5. Safety Assessment of Autonomous Cars

In this chapter, the generic methods on stochastic reachability analysis are applied to the safety assessment of autonomous vehicles. Ultimately, the presented safety verification module should assess various alternatively planned driving actions of autonomous cars according to their safety.

5.1. Introduction and State of the Art

First, some challenges of autonomous driving, i.e. driving without a human driver, are presented.

Autonomous Driving

A prerequisite of autonomous driving is the equipment of vehicles with sensors for the detection of their environment. More importantly, relevant information such as the position and velocity of other traffic participants has to be correctly extracted from the raw data streams of the sensors. This has to work properly in different weather conditions and even when unknown or unexpected objects are present. Besides the enormous challenge of detecting objects such as other vehicles or lane markings, it is also desirable to estimate the intentions of other traffic participants in order to derive the optimal behavior for the autonomous vehicle.

Human drivers are very good at recognizing other traffic participants, estimating their intention, and planning almost optimal driving actions. Besides these capabilities, humans can focus on the relevant information, handle unexpected situations, and automatically learn from previously unknown situations. It is obvious that researchers aim to partly implement those cognitive capabilities into an autonomously driving car. This has been tried in many research projects, including the collaborative research center *Cognitive Automobiles* [154], in which this work has been carried out.

One of the main objectives of the research on autonomous vehicles is the vision of accident-free driving by exclusion of human errors. Worldwide, the number of people killed in road traffic each year is estimated at almost 1.2 million, while the number of injured is estimated at 50 million [134, chap. 1]. However, autonomous cars will not be market-ready soon, such that mature driving capabilities of autonomous prototype vehicles will be incorporated into intelligent driver assistant systems of market-ready vehicles. It is e.g. desirable that a driver assistant system fully controls a vehicle when a crash is almost inevitable. Predicting

the probability of a crash is subject of the safety assessment developed for autonomous cars in this thesis.

Safety Assessment in Road Traffic

In order to assess the safety of a planned maneuver, the predicted motion of other traffic participants is vital for the identification of future threats. For this reason, the prediction of other traffic participants is one of the main objectives in this chapter. In contrast to this approach, non-predictive methods are based on the recording and evaluation of traffic situations that have resulted in dangerous situations; see e.g. [4]. However, such an approach is only suitable for driver warnings. Planned trajectories of autonomous cars cannot be evaluated with non-predictive methods since the consequences when following these trajectories have to be predicted.

Behavior prediction has been mainly limited to human drivers within the *ego vehicle* (i.e. the vehicle for which the safety assessment is performed). This is motivated by research on driver assistant systems which tries to warn drivers when dangerous situations are ignored. The majority of works on this topic use learning mechanisms (e.g. neural networks, autoregressive exogenous models [151, 174]), or filter techniques (e.g. Kalman filters [109, 136]). Another line of research is to detect traffic participants on selected road sections and predict their behavior for anomaly detection. Such automatic surveillance system has been realized with learning techniques such as clustering methods [86] or hidden Markov models [123]. The disadvantage of a prediction at fixed locations is that the predictions are specialized to this particular road segment and probably not generalizable to other traffic situations.

For the prediction of arbitrary traffic situations, simulations of traffic participants have been used [18, 84]. Due to the efficiency of single simulations, these approaches are already widely implemented in cars, e.g. to initiate an emergency braking maneuver based on measures like *time to collision* or *predicted minimum distance*. Simulations of traffic participants are also computed in microscopic traffic simulations [119, 162]. However, single simulations do not consider uncertainties in the measurements and actions of other traffic participants, which may lead to unsatisfactory collision predictions [108]. A more sophisticated threat assessment considers multiple simulations of other vehicles, considering different initial states and changes in their inputs (steering angle and acceleration). These so-called Monte-Carlo methods have been studied in [15, 31, 32, 52, 59, 60] for the risk analysis of road traffic and in [25, 26, 168] for the related topic of air traffic safety. A framework for the reduction of possible future scenarios of traffic situations, using motivations for the actions of drivers, is introduced in [46].

Another method to compute possible behaviors of traffic participants is reachability analysis as presented in Chap. 3. Safe motion of two vehicles is guaranteed if their reachable sets of positions do not intersect. In traffic scenarios, the reachable positions of a vehicle define the unsafe set of another vehicle, and vice versa. Reachable sets for vehicles and mobile robots have been investigated in [149, 165]. It has been shown that planned paths of autonomous vehicles are too often evaluated as unsafe by this method. This is because the reachable sets of other vehicles rapidly cover all positions the autonomous vehicle could possibly move to, which is demonstrated in Fig. 1.8(a).

A combination of reachability analysis and stochastic reachability analysis has been investigated in [77]. The reachable sets of traffic participants are used to find out which vehicles might have a crash. Next, the stochastic reachable sets are only computed for those traffic participants that might crash in order to save computational time. Note that the reachable sets in [77] are not over-approximative and that the algorithm for the computation is rather fuzzy. The stochastic reachable sets are described by Gaussian distributions, which are obtained by several linearized models. The concept of detecting relevant traffic participants by reachability analysis can be analogously applied to the concept in this work.

Contributions

In this chapter, *stochastic reachable sets* of traffic participants are computed as previously shown in Chap. 4. The stochastic information allows not only to check if a planned path of the ego vehicle may result in a crash, but also with which probability. Consequently, possible driving strategies of autonomous cars can be evaluated according to their safety. Traffic participants are predicted by Markov chains as presented in Sec. 4.3. There are three properties which are in favor of the Markov chain approach: The approach can handle the hybrid dynamics of traffic participants, the number of continuous state variables (position and velocity) is low, and Markov chains are computationally inexpensive when they are not too large.

The contributions in more detail are: A basic concept for the safety assessment of autonomous cars in Sec. 5.2. A mathematical model of traffic participants in Sec. 5.3 and their abstraction to Markov chains in Sec. 5.4. Further, a stochastic generation of driving commands, which is addressed in Sec. 5.5 for certain driving capabilities: road following, vehicle following, intersection crossing, and lane changing. It is also discussed how driving capabilities are unified and how to handle capabilities which are not implemented, such as parking. The driving command generation and the Markov chains allow the position distribution of other traffic participants to be predicted. How to compute the crash probability for the autonomous vehicle, given the position probabilities of other traffic participants, is presented in Sec. 5.6. In order to evaluate the Markov chain approach, it is tested against Monte Carlo simulation in Sec. 5.7. The introduced Markov chain approach is also tested in the autonomous vehicle *MUCCI* on a test ground, which is documented in Sec. 5.8. Finally, the chapter is summarized in Sec. 5.9.

5.2. Basic Concept

Clearly, autonomous driving requires a control loop containing a perception and a planner module; see Fig. 5.1. The perception module detects traffic situations and extracts relevant information, such as the road geometry as well as static and dynamic obstacles. In order to fulfill the driving task, the planner module computes trajectories that the autonomous car is tracking with the use of low-level controllers [173]. A major constraint for the trajectory planner is that the generated trajectories have to be safe, i.e. static and dynamic obstacles must not be hit. The task of circumventing static obstacles can be ensured by checking whether the static obstacle intersects with the vehicle body of the autonomous car when

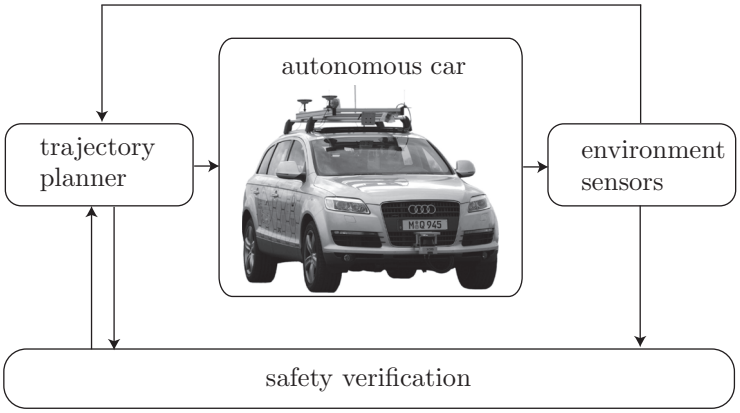


Fig. 5.1.: Conception of the safety assessment.

following the planned path. For dynamic obstacles, the safety assessment is much more intricate as their future actions are unknown. For this reason, sets of possible behaviors of other traffic participants are considered, which are checked with the planned path of the autonomous car in a dedicated safety verification module (see Fig. 5.1). Paths that fulfill the safety requirements are executed or are conservatively replanned otherwise, e.g. by braking the car.

The safety verification module which is described in this work requires the description of a traffic situation containing the following information gathered by the perception and planner modules:

- the planned trajectory of the autonomous car,
- the geometric description of relevant road sections,
- the position and geometry of static obstacles,
- as well as the position, velocity, and classification of dynamic obstacles.

Static obstacles are a special case of dynamic obstacles with zero velocity, and for that reason the discussion is continued for dynamic obstacles only. The classifier of the autonomous vehicle groups the dynamic obstacles ($\hat{=}$ traffic participants) into *cars*, *trucks*, *motorbikes*, *bicycles*, and *pedestrians*. As the measurement of positions and velocities of other traffic participants is uncertain, the presented approach allows the measured data to be specified by a probability distribution. However, it is required that all relevant traffic participants are detected. Given the information of the perception module, the future stochastic reachable set of all traffic participants is computed, from which the probability distribution of the position can be extracted. The positions with non-zero probability value belong to the set of reachable positions, which serves as a time varying unsafe set for the autonomous car[1].

If the reachable positions of other traffic participants do not intersect with the ones of the autonomous car for a prediction horizon t_f, safety can be guaranteed within the specified horizon. Where a crash is possible, the probability of the crash is computed from the probability distribution within the reachable set. This is illustrated in Fig. 5.2, where

[1]The ego car and the autonomous car refer to the same car in this thesis. However, it is possible that other vehicles are autonomous vehicles, too.

stochastic reachable sets are shown for the time intervals $\tau_1 = [0, t_1]$, $\tau_2 = [t_1, t_2]$ (dark color indicates high probability density). Within the time interval τ_1, a crash between both cars is impossible since their stochastic reachable sets do not intersect, while for the second time interval τ_2, the crash probability is non-zero. It is obvious that all necessary computations have to be faster than real time to allow an online application. In order to update the crash probability prediction after a time interval Δt based on new sensor values, its computation has to be faster than real time by a factor of $t_f/\Delta t$. This is illustrated in Fig. 5.3 for the reachable set of a single variable $x(t)$.

The mathematical model of other traffic participants used for the computation of their stochastic reachable sets is introduced next.

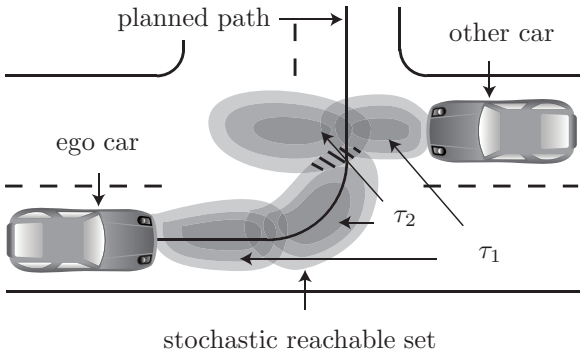


Fig. 5.2.: Stochastic reachable sets of traffic participants.

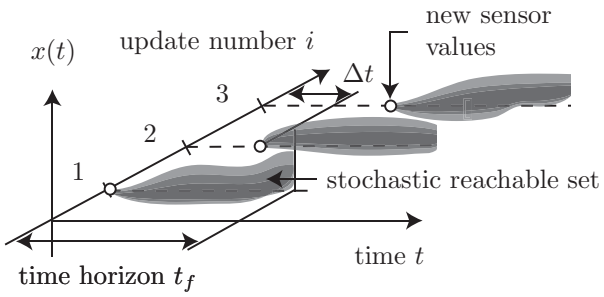


Fig. 5.3.: Repetitive computation of reachable sets.

5.3. Modeling of Traffic Participants

The presented safety assessment focuses on autonomous cars driving on a road network, i.e. the motion of traffic participants is constrained along designated lanes. On that account, the prediction of traffic participants is performed in two steps. Firstly, the lanes most likely followed by traffic participants are determined by high-level behaviors. Secondly, the dynamics of traffic participants along the corresponding paths on the lanes is considered. The same concept is applied in [77].

Possible paths of traffic participants are modeled by the finite set of high-level behaviors {*left turn*, *right turn*, *go straight*}, and {*left lane change*, *right lane change*} on a multi-lane road. Further high-level behaviors such as *parking* or *overtaking* can be included in the modeling scheme later. In the continuation of this work, it is planned to automatically derive such high-level behaviors by observation and clustering of traffic scenes. It is noted that the high-level behavior does not have to be exactly known, since one can also compute with probabilities of high-level behaviors.

In unstructured environments, such as parking spaces or pedestrian zones, the motion of vehicles/people cannot be described along paths. For these kinds of scenarios, the approach presented in [195] is suggested, which uses the same mathematical principles as presented later, but applies them to unstructured environments. The prediction in unstructured environments also serves as a fallback solution when the observed high-level behavior cannot

be categorized in one of the given or additionally learned high-level behaviors.

Note that the introduced model is only used for other traffic participants. The ego car does not have to be modeled since its future behavior is already determined by its trajectory planner.

5.3.1. Lateral Dynamics

The deviation along the paths of traffic participants is modeled by a piecewise constant probability distribution $f(\delta)$, where δ is the lateral deviation from a path. Possible paths of a road network section, as well as the deviation probability distribution $f(\delta)$, are shown in Fig. 5.4. The deviation probability can be adjusted to different classes of traffic participants: Bicycle riders are more likely to be found close to the curb, whereas cars and trucks are more likely to be driving in the center of a lane. A statistical analysis of lateral displacement of vehicles on a road can be found in [54]. The deviation probabilities are normalized to the width of the lanes so that typical distributions can be applied independently of the lane width.

In this thesis, the deviation probability is held constant over time and the more complex case of dynamically changing lateral distribution is the subject of future work. Another assumption is that the deviation probability is computed independently of the probability distribution along the path. This is a reasonable assumption since the task of following the desired path is more or less independent of the task of keeping the speed or the distance to other vehicles. Additionally, this assumption drastically simplifies the computation of probabilistic reachable sets of other traffic participants, because the lateral and longitudinal probability distribution can be computed separately in low dimensional spaces. Thus, given the lateral probability distribution $f(\delta)$ and the longitudinal probability distribution $f(s)$, the overall probability distribution is computed as $f(s, \delta) = f(s) \cdot f(\delta)$; see Fig. 5.4. The combined probability distribution is described in a curved, path-aligned coordinate system as also used in e.g. [60]. It is emphasized that the lateral and the longitudinal distribution $f(\delta)$ and $f(s)$ refer to the position s and the deviation δ of the volumetric center of the vehicles. However, for visualization reasons, all figures in this work show the density of the vehicle body, which takes the vehicle size into account; see Fig. 5.5.

The longitudinal probability distribution is obtained from a dynamical model which is explained in the next subsection.

5.3.2. Longitudinal Dynamics

For the longitudinal dynamics model, the position of a vehicle along a path s, the velocity v, and the absolute acceleration a have to be introduced. The acceleration command u is normalized and varies from $[-1, 1]$, where -1 represents full braking and 1 represents full acceleration. Further, the function $\rho(s)$ is introduced which maps the path coordinate s to the radius of curvature. The radius of the path determines the tangential acceleration a_T for a given velocity v and thus limits the normal acceleration a_N, since the absolute value of the combined accelerations has to be less than the maximum absolute acceleration $a^{\max}$. In addition, the acceleration dynamics changes at the switching velocity v^{sw}. The

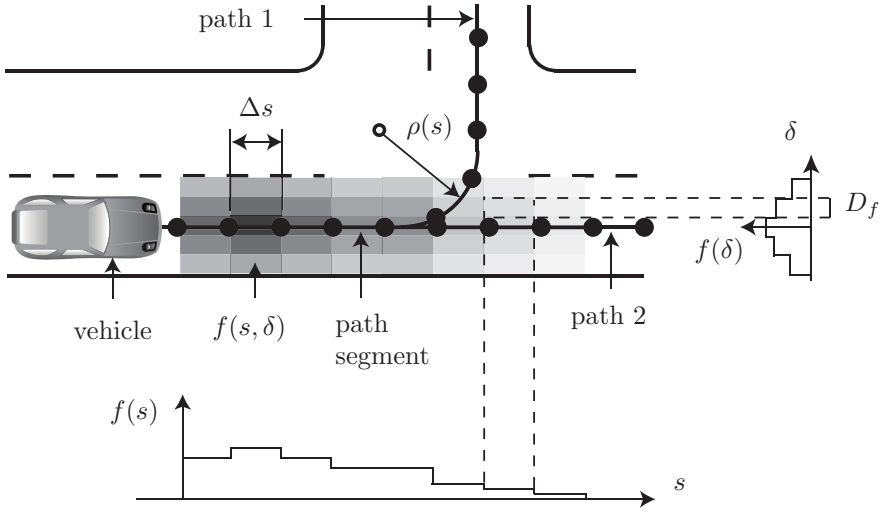


Fig. 5.4.: Position distribution $f(s,\delta) = f(s) \cdot f(\delta)$ along a path-aligned coordinate system, which is composed of the longitudinal and lateral distribution. D_f is a deviation segment and $\rho(s)$ is the radius of curvature along a path.

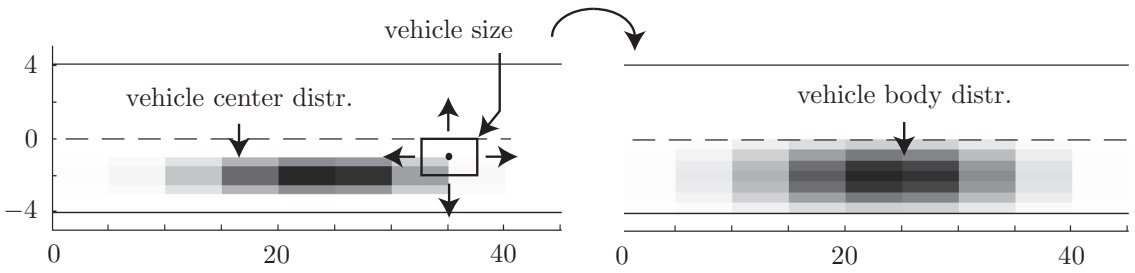


Fig. 5.5.: Probability distribution of the vehicle center and the vehicle body. The coordinate axes refer to positions in [m].

differential equations for the longitudinal dynamics are chosen as proposed in [60]:

$$\dot{s} = v, \quad \dot{v} = \begin{cases} a^{\max} u, & 0 < v \leq v^{\mathrm{sw}} \vee u \leq 0 \\ a^{\max} \frac{v^{\mathrm{sw}}}{v} u, & v > v^{\mathrm{sw}} \wedge u > 0 \\ 0, & v \leq 0 \end{cases} \tag{5.1}$$

subject to the constraint

$$|a| \leq a^{\max}, \text{ where } |a| = \sqrt{a_N^2 + a_T^2}, \; a_N = v^2/\rho(s), \; a_T = \dot{v}. \tag{5.2}$$

Backwards driving on a lane is not considered; see (5.1) ($\dot{v} = 0$, $v \leq 0$). The constraint in (5.2) models that the tire friction of a vehicle only allows a limited absolute acceleration $a^{\max}$ (Kamm's circle). The constants $a^{\max}$ and v^{sw} can be chosen according to the specific properties of the different classes of traffic participants. The values used in this thesis when not stated differently are listed in Tab. 5.1.

The differential equations in (5.1) are chosen exemplarily and can be easily exchanged against a different set of equations. For example, in previous publications [182, 183, 185,

Tab. 5.1.: Vehicle parameters.

	Car	Truck	Motorbike	Bicycle
$a^{\max}$ [m/s^2]	7	7	7	7
v^{sw} [m/s]	7.3	4	8	1

186, 188, 190], the vehicle model for the case $u > 0$ is $\dot{v} = (a^{\max} - c_2\, v^2)\, u$, where c_2 is a constant. This model considers aerodynamic drag, but does not consider that the acceleration force decreases with velocity since $m\, a \leq P^{\max}/v$, where m is the mass and $P^{\max}$ is the maximum acceleration power. Another possibility is to enhance the model according to Eidehall in (5.1) with aerodynamic drag so that the model for $v > v^{\mathrm{sw}} \wedge u > 0$ is $\dot{v} = \left(a^{\max} \frac{v^{\mathrm{sw}}}{v} - c_1\, v^2\right) u$, where v^{sw} and c_1 are constants. The different models are compared in Fig. 5.6 to a highly realistic vehicle model of the Audi Q7, which serves as the platform for the experimental vehicle *MUCCI* which is presented later. Note that the peaks in acceleration occur due to the torque converter of the automatic gear box in the Audi Q7. The model proposed by Eidehall and the one enhanced with aerodynamic drag match the realistic vehicle model best. However, since the model of Eidehall is accurate enough and simpler than the enhanced version, the model proposed in (5.1) is used throughout this thesis.

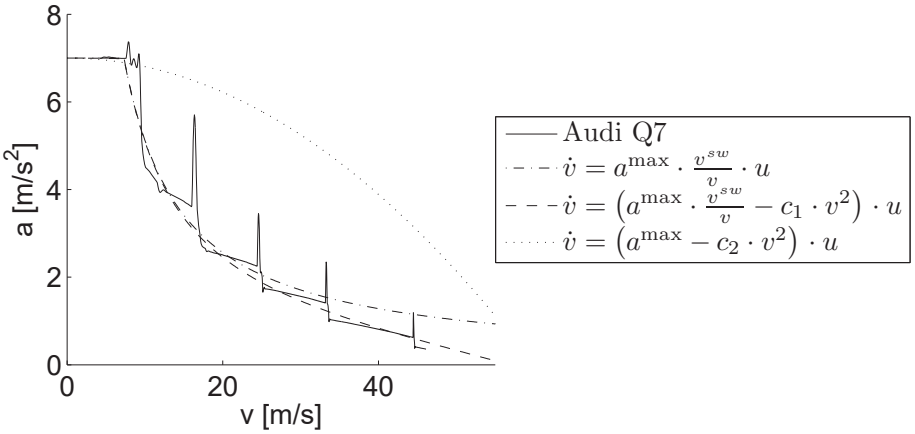


Fig. 5.6.: Maximum acceleration a of the Audi Q7 plotted over its velocity v; used parameters for the compared models: $a^{\max} = 7\,[\mathrm{m/s^2}]$, $v^{\mathrm{sw}} = 7.3\,[\mathrm{m/s}]$, $c_1 = 2.8e-4\,[\mathrm{m}]$, $c_2 = 1.9e-3\,[\mathrm{m}]$.

5.3.3. Violation of Traffic Regulations

The safety assessment approach presented in this thesis initially assumes that the traffic participants respect the traffic rules, e.g. they do not violate speed limits[2] or they do not drive in the wrong lanes (driving against oncoming traffic).

[2]In order to account for sporty drivers, the speed limit can be set higher than the official speed limit. The consideration of speed limits is presented in detail in Sec. 5.5.2.

The assumption that drivers stay in allowed lanes is considered by only allowing non-zero deviation probabilities within the span of the allowed lanes. Note that the determination of allowed lanes is not trivial since, for example, a vehicle may use a lane that is usually used by oncoming traffic if an obstacle has to be circumvented.

There are two strategies for considering violations of traffic regulations. One possibility is to assume that a non-conform driver is a reckless driver, e.g. the speed limit assumption, the assumption that this driver stays in allowed lanes, etc., is not considered anymore in the prediction. Another possibility is to only disable the speed limit regulation if only the speed limit is violated or only disable the allowed lanes assumption if this assumption is violated, etc. A satisfying answer for choosing the correct strategy is yet to be found.

The computation of the longitudinal probability distribution of traffic participants with Markov chains is presented next.

5.4. Abstraction of Traffic Participants to Markov Chains

The dynamic model of traffic participants (5.1) introduced in the previous section is hybrid with nonlinear continuous dynamics. Since this model is nonlinear, the enclosing hull method of Sec. 4.2 for the computation of stochastic reachable sets cannot be applied. However, due to the low dimensionality of the vehicle model, the Markov chain abstraction of Sec. 4.3 can be applied. The advantage of this approach is that the computationally intensive abstraction is computed offline. During the operation of the autonomous vehicle, the stochastic reachable sets of the Markov chains can be computed efficiently. It is again noted that the abstraction to Markov chains is only applied to other traffic participants, while the behavior of the ego vehicle is known from the trajectory planner.

In Sec. 4.3 on Markov chain abstraction, two methods were presented: Abstraction by Monte Carlo simulation and abstraction by reachability analysis. Monte Carlo simulation yields more accurate transition probabilities while reachability analysis yields a complete abstraction. It has also been shown that both methods can be combined so that both positive properties are unified. However, for the dynamics of traffic participants as specified in (5.1), there is no need for a complete abstraction. This is because the reachable position can be efficiently computed by two simulations, as shown in the next subsection. Thus, the probability that a crash may occur can be answered by the reachable positions, while the probability of a crash is answered by the stochastic reachable set. If there is contradicting information, i.e. a reachable set intersects, but the corresponding stochastic reachable set does not (due to an incomplete Markov chain abstraction), a low crash probability is assumed.

5.4.1. Reachable Set of Traffic Participants

This subsection presents an efficient computation of reachable sets for traffic participants as previously defined in Sec. 5.3. Writing the dynamics of a traffic participant as $\dot{x} = f^{TP}(x(t), u(t))$, where $x \in \mathbb{R}^2$ is the state and $u \in [-1, 1]$ is a Lipschitz continuous input,

the exact reachable set $\mathcal{R}^e(r)$ at time $t = r$ is defined as

$$\mathcal{R}^e(r) = \Big\{ x(r) = x(0) + \int_0^r f^{TP}(x(\tau), u(\tau))\, d\tau \Big| x(0) \in \mathcal{X}^0, u(\tau) \in [-1, 1] \Big\}.$$

In general, the exact reachable set of a system cannot be computed [106]. However, one can always compute an over-approximation as presented in Chap. 3. An example of the over-approximated reachable set of a traffic participant according to (5.1) for $u = 1$ and $t \in [0, 2]$ s is shown in Fig. 5.7 for two different initial sets. Additionally, sample trajectories starting from the initial set are shown, where the states at times $k \cdot \Delta t^*$, $k = 0 \dots 4, \Delta t^* = 0.5$ s are marked by a circle. If one is only interested in the reachable interval of the position and the velocity of a vehicle driving along a straight path, the following special case can be formulated:

Proposition 5.1 (Reachable Two-Dimensional Interval of the Vehicle State): Given is a vehicle driving along a straight path with dynamics subject to (5.1) and the initial condition $x(0) \in \mathcal{X}^0 = s(0) \times v(0)$ where $s(0) = [\underline{s}(0), \overline{s}(0)]$ and $v(0) = [\underline{v}(0), \overline{v}(0)]$ are the position and velocity intervals. The reachable, two-dimensional interval $\mathcal{X}(t) = [\underline{x}(t), \overline{x}(t)]$ of position and velocity is given by

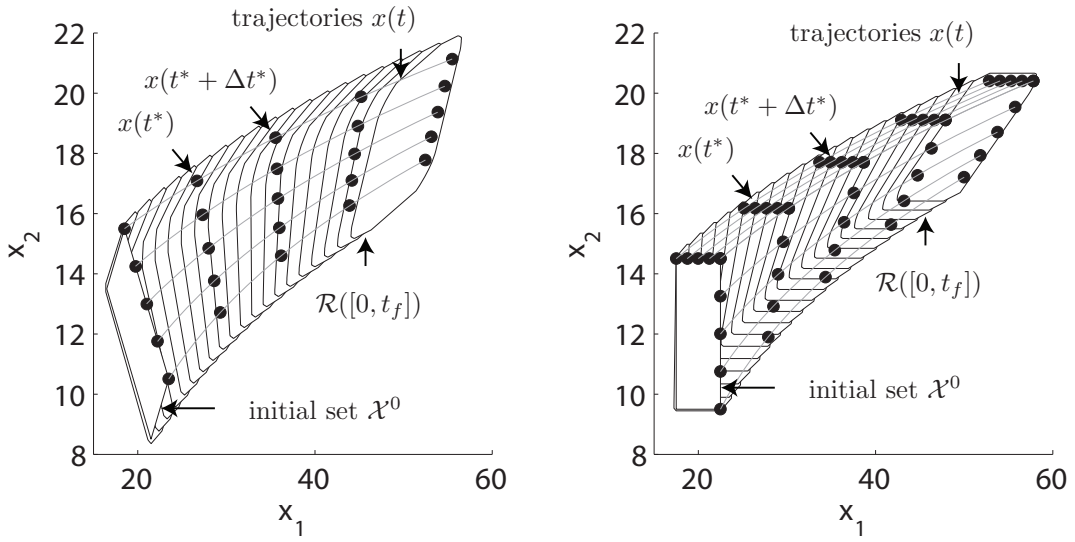
$$\begin{aligned} \underline{x}(t) &= \underline{x}(0) + \int_0^t f^{TP}(x(\tau), u(\tau))\, d\tau, \quad u(\tau) = -1 \\ \overline{x}(t) &= \overline{x}(0) + \int_0^t f^{TP}(x(\tau), u(\tau))\, d\tau, \quad u(\tau) = 1. \end{aligned}$$

□

The proof is omitted as it is obvious that the maximum position and velocity is obtained when the vehicle starts with the maximum initial position and velocity under full acceleration. The analogous argumentation holds for the lowest position and velocity. Note that this argumentation is only applicable if there exists an initial state that jointly contains the maximum initial position and velocity. This is always the case when the initial set is a two-dimensional interval, which is in contrast to the example of Fig. 5.7(a), for which Prop. 5.1 is not applicable. In this example, the maximum reachable position at different times is reached from trajectories starting from different initial states. However, if one is only interested in the reachable position – and the initial set is a two-dimensional interval, as shown in the example of Fig. 5.7(b), the result of Prop. 5.1 results in the exact reachable interval of the position coordinate.

If the path is curved, one has to consider the tire friction constraint in (5.2). For a given curvature profile $\rho(s)$, the minimum and maximum admissible input $\underline{u}(s)$ and $\overline{u}(s)$ can be obtained as presented e.g. in [167]. By exchanging $u(\tau) = -1$ with $u(\tau) = \underline{u}(s(\tau))$ and $u(\tau) = 1$ with $u(\tau) = \overline{u}(s(\tau))$ in Prop. 5.1, one can compute the reachable positions for a curved road. Speed limits on a road can be considered by cutting off the previously computed speed profile $v(s)$ at $v^{\max}$ by assigning $\overline{u}(s) = 0$ if $v(s) > v^{\max}$. Note that for a given input u (which is constant for a time step of the simulation), the position and velocity can be computed analytically:

Proposition 5.2 (Analytical Solution of the Longitudinal Dynamics): The analytical solution of the longitudinal dynamics of traffic participants in (5.1) for $u > 0$



(a) Initial values within a zonotope.

(b) Initial values within a two-dimensional interval.

Fig. 5.7.: Reachable sets of a vehicle for different initial sets. The vehicle model according to (5.1) with $a^{\max} = 7$ [m/s^2], $v^{\mathrm{sw}} = 7.3$ [m/s] is applied.

and $v > v^{\mathrm{sw}}$ is

$$s(t) = s(0) + \frac{(v(0)^2 + 2v^{\mathrm{sw}}\, u\, t)^{\frac{3}{2}} - v(0)^3}{3v^{\mathrm{sw}}\, u},$$
$$v(t) = \sqrt{v(0)^2 + 2v^{\mathrm{sw}}\, u\, t}.$$

The correctness can be easily verified by inserting the solution into (5.1). The analytical solution of the cases $0 < v < v^{\mathrm{sw}} \vee u < 0$ and $v \leq 0$ is trivial. □

In order to obtain the two-dimensional reachable positions on the road and not only the reachable positions along a path from the previous computation, it is generally assumed that the vehicle can laterally cover the whole lane if not stated differently. Another simple solution of reachable positions exists for vehicles with bounded absolute acceleration $a^{\max}$ on a two-dimensional plane [149].

The previously presented deterministic computations are supported by the Markov chains abstracting the original dynamics. Their generation is summarized in the following.

5.4.2. Offline Computations

Since the Markov chain does not have to be complete, its transition probabilities are computed by Monte Carlo simulation as shown in Sec. 4.3.2. In order to represent the movement of other traffic participants for the whole prediction horizon, the discretization region $\mathcal{X} = s \times v$ has to be properly chosen for the Markov chain abstraction. The maximum necessary region is as follows: The velocity interval ranges from standstill to the maximum considered speed $v = [0, v^{\max *}]$ and the position interval is $[0, v^{\max *} \cdot t_f + s^{\mathrm{detect}}]$, where t_f

is the fixed or maximal prediction horizon and s^{detect} the distance from which other traffic participants can be detected. However, due to efficiency reasons, smaller discretization regions $\mathcal{X}$ can be reasonable, too.

Based on the discretization, the transition matrices of the state are computed for points in time and time intervals as presented in Sec. 4.3.2. Besides different transition matrices for points in time and time intervals, transition matrices are also diversified by different choices of parameters in (5.1) which are listed in Tab. 5.1. The various transition matrices for different types of traffic participants are stored and loaded during the online procedure, which is addressed next.

5.4.3. Online Computations

During the online operation, a Markov chain for each detected traffic participant is instantiated whose state transition probability matrices $\tilde{\Phi}(\tau)$, $\tilde{\Phi}([0,\tau])$ are loaded from a database and τ is the time increment of Markov chains, see Sec. 4.3.2. For example, in a traffic scene with 2 cars and 1 bicycle, 2 Markov chains for the cars and 1 Markov chain for the bicycle are instantiated. The initial probability distributions $\tilde{p}(0)$ are generated according to the measurement uncertainties. Then, the Markov chain of each traffic participant is updated for the prediction horizon t_f according to (4.25) which is recapitulated for better readability:

$$\begin{aligned}\tilde{p}(t_{k+1}) &= \tilde{\Gamma}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}(t_k),\\ \tilde{p}([t_k, t_{k+1}]) &= \tilde{\Phi}([0,\tau])\,\tilde{p}(t_k),\end{aligned}$$

where $\tilde{\Gamma}(t_k)$ are the time varying input transition matrices which are generated from behavior models to be introduced in the next section. The times t_k are a short notation for $t_k = k \cdot \tau$.

Since the probabilities do not have to be computed in an over-approximative fashion as previously explained, one can cancel small probabilities and normalize the probability vector afterwards such that its sum is one. This procedure has the advantage that the computational time is reduced because the transition matrix and the probability vector are stored as a sparse matrix/vector which neglects zero entries. Special algorithms for sparse matrix multiplications allow a drastic increase in the efficiency of matrix multiplications, which are the faster the more zero entries exist (see e.g. [175]).

Heuristic 5.1 (Cancelation of Small Probabilities): Probabilities in $\tilde{p}$ which have a value of less than $\underline{p}$ are replaced by zeros so that one obtains $\tilde{p}^*$, which is normalized to the new probability vector $\tilde{p}_l = \tilde{p}_l^* / \sum_l \tilde{p}_l^*$. The value of $\underline{p}$ should be chosen in relation to the number of combined input and state cells $d \cdot c$ such that

$$\underline{p} = \frac{\varpi}{d \cdot c} \tag{5.3}$$

and ϖ can be freely chosen. □

5.5. Behavior Modeling

The dynamics of traffic participants has been modeled based on physical considerations in (5.1) and abstracted by Markov chains as described in the previous section. The Markov chains allow the computation of the probability distributions of other traffic participants when their sequence of input transition probabilities $\tilde{\Gamma}(t_k)$ is known, where the input values refer to the acceleration command. This sequence is also referred to as the behavior of traffic participants from now on. Besides the acceleration command, high-level decision probabilities such as the probability of taking a left turn or the probability of changing lane is also taken into account for the prediction of traffic participants. This is done by weighting possible paths of traffic participants according to high-level decision probabilities. In general, these probabilities are provided by other prediction algorithms, which often work with alternative methods such as Bayesian networks or neural networks. However, the computation of the high-level probability for changing lane is later addressed using the methods at hand. Besides this exception, only the generation of input transition matrices $\tilde{\Gamma}(t_k)$ is discussed below, because the high-level decision probabilities are provided by other software modules within the prototype vehicle [164].

Clearly, the input transition matrices $\tilde{\Gamma}(t_k)$ cannot be derived the same way as the state transition matrices $\tilde{\Phi}(\tau)$ and $\tilde{\Phi}([0,\tau])$, because the acceleration commands of other drivers cannot be described by differential equations. There are two main approaches for creating the input transition probability matrices $\tilde{\Gamma}(t_k)$. One possibility is to generate the transition probabilities based on heuristics. The other possibility is to learn the behavior of traffic participants based on traffic observations. In most works, those traffic observations are realized with static cameras (fixed position) and computer vision for the recognition of traffic participants in the camera image. Another possibility is to observe the traffic from a camera installed in a moving vehicle. Both configurations allow the recording of the trajectories of other traffic participants for the learning algorithms.

These trajectories can be clustered, resulting in motion primitives such as *left turn*, *lane change*, or *parking* [86]. A similar work focuses more on the probability distribution of trajectories within a cluster [88]. Those motion primitives could complement the paths of the traffic participants which are automatically generated from the road geometry and the finite set of decisions {*left turn*, *right turn*, *go straight*}, and {*left lane change*, *right lane change*} as introduced in Sec. 5.3. The advantage of this automatic clustering is that it covers more typical behaviors, so that one does not have to use the prediction algorithms for unstructured environments as a fallback solution so often.

The other objective for the recording of trajectories is to learn the behavior of traffic participants when following certain motion primitives. Alternatively, one can also learn behaviors directly from their two-dimensional movements on the road without grouping them into motion primitives. However, it is believed that the two-step approach of firstly learning motion patterns and secondly learning the dynamics along these motion patterns is more promising. In literature, various models are investigated in order to learn the behavior of traffic participants: Hidden Markov Models [123, 159], growing Hidden Markov Models [166], and switched ARX models [151].

Due to the lack of recorded trajectories of other traffic participants in various real world traffic situations, learning algorithms have not been applied, and heuristics are used in-

stead. The recording of the required trajectories by the prototype vehicles of the *Cognitive Automobiles* research project is part of future work, though. Unavailable trajectories of other traffic participants are also the reason why the applied heuristics have not yet been compared with real traffic data. The heuristic approach presented afterwards adapts the input probabilities (acceleration commands) based on the geometry of the road and the interaction with other traffic participants in *lane following*, *intersection crossing*, and *lane changing* situations. First, general properties of the input dynamics are introduced.

5.5.1. General Computation

The input transition values $\Gamma_i^{\alpha\beta}(t_k)$ are generated below, where the index i refers to the state and α, β are the final and initial value of the transition, respectively. For better readability, the update of the conditional input probabilities $q_i^{\beta}(t_k)$ according to the input transition values $\Gamma_i^{\alpha\beta}(t_k)$ is recalled from (4.22):

$$q_i^{\alpha}(t_k)' = \sum_{\beta} \Gamma_i^{\alpha\beta}(t_k) \cdot q_i^{\beta}(t_k).$$

The input transition probabilities $\Gamma_i^{\alpha\beta}$ are composed of two components. One component is a transition matrix Ψ which models the intrinsic behavior of the traffic participant when there are no priorities for certain input values. Priorities arise when e.g. a traffic participant is forced to brake due to a curve or a slower vehicle. Those priorities are modeled by a priority variable λ, which is the second component. The intrinsic transition matrix Ψ is introduced first and later combined with the priority λ.

The effect of the intrinsic transition matrix Ψ is discussed under the assumption that there are no priorities, such that $\Gamma_i^{\alpha\beta}(t_k) = \Psi^{\alpha\beta}$. Note that due to the intrinsic nature, $\Psi^{\alpha\beta}$ is independent of the time step t_k and the state value i. In order to generate a proper transition matrix Ψ, the normalization operator $\texttt{norm}()$ is introduced first:

$$\Psi^{\alpha\beta} = \texttt{norm}(\hat{\Psi}^{\alpha\beta}) := \frac{\hat{\Psi}^{\alpha\beta}}{\sum_{\alpha} \hat{\Psi}^{\alpha\beta}}.$$

The column sums of the resulting transition matrix Ψ are 1 in order to ensure that the multiplication with a probability vector results in a probability vector whose sum is 1. The transition probabilities of Ψ are set according to the heuristics that the bigger the change of the input[3], the more unlikely this change is. A transition matrix that considers this aspect is

$$\Psi^{\alpha\beta}(\gamma) = \texttt{norm}\big(\hat{\Psi}^{\alpha\beta}(\gamma)\big), \quad \hat{\Psi}^{\alpha\beta}(\gamma) = \frac{1}{(\alpha-\beta)^2 + \gamma}.$$

The parameter γ allows the gradual interpolation of the extreme cases $\lim_{\gamma\to 0} \Psi(\gamma) = I$ and $\lim_{\gamma\to\infty} \Psi(\gamma) = \frac{1}{c}\,\mathbf{1}$, where $\mathbf{1}$ is a matrix of ones and c is the number of inputs. Informally speaking, a low value of γ models drivers that rarely change their acceleration command, whereas a high value models drivers that often change their acceleration command. The higher the value of γ, the faster the initial input distribution converges to the steady state

[3] As the discrete inputs are numbered in increasing order according to the acceleration intervals, the difference between the input numbers is a measure of the change of the acceleration interval.

distribution, which is uniform over all inputs. This is illustrated in Fig. 5.8 for 3 inputs. High input numbers represent high positive acceleration, such that the first input $\mathbf{y} = 1$ represents full braking and the last input $\mathbf{y} = 3$ full acceleration. The initial probabilities are set to $P(\mathbf{y} = 1) = 0$, $P(\mathbf{y} = 2) = 0.8$, $P(\mathbf{y} = 3) = 0.2$ and the probabilities converge to $\frac{1}{3}$ as no prioritization is specified.

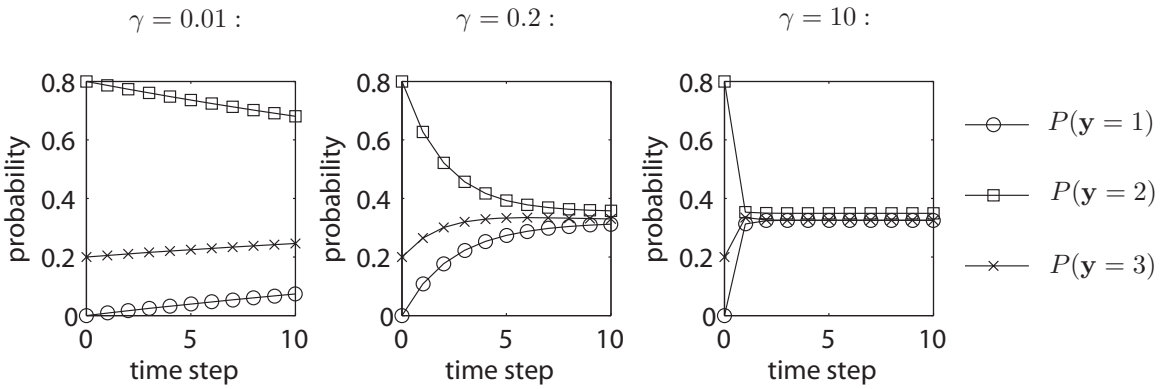


Fig. 5.8.: Input evolution for $\gamma = 0.01, 0.2, 10$.

Complementing the intrinsic transition matrix Ψ with the priority values of λ results in the input transition values $\Gamma_i^{\alpha\beta}$.

$$\begin{aligned} \Gamma_i^{\alpha\beta} &= \mathtt{norm}(\hat{\Gamma}_i^{\alpha\beta}), \\ \hat{\Gamma}_i^{\alpha\beta} &= \lambda_i^{\alpha} \cdot \Psi^{\alpha\beta}, \quad \forall i : \sum_{\alpha} \lambda_i^{\alpha} = 1,\ 0 \leq \lambda_i^{\alpha} \leq 1, \end{aligned} \tag{5.4}$$

where the state dependence of $\Gamma_i^{\alpha\beta}$ is solely modeled by the priority values λ_i^{α}, while the input dynamics matrix $\Psi^{\alpha\beta}$ is independent of the state. The above formula has the following special cases and properties:

- $\lambda_i^{\alpha} = 0$: Regardless of the intrinsic transition matrix Ψ, the input α of state i is prohibited ($q_i^{\alpha} = 0$).
- $\lambda_i^{\alpha} = \frac{1}{c}$, $\forall i, \alpha$ (c is the number of inputs): No input is prioritized, such that $\Gamma_i^{\alpha\beta} = \Psi^{\alpha\beta}$.
- $\Psi = I$ (I is the identity matrix): $\Gamma_i^{\alpha\beta} = I^{\alpha\beta}$, regardless of λ, such that the input probability is unchanged.
- $\Psi = \frac{1}{c}\mathbf{1}$ ($\mathbf{1}$ is a matrix of ones): The multiplication $\sum_{\beta} \Gamma_i^{\alpha\beta} \cdot q_i^{\beta}$ results in λ_i^{α}. Thus, a certain input probability distribution $q_i^{\alpha\prime} = \lambda_i^{\alpha}$ is enforced, regardless of the probability distribution of the previous time step.

Unfortunately, a formalism that generates values $\Gamma_i^{\alpha\beta}$ with the above-listed properties and additionally ensures the steady state solution $q_i^{\alpha}(t_{\infty}) = \lambda_i^{\alpha}$ has not been found. This solution would have the advantage that in the long run, drivers would always change their measured acceleration distribution to the one enforced by the priority values. The problem of creating a transition matrix which results in a certain steady state solution has been addressed in [117, 118]. However, there exist only solutions for so-called class $\mathcal{C}$ matrices and Γ does not belong to this class. Nevertheless, the steady state solution has values that

are at least similar to the priority values of λ and approximate the priority values better for higher γ values, so that for $\gamma \to \infty$: $q_i^\alpha(t_\infty) = \lambda_i^\alpha$. In a previous work of the author [186], the intrinsic transition matrix Ψ is not introduced, so that the input probability is $q_i^{\alpha\prime} = \lambda_i^\alpha$, which is equivalent to $\gamma \to \infty$.

It remains to compute the priority values $\lambda_i^\alpha(t_k)$ for different traffic situations. One aim is to alter λ so that constraints given by other traffic participants or the road geometry are met. The other considered effect for the priority values is that traffic participants have preferences for certain input values which are stored in the state independent motivation values μ^α ($\forall i : \mu_i^\alpha = \mu^\alpha$). The motivation values are equivalent to the priority values in the absence of constraints ($\lambda_i^\alpha = \mu_i^\alpha$).

In order to consider constraints, the event C of constraint satisfaction is introduced. Due to the uncertain modeling of traffic participants, constraint satisfaction is not guaranteed, such that the conditional probability of constraint satisfaction is introduced as $\eta_i^\alpha := P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha)$, where $\mathbf{z}$ and $\mathbf{y}$ is the random state and input of the Markov chain, respectively. Since constraints have to be met, the probability values η_i^α serve as an upper bound of the motivation values μ_i^α so that the priority values become

$$\lambda_i^\alpha = \begin{cases} \mu_i^\alpha, \text{ if } \mu_i^\alpha \leq \eta_i^\alpha \\ \eta_i^\alpha, \text{ otherwise.} \end{cases} \rightarrow \mu_i^{\alpha-1} := \mu_i^{\alpha-1} + \mu_i^\alpha - \eta_i^\alpha$$

In other words, μ_i^α is cut off at η_i^α and the cut-off probability is added to the next lower acceleration interval: $\mu_i^{\alpha-1} := \mu_i^{\alpha-1} + \mu_i^\alpha - \eta_i^\alpha$, which is also visualized in Fig. 5.9. The redistribution to lower acceleration intervals implies that a constraint is satisfied by lowering the acceleration. However, there are also situations such as overtaking maneuvers in which higher acceleration allows constraints to be satisfied. The integration of those maneuvers is the subject of future work.

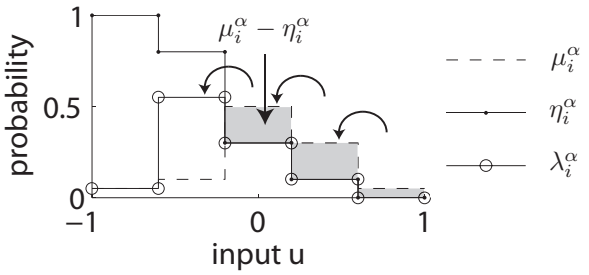


Fig. 5.9.: Combining a motivation distribution μ with a constraint distribution η.

The computation of constraint values for the capabilities *road following*, *vehicle following*, *intersection crossing* and *lane changing* is addressed in the next subsections.

5.5.2. Road Following

In this subsection, the computation of the constraint values $\eta^{\mathrm{road}}{}_i^\alpha$ with respect to acceleration limits and speed limits along curved paths is presented. For this, the possible velocity interval $[\underline{v}(s), \overline{v}(s)]$ resulting from the minimum and maximum possible acceleration in (5.2) is introduced. An additional labeling with brackets indicates the maximum absolute

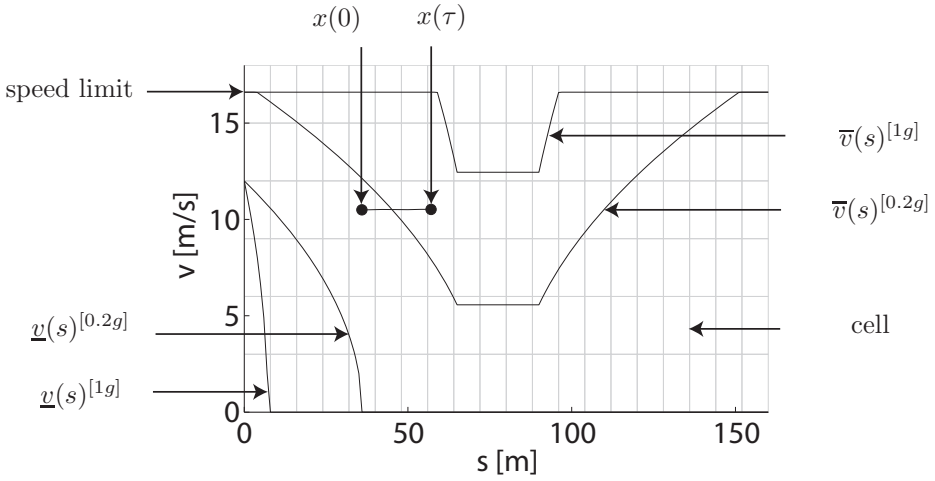


Fig. 5.10.: Velocity profiles for two straights connected by a $90°$ curve (15.5 m radius) and a speed limit of $v^{\max} = 16.7$ m/s $\hat{=}$60 km/h.

acceleration, e.g. $\overline{v}(s)^{[0.5g]}$ is the fastest possible velocity profile for $a^{\max} = 0.5g$ and g is the gravitational constant. Exemplary velocity profiles with a speed limit (possibly greater than the official speed limit to account for sporty drivers) are shown in Fig. 5.10.

The maximum acceleration constraint is violated when the velocity is outside the velocity profile bounds. However, this constraint may also be violated within the velocity bounds, when e.g. strongly accelerating within a curve while staying below $\overline{v}(s)$. Nevertheless, the event of constraint satisfaction C is defined such that it is true when the velocity is within the velocity profile bounds ($\underline{v}(s) \leq v \leq \overline{v}(s)$). This simplification is necessary for an efficient implementation and yields reasonable results within this framework, as shown later.

The compliance with the acceleration and the maximum velocity constraint for a state $\mathbf{z} = i$ and an input $\mathbf{y} = \alpha$ is approximately checked by a single simulation run[4]:

1. Simulate the vehicle for the time τ, starting from $x(0) = \texttt{center}(\mathcal{X}_i)$ under the effect of $u = \texttt{center}(\mathcal{U}^\alpha)$. The operator $\texttt{center}()$ returns the volumetric center of a set. Remember that $\mathcal{X}_i$ is the set of continuous states represented by the discrete state $\mathbf{z} = i$ and $\mathcal{U}^\alpha$ is the set of continuous inputs represented by the discrete input $\mathbf{y} = \alpha$.
2. Check whether the velocity is within the minimum and maximum velocity profile after one time increment τ (see also Fig. 5.10):

$$\underline{v}(s(\tau))^{[\bar{a}_d]} \leq v(\tau) \leq \overline{v}(s(\tau))^{[\bar{a}_d]}. \tag{5.5}$$

The constraint values are then obtained from the simulation results as

$$P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha, \mathbf{a} < \bar{a}_d) = \begin{cases} 1, & \text{if (5.5) holds} \\ 0, & \text{otherwise} \end{cases}.$$

Next, the probability distribution for applied accelerations $P(\mathbf{a} < \bar{a}_d)$ among all drivers is

[4]The lack of an over-approximative computation is appropriate since the reachable positions are computed separately in Prop. 5.1.

Tab. 5.2.: Discretization of the state and input space.

position interval s	$[0, 200]$ m
velocity interval v	$[0, 20]$ m/s
input interval $\mathcal{U}$	$[-1, 1]$
position segments	40
velocity segments	10
input segments	6
time increment τ	0.5 s

Tab. 5.3.: Behavior parameters.

γ	0.2
μ	$\begin{bmatrix}0.01 & 0.04 & 0.1 & 0.4 & 0.4 & 0.05\end{bmatrix}$
$q_i(0)$	$\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$ $(\forall i)$
$v^{\max}$	16 m/s

Tab. 5.4.: Initial state: Set with uniform distribution.

$s(0) \in$	$[2, 8]$ m
$v(0) \in$	$[12, 14]$ m/s

introduced, where $0 < \bar{a}_d \leq a^{\max}$ and $a^{\max}$ is the physically possible acceleration. The index d refers to a value within a finite set of selected absolute accelerations, e.g. $\{\bar{a}_1, \bar{a}_2, \ldots, \bar{a}_6\}$. Drivers that prefer a more comfortable ride have high probabilities for low values of $\bar{a}_d$ and the other way round for sporty drivers. The constraint values over all accelerations $\bar{a}_d$ are

$$\eta_i^\alpha = P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha) = \sum_d P(\mathrm{C}|\mathbf{z} = i, \mathbf{y} = \alpha, \mathbf{a} < \bar{a}_d) P(\mathbf{a} < \bar{a}_d).$$

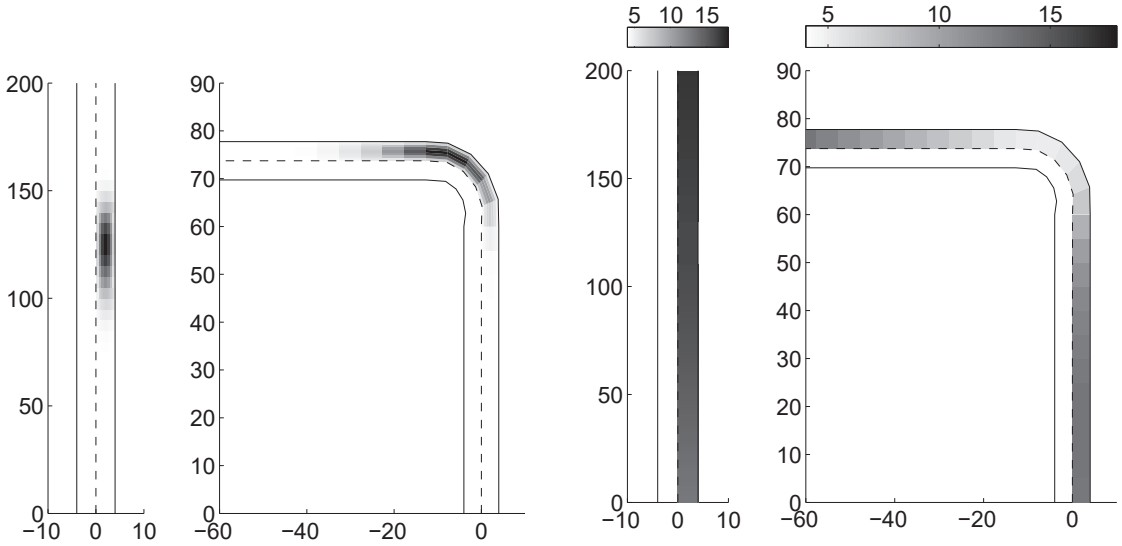
It is remarked that the probability of maximum applied acceleration $P(\mathbf{a} < \bar{a}_d)$ is not obtained online by observation of other drivers, but set according to an average distribution of all drivers. The effect of the constraint values η_i^α on road following is demonstrated for a vehicle that drives on a straight road and a curved road with the same initial states.

Example 5.1 (Straight versus Curved Road): The stochastic reachable sets of a vehicle is computed with identical initial states when following a straight and a curved road. The Markov chain for the vehicle is obtained from a discretization specified in Tab. 5.2. The parameters determining the behavior of the vehicle are shown in Tab. 5.3 and the initial state of the vehicle is listed in Tab. 5.4.

The probability distributions on the straight and curved road can be seen in Fig. 5.11(a) for $t \in [7.5, 8]$ s. The average velocity of the vehicle on different road segments in shown in Fig. 5.11(b), in which the color bar maps the gray tone to the average velocity. This figure nicely illustrates that the vehicle considers the speed limit of 16 m/s while decelerating in front of a curve and accelerating after it. The distributions of the longitudinal position, the velocity, and the input for the time interval $t \in [7.5, 8]$ s are plotted in Fig. 5.12. There, it can be observed that due to the curve, the vehicle on the straight road has traveled further.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.05 s for each scenario with a prediction horizon $t_f = 10$ s and cancelation of probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e-3$ (see Heuristic 5.1). □

The effect on the constraint values when following another vehicle is shown next.



(a) Probability distribution for $t \in [7.5, 8]$ s. The coordinate axes refer to positions in [m].

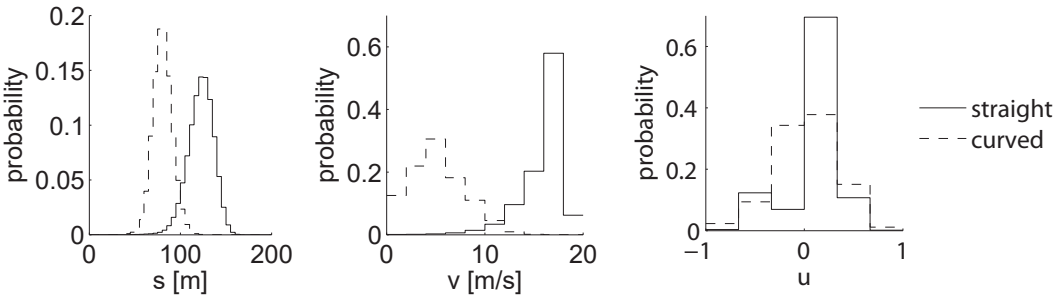
(b) Average velocity indicated by the color bar in [m/s] for $t \in [0, 10]$ s. The coordinate axes refer to positions in [m].

Fig. 5.11.: Probability distribution and average velocity on a straight and curved road.

5.5.3. Vehicle Following

Constraints not only arise from curved paths to be followed and speed limits to be respected, but also from other traffic participants. In this subsection, traffic participants following other traffic participants on the same lane are considered. The more complex cases of interaction at intersections and during lane changes are considered later.

Analogously to road following, the constraint values for vehicle following are computed based on simulations and heuristics. In order to distinguish the variables of the following vehicle from the ones of the leading vehicle, the variables of the following vehicle are denoted by a raised F (e.g. $\mathbf{z}^F$), and the variables for the leading vehicle by a raised L (e.g. $\mathbf{z}^L$). The constraint for the following vehicle is that its behavior should not cause



(a) Position histogram. (b) Velocity histogram. (c) Input histogram.

Fig. 5.12.: Histograms of position, velocity and input for $t \in [7.5, 8]$ s.

a crash. This is approximately checked by a single simulation[5], similarly as for the road following scenario:

1. Simulate both vehicles for the time interval $[0, \nu \cdot \tau]$, with constant $\nu \in \mathbb{N}^+$, starting from $x^F(0) = \texttt{center}(\mathcal{X}_i)$, $x^L(0) = \texttt{center}(\mathcal{X}_j)$ under the effect of $u^F = \texttt{center}(\mathcal{U}^\alpha)$, $u^L = \texttt{center}(\mathcal{U}^\beta)$.
2. Simulate a sudden and full brake beginning at $t = \nu \cdot \tau$ of the leading vehicle, to which the following vehicle reacts with a full brake, too. This behavior is simulated until the following vehicle has stopped at $t = t_S$.
3. Check if the following vehicle has crashed into the leading vehicle for $t \in [0, t_S]$.

The outcome of the simulation determines the conditional probability for satisfying the constraint of crashing with a probability of less than ϵ which is motivated by driver inattentiveness.

$$P(\mathrm{C}|\mathbf{z}^F = i, \mathbf{z}^L = j, \mathbf{y}^F = \alpha, \mathbf{y}^L = \beta, \Delta\mathbf{t} = \nu \cdot \tau) = \begin{cases} 1, & \text{no crash simulated} \\ \epsilon, & \text{otherwise,} \end{cases}$$

where $\Delta\mathbf{t}$ is the discrete random variable for the time interval with constant acceleration. Long time intervals model the behavior of foresighted drivers who adjust their acceleration early to changes of other drivers, while short time intervals represent sporty drivers. The probabilities $P(\Delta\mathbf{t} = \nu \cdot \tau)$ for time intervals in which the acceleration interval is unchanged, allows the computation of

$$P(\mathrm{C}|\underbrace{\mathbf{z}^F = i, \mathbf{z}^L = j, \mathbf{y}^F = \alpha, \mathbf{y}^L = \beta}_{D}) = \sum_{\nu} P(\mathrm{C}|D, \Delta\mathbf{t} = \nu \cdot \tau) \cdot P(\Delta\mathbf{t} = \nu \cdot \tau).$$

Note that the probability $P(\Delta\mathbf{t} = \nu \cdot \tau)$ is not obtained online by observation of other drivers, but set according to an average distribution of all drivers. The conditional probabilities $P(\mathrm{C}|D)$ are computed offline and stored according to the state and input indices in $\Theta_{ij}^{\alpha\beta}$ which allows the constraint values to be obtained for the vehicle following:

Proposition 5.3 (Constraint Vector for Vehicle Following): Under the assumption that $P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $P(\mathbf{z}^L = j, \mathbf{y}^L = \beta)$ are independent, one obtains

$$\eta_i^\alpha = \sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_j^{L\beta}. \qquad \square$$

Proof: After defining the events $\tilde{A} = (\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $\tilde{B}_j^\beta = (\mathbf{z}^L = j, \mathbf{y}^L = \beta)$, one can write:

$$\begin{aligned} P(\mathrm{C}, \tilde{A}) &= \sum_{j,\beta} P(\mathrm{C}|\tilde{A}, \tilde{B}_j^\beta) P(\tilde{A}, \tilde{B}_j^\beta) \\ &\overset{\text{independence}}{=} \sum_{j,\beta} P(\mathrm{C}|\tilde{A}, \tilde{B}_j^\beta) P(\tilde{A}) P(\tilde{B}_j^\beta) \end{aligned}$$

[5] The lack of an over-approximative computation is appropriate since the reachable positions are computed separately in Prop. 5.1.

$$\rightarrow \eta_i^\alpha = P(\mathrm{C}|\tilde{A}) = \frac{P(\mathrm{C},\tilde{A})}{P(\tilde{A})} = \sum_{j,\beta} \underbrace{P(\mathrm{C}|\tilde{A},\tilde{B}_j^\beta)}_{\Theta_{ij}^{\alpha\beta}} \underbrace{P(\tilde{B}_j^\beta)}_{p_j^{L\,\beta}}. \qquad \square$$

It is obvious that the independence assumption only approximates the joint probability $P(\tilde{A},\tilde{B}_j^\beta) \approx P(\tilde{A})P(\tilde{B}_j^\beta)$. Another issue is that the constraint value of the following vehicle for a single state i and input α is computed based on the complete probability distribution of the leading vehicle. The summation over all states j and inputs β of the leading vehicle in Prop. 5.3 leads to an averaging effect. The error caused by the averaging effect and the independence assumption are later evaluated in Sec. 5.7.6 when the Monte Carlo approach is compared to the Markov-chain approach.

In traffic situations with more than two vehicles on a lane, the simplification is made that each vehicle only reacts to the next vehicle driving in front and not to other vehicles.

The notation in Prop. 5.3 can be further simplified when using the probability vector $\tilde{p}$ as introduced in (4.23) containing all values p_j^β. This rewriting can be analogously applied to the constraint values of η:

$$\begin{aligned}\tilde{p}^T &= \begin{bmatrix} p_1^1 & p_1^2 & \dots & p_1^c & p_2^1 & p_2^2 & \dots & p_2^c & p_3^1 & \dots & p_d^c \end{bmatrix} \\ \tilde{\eta}^T &= \begin{bmatrix} \eta_1^1 & \eta_1^2 & \dots & \eta_1^c & \eta_2^1 & \eta_2^2 & \dots & \eta_2^c & \eta_3^1 & \dots & \eta_d^c \end{bmatrix}.\end{aligned}$$

After writing the interaction values $\Theta_{ij}^{\alpha\beta}$ into the interaction matrix

$$\tilde{\Theta} = \begin{bmatrix} \Theta_{11}^{11} & \Theta_{11}^{12} & \dots & \Theta_{11}^{1c} & \Theta_{12}^{11} & \Theta_{12}^{12} & \dots & \Theta_{12}^{1c} & \Theta_{13}^{11} & \dots & \Theta_{1d}^{1c} \\ \Theta_{11}^{21} & \Theta_{11}^{22} & \dots & \Theta_{11}^{2c} & \Theta_{12}^{21} & \Theta_{12}^{22} & \dots & \Theta_{12}^{2c} & \Theta_{13}^{21} & \dots & \Theta_{1d}^{2c} \\ \vdots & & & & & & & & & & \vdots \\ \Theta_{d1}^{c1} & \Theta_{d1}^{c2} & \dots & \Theta_{d1}^{cc} & \Theta_{d2}^{c1} & \Theta_{d2}^{c2} & \dots & \Theta_{d2}^{cc} & \Theta_{d3}^{c1} & \dots & \Theta_{dd}^{cc} \end{bmatrix},$$

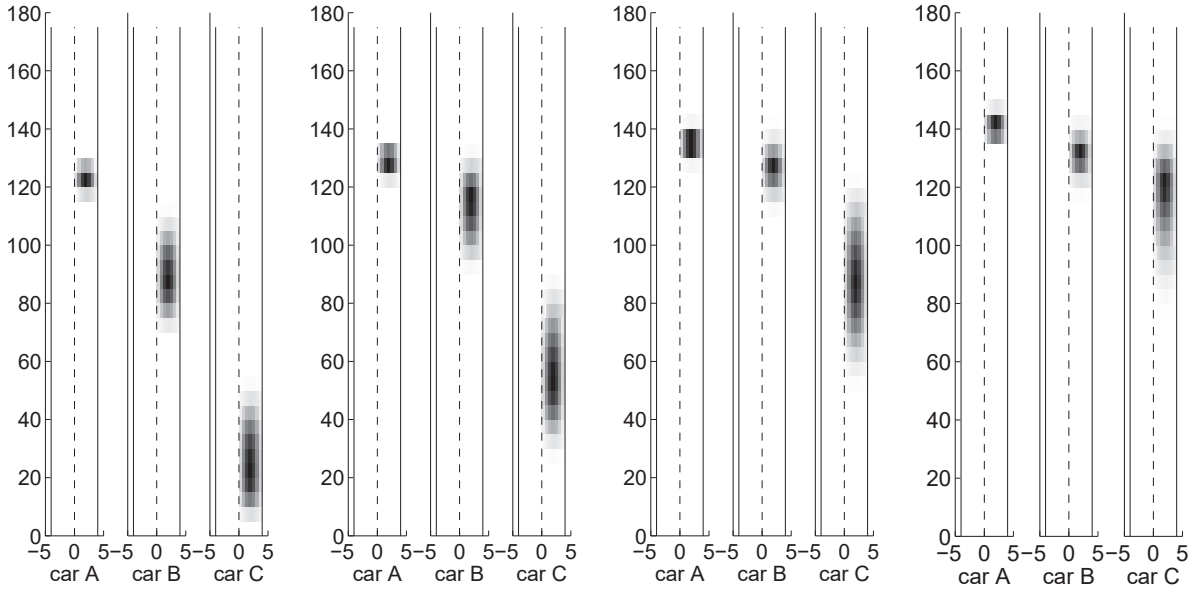
one can rewrite the computation of the constraint vector $\tilde{\eta}$ as $\tilde{\eta} = \tilde{\Theta}\,\tilde{p}^L$.

Note that the above equation is the only equation that has to be computed online in order to obtain $\tilde{\eta}$, since $\tilde{\Theta}$ is computed offline. When more than one constraint vector is active, e.g. the constraint vector for road following and vehicle following, the combined constraint vector is obtained by choosing the minimum constraint values elementwise ($\tilde{\eta} = \mathtt{min}(\tilde{\eta}^{\mathrm{road}}, \tilde{\eta}^{\mathrm{vehicle}})$) so that the constraints are simultaneously fulfilled in a probabilistic sense. The effect of the constraint vector is shown in the next example in which three cars drive in the same lane.

Example 5.2 (Vehicle Following): The interaction between vehicles driving in a lane is exemplarily shown for 3 cars driving one after the other. The cars are denoted by the capital letters A, B, and C, where A is the first and C the last vehicle in driving direction. Analogously to the previous example on road following, the Markov chain used for the vehicles B and C is obtained from the discretization in Tab. 5.2. Vehicle A is not computed based on a Markov chain, but predicted with a constant velocity of 3 m/s so that the faster vehicles B and C are forced to brake. The parameters determining the behavior of the vehicles B and C are as for the previous example in Tab. 5.3 and the initial state distributions of the vehicles $A - C$ are specified in Tab. 5.5.

The probability distributions for consecutive time intervals are plotted in Fig. 5.13. For visualization reasons, the position distributions are plotted in separate plots, although the vehicles drive in the same lane. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. The average velocity of the vehicles along the lane is shown in Fig. 5.14. The distributions of the position, velocity, and acceleration command for different vehicles and time intervals is shown in Fig. 5.15-5.17.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.21 s when canceling probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e{-}3$ (see Heuristic 5.1) for a prediction horizon of $t_f = 8$ s. □



(a) $t \in [0, 2]$ s. (b) $t \in [2, 4]$ s. (c) $t \in [4, 6]$ s. (d) $t \in [6, 8]$ s.

Fig. 5.13.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].

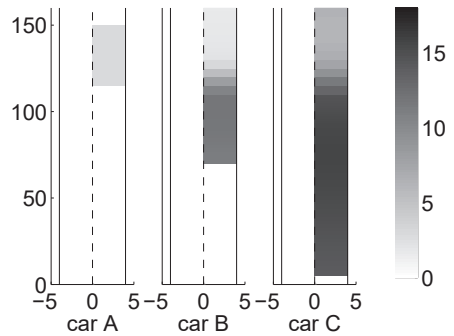
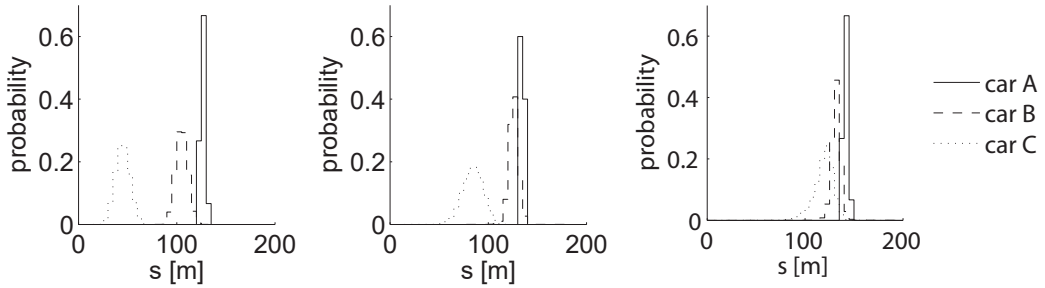


Fig. 5.14.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 8]$ s. The coordinate axes refer to positions in [m].

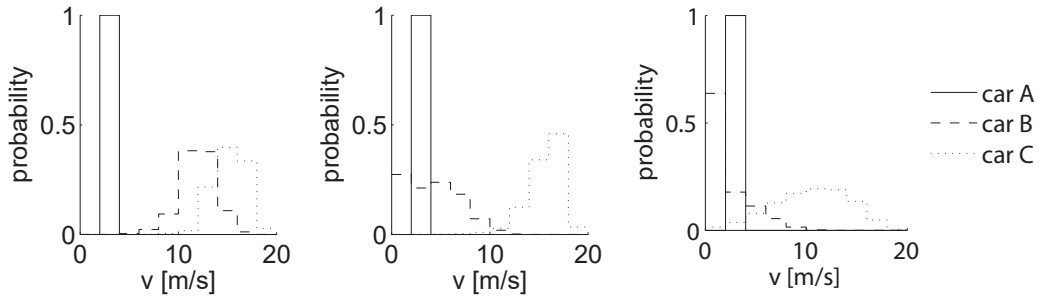
Tab. 5.5.: Initial state: Set with uniform distribution.

vehicle A	$s^A(0) \in [117, 123]$ m $v^A(0) \in [2, 4]$ m/s
vehicle B	$s^B(0) \in [72, 84]$ m $v^B(0) \in [10, 12]$ m/s
vehicle C	$s^C(0) \in [5, 17]$ m $v^C(0) \in [13, 15]$ m/s



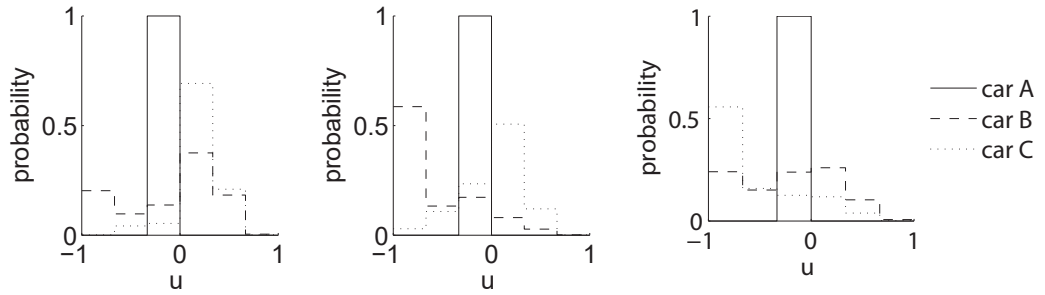
(a) $t \in [2, 2.5]$ s. (b) $t \in [4.5, 5]$ s. (c) $t \in [7, 7.5]$ s.

Fig. 5.15.: Position histograms for different time intervals.



(a) $t \in [2, 2.5]$ s. (b) $t \in [4.5, 5]$ s. (c) $t \in [7, 7.5]$ s.

Fig. 5.16.: Velocity histograms for different time intervals.



(a) $t \in [2, 2.5]$ s. (b) $t \in [4.5, 5]$ s. (c) $t \in [7, 7.5]$ s.

Fig. 5.17.: Input histograms for different time intervals.

The concept of vehicle following can be extended to scenarios at intersections with a few additional computations.

5.5.4. Intersection Crossing

The interaction results for two vehicles driving on a lane are extended to intersection scenarios. The behavior is differently computed for vehicles that have right of way and those that have not. The constraint vector of traffic participants having right of way are computed as previously described for road following and vehicle following, resulting in the combined constraint vector $\tilde{\eta} = \texttt{min}(\tilde{\eta}^{\text{road}}, \tilde{\eta}^{\text{vehicle}})$. The constraint vector for vehicles that

do not have right of way are computed in two phases (modes): *approaching* and *crossing*. First, the interaction at intersections is discussed for a simple example with two vehicles R (right of way) and N (no right of way) as depicted in Fig. 5.18, and extended later. Note that for each vehicle only one path through the intersection is considered for simplicity. This means no loss of generality, as further combinations of driving paths are computed the same way.

In *approaching* mode, a virtual vehicle V with zero velocity is put in front of the approaching vehicle where the path with right of way is crossing; see Fig. 5.18. This point is also referred to as the crossing position $s^{\text{cross},N}$ of vehicle N. The approaching vehicle N interacts with the virtual vehicle as previously described in Sec. 5.5.3 so that approaching a crossing is emulated by approaching a standing vehicle. As soon as the approaching vehicle N has entered the transition region to crossing mode (see Fig. 5.18), the mode *crossing* may become active. The length of the crossing region can be set by default to e.g. 10 m or be adapted to the intersection geometry if such information is available. Within the crossing region, the virtual vehicle V is replaced by a probabilistic virtual vehicle pV which has zero velocity, too, but its presence is modeled by the probability $p^{\text{cross}}(t_k)$ of crossing traffic. For each time interval $[t_k, t_{k+1}]$, the transition to *crossing* mode is activated for the fraction $1 - p^{\text{cross}}(t_k)$ of vehicle states within the crossing region. After the transition, the input probability distribution is reset in order to remove the deceleration behavior of the *approaching* phase.

The probability of crossing traffic p^{cross} is defined as the probability that a vehicle is crossing within the time span $\xi \cdot \tau$, where $\xi \in \mathbb{N}^+$ is a parameter that can be freely chosen. The crossing probability of one vehicle is computed as $p^{\text{cross}}(t_k) = \hat{p}(t_k - t_\xi) - \hat{p}(t_k)$, where $\hat{p}$ is the sum of position probabilities in front of the crossing. In the case of several crossing vehicles $1, \ldots, n$, the crossing probability is approximated by $p^{\text{cross}} = \texttt{min}(p^{\text{cross},1} + \ldots + p^{\text{cross},n}, 1)$ since the probability of crossing traffic cannot exceed 1.

It remains to extend the crossing procedure to the case when a vehicle simultaneously approaches a crossing and follows another vehicle. Analogously to the case when the vehicles are simultaneously constrained by the road and a vehicle driving ahead, it is sufficient to consider the constraint vector that is most restrictive. After introducing the constraint vector $\tilde{\eta}^{\text{inter}}$ for interaction with the virtual vehicle V, the overall constraint vector is computed as $\tilde{\eta} = \texttt{min}(\tilde{\eta}^{\text{road}}, \tilde{\eta}^{\text{vehicle}}, \tilde{\eta}^{\text{inter}})$.

The concept for intersection crossing is demonstrated by the following numerical example.

Example 5.3 (Intersection Crossing): In this example, it is assumed that each vehicle moves straight over the crossing. Additional possibilities of going left or right are neglected for simplicity, but can be computed analogously. The cars involved in this scenario are denoted by capital letters, where A, B have right of way and C, D do not have right of way; see Fig. 5.19. As for the previous examples, the Markov chains used for the vehicles are obtained according to the discretization in Tab. 5.2 and the parameters determining the behavior of the vehicles are in Tab. 5.3. The uniform distributions of initial states are specified in Tab. 5.6.

The probability distributions for selected time intervals are plotted in Fig. 5.19. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle.

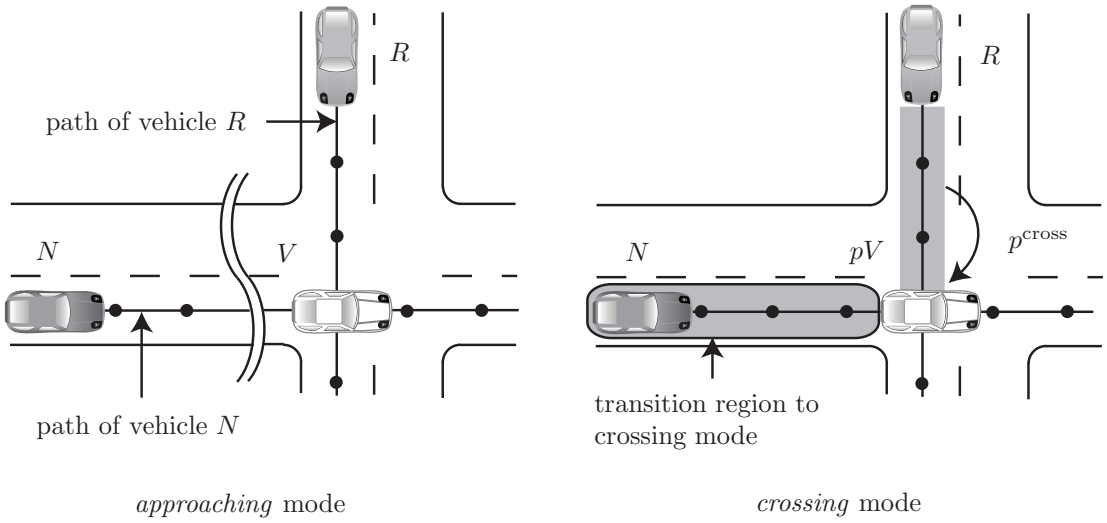
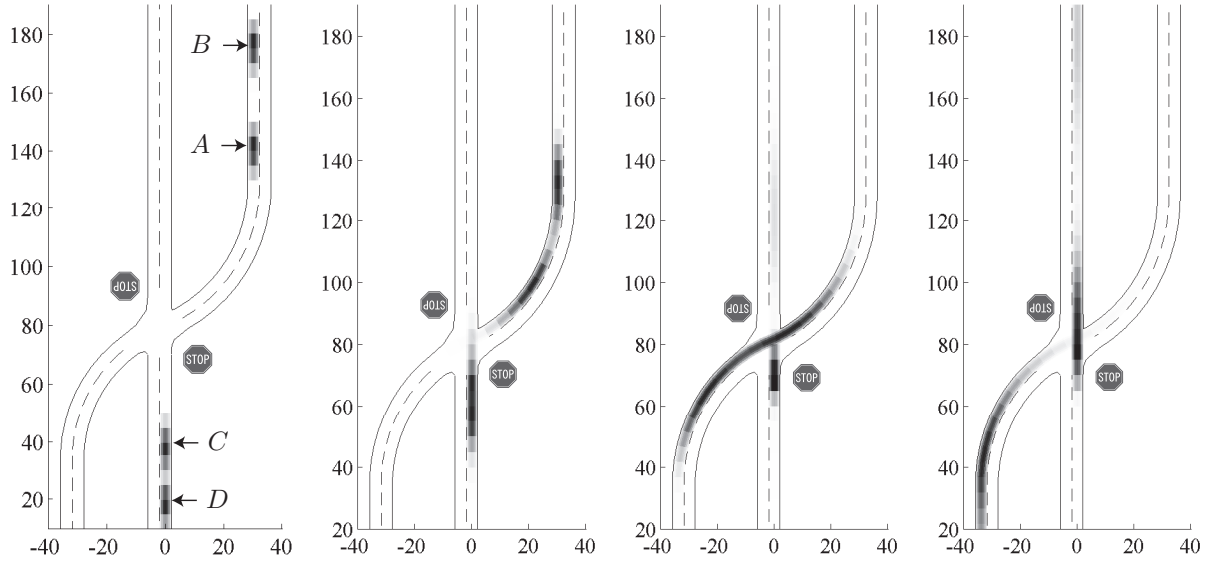


Fig. 5.18.: Intersection scenario: Approaching and crossing mode.

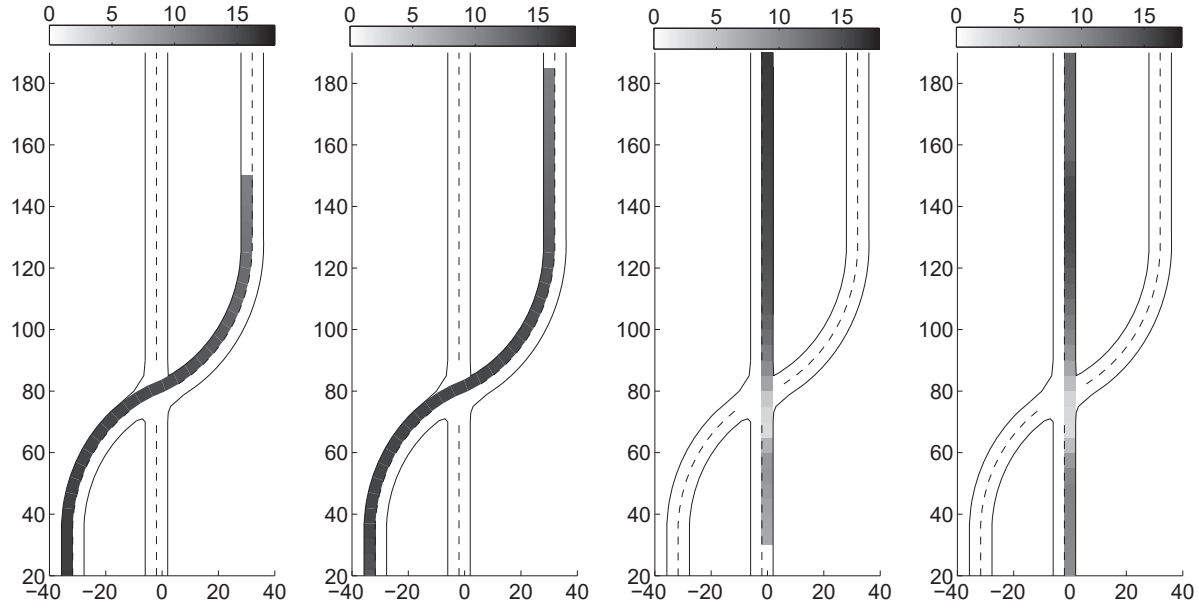
The average velocity of the vehicles along the lane is shown in Fig. 5.20. One can see that the vehicles without right of way are forced to brake. The crossing probability determined by the probability distributions of crossing vehicles A and B is plotted in Fig. 5.21.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 1.68 s when canceling probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e{-}3$ (see Heuristic 5.1) for a prediction horizon of $t_f = 11$ s. □



(a) $t \in [0, 0.5]$ s. (b) $t \in [3.5, 4]$ s. (c) $t \in [7, 7.5]$ s. (d) $t \in [10.5, 11]$ s.

Fig. 5.19.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].



(a) vehicle A. (b) vehicle B. (c) vehicle C. (d) vehicle D.

Fig. 5.20.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 11]$ s. The coordinate axes refer to positions in [m].

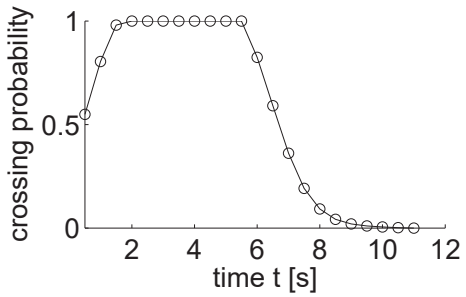


Fig. 5.21.: Probability p^{cross} of the virtual vehicle.

Tab. 5.6.: Initial state: Set with uniform distribution.

vehicle A	$s^A(0) \in [40, 52]$ m $v^A(0) \in [10, 12]$ m/s
vehicle B	$s^B(0) \in [5, 17]$ m $v^B(0) \in [10, 12]$ m/s
vehicle C	$s^C(0) \in [25, 37]$ m $v^C(0) \in [6, 8]$ m/s
vehicle D	$s^D(0) \in [5, 17]$ m $v^D(0) \in [8, 10]$ m/s

5.5.5. Lane Changing

Similar to the intersection crossing behavior, the lane change behavior requires a discrete decision. At intersections, the probability for crossing has to be modeled, and on a multi-lane road, the probability for a lane change has to be modeled. The lane change probability could be forwarded by an external algorithm. In the area of lane change recognition, most work focuses on lane change assistance for human drivers within the ego vehicle; see e.g. [120]. These systems warn the driver if a lane change is predicted and if the lane change is dangerous, where the latter problem is addressed in e.g. [89, 93]. Further, it has been shown that lane change prediction increases the acceptance of automatic cruise control (ACC) systems [65].

Lane change prediction of surrounding vehicles [45] is not so much investigated. Factors contributing to a lane change are discussed in [29]. Besides the prediction of the lane change decision, possible lane change trajectories using HMMs are generated in [126, 127].

In this work, another possibility to estimate the probability of a lane change maneuver is presented. This approach only uses data that has been computed from the Markov chain updates. In order to obtain an algorithm for traffic prediction under possible lane changes which can be computed online, some simplifications have to be made.

Simplifications and Restrictions

The consideration of lane changes adds a significant amount of complexity to the computation of stochastic reachable sets, as will be shown later. For this reason, certain restrictions for lane changes are proposed such that their consideration preserves the online capability of this approach. In order to discuss the restrictions, a scenario with 3 lanes is set up in Fig. 5.22. Note that vehicle A is the autonomous vehicle, while vehicles $B - F$ are surrounding traffic participants. The restrictions are as follows:

1. Lane changes are only considered for vehicles starting within a certain region around the autonomous vehicle: $s^A(0) - \underline{\eta} \leq s \leq s^A(0) + \overline{\eta}$, where $s^A(0)$ is the position of the autonomous vehicle at $t = 0$. Thus, lane changes are only considered for vehicles D and E in Fig. 5.22.
2. A vehicle does a lane change only once within the prediction horizon, i.e. changing two lanes or changing the lane and returning to the original lane is not considered.
3. There is no interaction between vehicles that may change to the same lane. Thus, the probability that vehicle D changes to the middle lane is computed independently of the probability that vehicle E changes to the middle lane.

From the second and third restriction follows that, in principle, the lane changes on a road with n lanes can be broken down to lane changes on several virtual roads with two lanes. For this reason, only two lanes with traffic participants $A - D$ are considered for lane changes from now on. It is remarked that it has not been checked if the assumptions comply with real traffic, which is part of future work. However, the first assumption is reasonable as human drivers also limit their prediction for possible lane changes to the vicinity of their own vehicle. The second assumption is justified as a single lane change takes about 6 seconds, which is similar to the prediction horizon t_f for lane change scenarios [129]. The last assumption does not strongly influence the overall result as the event that two vehicles change to the same lane within the prediction horizon t_f is rare. In the next subsection, the computation of the lane change probability is discussed.

Lane Change Probability Approximation

The following considerations are made for the situation in Fig. 5.22, but for two lanes with vehicles $A - D$, whereof vehicle D performs the lane change. This is no loss of generality as lane change maneuvers can be broken down to this case using the previously discussed assumptions. The probability of a lane change is heuristically obtained from three factors:

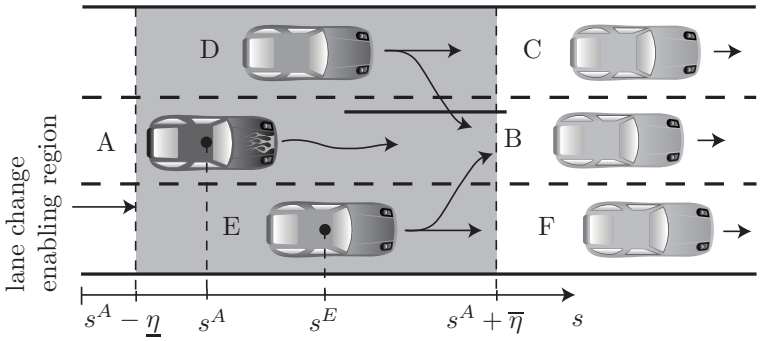


Fig. 5.22.: Considered actions for a three-lane scenario.

1. the motivation σ^{cl} for driving on the current lane,
2. the motivation σ^{nl} for driving on the neighboring lane,
3. the convenience σ^{conv} of the new following vehicle after the lane change is performed.

The motivation values σ^{cl} and σ^{nl} are obtained from the constraint values η_i^α of vehicle D following vehicle C or B, respectively. Similarly, the convenience value σ^{conv} is computed from the constraint values of vehicle A when following vehicle D after the lane change. As an intermediate step, the probabilities e^α of meeting the constraint event C for a given acceleration input are computed:

$$e^\alpha := P(\mathrm{C}|\mathbf{y}=\alpha) = \sum_i P(\mathrm{C}|\mathbf{z}=i,\mathbf{y}=\alpha)P(\mathbf{z}=i) = \sum_i \eta_i^\alpha \hat{p}_i,$$

where independency of $P(\mathbf{z}=i)$ and $P(\mathbf{y}=\alpha)$ is assumed as in Prop. 5.3. Next, the distributions $e^{C,\alpha}$, $e^{B,\alpha}$ and $e^{A,\alpha}$ for interaction with the vehicles A, B, and C are weighted by values stored in a vector $\overline{w}$ for the vehicles in front and a different vector $\underline{w}$ for the new following vehicle:

$$\sigma^{\text{cl}} = \sum_\alpha \overline{w}^\alpha e^{C,\alpha}, \quad \sigma^{\text{nl}} = \sum_\alpha \overline{w}^\alpha e^{B,\alpha}, \quad \sigma^{\text{conv}} = \sum_\alpha \underline{w}^\alpha e^{A,\alpha}.$$

For the motivation values σ^{cl}, σ^{nl}, the weights are chosen such that high positive acceleration has high weights, because the possibility of acceleration motivates driving in a particular lane. For the convenience value σ^{conv}, the weights are chosen such that braking has low values, since it is inconvenient for the new following vehicle to be forced to brake while all positive acceleration inputs have similar weights. Finally, the probability for a lane change p^{lc} is heuristically computed as

$$p^{\text{lc}} = \frac{2}{\pi} \arctan\left(b \frac{\sigma^{\text{nl}}}{\sigma^{\text{cl}}} \frac{\sigma^{\text{conv}}}{\sum \underline{w}} \right).$$

The heuristic is chosen such that a motivation ratio $\frac{\sigma^{\text{nl}}}{\sigma^{\text{cl}}}$ in favor of the neighboring lane as well as a high convenience value σ^{conv} motivates a lane change. The arctan function allows the modeling of the saturation ($\mathbb{R}^+ \to [0, 0.5\pi]$), but can also be replaced by a similar function. The summation $\sum \underline{w}$ is the maximum possible value for σ^{conv}, which normalizes the convenience value. This is not necessary for σ^{nl}, σ^{cl} as one is only interested in their ratio. Where there is no vehicle B or C, the corresponding σ values are set to $\sum \overline{w}$ or $\sum \underline{w}$. The parameter b is a tuning parameter that has to be found from traffic

observations. Besides the probability of a lane change, the lane change maneuver itself has to be considered, which is presented next.

Longitudinal and Lateral Probability Distributions

The probability for a lane change obtained from the previous scenario with vehicles $A - D$ strongly influences the future probability distribution. In order to refer to the cases when vehicle D drives on the left or right lane, the notation Dl and Dr is used, respectively. The longitudinal probabilities for the left and right lane are computed similarly as in previous scenarios. The extension is that the probability for a lane change decreases the probability of driving in one lane while it increases the probability in the other lane. Additionally, the input probability distribution after the lane change has to be changed to the distribution of the new lane by the input reset operator, which is defined as $\texttt{inputReset}(\Delta p_i^{\alpha}(t_k)) := \lambda_i^{Dr,\alpha}(t_k) \sum_{\alpha} \Delta p_i^{\alpha}(t_k)$ when changing to the right lane and $\Delta p_i^{\alpha}(t_k)$ are the probability values that are added to the right lane. The enhanced probability update of (4.25) when changing from left to right is

$$\begin{aligned}
\Delta\tilde{p}(t_k) &= p^{\mathrm{lc}}(t_k)\tilde{p}^{Dl}(t_k), \\
\tilde{p}^{Dl}(t_{k+1}) &= \tilde{\Gamma}^{Dl}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}^{Dl}(t_k) - \Delta\tilde{p}(t_k), \\
\tilde{p}^{Dr}(t_{k+1}) &= \tilde{\Gamma}^{Dr}(t_k)\,\tilde{\Phi}(\tau)\,\tilde{p}^{Dr}(t_k) + \texttt{inputReset}(\Delta\tilde{p}(t_k)),
\end{aligned}$$

while the updates for time intervals are identical to (4.25).

Besides the longitudinal probability distribution, the lateral distribution has to be changed as well. For this, the lane change is divided into phases with time span $\tau = t_{k+1} - t_k$. After each time step, the probabilities are shifted to the next phase until the lane change has been completed. This is illustrated in Fig. 5.25, where t^{lc} is the time passed since the lane change was initiated and the gray areas indicate the phases of driving in the initial and final lane. The probabilities of the lane change phases are denoted by $\hat{p}_l^{\mathrm{lc}}$ and the index refers to the l-th phase. In order to account for the uncertainty in the lateral position during a lane change, a deviation probability $f_l(\delta)$ is defined for each phase. The final deviation probability, which is spanned over the initial and neighboring lane, is computed as $f(\delta) = \sum_l \hat{p}_l^{\mathrm{lc}} \cdot f_l(\delta)$. The lane change probabilities and stochastic reachable sets for a lane change scenario are computed in the following example.

Example 5.4 (Lane Changing): The considered traffic situation is depicted in Fig. 5.23 with vehicles $A - D$, where vehicle A is the autonomous vehicle and a lane change is only considered for vehicle D. The traffic situation can easily be extended to three lanes as previously discussed. In order to motivate a lane change for vehicle D, the initial speed of vehicle C is chosen lower than for the vehicles A and B in the adjacent lane.

The parameters and settings are identical to the ones found in Tab. 5.2 and 5.3, except that the values of the conditional input probability $q_i(0)$ differ from vehicle to vehicle. The initial state distributions are presented in Tab. 5.7 and the weighting vectors for the lane change probability are $\overline{w} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 4 \end{bmatrix}$, $\underline{w} = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$ and $b = 0.11$.

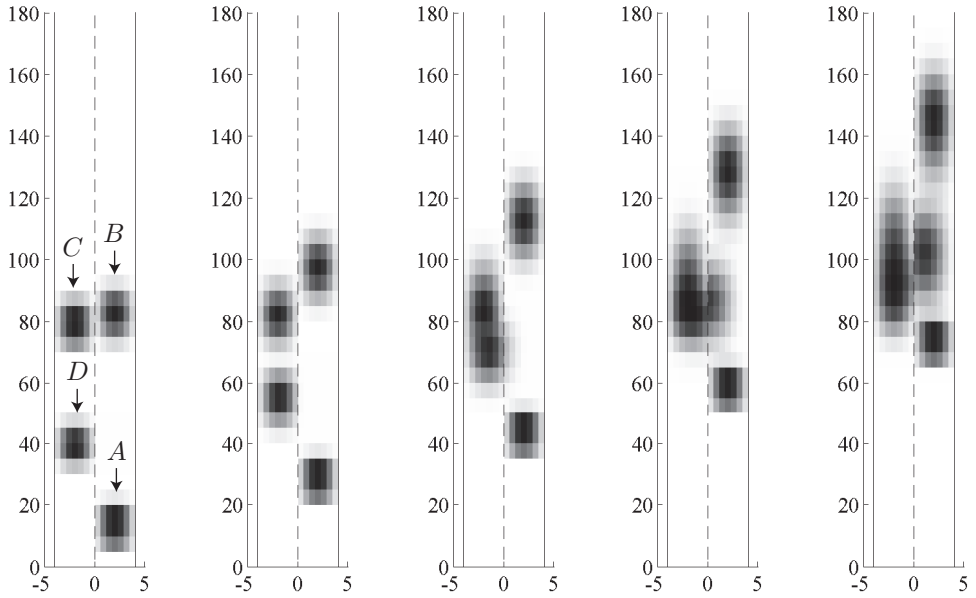
The probability distributions on the road are displayed in Fig. 5.23 for 5 selected time intervals. Dark regions indicate high probability, while bright regions represent areas of

Tab. 5.7.: Initial state: Set with uniform distribution.

vehicle A	vehicle B	vehicle C	vehicle D
$s^A(0) \in [7, 13]$ m $v^A(0) \in [14, 16]$ m/s	$s^B(0) \in [72, 84]$ m $v^B(0) \in [14, 16]$ m/s	$s^C(0) \in [72, 84]$ m $v^C(0) \in [4, 8]$ m/s	$s^D(0) \in [30, 40]$ m $v^D(0) \in [14, 16]$ m/s

low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. The average velocities are displayed in Fig. 5.24. Further, the normalized motivation values σ^{cl}, σ^{nl}, convenience value σ^{conv}, and lane change probability p^{lc} are plotted in Fig. 5.26 and the probability distribution $\hat{p}_l^{\text{lc}}$ of the lane change phases at $t = 4$ s is plotted in Fig. 5.25.

The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.43 s when canceling probabilities below $p^{\max} = 10/(d \cdot c) = 4.2e-3$ for a prediction horizon $t_f = 5$ s.



(a) $t \in [0, 0.5]$ s. (b) $t \in [1, 1.5]$ s. (c) $t \in [2, 2.5]$ s. (d) $t \in [3, 3.5]$ s. (e) $t \in [4, 4.5]$ s.

Fig. 5.23.: Position distribution for different time intervals. The coordinate axes refer to positions in [m].

5.5.6. Known Behavior

Finally, the case is considered when the future behavior of other traffic participants is known. The planned trajectory $\hat{x}(t)$ of other vehicles is available when e.g. autonomous vehicles broadcast their planned trajectories to other autonomous vehicles. The planned

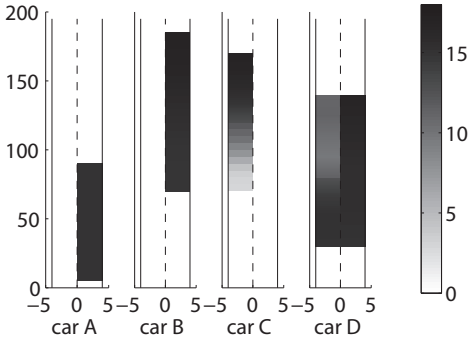


Fig. 5.24.: Average velocity indicated by the color bar in [m/s] for $t \in [0, 5]$ s. The coordinate axes refer to positions in [m].

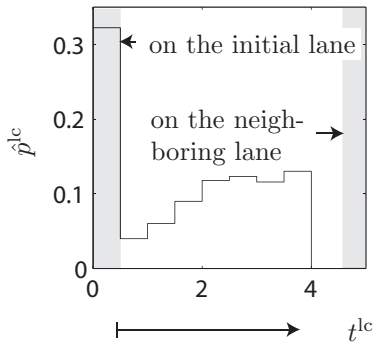


Fig. 5.25.: Probability of lane change phases at $t = 4$ s.

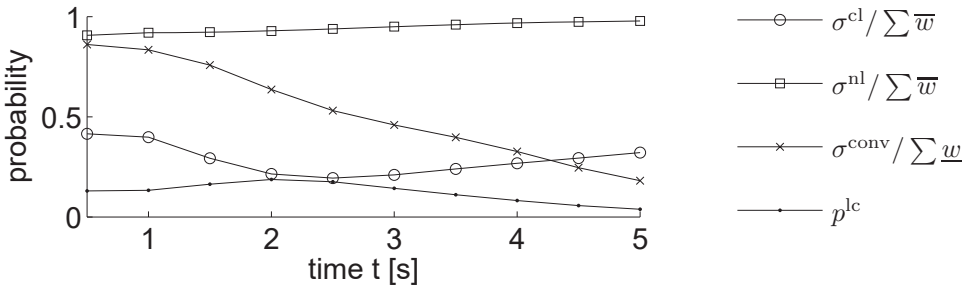
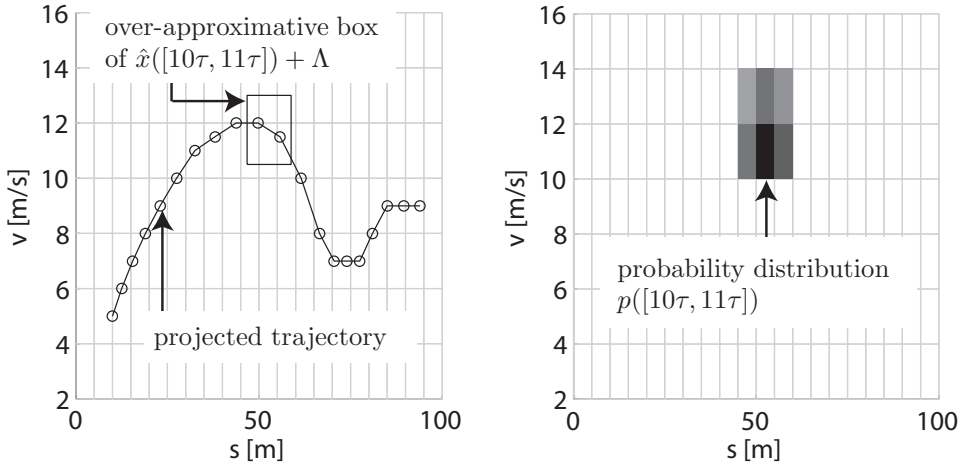


Fig. 5.26.: Motivation values.

trajectory of the ego vehicle is always known since the trajectory planner and the safety verification module are connected (see Fig. 5.1).

In practice, the planned trajectory cannot be perfectly realized, which is modeled by a bounded uncertainty Λ so that $x(t) \in \hat{x}(t) + \Lambda$, where the set Λ is translated by $\hat{x}(t)$. The connection to the probabilistic description of vehicles is established by first projecting the planned trajectory $\hat{x}(t)$ and the bounded uncertainty Λ onto the position and the velocity coordinate; see Fig. 5.27(a). It is assumed that the probability is uniformly distributed within Λ so that the probability of a state space cell is determined by the fraction of the volume in a cell. This is exemplarily presented for the uncertainty in Fig. 5.27(a), whose corresponding probability distribution within the discretized state space is shown in 5.27(b). The same procedure is performed for the input space.

In this section, the behavior of vehicles for road following, vehicle following, intersection crossing, and lane changing has been modeled. It remains to compute the probability of a crash of the ego vehicle with other vehicles. This is based on the probability distributions of all vehicles in the traffic scene, which is discussed next.



(a) Projected trajectory. (b) Discrete probability distribution.

Fig. 5.27.: Mapping of known behavior to discrete probabilities.

5.6. Crash Probability

In this section, the crash probability of the ego vehicle with all other vehicles in the traffic scene is estimated. The other question of guaranteeing whether a crash may occur or not, can be answered by checking if the reachable positions of the ego vehicle intersect with the ones of all other vehicles, which are computed as presented in Sec. 5.4.1. In the case of a conflict, i.e. the estimated crash probability is zero, but an intersection of reachable sets has been detected, a low crash probability is assumed.

Before the approach for the computation of the crash probability is presented, some assumptions are made which allow the crash probability as it is understood in this work to be defined. The purpose of the safety assessment of autonomous vehicles is to obtain a measure for the threat at different points in time when the planned trajectory is executed as planned. This plan should be strictly followed by the autonomous car, which implies that the errors made when following the planned trajectory are independent of the future behavior of other traffic participants. Note that the compliance to the current plan during the safety assessment is no contradiction to the continuous updates of the trajectory planner; see Fig. 5.1.

Another aspect is that the crash probability is computed independently of previous crash probabilities in this work. This has the advantage that for each point in time, a situation can be judged independently of previous occurrences. This is not the case, when computing the physical probability that a crash will happen. Consider a scenario in which two situations are equally dangerous at two different points in time. However, the probability that the vehicle crashes in the first situation is much greater than in the second situation, because a crash can only occur in the second situation, if the vehicle survived the first situation and crashes in the second situation. For this reason, it is assumed that the autonomous vehicle has not crashed until the investigated point in time or time interval. One can also say that the physically motivated crash probability is the total probability of a crash, while the definition used in this work is the conditional probability of a crash

under the condition that no crash has happened yet. Clearly, if the probability of a crash is low for all points in time, both definitions yield similar results.

The definition is detailed for a situation with only one other traffic participant. For a better distinction of variables from the other vehicle with the ones of the ego car, all variables referring to the ego car are indexed by a hat ($\hat{\square}$) from now on.

Definition 5.1 (Conditional Crash Probability): Given are the vectors $\hat{\xi}, \xi \in \mathbb{R}^\eta$ which uniquely define the position and orientation of the ego vehicle and another vehicle, respectively. The probability distributions at time t_k are denoted by $f(\xi, t_k)$ and $\hat{f}(\hat{\xi}, t_k)$. For both distributions, it is assumed that the ego vehicle has not yet crashed. The indicator function $\texttt{ind}(\xi, \hat{\xi})$ is 1 if the bodies of the ego vehicle and the other vehicle intersect and 0 otherwise. Under the previously discussed assumptions that the probability distributions $f(\xi)$ and $\hat{f}(\hat{\xi})$ are independent, and the ego vehicle has not crashed until the investigated point in time t_k, the conditional crash probability $p^{\text{crash}}(t_k)$ is defined as

$$p^{\text{crash}}(t_k) = \int_{\mathbb{R}^\eta} \int_{\mathbb{R}^\eta} f(\xi, t_k) \cdot \hat{f}(\hat{\xi}, t_k) \cdot \texttt{ind}(\xi, \hat{\xi})\, d\hat{\xi}\, d\xi. \qquad \square$$

In this work, ξ and $\hat{\xi}$ are two-dimensional position vectors of the vehicle centers. The orientation is indirectly specified since the vehicles are always oriented according to the tangential vector of their path.

This definition is extended to several traffic participants by computing the partial crash probabilities with other traffic participants. The final conditional crash probability is obtained by summing over all partial probabilities. Note that the above definition can only be applied to the ego vehicle and not to other traffic participants, since their probability distributions are no longer independent.

5.6.1. Computational Realization

For the implementation of the above definition, the probability distribution $f(\xi, t_k)$ of traffic participants has to be formalized. Since the path and the deviation along the path are segmented, and the probability distribution is uniform within one segment, one obtains a piecewise constant probability distribution in $\mathbb{R}^2$ for $f(\xi, t_k)$. In order to obtain regions of constant probability distribution with simple geometry, waypoints on the paths of other traffic participants are extracted every path segment distance. These waypoints are connected to a simplified path consisting of connected straight lines. The straight lines are also referred to as the simplified path segments s_g, where g is the path index. The deviation is also subdivided into segments d_h, where h is the deviation index. The segmentation of the path and the deviation can be observed in Fig. 5.28 and 5.4.

The region that is spanned when the path coordinate s is within s_g and the deviation coordinate δ is within d_h is denoted by $\mathcal{C}_{gh}$. The regions $\mathcal{C}_{gh}$ are trapezoids and the union of all trapezoids results in a path-aligned occupancy grid. A further region that is of interest, is the region that is occupied by the body of a vehicle when the center of the body is within $\mathcal{C}_{gh}$. This set is denoted by $\mathcal{B}_{gh}$ and its orientation equals the direction of the path segment g, as illustrated in Fig. 5.28.

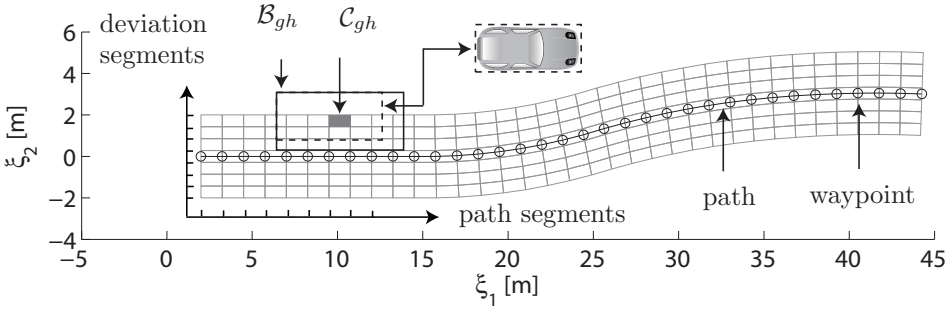


Fig. 5.28.: Exemplary path with occupancy regions.

In order to compute the crash probability, the probability p_{gh}^{pos} that the center c of a traffic participant is located in $\mathcal{C}_{gh}$ has to be computed first. Introducing the probability for a deviation segment $p_h^{\text{dev}} = P(\delta \in \mathit{d}_h)$ and the one for a path segment $p_g^{\text{path}} = P(s \in \mathit{s}_g)$, the probability that $(s \in \mathit{s}_e \wedge \delta \in \mathit{d}_f) \leftrightarrow c \in \mathcal{C}_{gh}$ is $p_{gh}^{\text{pos}} = p_g^{\text{path}} \cdot p_h^{\text{dev}}$ due to the independency assumption in Sec. 5.3. The deviation probabilities are fixed over time or time varying when considering a lane change maneuver.

In contrast to the deviation probabilities, the path segment probabilities have to be extracted from the joint probabilities of state and input p_i^{α} originating from the Markov chain computations. This is done by first summing over all inputs $\hat{p}_i = \sum_{\alpha} p_i^{\alpha}$. Each state space cell i represents a position and velocity interval s_e and v_m, such that $\mathcal{X}_i = \mathit{s}_e \times \mathit{v}_m$, where only the probability of the path segments s_e is of interest: $p_e^{\text{path}} = \sum_m P(s \in \mathit{s}_e, v \in \mathit{v}_m)$ and $P(s \in \mathit{s}_e, v \in \mathit{v}_m) \leftrightarrow P(x \in \mathcal{X}_i) = \hat{p}_i$.

Besides the probability that the center of a vehicle is in a certain trapezoid $\mathcal{C}_{gh}$, it is necessary to know the probability that two vehicle bodies intersect when the center of one of the bodies is in $\mathcal{C}_{gh}$ and that of the other one is in $\hat{\mathcal{C}}_{ef}$. This probability is denoted by p_{ghef}^{int} and is computed by uniformly gridding the uncertain sets $\mathcal{C}_{gh}$ and $\hat{\mathcal{C}}_{ef}$ as shown in Fig. 5.29. The grid points are possible centers of vehicle bodies and by counting the relative number of cases for which the bodies intersect, the probability p_{ghef}^{int} is obtained. Instead of gridding the set of vehicle centers, one can also conservatively assume that $p_{ghef}^{\text{int}} = 1$ if the sets of possible vehicle bodies $\mathcal{B}_{gh}$ and $\hat{\mathcal{B}}_{ef}$ intersect, and $p_{ghef}^{\text{int}} = 0$ otherwise. This computation has been used in [188] and generates overestimated crash probabilities. From now on, the computation of the crash probability according to the overestimated value of p_{ghef}^{int} is referred to as the *conservative* computation and the one according to the estimated value of p_{ghef}^{int} is referred to as the *relaxed* computation.

The probabilities p_{gh}^{pos} that centers are in certain regions $\mathcal{C}_{gh}$ and the probability p_{ghef}^{int} that two vehicle bodies intersect when their centers lie in $\mathcal{C}_{gh}$ and $\hat{\mathcal{C}}_{ef}$, allow the conditional crash probability according to Def. 5.1 to be computed by

$$p^{\text{crash}} = \sum_{g,h,e,f} p_{ghef}^{\text{int}} \cdot \hat{p}_{gh}^{\text{pos}} \cdot p_{ef}^{\text{pos}}. \tag{5.6}$$

The sum is taken over all possible combinations of g, h, e, f. This results in a huge number of possible combinations so that the following computational techniques are used to accelerate the computation.

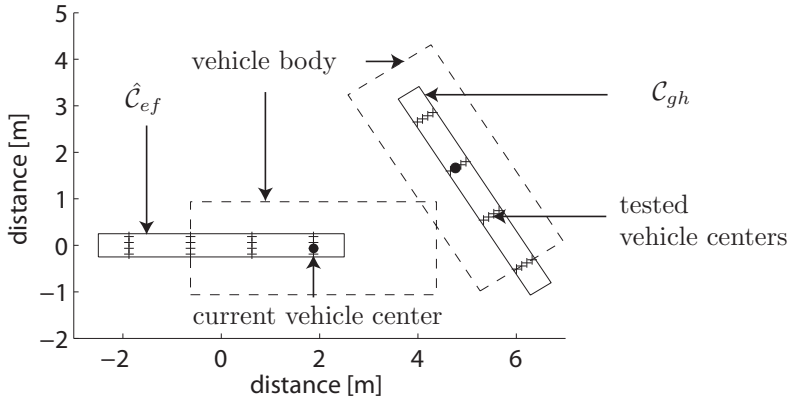


Fig. 5.29.: Estimating the intersection probability of vehicle bodies given the sets of vehicle centers.

5.6.2. Efficient Computation

In order to accelerate the computation of (5.6), pairs of vehicle body regions $(\mathcal{B}_{gh}, \hat{\mathcal{B}}_{ef})$ have to be found that intersect. Then, only a subset of combinations of indices g, h, e, f has to be considered in (5.6). In order to efficiently find intersecting pairs $(\mathcal{B}_{gh}, \hat{\mathcal{B}}_{ef})$, two steps are suggested:

1. First, it is checked if the vehicle bodies $\bigcup_f \mathcal{B}_{ef}$ of a certain path segment e can possibly intersect the vehicle bodies belonging to a path segment g of the ego car. For this, it is checked if the circles enclosing the set of vehicle bodies intersect, where circles are chosen since checking for their intersection is computationally cheap; see Fig. 5.30.
2. Next, the set of vehicle bodies $\mathcal{B}_{gh}$, $\hat{\mathcal{B}}_{ef}$ belonging to pairs of path segments passing the first test are again checked for intersection by the same procedure using enclosing circles.

Besides reducing the number of index combinations, the crash probability computation in (5.6) can be further accelerated by precomputing the probabilities p^{int}_{ghef}. Internally, the precomputed intersection probabilities p^{int} are stored by relative orientation and translation of uncertain centers $\mathcal{C}$. There are different look-up tables for p^{int}_{ghef} depending on checking intersection of the autonomous vehicle with a bicycle/bike, a car, or a truck.

5.6.3. Discussion

The computation of the crash probability based on the probability distribution of traffic participants within consecutive time intervals has the advantage, that no point of time is missed. For example, it prevents the tunneling effect which occurs when two vehicles cross each other at high velocity. In this case, the stochastic reachable sets of points in time might not intersect, while the stochastic reachable sets of time intervals intersect. The disadvantage of the computation with time intervals is that the uncertainty is greater than for points in time. This can lead to wrong crash probabilities as the following example shows.

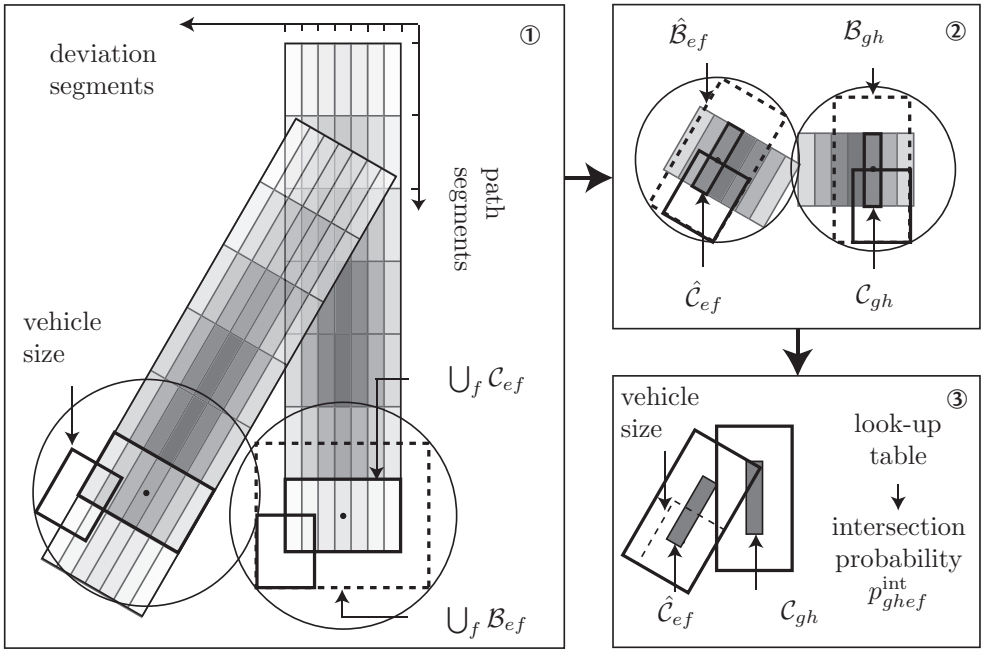


Fig. 5.30.: Crash probability obtained from stochastic reachable sets.

Example 5.5 (Crash Probability Evaluated for a Time Interval): Given are two vehicles driving one after the other with equal and constant velocity. The distance between the vehicles (bumper to bumper) is chosen as 10 m and the velocity is chosen as 30 m/s. For a time interval of $t \in [0, 0.5]$ s, the front bumper of the following vehicle and the rear bumper of the leading vehicle travel the distances as depicted in Fig. 5.31. Clearly, the crash probability is zero since both vehicles travel with equal velocity. However, since the corresponding bumpers of the following and leading vehicle are in the region between 10 m and 15 m with a probability of 1/3 in the time interval $t \in [0, 0.5]$ s, the crash probability is computed as 1/9 when using the independency assumption.

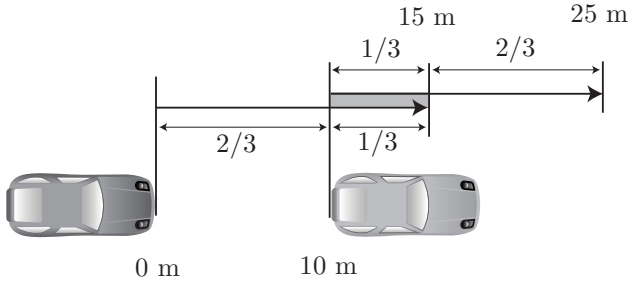


Fig. 5.31.: Crash probability example.

This error can be tackled by an extension, where in contrast to the previous approach, not only the position and orientation of the vehicles is considered, but the velocity is included, too. This is possible when using an enhanced look-up table in which the probabilities for a crash within the time interval $t \in [0, \tau]$ are stored based on the relative position, orientation, and the velocity intervals of both vehicles at $t = 0$. The disadvantage of the required look-up table is that it is very large and that reduction techniques such as

the exclusion of pairs of occupancy regions cannot be applied anymore. These two aspects easily make the extended approach about 100 times slower. However, this extension results in more accurate crash probabilities (see [194]). But after comparing the additional effort with the achieved improvement, this extension in not used here. The used approach is demonstrated for an overtaking scenario in the next example.

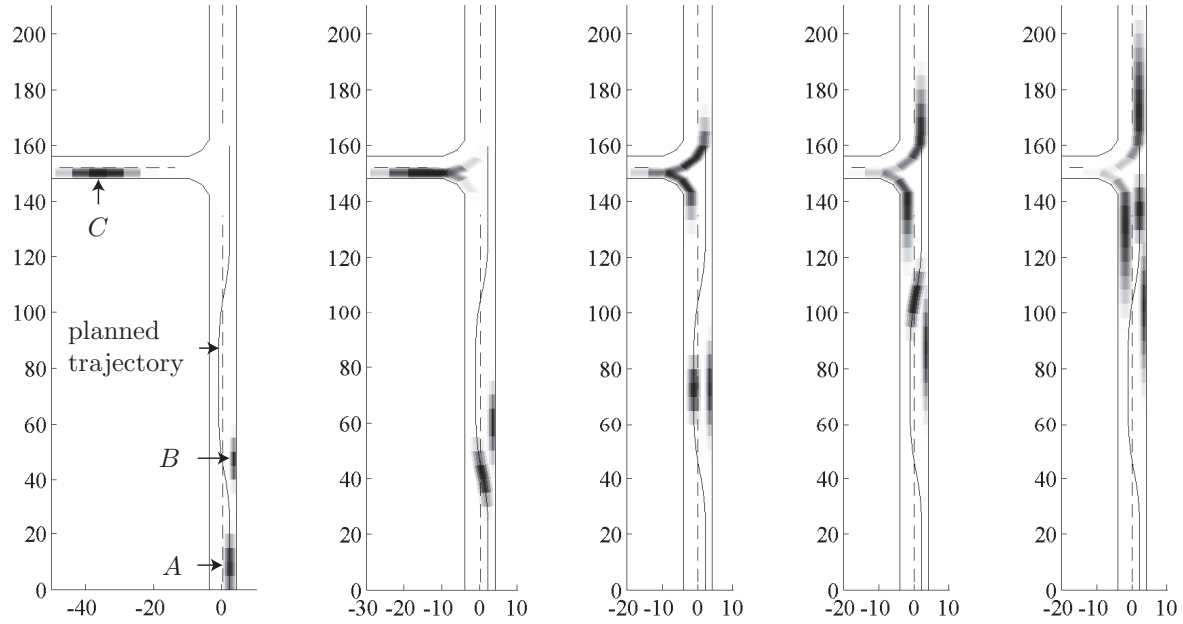
Example 5.6 (Crash Probability for an Overtaking Scenario): The crash probability of different time intervals according to the crash probability computation in (5.6) is computed for an overtaking scenario. In this scenario, the autonomous car A plans to overtake a bicycle B while another car C is approaching. The other car C enters the considered lane from a T-intersection and thus has the option of turning left or right. Since there is no probabilistic information on the turning behavior available, both options are considered with probability 1 such that the crash probabilities with the car are not computed too small. The parameters and settings are identical to the ones found in Tab. 5.2 and 5.3.

The probability distributions on the road are displayed in Fig. 5.32 for 5 selected time intervals. Dark regions indicate high probability, while bright regions represent areas of low probability. In order to improve the visualization, the colors are separately normalized for each vehicle. Since bicycle riders tend to ride close to the side of a lane, the lateral probability distribution is chosen differently to the one of cars. The crash probabilities of the autonomous car at different time intervals with different traffic participants is shown in Fig. 5.33. This plot shows the independent crash probabilities computed according to (5.6). The computational time on an AMD Athlon64 3700+ processor (single core) in Matlab was 0.26 s for the probability distributions and 0.49 s for the computation of the crash probabilities, resulting in 0.75 s for a prediction horizon of $t_f = 9$ s. A probability reduction with $p^{\max} = 3/(d \cdot c) = 1.25e-3$ was used.

More investigations on the accuracy of the crash probability computations are discussed in the next section on Monte Carlo simulation.

5.7. Comparison to Monte Carlo Simulation

In the previous sections of this chapter, a framework for the safety assessment of traffic situations using Markov chains was presented. In order to check these results and get an idea of the efficiency of the Markov chain approach, it is compared to Monte Carlo simulations in this section. The term Monte Carlo simulation or Monte Carlo method refers to methods that are based on random sampling and numerical simulation. Monte Carlo methods are especially popular for complex and highly coupled problems where other probabilistic approaches fail. One of the biggest applications of Monte Carlo simulation is risk analysis, such as the risk analysis of road traffic considered in this thesis. As already mentioned in the introduction of this chapter, Monte Carlo simulation has been applied to the risk analysis of road traffic [15, 31, 32, 52, 59, 60] and air traffic [25, 26, 168]. The results presented below have mainly been obtained from the supervised bachelor thesis of Alexander Mergel [194].



(a) $t \in [0, 0.5]$ s. (b) $t \in [2, 2.5]$ s. (c) $t \in [4, 4.5]$ s. (d) $t \in [6, 6.5]$ s. (e) $t \in [8, 8.5]$ s.

Fig. 5.32.: Position distribution for different time intervals in the overtaking scenario. The coordinate axes refer to positions in [m].

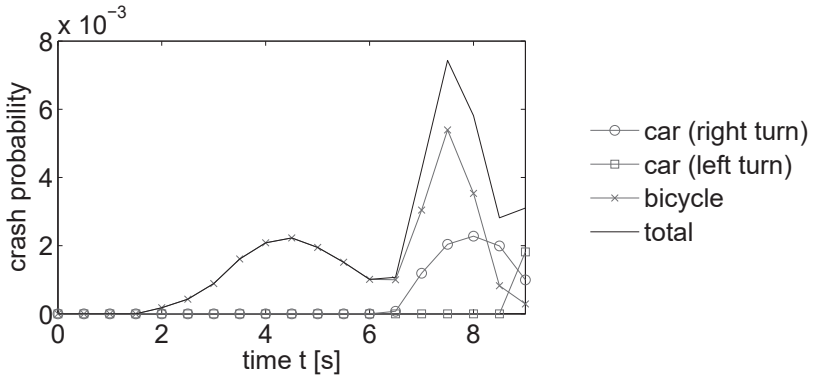


Fig. 5.33.: Crash probability of the overtaking scenario.

5.7.1. Basic Approach

There exists a huge variety of Monte Carlo methods and thus one cannot give a strict guidance on how to apply them in general. However, most methods exhibit the following scheme.

1. Define a domain of possible inputs and initial states.
2. Generate inputs and initial states randomly from the previously specified domains.
3. Perform a deterministic computation starting at the initial states subject to the randomly generated inputs.

4. Aggregate the results of the individual computations into the final result.

For the dynamic model of traffic participants (5.1), the domain of initial states is the set of initial positions and velocities, and the domain of inputs is the normalized range $[-1, 1]$ of acceleration commands. The initial states are randomly distributed according to the initial distribution. The input distribution is more complicated since it evolves over time so that the spectral density plays a major role in the outcome of the Monte Carlo simulation, which is discussed later in detail. The aggregation of the results differs from the two purposes pursued in this work.

One purpose is to compute the probability distribution of traffic participants for future points in time. When using importance sampling, i.e. the samples are created according to their probability distribution so that they have equal weight, the relative number of simulations ending up in certain state space cells determine the probability of reaching these state space regions. This procedure is exactly the same as for the determination of transition probabilities of Markov chains using Monte Carlo simulation; see Sec. 4.3.2. The other purpose is to compute the probability that the autonomous vehicle is crashing in a certain traffic scenario. The ratio of counted collisions N^{crash} to the number of simulations N_s determines the crash probability $p^{\text{crash}} = N^{\text{crash}}/N_s$ when using importance sampling. An alternative way of computing the crash probability is to determine the probability distributions of all traffic participants using Monte Carlo simulation first. In a second step, the crash probability is determined as for the Markov chain approach in Sec. 5.6. However, this procedure is not advantageous, as shown later.

An intrinsic property of Monte Carlo simulation is that the result of the computations is not deterministic, i.e. the result differs from execution to execution. Obviously, this is because the samples for the deterministic simulation are randomly generated. Thus, the probability distributions are possibly far from the exact solution. The good news, however, is that the mean error scales with $\frac{1}{\sqrt{N_s}}$ where N_s is the number of simulations. This result can be derived from Monte Carlo integration, which is briefly introduced in Appendix B. This is completely different when computing with Markov chains, where equal results are obtained for identical initial conditions since no random sampling is applied.

5.7.2. Random Number Generation

One of the most important tasks in Monte Carlo simulation is the generation of random numbers. This is a vast topic itself, dealing e.g. with the problem of how to generate random numbers from computers which are deterministic machines. For this reason, these random numbers are also called pseudo-random numbers and their degree of randomness is checked via certain tests; see e.g. [171]. Here, it is assumed that random numbers can be produced with sufficient quality from a uniform distribution in the interval $[0, 1]$. It remains to map these random numbers such that they are distributed according to a specified distribution.

In the Markov chain approach, the probability distributions are discrete. Each discrete probability represents the probability that a state or an input belongs to a certain cell of the discretized state or input space. In order to obtain a probability distribution in the continuous space, it is assumed that the probability distribution is uniform within a

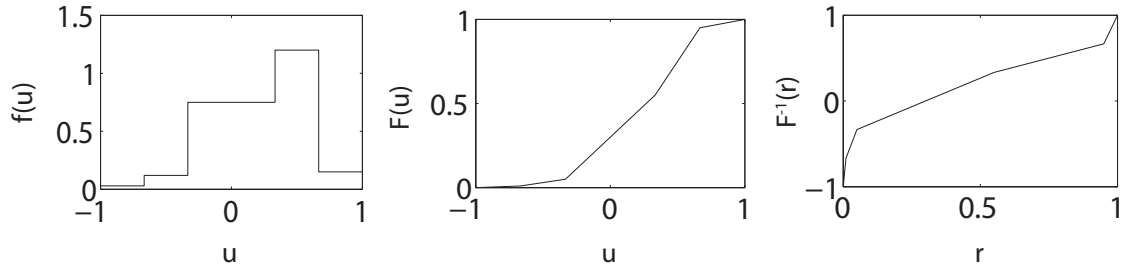
cell. This assumption has also been made for the generation of transition probabilities of Markov chains; see Sec. 4.3. Thus, the initial and input distribution is piecewise constant; see e.g. Fig. 5.34(a).

In order to compare the results of the Markov chain approach with the ones of the Monte Carlo simulation, random numbers according to the piecewise constant distributions have to be generated. The simplest approach for this task is the inverse transform method. This method is based on the inverse of the cumulative distribution, which always exists since only piecewise constant distributions are considered.

The method is illustratively described in Fig. 5.34 for a probability density function of the acceleration command u. First, the cumulative distribution of the input $F(u)$ is computed by integration of the probability density function $f(u)$. Next, the inverse of the cumulative distribution $F^{-1}(r)$ is computed, where r is the new argument. When generating uniformly distributed values of $\mathbf{r}$ within $[0, 1]$, the random values of $F^{-1}(\mathbf{r})$ are distributed according to the probability density function $f(u)$. Note that r refers to possible values of the random variable $\mathbf{r}$. The proof is straightforward:

$$P(\mathbf{u} \leq u) = P(F^{-1}(\mathbf{r}) \leq u) = P(\mathbf{r} \leq F(u)) = F(u).$$

The inverse transform method is used when generating the initial probability distribution of the state as well as for the random inputs. For the random inputs, one has to additionally consider correlations between time steps, which is discussed next.



(a) Probability density function. (b) Cumulative density function. (c) Inverse cumulative density function.

Fig. 5.34.: Inverse transform method for the generation of random samples.

5.7.3. Input Generation

In the Markov chain approach, not only the states, but also the inputs are generated by a Markov chain. More precisely, the conditional input distribution $q_i^\alpha = P(\mathbf{y} = \alpha | \mathbf{z} = i)$ is updated according to $q_i^\alpha([t_k, t_{k+1}]) = \sum_\beta \Gamma_i^{\alpha\beta}(t_k) q_i^\beta([t_{k-1}, t_k])$, where $q_i^\beta([t_{k-1}, t_k]) \in \{0, 1\}$ when using Monte Carlo simulation; see (4.22). Input samples for the Monte Carlo simulation are generated from the discrete probabilities $q_i^\alpha([t_k, t_{k+1}])$ as previously described in Sec. 5.7.2.

Another option to create random inputs is proposed by Broadhurst and Eidehall in works on Monte Carlo simulation of road traffic scenes [32, 52, 59, 60]. In these approaches, a

random trajectory is created first, and afterwards its likeliness is evaluated. The values of the input trajectories are created by an IID[6] process, but are no longer IID after a weight is assigned by the likeliness function. Before presenting the likeliness function (or also called goal function), some notations are introduced and recapitulated. The deviation from the lane center is denoted by δ, the velocity by v, the normal acceleration by a_N, the steering wheel angle by ϕ, the maximum velocity by $v^{\max}$, and the number of input values by $N_u = t_f/\tau$ (t_f is the time horizon and τ the time increment). The goal function is chosen in [32] to

$$g(u([0,t_f])) = -\sum_{k=1}^{N_u} \left(\lambda_1 \delta(t_k)^2 + \lambda_2 (v(t_k) - v^{\max})^2 + \lambda_3 a_N(t_k)^2 + \lambda_4 \phi(t_k)^2\right), \tag{5.7}$$

where λ_1–λ_4 are tuning parameters which punish the following: Large deviations from the lane center, velocity deviations from the allowed velocity, large accelerations, and steering wheel angles. The punishment of accelerations and steering angles ensures a comfortable ride for the passengers. The probability distribution of the input trajectories $f(u([0,t_f]))$ is assumed to be

$$f(u([0,t_f])) = c_n \exp(k_g \cdot g(u([0,t_f]))) \tag{5.8}$$

according to [32], where the previously introduced goal function is in the exponent. The value k_g is another tuning parameter and c_n is the normalization constant. The related publications [52, 59, 60] use almost the same model, but with different parameter values and the tangential acceleration a_T instead of the steering angle ϕ. It is remarked that the tuning parameters are set according to simulations and not learned from driving experiments in real traffic.

Since the input values are generated independently of previous values, consecutive input values are not strongly correlated despite the weighting. In contrast to this, the input values generated by a Markov chain are correlated by the input transition values in Γ. This is checked by the autocorrelation, i.e. the correlation of the signal against a time-shifted version of itself:

$$S(t,\tau) := E[\mathbf{u}(t)\mathbf{u}(\tau)] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u_t u_\tau f(u_t u_\tau)\, du_t\, du_\tau, \tag{5.9}$$

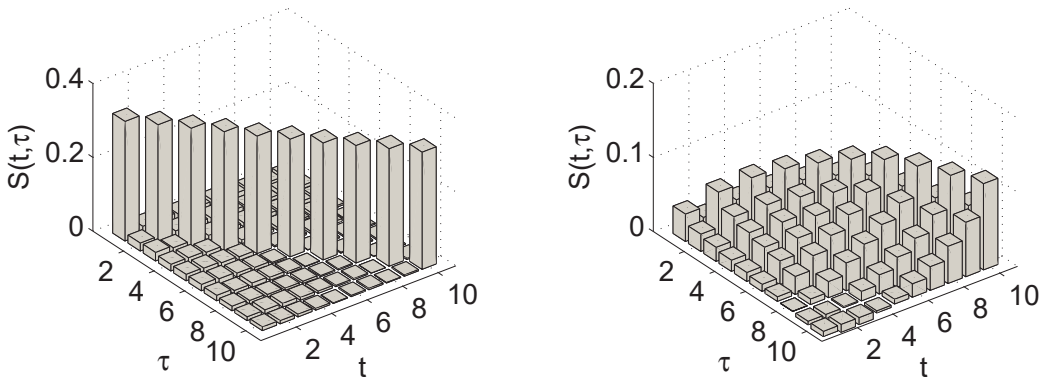
where $E[\,]$ is the expectation, $f(\,)$ is the probability density function, and u_t is a realization of the random variable $\mathbf{u}(t)$. The computation of the above integrals is approximated using Monte Carlo integration as shown in (B.1).

The autocorrelation values for the input trajectories generated from the Markov chain approach and the approach proposed by Broadhurst and Eidehall are plotted in Fig. 5.35. The input trajectories for the autocorrelation computation are uniformly sampled and generated with parameters listed in Tab. 5.8. Note that the parameters λ_1 and λ_4 are set to 0 since it is assumed that the vehicles drive perfectly on their path. There is almost no correlation between the inputs of different time steps in the approach proposed by Broadhurst and Eidehall; see Fig. 5.35(a). This is in contrast to the Markov chain approach, where the input signals are much more correlated, as shown in Fig. 5.35(b).

[6] IID: independent and identically distributed.

Tab. 5.8.: Parameters for input trajectory generation.

General		Broadhurst/Eidehall		Markov chain	
N_s	$1e5$	k_g	100	μ	$[0.01, 0.04, 0.25, 0.25, 0.4, 0.05]$
N_u	10	λ_1	0	$q^\alpha(0)$	[0, 0, 0.5, 0.5, 0, 0]
$v(0)$	15 ± 1 m/s	λ_2	$0.05/N_u/(1+v(0)^2)$	γ	0.2
$v^{\max}$	100/3.6 m/s	λ_3	$0.05/N_u/a^{\max 2}$		
τ	0.5 s	λ_4	0		



(a) Approach by Broadhurst, Eidehall. (b) Markov chain approach.

Fig. 5.35.: Autocorrelation of input trajectories.

Besides the autocorrelation, the spectral density of input signals generated by the Markov chain approach and the approach used in [32, 52, 59, 60] are compared for a deeper analysis. The spectral density describes how the energy of a signal is distributed over its frequency, where the energy of a signal $x(t)$ is defined as $\int_{-\infty}^{\infty} |x(t)|^2\, dt$ in signal processing. The spectral density is defined as $|X(f)|^2$, where $X(f)$ is the Fourier transform of $x(t)$. For the analysis of several instances of a stochastic signal, one is interested in the expectation of the random spectral density $E[|\mathbf{X}(f)|^2]$.

Proposition 5.4 (Average Spectral Density): The average of the spectral density $E[|\mathbf{U}(f)|^2]$ for input signals $u(t)$, which are piecewise constant for time τ, is computed as

$$\Phi(f) = E[|\mathbf{U}(f)|^2] = \tau^2 \mathtt{si}^2(\pi f \tau) \sum_{k=1}^{N_u} \sum_{l=1}^{N_u} E[\mathbf{u}_{t_k} \mathbf{u}_{t_l}] e^{-j2\pi f \tau (k-l)}. \qquad \square$$

The expectation $E[\mathbf{u}_{t_k} \mathbf{u}_{t_l}]$ is obtained from the autocorrelation (5.9) and $\mathtt{si}()$ is the sinc function which is defined as $\mathtt{si}(x) = \sin(x)/x$. A proof of the proposition can be found in Appendix A.2. The expectations of spectral densities $\Phi(f)$ are visualized in Fig. 5.36 and were computed based on the parameters in Tab. 5.8. It can be seen that the lower frequencies are more dominant in the Markov chain approach, while the frequency distribution of the approach by Broadhurst and Eidehall is close to an IID process with uniform distribution. Thus, the Markov chain approach better imitates the behavior of traffic participants

who are generally changing their acceleration with low frequency.

The two approaches described for modeling the acceleration command of a vehicle have not been verified with measured data from real traffic. However, the possibility to correlate input data and the stronger dominance of input signals with low frequency are arguments in favor of input trajectories generated by Markov chains. For this reason, only input trajectories generated from Markov chains are used here. The comparison of probability distributions obtained from the Markov chain and the Monte Carlo simulation approach are presented next.

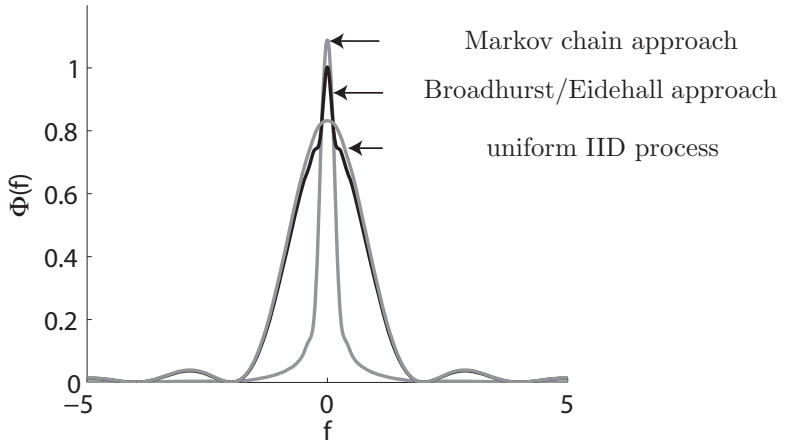


Fig. 5.36.: Average spectral density of input trajectories generated from different approaches.

5.7.4. Comparison of Probability Distributions

In this subsection, the input trajectories generated from Markov chains are used to compare the performance of the Markov chain approach with the Monte Carlo approach. The computed probability distributions are compared by the error or distance

$$d = \sum_i |p_i - p_i^{\mathrm{e}}| \cdot \mathtt{V}(\mathcal{X}_i),$$

where p^{e} is the exact probability distribution. The multiplication with the volume of the cells is required in order to compare results of different discretization. The probability distributions are compared for a braking maneuver and a maneuver where the acceleration is uncertain. In these two scenarios, three different Markov chain discretizations are used as listed in Tab. 5.9. The transition probabilities of the Markov chains are generated using Monte Carlo simulation[7]. Further, the non-zero probability reduction as suggested in Sec. 5.4.3 has been applied to the Markov chain approach with $p^{\max} = 3/(d \cdot c)$, where d and c is the number of discrete states and inputs, respectively. The Monte Carlo approach used in both scenarios is performed with 10^4 simulations.

The braking maneuver is chosen because the probability distribution of $\dot{v} = a^{\max}\,\mathbf{u}$ ($\mathbf{u} \in [\underline{u}, \overline{u}]$) (see (5.1)) has an exact solution, which is described in Example 4.1. In the braking scenario, only the velocities of the Markov chain and Monte Carlo approach

[7] 10^4 samples are generated for each cell.

are compared. The time horizon is chosen as $t_f = 5$ s and the time increment is $\tau = 0.5$ s. The acceleration command is uniform in $[-1/3, 0]$ and the initial speed is uniformly distributed within $v(0) = [17, 19]$ m/s. The exact solution, the Monte Carlo solution, and the solution of the Markov chains are compared: Markov chain A (coarse discretization) is compared in Fig. 5.37 and Markov chain B (fine discretization) in Fig. 5.38. In order to compare different discretizations, the bins of the Monte Carlo approach counting the number of samples are adjusted to the cells of the corresponding Markov chains. The computational times are shown in Tab. 5.10 and were obtained from an AMD Athlon64 3700+ processor (single core) using a Matlab implementation. The Monte Carlo simulation has been obtained using a Runge-Kutta solver for ordinary differential equations and the analytic solution presented in Prop. 5.2. One can see that the analytical solution is about 15 times faster than the Runge-Kutta solver.

Tab. 5.9.: State space discretizations for a position interval of $[0, 400]$ m and a velocity interval of $[0, 60]$ m/s.

discretization	position segments	position resolution	velocity segments	velocity resolution
A	80	5 m	30	2 m/s
B	80	5 m	120	0.5 m/s
C	320	1.25 m	120	0.5 m/s

Tab. 5.10.: Computational times of the braking scenario.

Monte Carlo (simulated)	Monte Carlo (analytical)	Markov chain A	Markov chain B
3.67 s	0.252 s	0.016 s	0.042 s

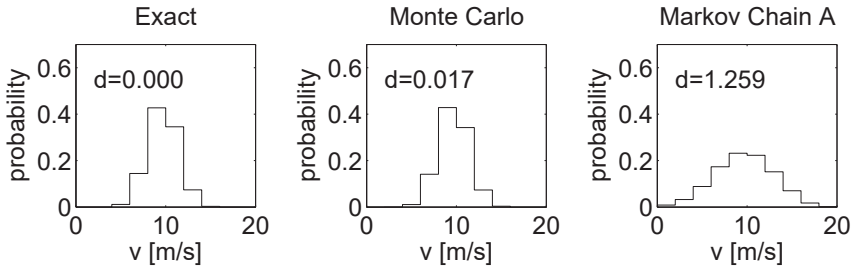


Fig. 5.37.: Braking scenario: Velocity distributions for a coarse discretization ($t = 5$ s).

In the second example, the input is generated from the input transition matrix Γ as described in Sec. 5.5.1 when following a straight road with speed limit. The parameters required to determine Γ and further parameters are listed in Tab. 5.11. The uniform initial position interval is $[2, 8]$ m and the initial velocity interval is $[15, 17]$ m/s. The resulting position and velocity distribution for a coarse discretization can be found in Fig. 5.39 and

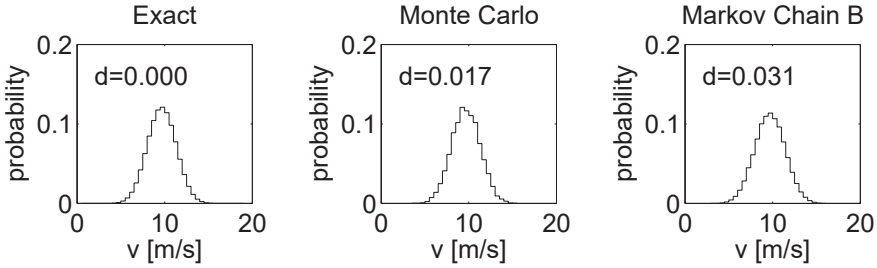


Fig. 5.38.: Braking scenario: Velocity distributions for a fine discretization ($t = 5$ s).

for a fine discretization in Fig. 5.40. Note that the Markov chain model B has a coarse discretization of the position and a fine one of the velocity so that the position result is shown in Fig. 5.39 and the velocity result in Fig. 5.40. Since there is no exact solution for this scenario, an almost exact solution was computed with Monte Carlo simulation using $5 \cdot 10^5$ samples. The computational times can be found in Tab. 5.12, which are again obtained from an AMD Athlon64 3700+ processor (single core) using a Matlab implementation. The Monte Carlo simulation has been obtained using the Runge-Kutta solver and the analytic solution as presented in Prop. 5.2. It can be observed that the Markov chain solution is faster than the analytically obtained Monte Carlo solution and that the discretization of the Markov chain C is fine enough to produce results that are more accurate than the Monte Carlo approach with 10^4 simulations.

Tab. 5.11.: Behavior parameters.

γ	0.2
μ	$\begin{bmatrix}0.01 & 0.04 & 0.25 & 0.25 & 0.4 & 0.05\end{bmatrix}$
$q_i(t=0)$	$\begin{bmatrix}0 & 0 & 0.5 & 0.5 & 0 & 0\end{bmatrix}$ $(\forall i)$
$v^{\max}$	100/3.6 m/s

Tab. 5.12.: Computational times of the road following scenario.

Monte Carlo (simulated)	Monte Carlo (analytical)	Markov chain A	Markov chain B	Markov chain C
3.44 s	0.578 s	0.030 s	0.110 s	0.417 s

Finally, it was analyzed if the quality of the probability distributions depends on the initial condition. As the vehicle model (5.1) is invariant under translations in position, it is only necessary to vary the initial velocity. The influence on the initial velocity on the distances d^{pos}, d^{vel} of the position and velocity is shown in Fig. 5.41. The Monte Carlo simulations are performed with 10^4 samples and the Markov chain approach was computed with the C model. In contrast to the previous computations, the speed limit of 100/3.6 m/s has been removed so that initial velocities above this speed can be investigated. It can be seen that the dependence on the initial velocity and thus on the initial state can be

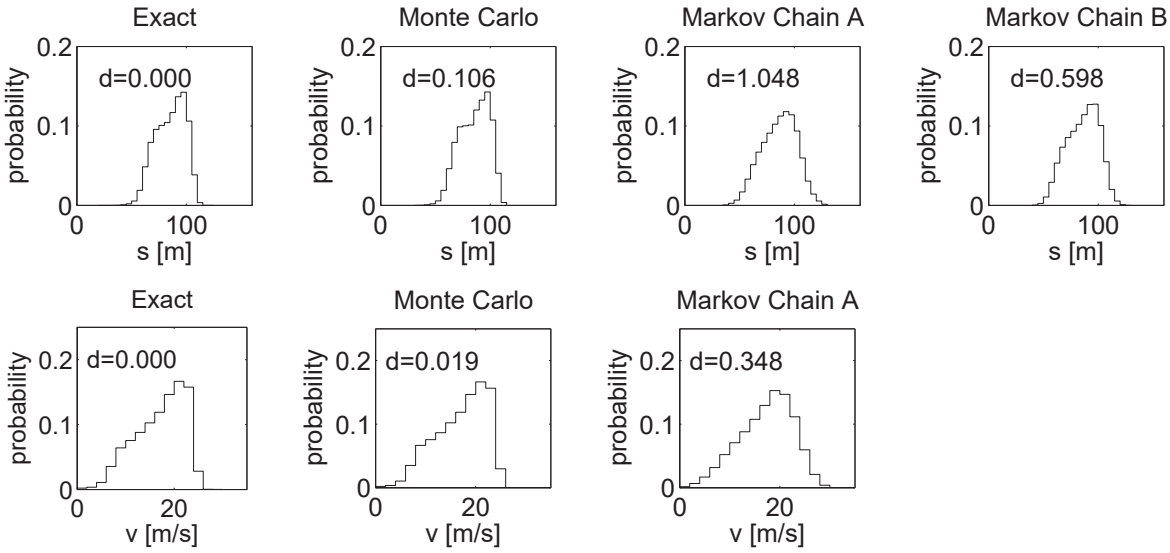


Fig. 5.39.: Road following scenario: Position and velocity distribution for a coarse discretization ($t = 5$ s).

neglected, meaning that the results in Fig. 5.39 and Fig. 5.40 are representative. In the next subsection, the crash probabilities obtained from the Monte Carlo simulation are compared to the ones obtained from the Markov chain approach.

5.7.5. Comparison of Crash Probabilities

One big advantage of Monte Carlo simulation is that the crash probability can be easily computed. When using importance sampling, it is obtained from the number of trajectories leading to a crash N^{crash} divided by the overall number of simulated trajectories: $p^{\text{crash}} = N^{\text{crash}}/N_s$. Note that trajectories causing a crash are not removed from the computation in order to obtain crash probabilities in compliance with Def. 5.1. The traffic participants are modeled by rectangles with a certain length and width. In order to efficiently determine if a crash occurs, the separating axis theorem is applied which allows the detection of intersections of rectangles with low computational costs [76]. The description of the implementation can be found in [194] and an extension considering the velocity of objects is presented in [58].

In the Markov chain approach, one has to compute the probability distribution of the traffic participants as an intermediate step and then compute the probability distribution as described in Sec. 5.6.

The crash probabilities are investigated for a scenario where the autonomous car drives behind another car. The autonomous car starts from the position 0 m with constant velocity 20 m/s and has a uniform position uncertainty of ±3 m. The vehicle driving in front has an uniform position in the interval of $[20, 25]$ m and the initial velocity is within $[15, 17]$ m/s. The other parameters are listed in Tab. 5.11. The crash probability of Markov chains is compared to the exact solution with a coarse and a fine discretization using model A and C (see Tab. 5.9). The (almost) exact solution is obtained from a Monte Carlo simulation with 10^5 samples. Besides two different Markov chain models, the two computational tech-

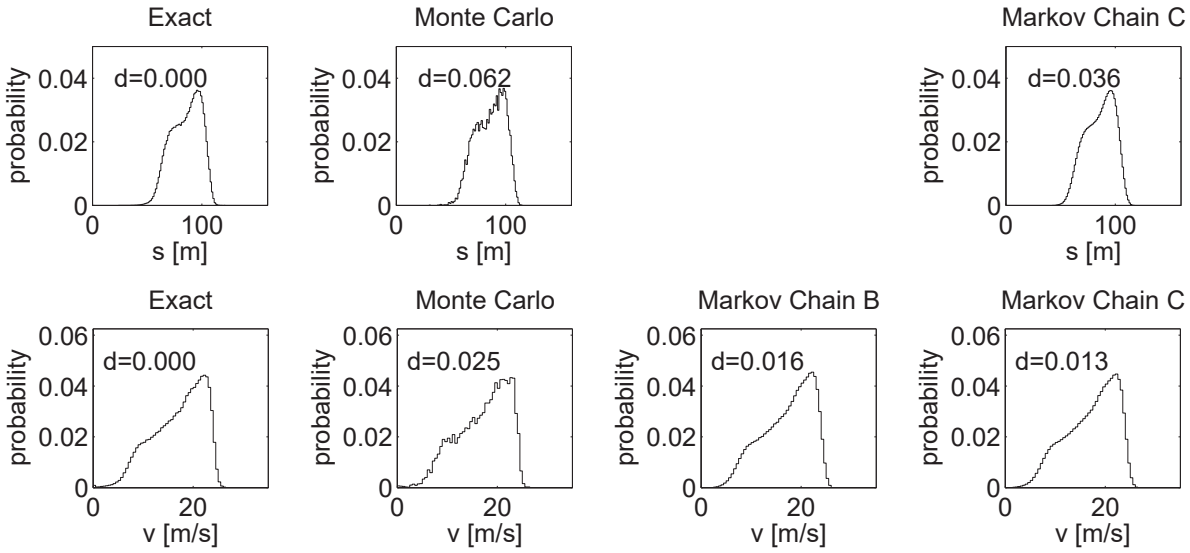
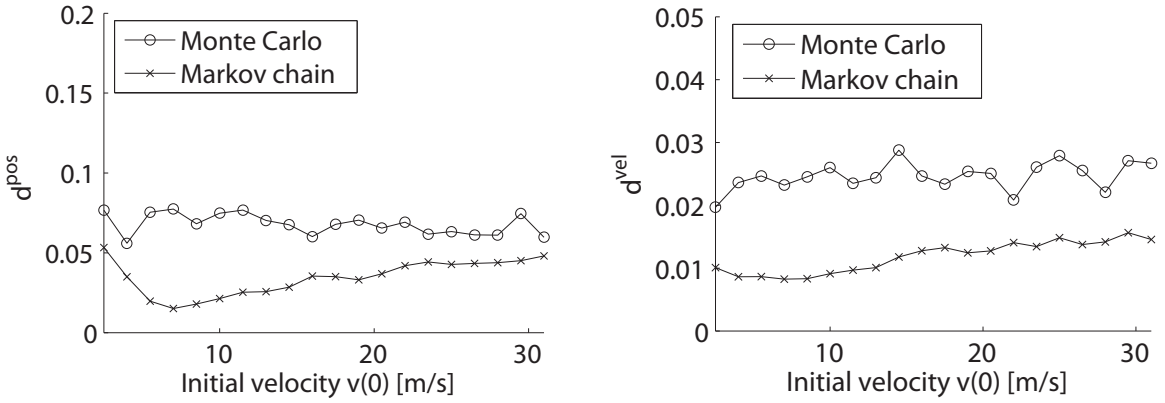


Fig. 5.40.: Road following scenario: Position and velocity distribution for a fine discretization ($t = 5$ s).



(a) Distance d of the position distribution.

(b) Distance d of the velocity distribution.

Fig. 5.41.: Distance d to the exact solution for different initial velocities.

niques for calculating the crash probability are compared: The conservative computation (con) with the relaxed computation (rel); see Sec. 5.6.1. In addition, the crash probability is computed based on the probability distribution at points in time (TP) and of time intervals (TI). This results in 4 different variations: TPcon, TIcon, TPrel, and TIrel. The crash probabilities p^{crash} for different time steps using the 4 different computing methods are shown in Fig. 5.42. It can be observed that the relaxed computation produces much better results than the conservative computation for the coarse model A. In the case of the fine model C, the conservative and relaxed computation produce similar results; however, in terms of the time interval solution the relaxed computation is slightly better. It is again remarked that only the time interval solution ensures that no dangerous situation is missed.

Besides different Markov chain models, Monte Carlo solutions were tested, too. Fig. 5.42(c) shows that the result is very accurate, even when only 10^3 or 10^2 samples are used. For

this reason, it can be clearly stated that the Monte Carlo simulation performs better than the Markov chain approach when the crash probability has to be computed. This is reconfirmed by the computational times in Tab. 5.13, where the Monte Carlo approach is more efficient. The computational times for the Markov chain approach are separated into the part for computing the probability distribution and the part that intersects the probability distributions to obtain the crash probability. The computations were performed on an AMD Athlon64 3700+ processor (single core) using a Matlab implementation.

Tab. 5.13.: Computational times of the crash scenario.

Markov chain				
	A (TP)	A (TI)	C (TP)	C (TI)
Prob. dist.	0.175 s	0.175 s	0.525 s	0.525 s
Intersection	0.042 s	0.107 s	0.169 s	0.394 s
Total	0.217 s	0.282 s	0.694 s	0.919 s
Monte Carlo simulation				
	1e2 (sim.)	1e3 (sim.)	1e2 (analy.)	1e3 (analy.)
Total	0.190 s	0.549 s	0.069 s	0.321 s

5.7.6. Examination of Interacting Vehicles

The interaction of two vehicles driving in the same lane has been addressed for the Markov chain approach in Sec. 5.5.3. The formalism to compute the constraint vector η of the following vehicle is described in Prop. 5.3, which is presented again for better readability:

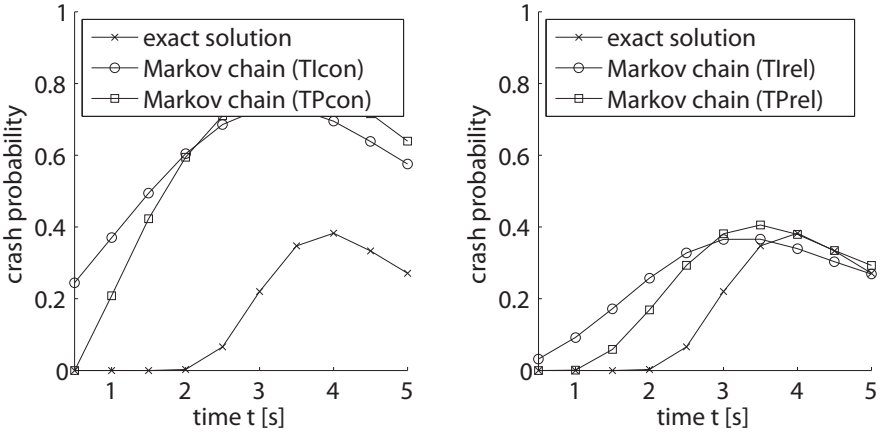
$$\eta_i^\alpha = \sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_j^{L\beta}. \tag{5.10}$$

In this formula, the constraint value η_i^α for a single state i and input α of the following vehicle is based on all states j and inputs β of the leading vehicle with probability distribution $p_j^{L\beta}$, causing an averaging effect. Another problem is that the above formula only holds if the joint probability $P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ of the following vehicle is independent of the one of the leading vehicle $P(\mathbf{z}^L = j, \mathbf{y}^L = \beta)$; see Prop. 5.3.

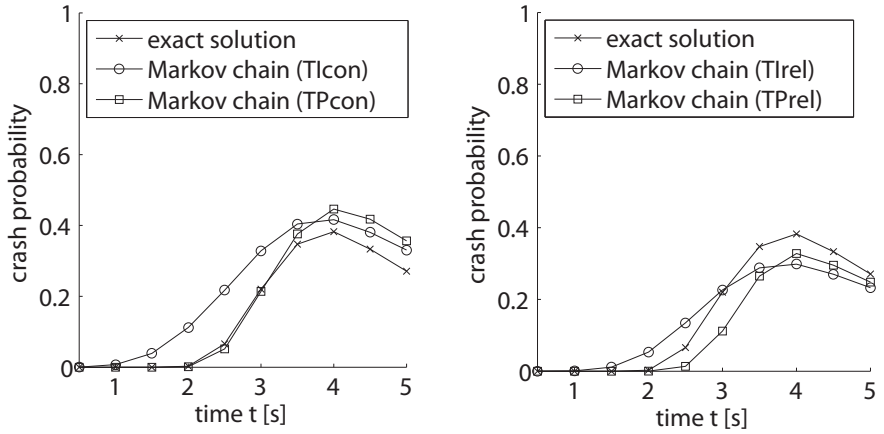
In the Monte Carlo approach, the independence assumption is not required. After introducing the events $A = (\mathbf{z}^F = i, \mathbf{y}^F = \alpha)$ and $B_j^\beta = (\mathbf{z}^L = j, \mathbf{y}^L = \beta)$ from Prop. 5.3, the modified computation of the constraint vector is

$$\eta_i^\alpha = P(C|A) = \frac{P(C,A)}{P(A)} = \frac{\sum_{j,\beta} P(C|A,B_j^\beta) P(A,B_j^\beta)}{P(A)} = \frac{\sum_{j,\beta} \Theta_{ij}^{\alpha\beta} p_{ij}^{F,L^{\alpha\beta}}}{p_i^{F\alpha}}, \tag{5.11}$$

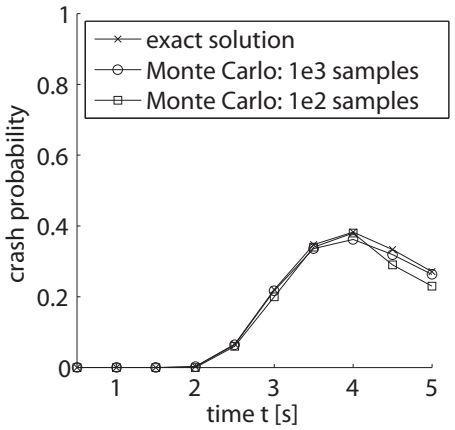
where $p_{ij}^{F,L^{\alpha\beta}} = P(\mathbf{z}^F = i, \mathbf{y}^F = \alpha, \mathbf{z}^L = j, \mathbf{y}^L = \beta)$. In the Monte Carlo approach, this probability is 1 for the cell indices the samples are located in, whereas the remaining values are 0. Thus, the Monte Carlo approach does not suffer from the independence assumption.



(a) Markov chain comparison (discretization A).



(b) Markov chain comparison (discretization C).



(c) Monte Carlo comparison.

Fig. 5.42.: Crash probabilities for different points in time.

It is also not suffering from the averaging effect since (5.11) is evaluated separately for each sample. The computation in (5.11) is not possible for the Markov chain approach since the probability $p_{ij}^{F,L^{\alpha\beta}}$ is not available. It would be available if the state space of the

Markov chain approach would be extended such that there are two dimensions (position and velocity) for one vehicle and the same two dimensions for the second vehicle. However, the number of dimensions would increase by 2, which results in an explosion in the number of discrete states due to the exponential growth of discrete states.

The consequences of the averaging effect and the independence assumption are investigated by emulating the behavior of the Markov chain approach by a Monte Carlo implementation. This is done by computing the constraint vector as described in (5.10), where the probability $p_j^{L\beta}$ is 1 for the cell indices the sample is located in and 0 otherwise. This result is compared to the exact result obtained from Monte Carlo simulation of (5.11), the result obtained from computing without interaction, and the result obtained when removing crashed samples in the Monte Carlo approach. The removal approach is performed by computing without any interaction and then removing pairs of crashing vehicles.

All results were obtained for a scenario with a leading and a following vehicle on a straight road. The parameters for the simulation are shown in Tab. 5.11, except that the speed limit is chosen to $v^{\max} = 16$ m/s. For the Markov chain computations, the discretization presented in Tab. 5.2 is used. The initial position and velocity of the following vehicle is uniformly distributed within $s^F(0) \in [2, 8]$ and $v^F(0) \in [10, 12]$ m/s. The initial intervals of the leading vehicle are $s^L(0) \in [12, 18]$ and $v^L(0) \in [10, 12]$ m/s. The samples of the different Monte Carlo simulations can be seen in Fig. 5.43, where the diagonal line[8] indicates when the following vehicle has crashed into the leading vehicle so that all samples above the diagonal line are collision-free. The figures show 10^3 samples at time $t = 5$ s. It can be observed that the emulated Markov chains contain a lot of crashed vehicles. This is mainly caused by the averaging effect and the independence assumption. A further emulation which investigates the error due to the independence assumption while fixing the averaging effect can be found in [194]. There are also some accidents in the exact solution because the stored values in $\Theta_{ij}^{\alpha\beta}$ are approximately computed so that not all trajectories starting in the corresponding cells are guaranteed to be collision-free; see Sec. 5.5.3. Clearly, in the sample removal approach, no accidents occur. The disadvantage of the sample removal method is that only accidents within the time horizon are canceled while a certain constellation of vehicles states might inevitably lead to a collision in the future.

The resulting marginal probability distributions of each vehicle are shown in Fig. 5.44 obtained with 10^4 samples after $t = 5$ s. Indeed, the Monte Carlo emulation resembles the probability distribution of the Markov chain approach. The plots also show that the intersection of probability distributions of the position does not necessarily imply that vehicles are crashing since e.g. the sample removal approach is collision-free. Thus, the approach for computing the collision probability out of the probability distribution of traffic participants as shown in Sec. 5.6 cannot be applied to two arbitrary traffic participants. However, it can be used to obtain the collision probability of the autonomous car with another vehicle, because their probability distributions are independent; see Def. 5.1.

The Monte Carlo simulations allowed a deeper insight into the advantages and disadvantages of the Markov chain approach which are discussed below.

[8] The diagonal line is shifted by the vehicle length of 5 m.

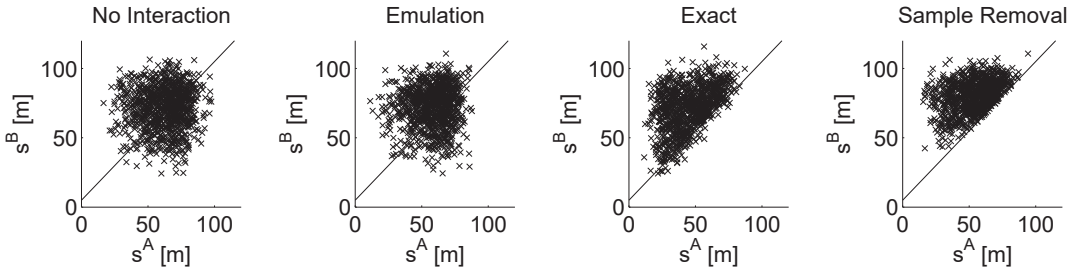


Fig. 5.43.: Joint position distribution of samples after $t = 5$ s.

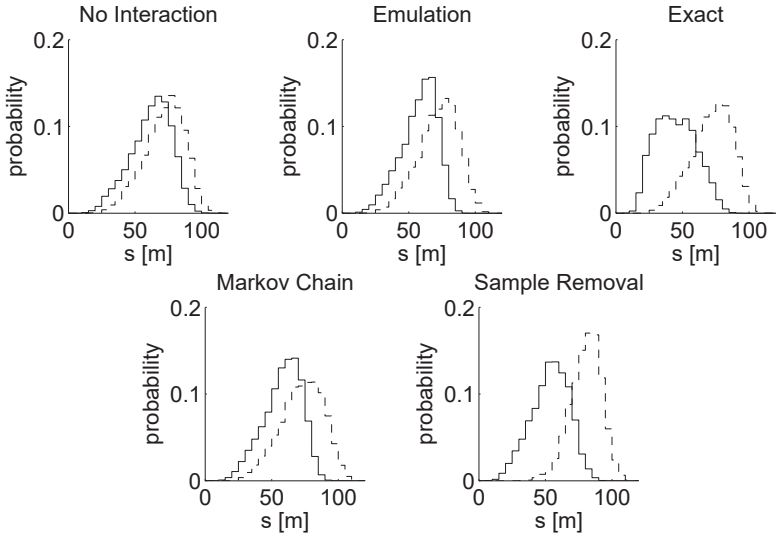


Fig. 5.44.: Marginal position distributions after $t = 5$ s.

5.7.7. Discussion

The Markov chain approach and the Monte Carlo approach have some inherent differences concerning their error sources. The main error in the Markov chain approach is introduced due to the discretization of the state and input space. The error in the transition probabilities can be made arbitrarily small since they are computed beforehand. Thus, the Markov chain approach has only systematic errors from the discretization but no stochastic errors since no random sampling is applied.

In the Monte Carlo approach, there are no systematic errors (no bias) because each simulation is correctly solved with the original dynamical system equations. However, the Monte Carlo simulation suffers under stochastic errors due to the sampling of the initial conditions and input sequences. Due to the stochastic errors, the resulting distributions and crash probabilities differ from execution to execution although the initial conditions and inputs are unchanged. This implies that the obtained results might be far off the exact solution – however, the likeliness of an extremely bad result is small and the mean error converges with $\frac{1}{\sqrt{N_s}}$, where N_s is the number of samples.

For a simple scenario, where a vehicle drives along a road, the resulting probability distributions of the Markov chain approach are slightly more accurate and faster than for the Monte Carlo simulation if an analytical solution exists. When no analytical solution exists, the Markov chain approach is at least about 10 times faster. Because there are many

matrix multiplications in the Markov chain approach, it can be significantly accelerated by using dedicated hardware such as DSPs (digital signal processors). However, when computing crash probabilities, the Monte Carlo approach clearly returns better results since it does not suffer from the discretization of the state space.

Another disadvantage of the Markov chain approach is that it is difficult to find an algorithm which accurately computes the probability distributions for interacting vehicles – a good solution is still to be found. For instance, one could investigate if it makes sense to remove probabilities in the Markov chain approach as it is done in the Monte Carlo approach when removing crashed samples. The errors in the current implementation could be explained by an emulation using Monte Carlo simulation, which shows that the Monte Carlo approach is more flexible. This might be of interest in unstructured environments or when there is no single plausible path for a traffic participant. In such cases, one could mix Monte Carlo related approaches with the Markov chain approach.

This is illustrated in Fig. 5.45, showing another vehicle which approaches a standing vehicle. This vehicle might wait behind the standing vehicle or pass by. There are three plausible paths in Fig. 5.45 and the partition of each path originating from the path and deviation segments is visualized. Path 1 represents the option that the vehicle waits and there are two alternatives paths 2 and 3 for passing the standing vehicle. When one is interested in the probability distribution of other vehicles to e.g. plan a safe trajectory, the more accurate and efficient Markov chain approach should be applied along these paths. If one is only interested in the probability of a crash, Monte Carlo simulation should be applied along the paths. However, Monte Carlo simulation should not be directly applied in 2-D scenarios since many samples end up at road borders or other obstacles, making this approach not very efficient. This is demonstrated for example in [194] and addressed in many publications, e.g. [15, 31, 32, 52, 59, 60].

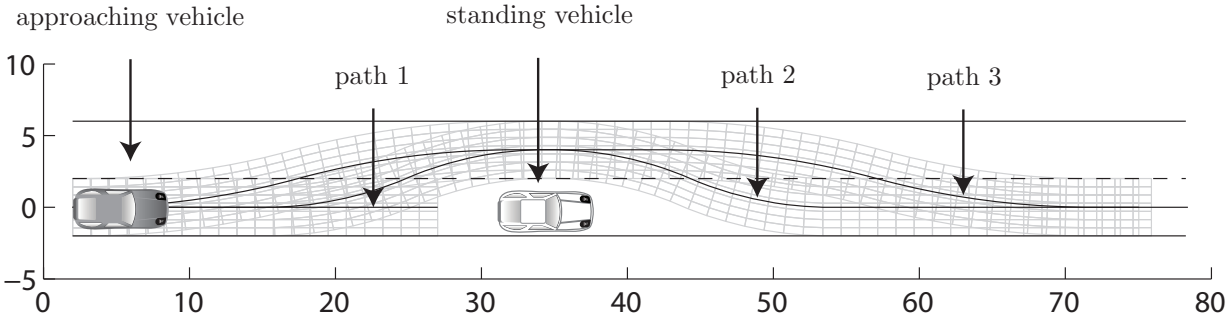


Fig. 5.45.: Evasion of a standing car with alternative paths.

5.8. Driving Experiment

In previous sections, the safety assessment framework for autonomous cars was demonstrated by numerical examples. In this section, the safety assessment is conducted based on real measurements from a driving experiment with the experimental vehicle *MUCCI* (Munich's Cognitive Car Innovation) [192, 198]. This vehicle has been converted from a standard Audi Q7 model to an autonomous vehicle within the research project *Cognitive*

Automobiles [154].

5.8.1. Experimental Vehicle MUCCI

The experimental vehicle *MUCCI* is equipped with several sensors in order to properly perceive the environment. The sensors used for the driving experiment described in this work are:

- Inertial measurement unit (IMU): This device measures the translational and rotatory acceleration of the vehicle. The rotatory accelerations are used to adjust the active camera platform such that it is aligned with the horizon, despite the roll and pitch of the vehicle.
- Active camera platform: The cameras are used to detect lane markings on the road which allows a representation of the road geometry to be built without any information from a navigation system with road network information. The shape of the lane markings is modeled by clothoids – a representation that has also been investigated for modeling possible paths of other traffic participants [197].
- Light detection and ranging (LIDAR): Other objects in the traffic scene are detected by this device, which measures the reflections of transmitted laser beams. After some preprocessing, the enclosing circle radius, the relative position, and the relative velocities of other traffic participants are obtained [161].

This equipment is also used in other vehicles of the *Cognitive Automobiles* project such as for *MuCAR-3*, developed by the University of the Bundeswehr München, and for a modified VW Passat developed by the Karlsruhe Institute of Technology. The VW Passat participated in the 2007 Darpa Urban Challenge [92]. Besides similar sensor technologies, all vehicles share a common computer hardware and software framework [172]. The software is organized around the real time database *KogMo-RTDB* [74, 75]. This centralized architecture has shown great benefits, since data produced by one software module is often used by several other modules. For example, the pose and velocity of other traffic participants is used by the vision software, the trajectory planner, and the safety assessment module.

Data is written into and read from the real time database by a C or C++ interface. The results of the safety assessment algorithms previously presented in Sec. 5.4-5.7 were implemented in Matlab. In order to use the algorithms within the software framework of the autonomous car, the code has been rewritten using the C++ language. Analogously to the Matlab implementation, the fact that the transition matrices of the Markov chains are sparse and that most values of the probability vector are 0 has been taken advantage of.

5.8.2. Scenario

The algorithms developed for the safety assessment of the autonomous vehicle have been tested with sensor information recorded at the University of the Bundeswehr München in Neubiberg, Germany. An aerial image of the test location, on which the driven road

section is highlighted, can be found in Fig. 5.46. The test scenario was driven with two autonomous cars and a human-driven car as depicted in Fig. 5.46.

At the beginning of the scenario, the human-driven car is the leading vehicle, followed by *MuCAR-3*, which in turn is followed by *MUCCI* (see location ① and ②). At position ③, the autonomous vehicle *MuCAR-3* decides to overtake the human-driven car. During the overtaking maneuver, the autonomous car *MUCCI* closes the gap to the human-driven car (location ④). The scenario is finished as soon as *MuCAR-3* has successfully ended its overtaking maneuver at location ⑤.

The safety assessment has been performed for *MUCCI*, but could in principle be ported to *MuCAR-3* due to the common hardware and software framework. However, the porting would still require several modifications because the software modules reading and writing to the common real time database differ from vehicle to vehicle. Screenshots of the video image of *MUCCI* and the graphical user interface (GUI) of the safety assessment software are shown in Fig. 5.47 at different locations marked in Fig. 5.46. The GUI is not directly connected to the safety verification algorithms, but gathers the displayed information from the real time database. Data that is written from the safety verification module to the real time database include the predicted probability distributions of other traffic participants, the computation time for the prediction, and the crash probabilities over time.

The computation time was about 0.1 s when one vehicle was detected and about 0.15 s when two vehicles were detected using the computer hardware described in [75]. The prediction horizon was fixed to 5 s such that the prediction was about 30-50 times faster than real time.

The main purpose of this test drive was to demonstrate the interaction with other software modules connected through the real time database. Another aspect was to show the real time capabilities of the used algorithms. However, due to the not yet fully achieved capability of driving in real traffic, the compliance of the predictions with real traffic behavior could not be tested. This shortcoming and possible future work for handling more complicated scenarios is discussed next.

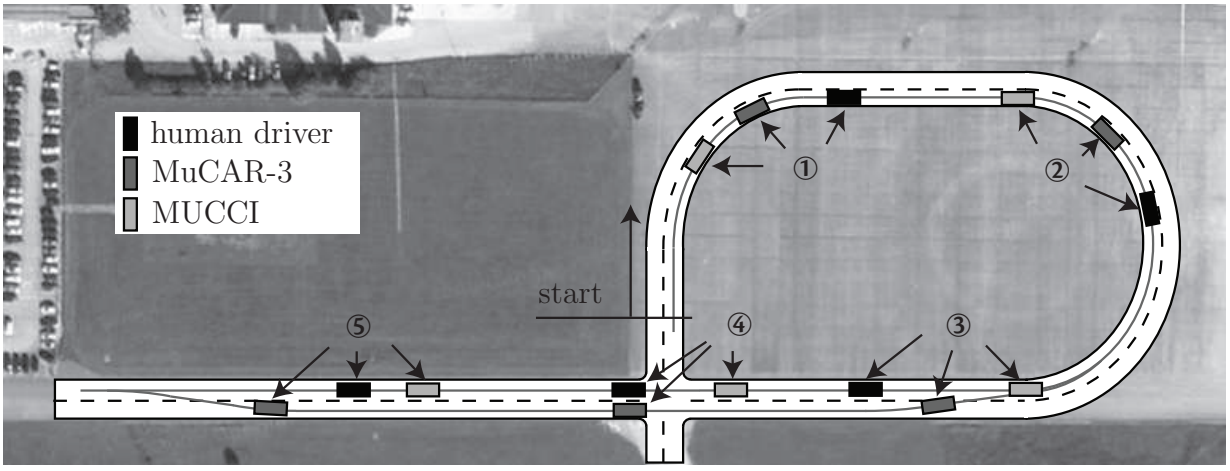
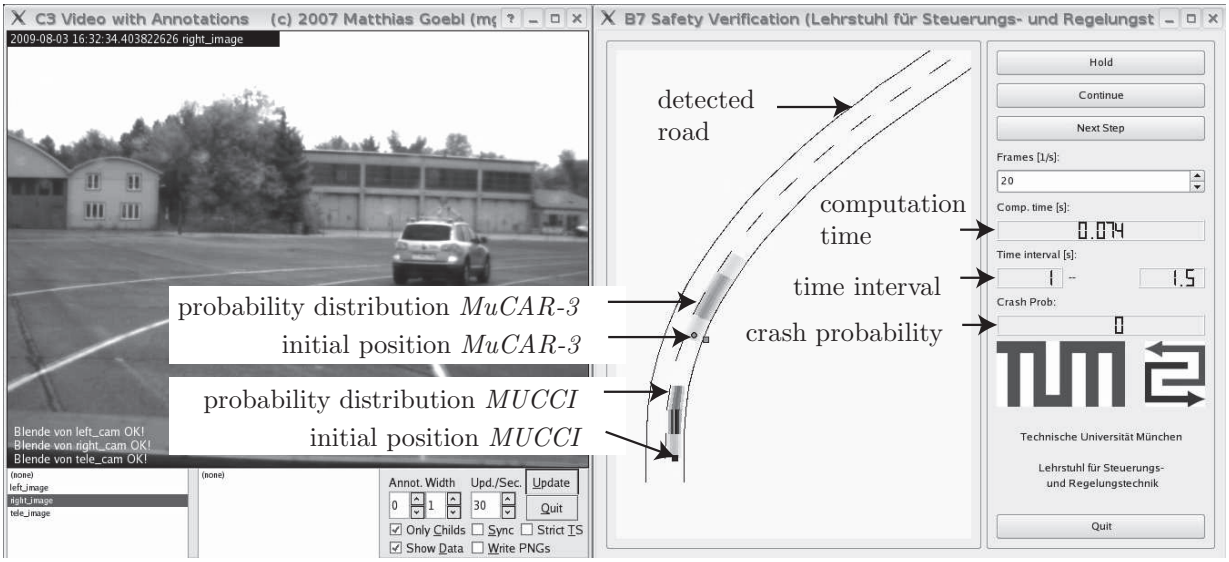


Fig. 5.46.: Test drive scenario.



(a) Location ①.

(b) Location ②.

(c) Location ③.

(d) Location ④.

(e) Location ⑤.

Fig. 5.47.: Screenshots of the test drive scenario.

5.9. Summary

A framework for the safety assessment of autonomous vehicles with respect to the planned driving actions has been developed. Thereby, the whole chain – consisting of modeling (vehicle model, stochastic behavior model), computational realization (Markov chain abstraction, crash probability computation), comparison to alternative approaches (Monte Carlo simulation), and testing (test drive of a prototype vehicle) – has been processed.

In the modeling section, the concept has been introduced that each traffic participant has one or several paths to be followed. Due to the concept of defining possible paths for each

vehicle, the motion along a path can be described by the position and the velocity only. The deviation probability distribution along the paths is modeled time invariant, except for lane change maneuvers. A dynamic adaption of the deviation probability distribution is subject to future work. The probability distribution of the path coordinate is obtained from an abstraction of traffic participants to Markov chains. Since the longitudinal dynamics of traffic participants consists of position and velocity only, the abstraction to Markov chains is computationally feasible.

A model considering arbitrary motions on a two-dimensional plane would be too high-dimensional for a practical Markov chain abstraction. However, Monte Carlo simulations also suffer from computational complexity in fully two-dimensional scenarios. When a single path does not sufficiently describe a situation, multiple paths could compensate for the shortcomings of the path concept; see Fig. 5.45. Alternative paths, which are not generated from the center of possible lanes could be obtained by Monte Carlo simulation or optimization techniques with alternative cost functions, realizing different types of drivers (e.g. safe or sporty drivers). The advantage of multiple paths with deviation probability is that many fewer paths have to be considered than in a pure Monte Carlo simulation of a two-dimensional scenario.

A particularity of the Markov chain abstraction for the longitudinal vehicle dynamics is that the abstraction does not have to be complete. The completeness can be achieved by additional simulations of the vehicle dynamics which result in lower and upper bounds of positions and velocities. Since the Markov chain computations can be incomplete, small probabilities for certain state and input values are canceled, followed by a normalization of the remaining probability values. This procedure allows fewer probabilities to be stored and further reduces the computational burden due to the sparse representations of probability values.

The stochastic acceleration values of other traffic participants have been modeled by another Markov chain. The transition probabilities of the acceleration values are based on simulations and heuristics. A learning approach which directly incorporates recorded traffic data would be a reasonable alternative, which becomes realizable when the detection of traffic participants and their tracking is more reliable in the future. For the heuristic approach, different stochastic models for typical patterns such as road following, vehicle following, intersection crossing, and lane changing have been suggested. Other behaviors that do not fall into these categories are handled by the unstructured environment approach used for a mobile robot in human-populated areas [195].

The computed probability distributions of the Markov chains are the basis for the computation of the crash probability. In this work, the probability of a crash is computed under the assumption that the autonomous vehicle has not crashed until the investigated point in time (conditional crash probability). This has the advantage that the crash probability of a particular time interval can be assessed independently from previous occurrences. Computational techniques for the efficient computation of the conditional crash probability have been presented.

The obtained probability distributions and crash probabilities have been compared to a Monte Carlo implementation. This comparison not only reveals implementation errors, but also allows the strengths and weaknesses of the Markov chain abstraction to be assessed. One of the main advantages of the Markov chain abstraction is the better efficiency in

computing probability distributions, especially when no analytical solution is available for the vehicle model. The main disadvantages of the Markov chain approach compared to the Monte Carlo simulation are that it computes worse crash probabilities and is less flexible. The worse flexibility is especially evident when the interaction between traffic participants is considered.

It is finally remarked that the probability distribution of other vehicles could not only be used to estimate the threat of a planned trajectory, but might also be used by the trajectory planner for a goal-oriented adaption of existing trajectories to safer trajectories.