_l - \sin(\Delta\Psi)b_w|,$$
$$\Delta b_{w,\Psi} = |(1 - \cos(\Delta\Psi))b_w - \sin(\Delta\Psi)b_l|.$$

The uncertain position of the center of mass is also considered by enlarging the body size, while the center of the vehicle body is shifted to $\mathtt{center}(\mathcal{R}_s)$. The enlargement due to uncertain positions can be obtained by rotating the set of positions by $-\Psi_c$ and computing an enclosing interval for each dimension. The box enclosure operation $\mathtt{box}()$ can be efficiently done for zonotopes, which are used for reachable set computations.

$$\begin{pmatrix}[-\Delta b_{l,s}, \Delta b_{l,s}]\\ [-\Delta b_{w,s}, \Delta b_{w,s}]\end{pmatrix} = \mathtt{box}(T(\mathcal{R}_s - \mathtt{center}(\mathcal{R}_s)),$$
$$T = \begin{pmatrix}\cos(-\Psi_c) & \sin(-\Psi_c)\\ -\sin(-\Psi_c) & \cos(-\Psi_c)\end{pmatrix}$$



(a) Uncertain orientation. (b) Uncertain center of mass.

Fig. 5. Enlargement of the vehicle occupation due to uncertain orientation and position.

Once the set of possible occupation is obtained, one has to check if collisions with other vehicles or road boundaries exist, for which many efficient algorithms exist. In the previous work, the occupations at time intervals have been overapproximated by axis-aligned boxes, and only if axis-aligned boxes intersect, is the more elaborate collision check of oriented boxes computed [9]. Another method is to inscribe the occupied sets by several circles, which are the most efficient representation for collision checks [18].

VI. Numerical Example

In this section we present the usefulness of the reachable set computations for the online verification of traffic scenes. It is again mentioned that the reachability results can also be used to evaluate the performance of controllers under sensor noise and uncertain initial states.

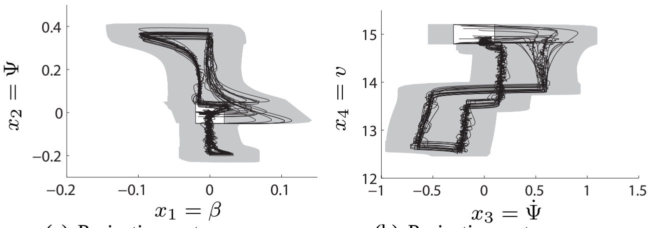
The considered scenario presents an evasive maneuver of the autonomous car caused by a pedestrian who steps into the road without respecting oncoming traffic. In addition, the autonomous car has to avoid a collision with an oncoming car while respecting the road limits during the evasive maneuver. The first part of the maneuver is a combined braking and steering maneuver with lateral acceleration of ± 0.8g and longitudinal acceleration of -0.2g, where g is the gravity constant. This part of the maneuver is at the limit of maximum possible tire force, whereas the second part induces smaller acceleration values, since it is only required to steer back into the original lane before hitting oncoming traffic.

The vehicle, sensor, and control parameters of the numerical example are listed in Tab. I. The vehicle parameters are taken from a 1986 Pontiac 6000 STE sedan [19] and the standard deviation σ of the sensor noise for u_x, u_y, and u_Ψ is taken from the Applanix POS LV platform, whose specifications can be found online. The standard deviation for the other noise sources are reasonably chosen. The interval of possible sensor noises is chosen as the 4σ interval, which corresponds to a probability of 0.99994 that the sensor noise is in the interval when it is assumed to be Gaussian. The set of initial states for the considered maneuver is $\mathcal{R}(0) = [-0.02, 0.02]$ rad $\times[-0.05, 0.05]$ rad $\times[-0.3, 0.3]$ rad/s $\times[14.8, 15.2]$ m/s $\times[-0.2, 0.2]$ m $\times[-0.5, -0.1]$ m. The body size of the autonomous car is $b_l = 4.5$ m, $b_w = 1.8$ m and the time step for updating the reference trajectory is $r = 0.01$ s.

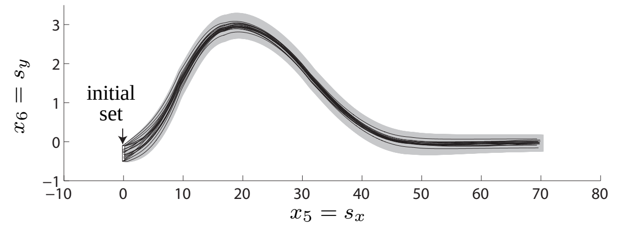
TABLE I
Vehicle parameters.

vehicle parameters				
m	I_z	$C_f = C_r$	l_f	l_r
1573 kg	2873 kg m^2	$80e4$ N/rad	1.1 m	1.58 m
sensor noise intervals, $\beta = [-1, 1]$				
u_x	u_y	u_Ψ	$u_{\dot\Psi}$	u_v
0.08β m	0.08β m	$\frac{0.2\pi}{180}\beta$ rad	$\frac{0.2\pi}{180}\beta$ rad/s	0.08β m/s
control parameters				
k_1	k_2	k_3	k_4	k_5
1	10	2	1	10

The reachable sets for different projections are shown in Fig. 6. It can be seen that all example simulations are enclosed by the reachable set. The simulations indicate that the overapproximation of the position deviation (s_x, s_y) as well as the other coordinates is small. In Fig. 7, the occupancy of the autonomous car and the oncoming car are shown. For the oncoming car it is assumed that the maximum speed is 20% higher than the speed limit of 15 m/s, the maximum acceleration is 0.7 g, and the vehicle does not drive backwards. The initial longitudinal position and velocity are $[110, 120]$ m and $[13, 15]$ m/s. It is assumed that the other vehicle only uses its own lane, but possibly occupies the full width of the lane. Due to braking, the vehicle may stop at 108 m, which marks the rightmost occupancy. Note that we compute all possible behaviors of the other vehicle within its acceleration bounds. Based on the occupancy set of both vehicles, it can be concluded that the autonomous car stays within the road limits from -1.75 m to 5.25 m (lane width 3.5 m) and avoids a crash with the oncoming vehicle.



(a) Projection onto x_1, x_2. (b) Projection onto x_3, x_4.



(c) Projection onto x_5, x_6 (“road coordinates“).

Fig. 6. Reachable set for the evasive maneuver. The white set shows the set of initial states, black lines show example simulation results.

All computations have been performed on an Intel i7

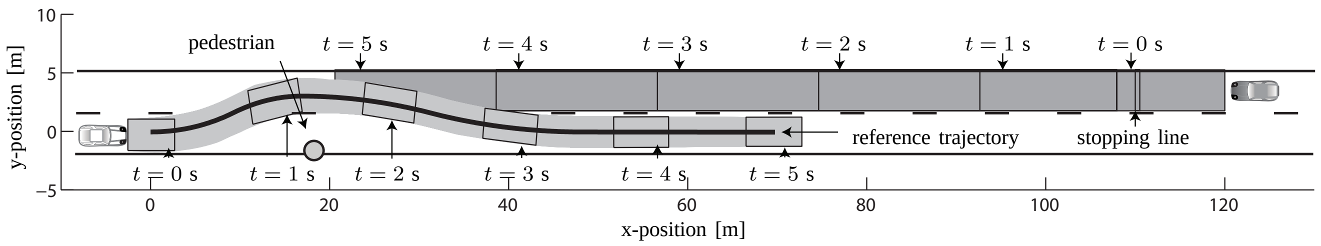


Fig. 7. Occupancy set for the evasive maneuver. The gray regions show the occupancies of the autonomous and the other car, the black lines shows the path the autonomous car should follow. The occupancy at selected times is indicated by black boxes.

Processor with 1.6 GHz and 6 GB memory in MATLAB. The reachable set computation of the parallelizable part took 1.56 s (process A) using 4 cores and 2.24 s for the part that cannot be parallelized (process B). Since process B can start after process A has the result for the first time interval, the overall computation time for the reachable sets is 2.24 s. The computation for transforming the reachable set to the occupancy of the vehicle (process C) took 0.46 s and the collision check took 0.25 s (process D). However, process C and D are post-processing steps that can start as soon as the first reachable set has been computed, such that these two processes do not add computation time, so that only the 2.24 s of process B matter. Considering that the maneuver took 5 s, the current implementation is about 2 times faster than real time. By implementing the algorithm in C++ on a real vehicle, it is expected that the efficiency will be significantly improved.

VII. Conclusions and Future Work

We have presented an algorithm that can predict all possible behaviors of an autonomous car given a dynamic model, a set of initial states, and bounds on sensor noise. The set of all possible trajectories of a vehicle makes it possible to detect crashes of traffic scenes. It has been demonstrated that the approach can be computed faster than real time, where further improvements can be expected by implementing the algorithms in C++. Other future work involves considering uncertain vehicle parameters, such as the tire-road friction coefficient, or the weight distribution due to loading the trunk or longitudinal accelerations.

Acknowledgements

The authors gratefully acknowledge partial financial support by the NSF Cyberphysical Systems Program, Award CNS1035813.

References

[1] 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.

[2] J. van den Berg, "Path planning in dynamic environments," Ph.D. dissertation, Utrecht University, 2007.

[3] K. Savla, F. Bullo, and E. Frazzoli, "The coverage problem for loitering Dubins vehicles," in *Proc. of the 46th IEEE Conference on Decision and Control*, 2007, pp. 1398–1403.

[4] 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.

[5] A. Kwok and S. Martínez, "A coverage algorithm for drifters in a river environment," in *Proc. of the American Control Conference*, 2010, pp. 6436–6441.

[6] C. Tomlin, I. Mitchell, and R. Ghosh, "Safety verification of conflict resolution maneuvers," *IEEE Transactions in Intelligent Transportation Systems*, vol. 2, pp. 110–120, 2001.

[7] J. H. Gillulay, H. Huangy, M. P. Vitusy, and C. J. Tomlin, "Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2010, pp. 1649–1654.

[8] S. M. Loos, A. Platzer, and L. Nistor, "Adaptive cruise control: Hybrid, distributed, and now formally verified," in *Proc. pf the 17th International Symposium on Formal Methods*, ser. LNCS 6664. Springer, 2011, pp. 42–56.

[9] 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.

[10] R. Rajamani, *Vehicle Dynamics and Control*, F. F. Ling and E. F. Gloyna, Eds. Springer, 2006.

[11] G. Lafferriere, G. J. Pappas, and S. Yovine, "Symbolic reachability computation for families of linear vector fields," *Symbolic Computation*, vol. 32, pp. 231–253, 2001.

[12] M. Berz and G. Hoffstätter, "Computation and application of Taylor polynomials with interval remainder bounds," *Reliable Computing*, vol. 4, pp. 83–97, 1998.

[13] 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.

[14] M. Althoff, "Reachability analysis and its application to the safety assessment of autonomous cars," Dissertation, TU München, 2010, URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20100715-963752-1-4.

[15] W. Kühn, "Rigorously computed orbits of dynamical systems without the wrapping effect," *Computing*, vol. 61, pp. 47–67, 1998.

[16] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control*, ser. LNCS 3414. Springer, 2005, pp. 291–305.

[17] N. Ramdani, N. Meslem, and Y. Candau, "Computing reachable sets for uncertain nonlinear monotone systems," *Nonlinear Analysis : Hybrid Systems*, vol. 4, pp. 263–278, 2010.

[18] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 518–522.

[19] J. Guldner, W. Sienel, H.-S. Tan, J. Ackermann, S. Patwardhan, and T. Bünte, "Robust automatic steering control for look-down reference systems with front and rear sensors," *IEEE Transactions on Control Systems Technology*, vol. 7, pp. 2–11, 1999.