Comparison of Markov Chain Abstraction and Monte Carlo Simulation for the Safety Assessment of Autonomous Cars

Matthias Althoff and Alexander Mergel

***Abstract*—The probabilistic prediction of road traffic scenarios is addressed. One result is a probabilistic occupancy of traffic participants, and the other result is the collision risk for autonomous vehicles when executing a planned maneuver. The probabilistic occupancy of surrounding traffic participants helps to plan the maneuver of an autonomous vehicle, whereas the computed collision risk helps to decide if a planned maneuver should be executed. Two methods for the probabilistic prediction are presented and compared: 1) Markov chain abstraction and 2) Monte Carlo simulation. The performance of both methods is evaluated with respect to the prediction of the probabilistic occupancy and the collision risk. For each comparison test, we use the same models that generate the probabilistic behavior of traffic participants, where the generation of these data is not compared with real-world data. However, the results independently show the behavior generation that Markov chains are preferred for the probabilistic occupancy, whereas Monte Carlo simulation is clearly preferred for determining the collision risk.**

***Index Terms*—Autonomous cars, behavior prediction, crash probability, Markov chains, Monte Carlo simulation, probabilistic occupancy, safety assessment, threat level.**

I. Introduction

ONE OF the main objectives of research on autonomous vehicles is to realize the vision of accident-free driving by excluding human error. This can be done by fully autonomous vehicles or partly autonomous vehicles, which only take over the control when an accident is or is almost inevitable.

To assess the safety of a planned maneuver, the prediction of traffic participants is vital for the identification of future threats. In contrast to predictive approaches, nonpredictive methods are based on the recording and evaluation of traffic situations that have resulted in dangerous situations (see, e.g., [1] and [31]). However, this approach is only suitable for driver warnings. Planned trajectories of autonomous cars cannot be evaluated with this method since the consequences when following these trajectories have to be predicted.

Behavior prediction has mainly been 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 that warn drivers when dangerous situations are ignored. The majority of works on this topic use learning mechanisms, such as neural networks and autoregressive exogenous models [37], [42], or filter techniques, such as Kalman filters [26], [33]. Another line of research is to detect traffic participants on selected road sections and learn motion patterns for anomaly detection. The developed learning techniques for motion patterns are, e.g., clustering methods [20], hidden Markov models [30], and growing hidden Markov models [39]. The disadvantage of a prediction at fixed locations is that the predictions are specialized to this particular road segment and are probably not representative of other traffic situations.

For the prediction of arbitrary traffic situations, simulations of traffic participants have been used [5], [18]. 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. Simulations of traffic participants are also computed in microscopic traffic simulations [28], [38]. However, single simulations do not consider uncertainties in the measurements and actions of other traffic participants, which may lead to unsatisfactory collision predictions [25].

A more sophisticated threat assessment considers multiple simulations of other vehicles, considering different initial states and changes in their inputs (steering angle and acceleration). Multiple simulations have been used in [22] to identify collision-free maneuvers for triggering emergency braking. If the simulations are randomly generated, then one computes by a so-called Monte Carlo method, which has been studied in [4], [7]–[9], and [13] for the risk analysis of road traffic and in [6] and [40] for the related topic of air traffic safety.

Another method to compute the probabilistic behaviors of traffic participants is to abstract their behavior. One approach is to linearize the system dynamics and compute with Gaussian distributions since the distribution remains Gaussian after a linear transformation [24]. Several linearizations representing different operation modes, such as stopping or accelerating, are computed in [17]. The operation modes are probabilistically switched, and the switching is Markovian. Another approach

Manuscript received April 12, 2010; revised August 26, 2010, December 15, 2010, and April 21, 2011; accepted May 5, 2011. Date of publication June 13, 2011; date of current version December 5, 2011. This work was supported in part by Deutsche Forschungsgemeinschaft (German Research Foundation) within the Transregional Collaborative Research Center 28 "Cognitive Automobiles." The Associate Editor for this paper was B. De Schutter.

M. Althoff is with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: malthoff@ece.cmu.edu).

A. Mergel is with the Institute of Automatic Control Engineering, Technische Universität München, 80290 München, Germany (e-mail: alexander.mergel@mytum.de).

Digital Object Identifier 10.1109/TITS.2011.2157342

is to abstract traffic participants by Markov chains, as presented in previous works [2], [3].

A. Contributions

The main contribution of this paper is to compare the Monte Carlo simulation in [4], [7]–[9], and [13] with the Markov chain abstraction in [2] and [3] according to their performance in the probabilistic prediction of traffic situations. However, to compare both techniques, an appropriate behavior model has to be found to generate acceleration commands. For the first time, the behavior models used in [2], [3], [7]–[9], and [13] are analyzed according to the autocorrelation and the average spectral density. In addition, the abstraction of Markov chains is computed via simulation techniques instead of reachability analysis, as in [2] and [3], for better accuracy. Accuracy has further been improved in the Markov chain approach by resolving an unnecessary overapproximation in the crash probability computation and by using a more accurate vehicle model. The efficiencies of the Markov chain computations and the Monte Carlo simulations have been increased as well. The Markov chain computation has been sped up by reformulating it to a sparse matrix multiplication and applying on-the-fly cancellation of nonrelevant probabilities. The Monte Carlo simulations have been sped up by computing with analytical solutions instead of numerical solvers.

B. Organization

In Section II, we give a detailed problem description. The modeling of traffic participants for the prediction is presented in Section III. We introduce the Markov chain abstraction and the Monte Carlo simulation in Sections IV and V, respectively. The probabilistic generation of acceleration commands is presented in Section VI. The remainder of this paper is about the assessment of the Markov chain abstraction and the Monte Carlo simulation. First, the results for the prediction of the probabilistic occupancy are compared in Section VII. Second, the predicted crash probabilities are compared in Section VIII.

II. Problem Description

To get a better idea of the addressed problem, the required information and the objectives are subsequently described.

A. Required Information

The presented safety assessment requires the following information:

1) the planned trajectory of the autonomous car;
2) the geometric description of the relevant road sections;
3) the position and geometry of static obstacles;
4) the position, velocity, and classification of traffic participants (into cars, trucks, motorbikes, bicycles, and pedestrians).

Note that the gathering of this data is not the subject of this work. The planned trajectory of the ego car is known since it is internally planned. The geometric description of the relevant road sections can be extracted from off-the-shelf navigation systems and fused with lane detection software. For lane detection, one usually uses light detection and ranging (LIDAR) sensors and cameras [29]. The same sensors are generally used for the detection of static obstacles. The detection of traffic participants and the estimation of their position and velocity are challenging tasks [10], [32]. Since these data are difficult to infer, the result of dynamic object detection is usually described by a probability distribution.

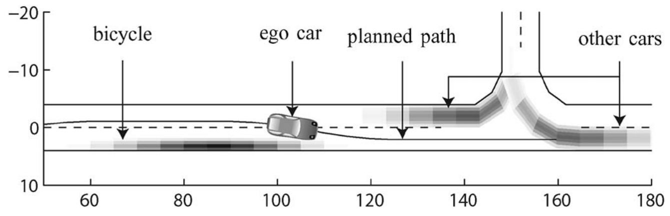

Fig. 1. Probabilistic occupancy of other traffic participants at a certain point in time. The result can be used for path planning of the ego car to avoid areas of high probability of occupancy.

The specialty of the proposed methods for safety assessment is that they can compute with the uncertainties provided by the dynamic object detection. The probability distributions of measurements can be arbitrary and may have multiple maxima due to multiple object hypotheses. In addition to this uncertainty, there is more uncertainty regarding the future behavior of other traffic participants.

B. Objectives

Incorporating the uncertainty from the obstacle detection and the future behavior, this work predicts the future probabilistic occupancy of obstacles on the road. The second result of this work is the probability of a crash for the autonomous vehicle when following its planned path.

The probabilistic occupancy of other traffic participants allows to optimize the planned paths such that the autonomous car does not come to close to other traffic participants. This is illustrated in Fig. 1. The crash probability helps to decide if a planned trajectory should be executed or not. If several trajectories are planned, then the safety assessment can identify the trajectory with the least crash probability. The interaction of the safety assessment module with the sensor information and the trajectory planner is illustrated in Fig. 2. To update the prediction for $t \in [0, t_f]$ after a time interval (TI) Δt based on new sensor values, the computation has to be faster than real time by a factor of $t_f/\Delta t$.

III. Modeling of Traffic Participants

This work focuses on the safety assessment of autonomous cars driving on a road network, i.e., the motion of traffic participants is constrained along designated roads. On that account, the motion of traffic participants is modeled such that traffic participants follow certain paths up to a certain accuracy. An alternative modeling for probabilistic prediction in unstructured environments can be found, e.g., in [35].

The generated paths are located in the centers of lanes that can be followed. Possible lanes to be followed are extracted

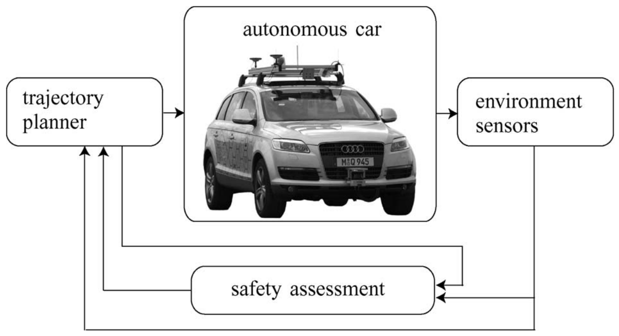

Fig. 2. Concept of safety assessment. The environment sensors provide road geometry, static obstacles, and traffic participants. This information is forwarded to the trajectory planner and the safety assessment module. The trajectory is only executed when approved by the safety assessment module.

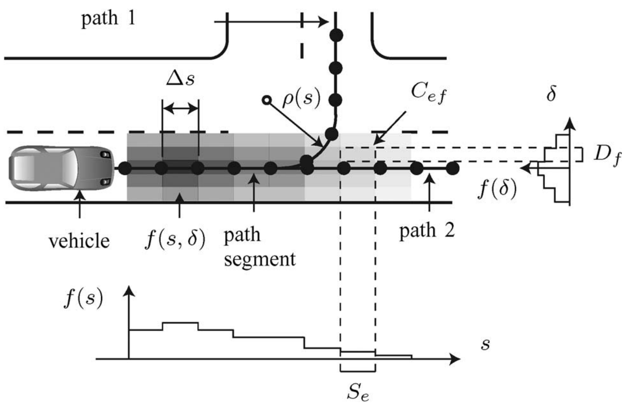

Fig. 3. Probability distribution of the position of a vehicle along a path-aligned coordinate system.

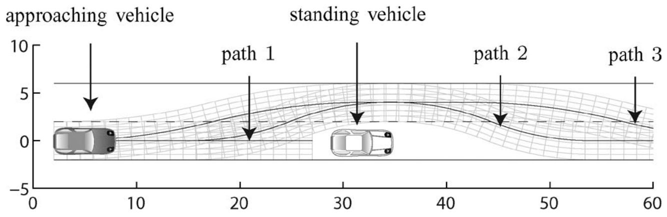

Fig. 4. Evasion of a standing car with alternative paths.

from detected road sections by checking all possible driving options, such as left turn, right turn, go straight, left lane change, and right lane change. Possible driving paths of an exemplary road section using the aforementioned driving options are shown in Fig. 3. In some traffic situations, additional paths that use lanes for oncoming traffic are required. This is the case when an obstacle (e.g., standing car) blocks a lane. Possible paths for such a scenario are shown in Fig. 4 and can be obtained with the same methods used in the trajectory planner but applied to other traffic participants instead of the ego car. The concept of generating paths that are followed by a certain accuracy is also suggested in [17].

A. Lateral Distribution

The deviation along generated paths is modeled by a piecewise constant probability density function $f(\delta)$, where δ is the lateral deviation from a driving path, as shown in Fig. 3. The deviation probability of other traffic participants at the beginning of the prediction is set according to the measurement uncertainty of the obstacle detection. This distribution can be cross faded over time to a distribution that has been obtained from an averaging of traffic observations (see [11]).

B. Longitudinal Distribution

In contrast to the path deviation, the longitudinal dynamics of traffic participants along a path can be much better described by a mathematical model. To formulate this model, the volumetric center of traffic participants along a path is denoted by s, the velocity by v, and the absolute acceleration by a. 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, and the normal acceleration is denoted by a_N. The differential equations for vehicle dynamics are

$$\dot{s} = v, \quad \dot{v} = \begin{cases} a^{\max} u, & 0 < v \le v^{\mathrm{sw}} \vee u \le 0 \\ a^{\max} \frac{v^{\mathrm{sw}}}{v} u, & v > v^{\mathrm{sw}} \wedge u > 0 \\ 0, & v \le 0 \end{cases} \tag{1}$$

subject to the constraints

$$\begin{aligned} & v \le v^{\max} \\ & |a| \le a^{\max}, \quad |a| = \sqrt{a_N^2 + a_T^2}, a_N = v^2/\rho(s), a_T = \dot{v}. \end{aligned} \tag{2}$$

Backwards driving on a lane is not considered; see (1) ($\dot{v} = 0, v \le 0$). The positive acceleration dynamic changes at the switching velocity v^{sw}. The dynamics for $0 < v \le v^{\mathrm{sw}} \vee u \le 0$ is based on tire friction, whereas the other dynamics is based on engine power, which limits the acceleration when the torque for $v > v^{\mathrm{sw}}$ is not causing wheel spin anymore. The constraint $v \le v^{\max}$ in (2) models the speed limit. The other constraint models that the tire friction only allows a limited absolute acceleration $a^{\max}$ (Kamm's circle). The constant $a^{\max}$ is chosen as $a^{\max} = 7$ [m/s^2], and v^{sw} is chosen according to the different classes of traffic participants.[1]

The proposed model for the longitudinal dynamics is almost the same as used in [13]. The difference there is that the acceleration command resulting in zero acceleration is nonzero. It is also remarked that the model in this paper differs from the previous work in [2]. The new model is more accurate, which can be observed by plotting the acceleration over the velocity and comparing it with the result of a high-fidelity simulation[2] of an Audi Q7 in Fig. 5. Note that the peaks in acceleration occur due to the torque converter of the automatic gear box.

A specialty of the proposed longitudinal dynamics is that an analytical solution exists for the constant u. This is beneficial for the Monte Carlo simulation, as introduced later.

[1]Used values in this work: Car: 7.3 [m/s], truck: 4 [m/s], motorbike: 8 [m/s], bicycle: 1 [m/s].

[2]Used software: veDYNA.

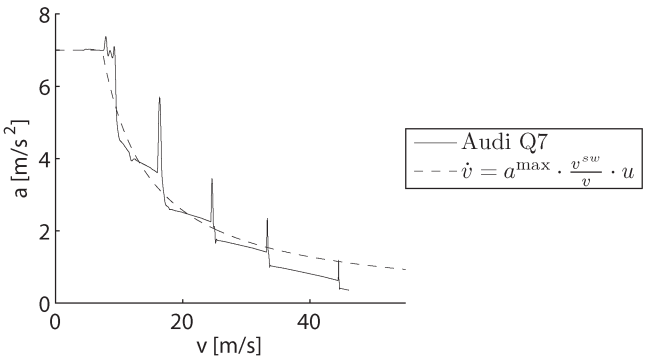

Fig. 5. Maximum acceleration a of the Audi Q7 plotted over its velocity v. Used parameters: $a^{\max} = 7$ [m/s^2], and $v^{\rm sw} = 7.3$ [m/s].

Proposition 1 [Analytical Solution of (1)]: The analytical solution of the longitudinal dynamics of traffic participants in (1) for $u > 0$ ($u = \texttt{const}$) and $v > v^{\rm sw}$ is

$$s(t) = s(0) + \frac{\left(v(0)^2 + 2v^{\rm sw}ut\right)^{\frac{3}{2}} - v(0)^3}{3v^{\rm sw}u}$$
$$v(t) = \sqrt{v(0)^2 + 2v^{\rm sw}ut}.$$

The analytical solutions of the cases $0 < v < v^{\rm sw} \vee u < 0$ and $v \leq 0$ are trivial. ■

The correctness can easily be verified by inserting the solution into (1).

C. Combined Distribution

The driver task of velocity control and lane keeping can be assumed to be fairly independent (see [14] and [21]). The same observation holds for autonomous vehicles whose vehicle control is separately developed for lateral and longitudinal dynamics [34] or whose control has negligible dependency with respect to the safety assessment in road traffic. Thus, it is assumed that the longitudinal probability distribution $f(s)$, which is obtained from the longitudinal dynamics (1), is independent from the lateral distribution so that the overall distribution is $f(s, \delta) = f(s) \cdot f(\delta)$, as indicated in Fig. 3. It is emphasized that the lateral and longitudinal distributions $f(\delta)$ and $f(s)$ refer to the volumetric center of the vehicle bodies.

In this paper, the lateral and longitudinal probability distributions are modeled as a piecewise constant distribution, as shown in Fig. 3. An interval with constant probability distribution along a path is denoted by S_e and by D_f for a deviation interval. The region with constant probability distribution spanned by S_e and D_f is denoted by C_{ef} (see Fig. 3), which is required later.

The vehicle bodies and the occupancy of pedestrians on the road are modeled by rectangles whose size varies between the different types of traffic participants (cars, trucks, motorbikes/ bicycles, and pedestrians).

IV. Markov Chain Abstraction

First, Markov chain abstraction is considered for predicting the longitudinal probability distribution. The techniques for the abstraction of dynamic systems with continuous state variables to Markov chains are manifold, where many of them couple the time increment with the accuracy of the abstraction; see, e.g., [19] and [23]. However, this coupling is unfavorable in terms of real-time applicability as a required approximation accuracy may lead to short time increments of probability updates—and consequently to too many update iterations that cannot be handled in real time. For this reason, the update intervals and the approximation accuracy are decoupled as in [27] and [36].

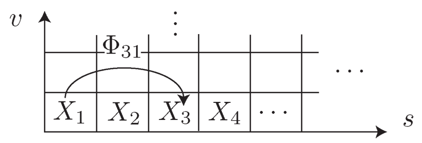

Fig. 6. Discretization of the state space.

The abstraction, which is computed offline, allows the dynamics of a traffic participant (1) to be described by a Markov chain. During online operation, a Markov chain is instantiated for each relevant traffic participant around the ego vehicle.

A Markov chain is a stochastic dynamic system with discrete state $z \in \mathbb{N}^+$. There are discrete time and continuous time Markov chains. In this paper, discrete-time Markov chains are used such that $t \in \{t_1, t_2, \ldots, t_f\}$, where t_f is the prediction horizon, and $t_{k+1} - t_k = T \in \mathbb{R}^+$ is the time step increment. The current state of Markov chains is not exactly known, but probabilities $p_i = P(z = i)$ describe that the system is in state $z = i$, which are combined to a probability vector p for all states. By definition, the probability vector for the next time step t_{k+1} is a linear combination of the probability vector of the previous time step t_k, i.e.,

$$p(t_{k+1}) = \Phi p(t_k) \tag{3}$$

where Φ is referred to as the transition matrix.

Since the original system has continuous state variables, regions of the continuous state space have to be assigned to discrete states. In this paper, a region $X \subset \mathbb{R}^2$ of the continuous state space of (1) is discretized in orthogonal cells of equal size, resulting in rectangular cells (see Fig. 6). The state space cells are denoted by X_i, where the index i refers to the value of the corresponding discrete state. In an analogous way, the region $U = [-1, 1]$ of the input space of (1) is discretized into intervals U^α. The index α refers to the value of the discrete input y. To distinguish between indices referring to discrete state or input values, state indices are subscripted and Latin, where input indices are superscripted and Greek. To obtain a continuous distribution from the probability vector p, it is assumed that the probability distribution is uniform within each cell.

A. Transition Probabilities of the Markov Chain

The transition probabilities store the probabilities that the discrete state changes from j to i, i.e., $\Phi_{ij} = P(z(t_{k+1}) = i|z(t_k) = j)$. In this paper, the transition probabilities depend on the value of the discrete input y as well. For this reason, a different transition probability matrix Φ^α is computed for each discrete input value α such that $\Phi^\alpha_{ij} = P(z(t_{k+1}) = i, y(t_{k+1}) = \alpha|z(t_k) = j, y(t_k) = \alpha)$.

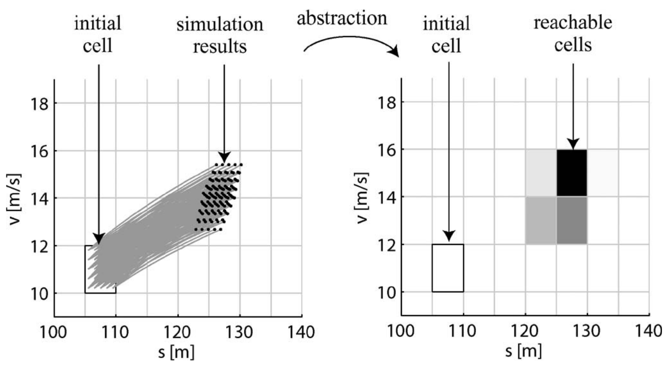

Fig. 7. Simulation results of the original system (left) and the corresponding probabilistic reachable set of the abstracting Markov chain (right). Both results are obtained using the same initial and input cells. A transition to a cell is more likely the darker the color of the cell is.

The transition probabilities are obtained by running N_j^α simulations starting from the initial cell X_j under input $u \in U^\alpha$. The number N_{ij}^α of those simulations ending up in cell X_i determines the transition probability $\Phi_{ij}^\alpha = N_{ij}^\alpha / N_j^\alpha$.

The input values are held constant during the simulation but have varying values from simulation to simulation. Since it is assumed that the initial states and the inputs are uniformly distributed within cells, the initial states and the inputs are drawn from a uniform grid within the cells, as illustrated in Fig. 7.

Another possibility to abstract a continuous dynamics to a Markov chain is via reachability analysis, as described in [2]. This technique is less accurate in terms of the resulting transition probabilities but makes it possible to compute complete abstractions, i.e., all states reachable by the abstracted Markov chain (cells with nonzero probability) cover all possible trajectories of the original system. Completeness makes it possible to guarantee that no crash occurs when the crash probability is 0. However, a crash-free trajectory can also be guaranteed when the intersection between reachable road sections of traffic participants and the autonomous car are empty. Reachable road sections can be computed, as proposed in [2]. Vehicles that do not pose a threat can be ignored for the probabilistic computations, which are also proposed in [17]. Since completeness is not required, the more accurate simulation technique is applied.

B. Computing Stochastic Reachable Sets Using Markov Chains

The transition probabilities allow to compute the state probabilities, as shown in (3). The difference, however, is that the transition probabilities incorporate the input probability as well. To use the update scheme of a Markov chain, the state probability vector has to be redefined. Denoting the joint probability of a state value and an input value as $p_i^\alpha := P(z = i, y = \alpha)$, the combined probability vector is defined as

$$\tilde{p}^T = \begin{bmatrix} p_1^1 & p_1^2 & \dots & p_1^c & p_2^1 & \dots & p_2^c & p_3^1 & \dots & p_d^c \end{bmatrix}$$

where d is the number of states, and c is the number of inputs. Given this probability vector, the transition values have to be organized in the transition matrix as

$$\tilde{\Phi} = \begin{bmatrix} \Phi_{11}^1 & 0 & \cdots & 0 & \cdots & \Phi_{1d}^1 & 0 & \cdots & 0 \\ 0 & \Phi_{11}^2 & \cdots & 0 & \cdots & 0 & \Phi_{1d}^2 & \cdots & 0 \\ \vdots & & & & & & & \vdots & \\ 0 & 0 & \cdots & \Phi_{d1}^c & \cdots & 0 & 0 & \cdots & \Phi_{dd}^c \end{bmatrix}.$$

The rewritings allow to update the probabilities by $\tilde{p}(t_{k+1}) = \tilde{\Phi}\tilde{p}(t_k)$. The transition matrix $\tilde{\Phi}$ is sparse, i.e., it does not contain many nonzero values, which allows to increase the efficiency of the matrix multiplication (see, e.g., [43]). The efficiency is even more increased when canceling small probabilities in $\tilde{p}$ after each time step t_k and normalizing $\tilde{p}$ afterward such that its sum is 1. This is reasonable since completeness is not required. The value $\underline{\tilde{p}}$, below which probabilities are canceled, is set by the minimum probability density $\underline{\xi}$, which is uniform within each cell according to previous assumptions so that

$$\underline{\tilde{p}} = s^\Delta v^\Delta u^\Delta \underline{\xi} \tag{4}$$

where s^Δ, v^Δ, and u^Δ are the cell lengths of position, velocity, and input value. The default value of $\underline{\xi}$ is 0 (no cancelation), and good results have been obtained with $\underline{\xi} = 1/16 \cdot 10^{-3}$. Higher values of $\underline{\xi}$ reduce computation time while decreasing the accuracy.

V. Monte Carlo Simulation

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) Generate inputs and initial states randomly.
2) Perform a deterministic computation starting at the initial states subject to the randomly generated inputs.
3) Aggregate the results of the individual computations into the final result.

In this paper, the initial states and the input values have a piecewise constant probability distribution to compare the results with the Markov chain computations. The random values are obtained by the inverse transform method (see [41]) according to the given piecewise constant probability distributions. The deterministic computations for the longitudinal dynamics are simply computed as presented in Proposition 1. The aggregation of results is discussed in more detail below.

A. Aggregation of Results

The aggregation of the results depends on the purpose of the Monte Carlo simulation. For the computation of the probabilistic occupancy, it is checked in which segment (seg) S_i of the followed path a simulation ends. For the computation of the crash probability, it is checked if a crash occurs. The detection of these events is formalized by indicator functions that use the general vector x containing all the variables of the Monte Carlo simulation, i.e.,

$$\mathtt{ind}_i^{\mathtt{seg}}(x) = \begin{cases} 1, & \text{if } s(x) \in S_i \\ 0, & \text{otherwise} \end{cases}$$

$$\mathtt{ind}^{\mathtt{crash}}(x) = \begin{cases} 1, & \text{crash} \\ 0, & \text{otherwise.} \end{cases}$$

Since in this work, all the simulations have equal weight, the probabilities of occupying a path region or causing a crash are obtained by the relative number of simulations for which the indicator function is 1.

B. Error Analysis

An intrinsic property of the 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 $1/\sqrt{N_s}$, where N_s is the number of simulations. This is a well-known result from the Monte Carlo integration [41] of probability density functions $f(x)$ and holds for the problems discussed herein. This can be shown by reformulating the computations as a Monte Carlo integration using the indicator functions. The probability $\hat{p}_i$ that the path position s is in S_i and the crash probability p^C are computed as

$$\hat{p}_i = \int \mathtt{ind}_i^{\mathtt{seg}}(x) f(x) dx, \quad p^C = \int \mathtt{ind}^{\mathtt{crash}}(x) f(x) dx.$$

VI. Generation of Acceleration Commands

An important influence on the outcome of the probabilistic predictions is given by the sequence of probabilistic acceleration commands over time. Two methods for the generation of acceleration commands are presented and then compared. The first method is a Markov chain approach, as presented in [2]. The other method is an approach based on Monte Carlo simulation, as presented in [8], [9], and [13]. In all the approaches, the input value is held constant for a certain time span. In this paper, this time span is chosen as for the time increment T of the Markov chain abstraction. The comparison of the generated acceleration commands to real-world measurements is future work.

A. Markov Chain Approach

In [2], not only the state transitions but also the probabilistic acceleration values are generated by Markov chains. As previously mentioned, the acceleration commands are held constant during the TIs $[t_k, t_{k+1}]$ and are instantly changed at times t_k. The probability that the input value changes for a given state value is described by the input transition values $\Gamma_i^{\alpha\beta}(t_k) := P(z(t_k') = i, y(t_k') = \alpha | z(t_k) = i, y(t_k) = \beta)$, where $t_k' = t_k + \delta t$, and δt is an infinitesimal time step. Those transition values are combined into an input transition matrix $\tilde{\Gamma}(t_k)$ similarly as for the state transition matrix $\tilde{\Phi}$, i.e.,

$$\tilde{\Gamma} = \begin{bmatrix} \Gamma_1^{11} & \Gamma_1^{12} & \cdots & \Gamma_1^{1c} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \Gamma_1^{21} & \Gamma_1^{22} & \cdots & \Gamma_1^{2c} & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & & & & & & & \vdots & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 & \Gamma_d^{c1} & \cdots & \Gamma_d^{cc} \end{bmatrix}.$$

In contrast to the computation of the transition matrix $\tilde{\Phi}$ for the states, the transition matrices $\tilde{\Gamma}(t_k)$ for the input cannot be computed based on a reasonably simple dynamic model (due to the complexity of human behavior or decision systems of autonomous vehicles). As a consequence, the transition matrices $\tilde{\Gamma}(t_k)$ have to be learned by observation or set by a combination of offline simulations and heuristics, where the latter is used in [2] considering the constraints of the vehicle model in (2).

Combining the input and the state transition matrix yields the extended formula of the Markov chain update for time varying input probabilities, i.e.,

$$\tilde{p}(t_{k+1}) = \tilde{\Gamma}(t_k)\tilde{\Phi}\tilde{p}(t_k). \tag{5}$$

B. Trajectory Weighting

Another way of creating random inputs is proposed in the works on Monte Carlo simulation of road traffic scenes [8], [9], [13]. In these approaches, a random trajectory is created first, and afterward, its likeliness is evaluated, i.e., a weight is assigned to it. The values of the input trajectories are created by an independent identically distributed (i.i.d.) process, but the final process is no longer i.i.d. after a weight is assigned by the likeliness function. The inputs created in the aforementioned works are for the acceleration and the steering, whereas only the acceleration is of interest in this work. After removing the steering-related aspects, the so-called goal function for the computation of the likeliness of the input trajectory is computed in [9] and [13] as

$$g(u) = -\sum_{k=1}^{N_u} \left(\lambda_1 \left(v(t_k) - v^{\max}\right)^2 + \lambda_2 a_T(t_k)^2 + \lambda_3 a_N(t_k)^2\right)$$

where $\lambda_1 - \lambda_3$ are the tuning parameters, which punish velocity deviations from the allowed velocity, and large normal as well as tangential accelerations. The probability distribution of the input trajectories $f(u)$ is assumed to be $f(u) = c_n \exp(k_g \cdot g(u))$, 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.

C. Comparison

Continuous input values generated from the Markov chain approach and the trajectory weighting approach are compared according to their autocorrelation and average spectral density. Continuous input values are obtained from the discrete inputs of the Markov chain approach by the previously mentioned inverse transform method. The input trajectories for both approaches

TABLE I
PARAMETERS FOR INPUT TRAJECTORY GENERATION

General	
$v^{\max}$	$100/3.6$ m/s
T	0.5 s
Trajectory weighting; see [13]	
k_g	100
λ_1	$0.05/N_u/(1+v(0)^2)$
λ_2	$0.05/N_u/(a^{\max})^2$
λ_3	$0.05/N_u/(a^{\max})^2$
Markov chain; see [2]	
m	$[0.01, 0.04, 0.25, 0.25, 0.4, 0.05]$
$q(0)$	$[0,\ 0,\ 0.5,\ 0.5,\ 0,\ 0]$
γ	0.2

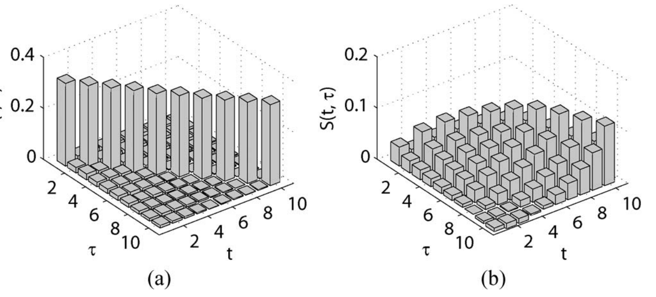

Fig. 8. Autocorrelation of input trajectories. (a) Trajectory weighting approach. (b) Markov chain approach.

are generated with the parameters listed in Table I for $N_u = 10$, where N_u is the number of time steps. The Monte Carlo simulation parameters are taken from [8]. The initial velocity is uniformly distributed within $v(0) = 15 \pm 1$ m/s, which affects the input generation due to the speed limit. The considered time horizon is $t_f = 5$ s, and the number of computed simulations is $N_s = 10^5$.

1) Autocorrelation: The autocorrelation, i.e., the correlation of the signal against a time-shifted version of itself, is defined as

$$S(t,\tau) := E\left[\mathbf{u}(t)\mathbf{u}(\tau)\right] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u_t u_\tau f(u_t u_\tau) du_t du_\tau \quad (6)$$

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 preceding integrals is approximated using Monte Carlo integration.

The autocorrelation values for the input trajectories are plotted in Fig. 8. There is almost no correlation between the inputs of different time steps in the trajectory weighting approach, as shown in Fig. 8(a). This is in contrast to the Markov chain approach, where the input signals are much more correlated [see Fig. 8(b)].

2) Average Spectral Density: In addition to the autocorrelation, the spectral density of input signals is 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 $\int_{-\infty}^{\infty}|x(t)|^2dt$ 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

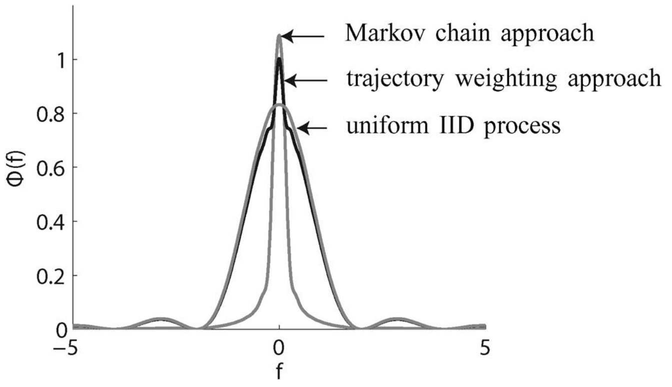

Fig. 9. Average spectral density of input trajectories.

of a stochastic signal, one is interested in the expectation of the random spectral density $E[|\mathbf{X}(f)|^2]$.

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

$$\Phi(f) = T^2\mathtt{si}^2(\pi f T)\sum_{k=1}^{N_u}\sum_{l=1}^{N_u} E\left[\mathbf{u}(t_k)\mathbf{u}(t_l)\right]e^{-j2\pi f T(k-l)}$$

where N_u is the number of time steps, and $\mathtt{si}(x) = \sin(x)/x$ is the sinc function. ■

Proof: The input signal $u(t)$ is a constant within consecutive TIs $[t_k, t_{k+1}]$, where $t_{k+1} - t_k = T$. This can be written as

$$u(t) = \sum_{k=1}^{N_u} u(t_k)\mathtt{rect}\left(\frac{t-(k-0.5)T}{T}\right)$$

where $\mathtt{rect}(t)$ is 1 for $-0.5 < t < 0.5$ and 0 otherwise.

The Fourier transform makes it possible to formulate the following relation between the time and frequency domains:

$$\mathtt{rect}\left(\frac{t-(k-0.5)T}{T}\right) \circ\!\!-\!\!\bullet\ jT\mathtt{si}(\pi f T)e^{-j2\pi f(k-0.5)T}.$$

For the whole input signal, the Fourier transform is $U(f) = jT\mathtt{si}(\pi f T)e^{j\pi f T}\sum_{k=1}^{N_u} u(t_k)e^{-j2\pi f k T}$, from which follows the spectral density $\Phi(f)$. ■

The expectation $E[\mathbf{u}(t_k)\mathbf{u}(t_l)]$ is obtained from the autocorrelation in (6). The expectations of spectral densities $\Phi(f)$ are visualized in Fig. 9. It can be seen that the lower frequencies are more dominant in the Markov chain approach, whereas the frequency distribution of the trajectory weighting approach is close to an i.i.d. process with uniform distribution.

A general observation in road traffic is that drivers change their acceleration with low frequency and that their inputs are highly correlated. In both tests, the autocorrelation and the average spectral density test, the acceleration inputs created by the Markov chain showed more realistic behavior.

VII. Comparison of Probabilistic Occupancy

In this section, acceleration commands from the previously presented Monte Carlo and Markov chain approaches are used

TABLE II
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	320	1.25 m	120	0.5 m/s

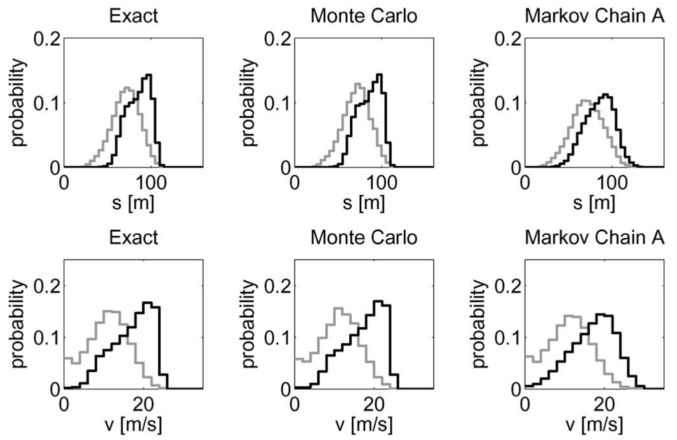

Fig. 10. Road following scenario. Position and velocity distributions for a coarse discretization ($t = 5$ s). Inputs generated by the Markov chain approach result in the black distribution, whereas the Monte Carlo generated inputs result in the gray distribution.

to compare the probability distribution of traffic participants. For each comparison, the same model for generating the acceleration commands is used, which ensures comparability of the distributions. Since the final probabilistic occupancy is simply obtained by $f(s, \delta) = f(s) \cdot f(\delta)$, where $f(\delta)$ is heuristically obtained, the following comparison is about $f(s)$ and the distribution of the velocity $f(v)$ as an auxiliary result.

The computed piecewise constant probability distributions of $f(s)$ and $f(v)$ with probabilities p_i for certain intervals are compared by the distance

$$d = \sum_i |p_i - p_i^{\mathrm{e}}| \, V(X_i)$$

where p^{e} is the exact probability distribution. The multiplication with the volume $V(X_i)$ of the cells X_i is required to compare the results of different discretizations.

For the numerical experiments, two different Markov chain discretizations are used, as listed in Table II. The cancelation of small probability values is performed as suggested in (4) with $\underline{\xi} = 1/16 \cdot 10^{-3}$. The Monte Carlo approach used in the numerical experiment is performed with 10^4 simulations. Since there is no exact solution for the presented scenario, an almost exact solution was computed with the Monte Carlo simulation using 10^7 samples. The acceleration command is generated by a Markov chain using the parameters in Table I. The initial position and the initial velocity are uniformly distributed in the intervals [2, 8] m and [15, 17] m/s, respectively. The probability distributions are compared at $t = 5$ s.

The resulting position and velocity distribution for different discretizations and inputs can be found in Figs. 10 and 11. The distance measure d for all tested combinations of input generations, prediction techniques, and discretizations is listed in Table III. Since the Monte Carlo approach delivers different results for each run, 100 runs have been computed, and the minimum, maximum, and mean values are shown. The results are fairly independent from the input generation, except that the Markov chain approach performs better when Monte Carlo generated inputs are used. This is because the probability of low velocities is high for those inputs so that the Markov chain specific problem of flattening distributions is limited to high velocities. The computational times can be found in Table IV, which are 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 Proposition 1. Ultimately, the Markov chain solution is faster than the analytically obtained Monte Carlo solution, and the discretization of the Markov chain B is fine enough to produce results that are more accurate than the Monte Carlo approach with 10^4 simulations.

Finally, it was analyzed if the quality of the probability distributions depend on the initial condition. As the vehicle model (1) is invariant under translations in position, it is only necessary to vary the initial velocity. The influence on the initial velocity of the distances d^{pos} and d^{vel} of the position and velocity is shown

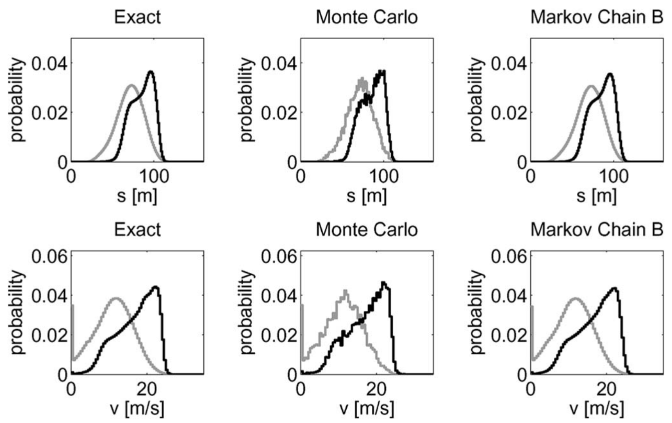

Fig. 11. Road following scenario. Position and velocity distributions for fine discretization ($t = 5$ s). Inputs generated by the Markov chain approach result in the black distribution, whereas the Monte Carlo generated inputs result in the gray distribution.

TABLE III
DISTANCE MEASURE d

		res. A position	res. A velocity	res. B position	res. B velocity
Input generation: Markov chain					
Monte Carlo:	min	0.0594	0.0218	0.0500	0.0166
	max	0.2393	0.0783	0.0905	0.0331
	mean	0.1327	0.0512	0.0677	0.0259
Markov chain:		1.0882	0.3425	0.0346	0.0121
Input generation: Monte Carlo					
Monte Carlo:	min	0.0885	0.0212	0.0573	0.0186
	max	0.2169	0.0985	0.0936	0.0338
	mean	0.1485	0.0518	0.0752	0.0261
Markov chain:		0.8438	0.1272	0.0165	0.0056

TABLE IV
COMPUTATIONAL TIMES OF THE ROAD FOLLOWING SCENARIO

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

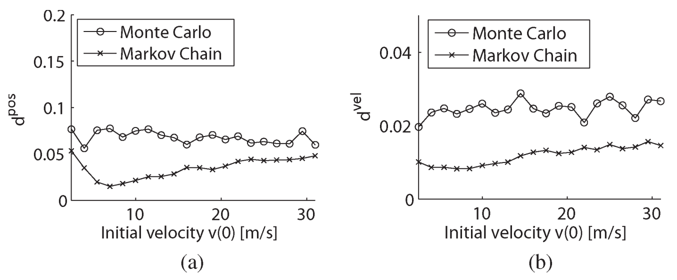

Fig. 12. Distance d to the exact solution for different initial velocities. (a) Distance d of the position distribution. (b) Distance d of the velocity distribution.

in Fig. 12. The Monte Carlo simulations are performed with 10^4 samples, and the Markov chain approach was computed with the B model. In contrast to 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 neglected, meaning that the results in Figs. 10 and 11 are representative. The small dependence on the initial state makes it possible to tune the discretization based on a single initial distribution. If the distance measure d of the position or velocity is too high, then one can make the corresponding discretization finer until the desired accuracy is achieved, which should approximately hold for all the other conditions.

VIII. Comparison of Crash Probabilities

In this paper, the crash probability of different TIs is computed independently of previous crash probabilities. This has the advantage that, for each TI, 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 (TP). However, the probability that the vehicle crashes in the first situation is greater than in the second situation. This is 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 TI.

The crash probability is computed for consecutive TIs since one may miss a high crash probability when only computing for TP. This is achieved in the Monte Carlo simulation by computing for sufficient intermediate TP within a TI. When using Markov chains, an additional transition matrix for TIs is computed offline as $\tilde{\Phi}^{\texttt{interval}} = (1/\tilde{n})\sum_{k=1}^{\tilde{n}} \tilde{\Phi}^{(\tilde{t}_k)}$, where $\tilde{\Phi}^{(\tilde{t}_k)}$ are transition probability matrices for intermediate TP $\tilde{t}_k$. The probability distribution for TIs is then computed as $\tilde{p}(t_k)^{\texttt{interval}} = \tilde{\Phi}^{\texttt{interval}}\tilde{p}(t_k)$, where $\tilde{p}(t_k)$ is computed in (5).

A. Computation of Crash Probabilities

The computation of crash probabilities is separately discussed for the Markov chain abstraction and the Monte Carlo simulation.

1) Markov Chain Abstraction: When using the Markov chain abstraction, one has to compute the probabilistic occupancy as an intermediate step. To describe the probabilistic occupancy, some additional notations have to be introduced. The region spanned by the path position interval S_e and the deviation interval D_f is denoted by C_{ef} (see Fig. 3). The probability that the center of a vehicle is in a region C_{ef} is denoted by $\hat{p}_{ef}$. The probabilities for the autonomous car are indicated by the superscript ego (e.g., $\hat{p}_{gh}^{\mathsf{ego}}$). It is required to additionally introduce the probability p_{ghef}^{int} that two bodies of the ego vehicle and another vehicle intersect when the center of the ego vehicle body is in C_{gh}^{ego} and that of the other vehicle in C_{ef}. This probability is computed by gridding the regions C_{gh}^{ego} and C_{ef} and testing for all combinations of centers if an intersection of vehicle bodies exists. The relative number of intersections provides the intersection probability. This computation is expensive so that the intersection probabilities are precomputed offline by storing the results for different relative positions, orientations, and types of traffic participants in a database. In the previous work [2], p_{ghef}^{int} was conservatively chosen to 1 when an intersection is possible.

Using the introduced variables, the crash probability between the autonomous vehicle and another vehicle can be formulated as

$$p^C = \sum_{g,h,e,f} p_{ghef}^{\mathrm{int}} \hat{p}_{gh}^{\mathsf{ego}} \hat{p}_{ef}.$$

The summation over all possible indices g, h, e, and f is computationally expensive. For this reason, techniques to effectively search for combinations of path and deviation indices, which potentially cause a vehicle body intersection, are presented in the previous work [2].

2) Monte Carlo Simulation: The crash probability for the Monte Carlo approach is simply obtained by summing up the probabilities of simulations that have crashed. Note that simulations resulting in a crash are not removed from the computation to obtain crash probabilities in compliance with the aforementioned definition of the crash probability.

It is crucial that the detection of a crash is computationally cheap. Crashes are detected by checking if the rectangular vehicle bodies intersect. An efficient method to detect the intersection of two rectangles is the separating axis theorem [16]. An extension considering the velocity of objects is presented in [12].

B. Crash Scenario

The crash probabilities are investigated for a scenario where the autonomous car drives behind another car in the same lane. The autonomous car starts from the position 0 m with constant velocity 20 m/s and has uniform position uncertainty of ±3 m. The vehicle driving in front has a uniform position uncertainty of [20, 25] m, and the initial velocity is within [15, 17] m/s. The other parameters are as listed in Table I, and the considered time horizon is $t_f = 5$ s. The (almost) exact solution is obtained from a Monte Carlo simulation with 10^5 simulations.

The crash probability of the Markov chain approach is compared with the exact solution with coarse and fine discretizations using models A and B (see Table II) and for TPs as well as

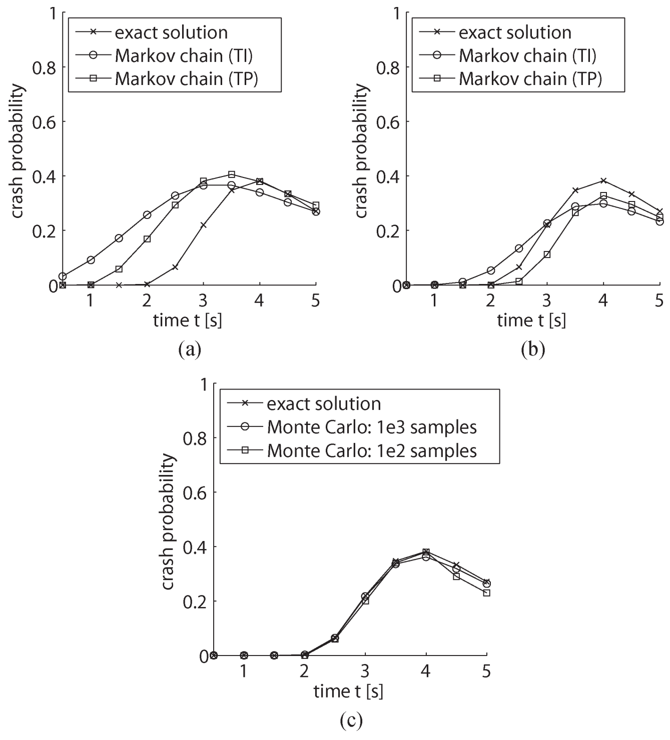

Fig. 13. Crash probabilities for different TPs. (a) Markov chain comparison (discretization A). (b) Markov chain comparison (discretization B). (c) Monte Carlo comparison.

TABLE V
COMPUTATIONAL TIMES OF THE CRASH SCENARIO

Markov chain				
	A (TP)	A (TI)	B (TP)	B (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

TIs. The crash probabilities p^C for different time steps/TIs are shown in Fig. 13. It can be observed that the fine discretization produces much better results than the coarse discretization.

In addition to different Markov chain models, Monte Carlo solutions were tested for varying number of samples [see Fig. 13(c)]. The results show that the crash probability is very accurate, even when only 10^3 or 10^2 samples are used. For this reason, it can clearly be 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 Table V, 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.

IX. CONCLUSION

The Markov chain and Monte Carlo approaches 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. Consequently, the Markov chain approach has only systematic errors from the discretization but no probabilistic errors since no random sampling is applied. Thus, the resulting probabilities are deterministically computed so that the results can be repeated.

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 probabilistic errors due to the sampling of the initial conditions and the input sequences. Due to the probabilistic errors, the resulting distributions and crash probabilities differ from execution to execution under unchanged initial conditions. 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 $1/\sqrt{N_s}$, where N_s is the number of samples.

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 ten times faster. Because there are many matrix multiplications in the Markov chain approach, it can significantly be accelerated by using dedicated hardware, such as 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.

The results can directly be implemented in an autonomous car. A screenshot of the probabilistic prediction in the test vehicle MUCCI [15] is shown in Fig. 14.

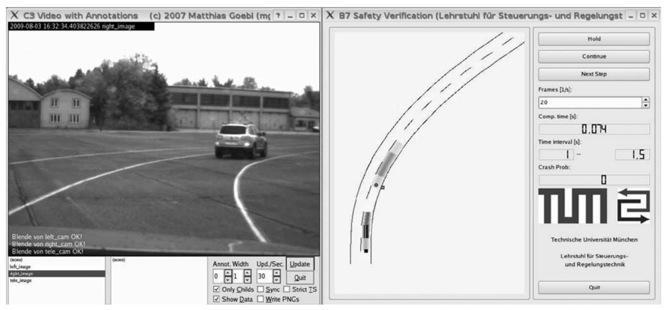
C3 Video with Annotations (c) 2007 Matthias Goebl
B7 Safety Verification (Lehrstuhl für Steuerungs- und Regelungstechnik)
Hold
Continue
Next Step
Frames [1/s]:
20
Comp. time [s]:
0.074
Time interval [s]:
1
1.5
Crash Prob.:
0
Technische Universität München
Lehrstuhl für Steuerungs- und Regelungstechnik
Quit
Annot. Width
Upd./Sec
Update
Only Childs
Sync
Strict TS
Show Data
Write PNGs

Fig. 14. Screenshot of a test drive. The probabilistic occupancy of the other car is computed via the Markov chain abstraction.

REFERENCES

[1] M. Abdel-Aty and A. Pande, "ATMS implementation system for identifying traffic conditions leading to potential crashes," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 78–91, Mar. 2006.

[2] M. Althoff, O. Stursberg, and M. Buss, "Model-based probabilistic collision detection in autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 299–310, Jun. 2009.

[3] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment of driving behavior in multi-lane traffic for autonomous vehicles," in *Proc. IEEE Intell. Vehicles Symp.*, 2009, pp. 893–900.

[4] K. Aso and T. Kindo, "Stochastic decision-making method for autonomous driving system that minimizes collision probability," in *Proc. FISITA World Automotive Congr.*, 2008.

[5] A. Barth and U. Franke, "Estimating the driving state of oncoming vehicles from a moving platform using stereo vision," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 560–571, Dec. 2009.

[6] H. A. P. Blom, J. Krystul, and G. J. Bakker, *Free Flight Collision Risk Estimation by Sequential Monte Carlo Simulation*. New York: Taylor & Francis, 2006, ch. 10, pp. 247–279.

[7] A. E. Broadhurst, S. Baker, and T. Kanade, "A prediction and planning framework for road safety analysis, obstacle avoidance and driver information," in *Proc. 11th World Congr. Intell. Transp. Syst.*, Oct. 2004.

[8] A. E. Broadhurst, S. Baker, and T. Kanade, "Monte Carlo road safety reasoning," in *Proc. IEEE Intell. Vehicles Symp.*, 2005, pp. 319–324.

[9] S. Danielsson, L. Petersson, and A. Eidehall, "Monte Carlo based threat assessment: Analysis and improvements," in *Proc. IEEE Intell. Vehicles Symp.*, 2007, pp. 233–238.

[10] M. S. Darms, P. E. Rybski, C. Baker, and C. Urmson, "Obstacle detection and tracking for the urban challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 475–485, Sep. 2009.

[11] P. P. Dey, S. Chandra, and S. Gangopadhaya, "Lateral distribution of mixed traffic on two-lane roads," *J. Transp. Eng.*, vol. 132, no. 7, pp. 597–600, Jul. 2006.

[12] D. Eberly, "Dynamic collision detection using oriented bounding boxes," Geometric Tools, Inc., Scottsdale, AZ, 2002.

[13] A. Eidehall and L. Petersson, "Statistical threat assessment for general road scenes using Monte Carlo sampling," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 1, pp. 137–147, Mar. 2008.

[14] M. Gabibulayev and B. Ravani, "A stochastic form of a human driver steering dynamics model," *J. Dyn. Syst., Meas., Control*, vol. 129, no. 3, pp. 322–336, May 2007.

[15] M. Goebl, M. Althoff, M. Buss, G. Färber, F. Hecker, B. Heißing, S. Kraus, R. Nagel, F. Puente León, F. Rattei, M. Russ, M. Schweitzer, M. Thuy, C. Wang, and H.-J. Wünsche, "Design and capabilities of the Munich cognitive automobile," in *Proc. IEEE Intell. Vehicles Symp.*, 2008, pp. 1101–1107.

[16] S. Gottschalk, M. C. Lin, and D. Manocha, "OBBTree: A hierarchical structure for rapid interference detection," *Comput. Graph.*, vol. 30, pp. 171–180, 1996.

[17] D. H. Greene, J. J. Liu, J. E. Reich, Y. Hirokawa, T. Mikami, H. Ito, and A. Shinagawa, "A computationally-efficient collision early warning system for vehicles, pedestrian and bicyclists," in *Proc. 15th World Congr. Intell. Transp. Syst.*, 2008.

[18] J. Hillenbrand, A. M. Spieker, and K. Kroschel, "A multilevel collision mitigation approach—Its situation assessment, decision making, and performance tradeoffs," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 528–540, Dec. 2006.

[19] J. Hu, M. Prandini, and S. Sastry, "Aircraft conflict detection in presence of a spatially correlated wind field," *IEEE Trans. Intell. Transp. Syst.*, vol. 6, no. 3, pp. 326–340, Sep. 2005.

[20] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, "A system for learning statistical motion patterns," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, Sep. 2006.

[21] T. Jürgensohn, "Control theory models of the driver," in *Modelling Driver Behaviour in Automotive Environments*, P. C. Cacciabue, Ed. New York: Springer-Verlag, 2007, pp. 277–292.

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

[23] X. Koutsoukos and D. Riley, "Computational methods for reachability analysis of stochastic hybrid systems," in *Hybrid Systems: Computation and Control*. New York: Springer-Verlag, 2006, pp. 377–391.

[24] A. Lambert, D. Gruyer, G. S. Pierre, and A. N. Ndjeng, "Collision probability assessment for speed control," in *Proc. 11th Int. IEEE Conf. Intell. Transp. Syst.*, 2008, pp. 1043–1048.

[25] K. Lee and H. Peng, "Evaluation of automotive forward collision warning and collision avoidance algorithms," *Vehicle Syst. Dyn.*, vol. 43, no. 10, pp. 735–751, Oct. 2005.

[26] C.-F. Lin, A. G. Ulsoy, and D. J. LeBlanc, "Vehicle dynamics and external disturbance estimation for vehicle path prediction," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 3, pp. 508–518, May 2000.

[27] J. Lunze and B. Nixdorf, "Representation of hybrid systems by means of stochastic automata," *Math. Comput. Model. Dyn. Syst.*, vol. 7, no. 4, pp. 383–422, Dec. 2001.

[28] J. Maroto, E. Delso, J. Félez, and J. M. Cabanellas, "Real-time traffic simulation with a microscopic model," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 513–527, Dec. 2006.

[29] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: Survey, system, and evaluation," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 20–37, Mar. 2006.

[30] B. T. Morris and M. M. Trivedi, "Learning, modeling, and classification of vehicle track patterns from live video," *IEEE Trans. Intell. Transp. Syst.*, vol. 9, no. 3, pp. 425–437, Sep. 2008.

[31] H. Ning, W. Xu, Y. Zhou, Y. Gong, and T. S. Huang, "A general framework to detect unsafe system states from multisensor data stream," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 1, pp. 4–15, Mar. 2010.

[32] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *IEEE Trans. Veh. Technol.*, vol. 59, no. 3, pp. 1172–1182, Mar. 2010.

[33] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 3, pp. 549–562, Sep. 2007.

[34] R. Rajamani, *Vehicle Dynamics and Control*. New York: Springer-Verlag, 2005.

[35] F. Rohrmüller, M. Althoff, D. Wollherr, and M. Buss, "Probabilistic mapping of dynamic obstacles using Markov chains for replanning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2008, pp. 2504–2510.

[36] J. Schröder, *Modelling, State Observation and Diagnosis of Quantised Systems*. New York: Springer-Verlag, 2003.

[37] S. Sekizawa, S. Inagaki, T. Suzuki, S. Hayakawa, N. Tsuchida, T. Tsuda, and H. Fujinami, "Modeling and recognition of driving behavior based on stochastic switched ARX model," *IEEE Trans. Intell. Transp. Syst.*, vol. 8, no. 4, pp. 593–606, Dec. 2007.

[38] T. Toledo, "Integrated driving behavior modeling," Ph.D. dissertation, Mass. Inst. Technol., Cambridge, MA, 2003.

[39] D. Vasquez, T. Fraichard, and C. Laugier, "Incremental learning of statistical motion patterns with growing hidden Markov models," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 403–416, Sep. 2009.

[40] A. L. Visintini, W. Glover, J. Lygeros, and J. Maciejowski, "Monte Carlo optimization for conflict resolution in air traffic control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, pp. 470–482, Dec. 2006.

[41] S. Weinzierl, "Introduction to Monte Carlo methods," NIKHEF Theory Group, Amsterdam, The Netherlands, 2000.

[42] Y. U. Yim and S.-Y. Oh, "Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction," *IEEE Trans. Veh. Technol.*, vol. 53, no. 4, pp. 1076–1084, Jul. 2004.

[43] R. Yuster and U. Zwick, "Fast sparse matrix multiplication," *ACM Trans. Algorithms*, vol. 1, no. 1, pp. 2–13, Jul. 2005.

Matthias Althoff received the Diploma Engineering degree in mechanical engineering and the Ph.D. degree in electrical engineering from Technische Universität München, München, Germany, in 2005 and 2010, respectively.

He is currently a Postdoctoral Researcher with the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA. His research interests include (probabilistic) reachability analysis of continuous and hybrid systems and safety analysis of autonomous cars.

Alexander Mergel received the Diploma Engineering degree in electrical engineering from Technische Universität München, München, Germany, in 2010.

He is currently with the Institute of Automatic Control Engineering (LSR), Technische Universität München. His research interests include Monte Carlo simulation, safety assessment of autonomous cars, and optimal control of hybrid systems.