Parallel Multi-Hypothesis Algorithm for Criticality Estimation in Traffic and Collision Avoidance

Eduardo Sánchez Morales[1], Richard Membarth[2], Andreas Gaull[1], Philipp Slusallek[2], Tobias Dirndorfer[3], Alexander Kammenhuber[3], Christoph Lauer[3], and Michael Botsch[1]

Abstract— **Due to the current developments towards autonomous driving and vehicle active safety, there is an increasing necessity for algorithms that are able to perform complex criticality predictions in real-time. Being able to process multi-object traffic scenarios aids the implementation of a variety of automotive applications such as driver assistance systems for collision prevention and mitigation as well as fall-back systems for autonomous vehicles.**

We present a fully model-based algorithm with a parallelizable architecture. The proposed algorithm can evaluate the criticality of complex, multi-modal (vehicles and pedestrians) traffic scenarios by simulating millions of trajectory combinations and detecting collisions between objects. The algorithm is able to estimate upcoming criticality at very early stages, demonstrating its potential for vehicle safety-systems and autonomous driving applications. An implementation on an embedded system in a test vehicle proves in a prototypical manner the compatibility of the algorithm with the hardware possibilities of modern cars. For a complex traffic scenario with 11 dynamic objects, more than 86 million pose combinations are evaluated in $21\,$ms on the GPU of a Drive PX 2.

I. Introduction and motivation

The number of vehicles on the road is constantly increasing. According to Sousanis [1], the billion units mark was passed in 2010, and, with the current growth rate, the tendency appears to remain the same for years to come. With an increasing number of vehicles on the road, there is a rising interest and necessity of analyzing and planning trajectories in complex multi-modal traffic scenarios. One goal for trajectory planning is the implementation of vehicle active safety systems for collision avoidance and mitigation [2]. Globally, there are more than 1.2 million traffic-accident related fatalities per year [3]. Another goal is to enable fall-back or self-supervising systems for autonomous vehicles.

In this paper, we define "complexity" as the combination of the following factors: an undetermined number of moving objects to be considered, a large and undefined number of available options or "trajectories" these objects can follow, and an extensive variety of roads where vehicles can drive on. Because of this complexity, the amount of information that has to be acquired, processed, and assessed for autonomous driving and vehicle active safety systems is accordingly huge.

[1]Eduardo Sánchez Morales, Michael Botsch, and Andreas Gaull are with the Department of Vehicle Safety, CARISSMA, Technische Hochschule Ingolstadt, Ingolstadt, Germany.

[2]Richard Membarth and Philipp Slusallek are with the German Research Center for Artificial Intelligence (DFKI) and Saarland University, Saarland Informatics Campus, Saarbrücken, Germany.

[3]Tobias Dirndorfer, Alexander Kammenhuber, and Christoph Lauer are with the AUDI AG, Development Automated Driving, Ingolstadt, Germany.

One possible solution are machine learning methods. A major advantage of these methods is the ability to analyze huge amounts of data in short time periods. However, they also pose new challenges: First, a comprehensive dataset is required in order to train machine learning models. The compilation of a diverse and complete enough database, that ensures that each and every open street scenario is covered, results in an impractical approach due to the time and resources this would cost. Current work in this area focuses only on a limited set of scenarios [4]. Second, data labeling can be problematic if a supervised machine learning method is chosen. Automated processes might not have the required classification accuracy for automotive safety systems and manual processes are very costly and time consuming. Third, most machine learning models are not interpretable. Rapidly evolving applications like cyber security profit greatly from methods like deep learning [5]. Yet, ethical and legal implications of false-positive actuations make machine learning methods inadequate for safety-critical applications. Research on algorithm validation for automated driving functions focuses on a limited set of scenarios, too [6].

Considering the previous points, we favor a model-based approach. Here, the biggest challenge is the processing of complex and costly computations, which results in long execution times. However, we will show that we can obtain execution times comparable to machine learning methods, but with deterministic outputs. Since all objects (vehicles and pedestrians) are mobile, no information from previous simulations can be reused: each traffic scenario has to be simulated every time from ground up. Even when the objects can be tracked over time, the trajectories that they are able to follow do change over time, altering the final outcome of the current traffic situation. Considering this and the runtime constraints for applications for vehicle safety systems, it is imperative to design the algorithm for parallel processing.

In this paper, we introduce an approach that deals with the task of traffic analysis for criticality estimation and motion planning in a fully model-based fashion. As a result, a large number of possible trajectories for all traffic participants have to be computed and assessed. For this work, we define "criticality" as the probability that the current traffic situation will lead to a collision for the own (EGO-) vehicle. The huge advantage of such an approach is that it can be easily tested, modified, and validated in simulation, which are key components for vehicle active safety algorithms. The challenge of this approach is the high computational costs related to predicting trajectories in multi-object scenarios.

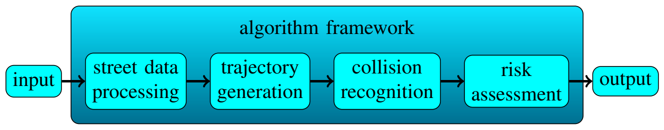

Fig. 1. Architecture of the algorithm and sub-process division.

However, we will show that such an approach can run—when executed in parallel—in 21 ms on embedded hardware that is available for modern cars.

This paper makes the following contributions:

1) We present an algorithm for criticality estimation of complex, multi-modal traffic scenarios, that is novel for combining (i) vehicle models, steering controllers, acceleration profiles, and mechanical latencies for realistic trajectory generation (Section IV-B), (ii) the possibility of accurate modeling the geometry of objects for an accurate collision recognition (Section IV-C), (iii) valid approximations for the conditional probabilities of collisions (Section IV-D) (iv) stochastic algorithm outputs and (v) all this while maintaining a highly parallelizable algorithm architecture. 2) We present an efficient mapping of the proposed algorithm to embedded systems utilizing data-parallel GPU processing and unified system memory (Section V). 3) We show that our algorithm is able to simulate over 2000 trajectories, to recognize collisions between objects and to estimate the criticality of traffic scenarios in 21 ms (Section VI). This requires to evaluate more than 86 million pose combinations of the EGO-vehicle and the other mobile objects (Collision Objects - COs).

II. Related Work

Trajectory planning and risk assessment are topics of continuous research in the automotive area. These overlap, but differ on their main task. The first one focuses on finding a route A to B that fulfills certain criteria that can include collision avoidance, but tends to neglect certain possible trajectories for sake of efficiency. The second one focuses on estimating the criticality of a traffic situation, for which the future states of the objects are *somehow* planned and/or predicted.

The work at hand is based on existing basic ideas like trajectory generation and collision recognition. It addresses key critical points of existing approaches that a) have been avoided by other authors because of the complexity or computational cost, b) were not considered combined until now and c) impact noticeably the final criticality estimation. The algorithm also differentiates itself by its flexible, highly parallelizable architecture, which is a base requirement for GPU-parallel mapping. These critical points are addressed following by comparing the present work with related publications.

Broadhurst et al. [7] present a method for *reasoning* how vehicles move, when in presence of other traffic participants. The authors neglect several relevant aspects of the trajectory generation, such as mechanical latencies and limits. A purely geometrical model is used for describing vehicle motion, which generates non-driveable trajectories. This and the use of constant steering instead of a steering controller shift greatly the scenario criticality as shown in Figure 7. The use of random components also complicates the algorithm validation for use in vehicle safety functions.

Broadhurst et al. [7] is a very adequate example of why it is necessary to approximate conditional probabilities as we will shown in Section IV-D.2. As stated by Broadhurst et al. in Section II-F, the collision probability is calculated sequentially, which is time consuming. Their algorithm applied to a single object operates at 2 Hz, while the algorithm presented in this work processes $30,869$ trajectory combinations of 4 objects (EGO + 3 CO) in the same time period.

Lefèvre et al. [8] present a comprehensive survey on existing criticality estimation methods. They perform semantic classification and show trade-offs of the different classes. The results of this paper indicate that by using a smart, parallelizable algorithm structure, it is possible to obtain criticality estimations with a quality similar to that of *interaction-aware* motion models (Section IV-D), but using *maneuver-based* models (Section IV-B), which opens the door for real-time risk assessment.

Ziegler et al. [9] and Ferguson et al. [10] present real-world tested motion planning algorithms. Both rely on representing the motion planning as a constrained optimization task, where the trajectories are represented by cost functions that include a “no collision” constraint. The trajectory given by the optimized cost function is driven, even if it is the *only* collision-free trajectory. This implies that errors in the sensor information or actuator control, unexpected CO behavior, modeling inaccuracies or wrong assumptions could lead to collisions. So, in highly dense traffic situations, unnecessary risks might be taken since only a small subset of the physically feasible trajectories are considered. That a trajectory is *unlikely* to happen, does not mean that it *cannot* happen.

Note that the goal of the present algorithm is *not* to completely substitute current path planning or collision recognition approaches. Redundancy is a best practice where humans can be harmed due to system malfunction. As stated in Section I, the present algorithm can serve as a fall-back, supervising or plausibility system for both autonomous and human-driven vehicles.

III. Multi-Hypotheses Approach

For estimating the criticality of a traffic scenario, an algorithm that can model interactions between multiple objects present in an area of interest is necessary. Not knowing exactly which trajectory will be followed by each of these objects generates uncertainty about the future. However, this uncertainty can be modeling by generating multiple hypotheses for each object. This addresses the possible motion options of the objects. Considering the previous, this algorithm was designed as a fully model-based multi-hypothesis algorithm.

A model-based approach has key benefits for passive and active vehicle safety systems. First, it enables the future integration of infrastructural elements (such as more lanes) and a larger number of static and dynamic objects in the computation. This could improve the precision, reliability and relevance of the obtained results. Second, this approach is based on domain knowledge and widely accepted mathematical models. This ensures deterministic outputs, allowing an easy validation of the algorithm. Finally, the model-based generation of statistical quantities for prediction tasks is important for automotive safety systems.

We define a “hypothesis” as the combination of a specific acceleration profile and a path that correlate over time; that is, one unique trajectory. A main advantage of the multi-hypothesis strategy is that the number of threads that are executed in parallel is easy to influence, as it depends on easily adjustable parameters such as the number of hypotheses to be simulated or the amount of objects to be considered. With this in mind, it is important to note that the more hypotheses that are simulated, the better coverage of all possible states that an object might traverse in the near future. Therefore, the maximum possible accuracy of the algorithm can be scaled depending on available parallel computing resources.

The algorithm is composed by four modules which have to be executed in a given order, and inside these modules, there are tasks that can be executed in parallel. The modules and the computational structure of the algorithm can be seen in Figure 1.

To manage the parallel execution of the algorithm, a set of matrices were structured to keep the information traceable and independent, and nested indexes had to be generated to allow the information to be retrieved. These are module-specific and it is precisely these indexes that express the parallelization possibilities of the corresponding tasks. This will be explained in the corresponding modules.

IV. Algorithm Process

In order to detect collisions between two objects, their pose (coordinates of center of gravity and orientation) over time (trajectory) and their shape has to be known. This means that each and every prediction has to go through all four modules, which will be explained in the following.

A. *Street Data Processing*

One of the constraints for the trajectory generation is the road infrastructure, meaning that the vehicles are bound to it under conventional traffic circumstances (e. g. not going off-road). Because of this, the trajectories will be generated according to the lanes present on the driveable road. Considering that the focus of this work is not the street modeling, the assumption is made that the lane information is delivered from the sensors to the algorithm in the form of sets of three pairs of (x, y) coordinates. Using exteroceptive sensors in vehicles, this representation is realized in modern cars. Each one of these sets corresponds to a specific lane divider, and each lane is delimited by exactly two lane dividers. Adjacent lanes share one lane divider.

From each set, it is assumed that two coordinates correspond to the closest and farthest detected points of the lane divider, and the third one is any point in between. From

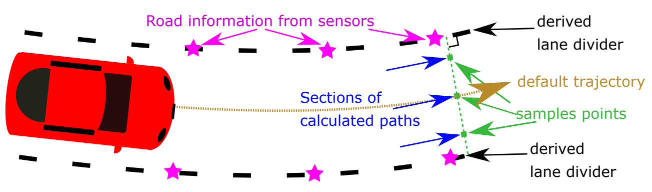

Fig. 2. Graphical depiction of the trajectory generation and lane divider derivation.

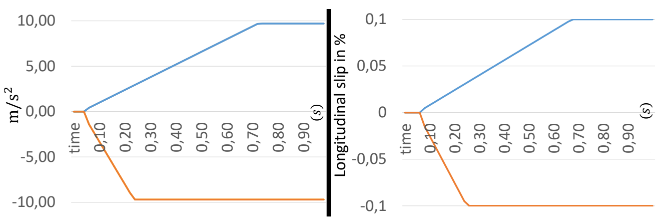

Fig. 3. Examples of acceleration profiles for the one-track (OT) model (left) and longitudinal slip profiles for the two-track (TT) model (right).

these three points and using the Gauss-Jordan method [11], a second degree equation is obtained, which corresponds to the mathematical representation of the associated lane divider. A graphical explanation of this can be seen in Figure 2.

The algorithm is designed to consider up to three lanes: the own (EGO-), immediate left, and immediate right lanes. If the sensors do not deliver information for neighbouring lanes, they will not be considered, and no trajectory will be generated in the corresponding area. If the information of the EGO-lane is missing, a “virtual” lane will be generated according to the current vehicle state and applicable legislation [12].

Having the mathematical representation of the driveable road, the vehicles are associated with their corresponding lanes. The EGO-vehicle will always be placed on the EGO-lane. All other vehicles are associated with the lanes according to the location of their estimated center of gravity (more on vehicle parameters later). Pedestrians are not bound to the road infrastructure, so no association is performed for them.

For this module, up to 4 threads can be computed in parallel, one for each lane divider.

B. *Trajectory Generation*

Having the mathematical representation of the road (Section IV-A), many hypotheses for each object are computed by means of motion models as described in the following.

1) Motion Models for Vehicles: Under tractive driving (i. e. not exceeding the grip limits of the tires), longitudinal and lateral dynamics of the vehicles are coupled. For this, motion models are used for generating the trajectories of the vehicles.

a) The trajectories of the EGO-vehicle are computed using the TT model [13], since all relevant information is known. Its mathematical representation is given by

$$\dot{\boldsymbol{x}} = \begin{bmatrix}\dot{v} & \dot{\beta} & \ddot{\psi}\end{bmatrix}^T = \boldsymbol{f}\left(v, \beta, \dot{\psi}, \boldsymbol{F}_{lij}, \boldsymbol{F}_{sij}\right),$$

where β is the sideslip angle, ψ is the yaw angle in global frame, v is the current velocity over ground, $\boldsymbol{F}$ represents the tire forces, l and s indicate longitudinal and side forces, and indices i and j denote the front, rear, left, and right tire.

b) For all other vehicles, a OT model [14] is chosen since, up to now, it is not possible to acquire all the information required for the TT model implementation in real-time. The used OT model is given by

$$\begin{bmatrix}\dot{\beta}\\ \ddot{\psi}\end{bmatrix}=\begin{bmatrix}-\frac{c_{\alpha f}+c_{\alpha r}}{mv} & \frac{c_{\alpha r}l_r-c_{\alpha f}l_f}{mv^2}-1\\ \frac{c_{\alpha r}l_r-c_{\alpha f}l_f}{I_z} & -\frac{c_{\alpha f}l_f^2+c_{\alpha r}l_r^2}{I_z v}\end{bmatrix}\begin{bmatrix}\beta\\ \dot{\psi}\end{bmatrix}+\begin{bmatrix}\frac{c_{\alpha f}}{mv}\\ \frac{c_{\alpha f}l_f}{I_z}\end{bmatrix}\delta_f,$$

where $c_{\alpha f}$ and $c_{\alpha r}$ are the front and rear tire cornering stiffness, l_f and l_r indicate the distance from the center of gravity of the vehicle to the front and rear axles, m and I_z are the mass and moment of inertia of the vehicle, and δ_f is the front steering angle at the tires.

For both motion models, once having the corresponding state variables, the pose of the vehicle is calculated by means of Euler integration as described by the following equations:

$$\begin{bmatrix}v_{V,n+1}\\ \beta_{n+1}\end{bmatrix}=\begin{bmatrix}v_{V,n}\\ \beta_n\end{bmatrix}+\begin{bmatrix}\dot{v}_{V,n}\\ \dot{\beta}_n\end{bmatrix}\tau,$$

$$\psi_{V,n+1}=\psi_{V,n}+\dot{\psi}_{V,n}\tau+\ddot{\psi}_{V,n}\frac{\tau^2}{2},$$

$$\begin{bmatrix}a_{x,V,n+1}\\ a_{y,V,n+1}\end{bmatrix}=\begin{bmatrix}c(\beta_n)\\ s(\beta_n)\end{bmatrix}\dot{v}_{V,n}+\begin{bmatrix}-s(\beta_n)\\ c(\beta_n)\end{bmatrix}v_{V,n}\left(\dot{\beta}_n+\dot{\psi}_{V,n}\right),$$

$$\begin{bmatrix}x_{V,n+1}\\ y_{V,n+1}\end{bmatrix}=\begin{bmatrix}x_{V,n}\\ y_{V,n}\end{bmatrix}+\begin{bmatrix}c(\psi_{V,n}+\beta_n)\\ s(\psi_{V,n}+\beta_n)\end{bmatrix}v_{V,n}\tau +\begin{bmatrix}c(\psi_{V,n})\\ s(\psi_{V,n})\end{bmatrix}a_{x,V,n}\frac{\tau^2}{2}+\begin{bmatrix}-s(\psi_{V,n})\\ c(\psi_{V,n})\end{bmatrix}a_{y,V,n}\frac{\tau^2}{2},$$

where s and c are the sine and cosine functions, subscript V denotes a vehicle, τ is the time lapse between instances $n+1$ and n, $[x_V, y_V]^T$ are the coordinates in global frame, $a_{x,V}$ and $a_{y,V}$ are the accelerations in vehicle coordinate frame. The input $\dot{v}_V$ will be addressed in Section IV-B.2.

The mass of the vehicles plays an important role in traffic accidents [15] and it is a parameter for the OT model. Because of this, the COs are classified according to their dimensions to assign them a mass value. For this, it is assumed that the on-board sensors of the EGO-vehicle can deliver approximate information about the dimensions of the COs. If the height of the COs cannot be estimated, 6 classes are used: quadricycle, supermini, small family car, large family car, executive, and multi-purpose vehicle. Otherwise, 2 extra classes are used: off-roader and cargo. This accounts for the different length/width-to-weight ratios that taller vehicles have, when compared to the other classes. The classes are chosen according to Van Miert [16] and average values are obtained from Heydinger et al. [17]. Typical values for tire parameters and vehicle geometry are obtained from Isermann [18].

2) Variation for the Vehicle Trajectory Generation: The generation of multiple hypotheses requires to influence both longitudinal and lateral dynamics of the vehicles.

a) For the longitudinal motion, either the proper acceleration or longitudinal slip is sampled. Two samples are the maximum and minimum of the corresponding range, and one is zero (vehicle is cruising). The remaining samples are equally distributed in the negative region to favor a reduction of kinetic energy of the bodies. This is very robust in critical situations. The positive area is also sampled to address cases where accelerating would avoid an accident, like a rear-end collision. The ranges go from $-9.7\frac{\mathrm{m}}{\mathrm{s}^2}$ to $9.7\frac{\mathrm{m}}{\mathrm{s}^2}$ (OT) and the longitudinal slip from -0.1 to 0.1 (TT). Furthermore, mechanical latencies and jerks are introduced to generate profiles that are present in road vehicles. This aids to maintain realistic trajectories. The number of profiles are equal for the EGO-vehicle and COs. The output of the profiles is $\dot{v}_V$, which is an input for the corresponding motion model. Examples of these profiles can be seen in Figure 3.

b) To get feasible and realistic trajectories, a controller is designed and implemented for lateral dynamics. First, a reference trajectory is generated with a motion model and the expected driver behavior. That is, the vehicles keep driving with their current acceleration towards the middle of their current lane. Along this trajectory, the available lanes are sampled at three predefined instances of the prediction time (1.0s, 1.5s and 2.0s). The sample points are equally distributed on each lane and perpendicular to the lane divider. Three samples are made on the own lane and two samples are made on the neighboring lanes. This reflects the expected behavior of the vehicles: it is more likely that they will stay on their current lane, rather than steer towards the neighboring lanes. Out of these sample points, the path sections are generated. These sections go over the sample points and are parallel to the lane dividers (see Figure 2). Three of these path sections (one from each sampling instance) represent one complete path. The path sections are the input for the controller and the output is the steering angle at the front wheels δ_f. This is then used as input for the correspondent motion model. It was decided to generate more complete paths for the EGO-vehicle than for the COs. This takes into account that one is able to influence the EGO-vehicle, but not the COs.

The lateral controller has been optimized using a large number of simulations and is expressed mathematically by

$$\delta_f=\left(\left|(-0.018\cdot v_V+1.5)\,d_{\text{path}}\right|+0.5\right)d_{\text{path}} +\left(-\left|(-0.018\cdot v_V+1.5)\,d_{\text{path}}\right|+9.5\right)(3.8197\cdot e_\psi),$$

where d_{path} is the closest distance between the predicted center of gravity of the vehicle and the path section (distance error), and e_ψ is the difference between ψ_V and the direction of the section of the calculated path (yaw error). So, the controller is realized based on a predicted error and two P-controllers. A graphical description of the functioning principle of the lateral controller can be seen can be seen in Figure 4.

To optimize the performance of the controller, its design also includes features such as dynamically weighting d_{path} and e_ψ, and dynamically adjusting the prediction time for the position of the vehicle according to v_V. A maximum steering angle and maximum steering rate are also included for considering the mechanical limitations of the road vehicles and driver capabilities [19].

3) Motion Model for Pedestrians: Research on pedestrian motion modeling has been done by Schlake [20]. Factors

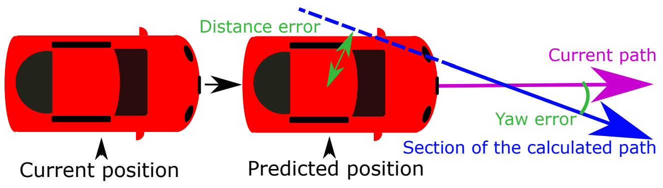

Fig. 4. Principle of the lateral dynamics controller.

like emergency situations and pedestrian density do affect the motion of pedestrians, but these motion models focus on in-building situations. In open-air environments, the movement of pedestrians is not bound to the road infrastructure. Because of this, the pedestrian motion is expressed by a kinematic model with no controller:

$$\begin{bmatrix} x_{P,n+1} \\ y_{P,n+1} \end{bmatrix} = \begin{bmatrix} x_{P,n} \\ y_{P,n} \end{bmatrix} + \begin{bmatrix} c(\psi_P) \\ s(\psi_P) \end{bmatrix} v_{P,n}\tau + \begin{bmatrix} c(\psi_P) \\ s(\psi_P) \end{bmatrix} \dot{v}_{P,n}\frac{\tau^2}{2},$$

where subscript P denotes a pedestrian.

For pedestrians, the samples are equally distributed from 0° to 360° for ψ_P and from $-12\frac{\text{m}}{\text{s}^2}$ to $12\frac{\text{m}}{\text{s}^2}$ for $\dot{v}_P$. The velocity v_P is limited to $2.7\frac{\text{m}}{\text{s}}$ assuming that pedestrian velocities on open roads range from slow to running, rather than sprinting [21]. The number of samples of ψ_P is equal to the number of complete paths for COs and the number of samples of $\dot{v}_P$ is equal to the number of profiles for vehicles.

4) Parallelization of Trajectory Generation: A key for computing so much information in parallel is to generate the trajectories so that they are completely independent from each other. In Section IV-B, the generation of trajectories by combining acceleration profiles and paths is explained. It is precisely this combination that allows the generation of a large number of independent trajectories.

The total number of trajectories r_{tra} that can be simulated in parallel is equal to

$$r_{\text{tra}} = (o \cdot h_{\text{acc}} \cdot h_{\text{CO,str}}) + (h_{\text{acc}} \cdot h_{\text{EGO,str}})\,,$$

where o is the total number of COs considered, h_{acc} is the number of acceleration profiles, $h_{\text{CO,str}}$ is the number of complete paths for the COs, and $h_{\text{EGO,str}}$ is the number of complete paths for the EGO-vehicle.

In this paper, the number of trajectories for the EGO-vehicle is equal to 2058. This results from the linear combination of the path sections ($h_{\text{EGO,str}} = 7^3$) which are also combined with 6 acceleration profiles. As stated in Section IV-B.2, one is not able to influence the COs. For this reason, no combination of the path sections is made for COs. This means that each CO has 7 complete paths, which are combined with 6 acceleration profiles to generate 42 trajectories. For example, 2478 trajectory generations can be executed in parallel when $o = 10$.

It should be noted that the used motion models are represented by differential equations that are solved numerically for each time instance of each trajectory (Section IV-B.1). This means that the current position has to be known in order to calculate the future position of an object. Following this train of thought, one limitation of parallelizing is that it is not possible to simulate all time instances of one single trajectory in parallel—they have to be calculated sequentially.

C. Collision Recognition

1) Object Modelling with Polygons: Once the trajectories are generated, it is possible to check if a collision between the EGO-vehicle and a CO occurs. For this, the complete set of hypotheses of each CO is combined with each and every hypothesis of the EGO-vehicle. Both objects are then modeled as polygons, and it is checked at each time instance whether they overlap or not. If the polygons overlap, this indicates that a collision occurs at the given time instance. Well known point in polygon strategies exist to test for this [22], [23]. The method requires to check the overlapping twice per time instance: EGO over CO, and CO over EGO. The polygons can be as complex as necessary. The more complex they are, the better the objects are described, but the more runtime is required. This is very relevant for the computational resources, since the collision recognition is the module that takes the most runtime and the overlapping check is the function that is called most frequently.

Combinations of hypotheses between COs are not taken into account, seeing that it is not the focus of this work to predict collisions that do not involve the EGO-vehicle.

2) Parallelization of Collision Recognition: As mentioned in Section IV-B.4, all trajectories are completely independent from each other. Thus, any combination resulting from them is independent as well to be simulated and evaluated by different processing units in parallel. The resulting number of trajectory combinations r_{col} that could be simulated in parallel is given by

$$r_{\text{col}} = o \cdot h_{\text{CO,str}} \cdot h_{\text{EGO,str}} \cdot h_{\text{acc}}^2.$$

As an example, for a case with $o = 10$, $h_{\text{acc}} = 6$, $h_{\text{EGO,str}} = 343$ and $h_{\text{CO,str}} = 7$, $864,360$ trajectory combinations are executed in parallel in this module in 21 ms.

In this work, a maximum prediction time of 2 s and a discretization time of 20 ms is used for any given scenario. This means that the overlapping check function will be called 200 times for each trajectory combination, yielding a total number of $172,872,000$ calls per scenario for this routine alone that are executed in 21 ms for this example.

The length and width used for cars in the class with the smallest vehicles is 3.13 m and 1.46 m, respectively. Assuming the two specific collision cases of a T-bone and a purely longitudinal one, the vehicles need a relative velocity of $229\frac{\text{m}}{\text{s}}$ and $313\frac{\text{m}}{\text{s}}$ accordingly for them to drive "through" each other and the collision to be undetected. It is known to the authors that the collision configurations are infinite, and that special cases, such as a small overlap, do happen. However, this shows that the discretization time covers a very comprehensive range of street scenarios.

D. Risk Assessment

The risk assessment is based on the anticipation time and criticality. We define “anticipation time” as the time in advance that the criticality of a situation can be recognized.

For this, once it is determined that a combination of trajectories leads to a collision, the probabilities of these trajectories are multiplied, obtaining the probability that this combination will occur. Assuming statistical independence (Section IV-D.2), the probability p_{cra} that a traffic scenario will lead to a collision is given by

$$p_{\text{cra}} = \sum_{i=1}^{r_{\text{EGO}}} \sum_{j=1}^{r_{\text{CO}}} I(i,j) \cdot p_{\text{EGO},i} \cdot p_{\text{CO},j},$$

where $I(i,j)$ is the indicator function that yields one if and only if the i-th EGO and the j-th CO trajectory lead to a collision and zero otherwise, $r_{\text{EGO}} = h_{\text{acc}} \cdot h_{\text{EGO,str}}$, $r_{\text{CO}} = o \cdot h_{\text{acc}} \cdot h_{\text{CO,str}}$, $p_{\text{EGO},i}$ is the probability of the i-th EGO trajectory, and $p_{\text{CO},j}$ is the probability of the j-th CO trajectory.

1) Trajectory Probability Calculation: To get the probability that a specific hypothesis can occur, it is scored against the reference trajectory mentioned in Section IV-B.2. The lower the score, the lower the occurrence probability. The scoring value n_{h} is given as follows:

$$n_{\text{h}} = \frac{[w_{\text{acc}} \cdot n_{\text{acc}}] + [w_{\text{str}} \cdot d_{\text{str}}]}{c_{\text{com}} \cdot c_{\text{cou}}},$$

where c_{com} and c_{cou} are penalizing factors for the complexity of the maneuver and for entering lanes with counter traffic; n_{acc} and d_{str} are the scoring factors for the acceleration profile and the path, and w_{acc} and w_{str} are fixed weighting factors for the acceleration and steering correspondingly. The scoring and penalizing factors steer the occurrence probability of the hypotheses, thus the criticality of the traffic scenario as well. Should colliding trajectories have a higher occurrence probability, the criticality of the scenario will be higher too. One way of optimizing the parameters in practice, is according to the expected passenger injury for each type of collision. The used parameters are chosen using domain knowledge.

Once all trajectories of an object are scored, the scores are normalized with respect to the L^1-norm. The resulting values are considered as occurrence probabilities of the trajectories. This process is repeated for each object.

2) Conditional Probability: In Section IV-D.1, an occurrence probability for each hypothesis of each object is calculated. Having multiple COs that can collide with EGO at different time instances, the conditional probability of these collisions has to be considered. The implementation of conditional probabilities could give a marginal benefit for representing the behavior of the criticality, but the required mathematics prevent the algorithm from being parallelizable.

To maintain the algorithm parallelizable, the conditional probability is approximated. For this, it is assumed that an EGO-CO hypothesis combination can occur if and only if no other collision occurred before along the corresponding EGO-trajectory. Thus, the EGO-CO combinations are considered to be independent, and their probabilities are scaled down in chronological order. That is, the first hypothesis combination to occur in time maintains its estimated probabilities ("collision" and "no collision"). Then, the probability that the next hypothesis combination occurs, is equal to the "no collision" probability of the first combination. Both "collision" and "no collision" probabilities are then scaled down accordingly. This process continues until all the probabilities of all the combinations are scaled.

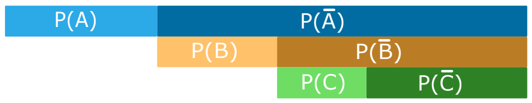

Fig. 5. Graphical exemplification of the implemented approximation of conditional probability.

The following equations explain this further and a graphical representation of this can be seen in Figure 5

$$P\left(B \cap \bar{A}\right) \approx P\left(\bar{A}\right) \cdot P\left(B\right)$$
$$P\left(C \cap \bar{B} \cap \bar{A}\right) \approx P\left(\bar{A}\right) \cdot P\left(\bar{B}\right) \cdot P\left(C\right).$$

V. Algorithm Mapping to Embedded Systems

The proposed algorithm is designed such that it can be executed in parallel by thousands of threads. While each module in Figure 1 depends on the results of its predecessor, the modules themselves are highly parallelizable. We target embedded systems that offer embedded, on-board GPUs as execution platform. For automotive applications, there is a variety of hardware platforms offered by different manufacturers such as Renesas (R-Car), NXP (i.MX), Texas Instruments (OMAP Jacinto), Qualcomm (Snapdragon), Intel (GO), or NVIDIA (Jetson).

For real-time performance, it is essential to map the algorithm efficiently to the available hardware resources and to make best use of them. This requires typically low-level programming and ties the implementation to one specific architecture. To avoid this, we use the AnyDSL[1] framework that allows to separate low-level hardware-specific aspects from the high-level algorithm description [24], [25]. AnyDSL supports code generation for GPUs by generating CUDA and OpenCL. The algorithm description for CPU and GPU are the same, only a hardware-specific mapping is required for each target platform:

- Iteration Logic: Defines in which order data is processed in each module. This can be sequential on the CPU vs. data-parallel on the GPU.
- Hardware Intrinsics: Many trigonometrical functions such as sine or cosine can be mapped to much faster hardware-accelerated versions on the GPU.
- Memory Hierarchy: Modern CPUs and GPUs have a deep memory hierarchy. Some of them require explicit programming.
- Memory Management: CPU and GPU share the same physical memory in most embedded systems.

In our implementation, we exploit all those hardware-specific features mapping all modules to the GPU. In particular, the resulting implementation executes all modules in a data-parallel fashion, makes extensive use of hardware

[1]`https://anydsl.github.io`

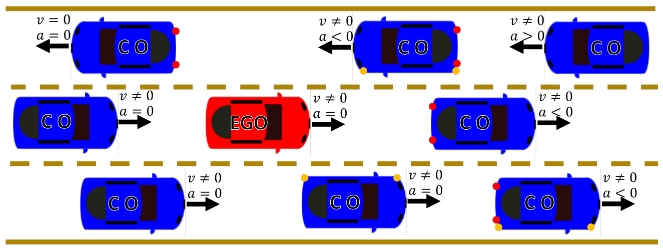

Fig. 6. Example of a complex traffic scenario that can be handled by the presented algorithm.

intrinsics for the collision recognition (sine, cosine, tangent), and requires no data-transfers between CPU and GPU, exploiting unified CPU/GPU memory. For deployment, we use Thrift[2] to retrieve input data from the vehicle and return the results of to our algorithm.

VI. Evaluation and Results

A. Algorithm Outputs

The evaluation is performed first by designing a set of 20 different simulated scenarios that cover in a wide, general manner possible traffic situations. These scenarios include combinations of one, two, and three lanes, COs as static and moving objects, as well as counter traffic and vehicles approaching from behind. The Figure Figure 6 shows an example of a complex traffic scenario that can be handled by the proposed algorithm. The second part of the evaluation of the algorithm includes 547 real-life traffic situations. For this, a vehicle is driven on open roads and the information obtained from the sensors is collected and used as an input for the algorithm for off-line evaluation.

An evaluation metric of the algorithm stability is the development of the predicted criticality over time. A smooth, progressive development indicates the absence of misjudged collisions (false-positives). It includes in a compact and understandable manner the most relevant information about the traffic scenario for passive and active vehicle safety systems. This because features like false-positives and triggering thresholds are recognized easily.

The obtained results indicate that the algorithm is capable of detecting unavoidable collisions with an anticipation time that depends on the scenario that is being evaluated. When the traffic is purely longitudinal, this anticipation time is long enough to influence the vehicle dynamics. When cross-traffic is present, this anticipation time is long enough to activate passive safety systems. For the designed scenarios where static objects were placed in different arrangements in front of the EGO-vehicle, the average anticipation time was 932 ms; and for the scenarios where the COs were in motion, it ranged from 660 ms to 1800 ms. These results contrast with the 300 ms that occur in a simple scenario without road modeling. The large anticipation times could aid a better triggering of multistage airbags, thus preventing passenger injuries [26]. This is specially interesting for lateral

[2]`https://thrift.apache.org`

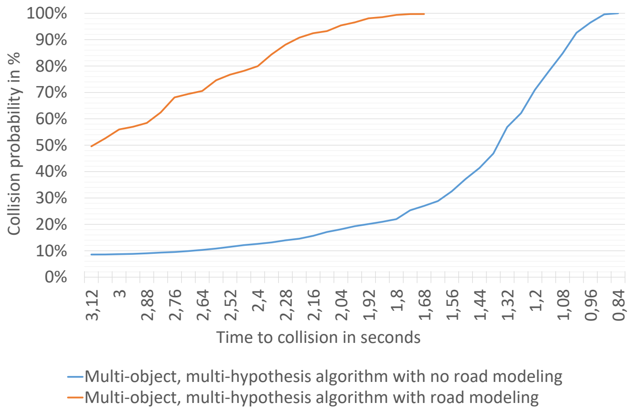

Fig. 7. Development over time of the predicted collision probability of a single test scenario when evaluated by two algorithms.

airbags [27]. Slow actuators could benefit from this too, since the extra time helps to compensate mechanical latencies.

One of the pillars of our algorithm is the combination of road modeling, vehicle motion models and the lateral dynamics controller. This ensures that the generated trajectories are not only drivable by vehicles, but that they are meaningful as well. Figure 7 shows a good comparison of this. For evaluation purposes, the controller module is disabled. This provokes that the trajectory generation does not take into consideration the drivable road, and that some of the trajectories tagged as "collision-free" actually go outside the available road infrastructure.

An extremely important result is that there are zero false-positive outputs for the evaluated scenarios. This demonstrates that the algorithm possesses a high degree of reliability and robustness. This is specially relevant when deciding triggering points of passive and active vehicle safety systems.

An additional benefit of the algorithm is the output of possible escape routes, that is, trajectories that could aid to avoid an oncoming collision. This derives from two factors. First, as stated in Section IV-B, all the trajectories are generated by the use of motion models combined with the aid of realistic acceleration profiles and a controller for lateral dynamics. This means that all generated and simulated trajectories can be driven by a vehicle. Second, as described in Section IV-D, the complete probability spectrum of the EGO-trajectories is known, so the most adequate one can be chosen. As stated in Section III, it is important to note that the resolution of the algorithm depends highly on the amount of acceleration profiles and paths (hypotheses).

B. Execution Time

For evaluation, we use the Drive PX 2 development board from NVIDIA. The Drive PX 2 has a CPU with four ARM Cortex A57 cores and two NVIDIA Denver cores as well as a Tegra X2 GPU (GP10B) with 256 cores and a dedicated GPU (GP106) with 1152 cores, both based on the Pascal architecture. While the Tegra X2 GPU shares the main memory of the CPU, the dedicated GPU has its own memory.

We consider two scenarios: The first scenario (S1) considers 3 COs in addition to the EGO-vehicle while the second scenario (S2) considers 10 COs in addition to the EGO-vehicle. We use 6 acceleration profiles, which are combined with 7 and 343 paths for CO and EGO-vehicle, respectively. For each scenario, we compute the trajectories for the next 2 seconds with a resolution of 20 ms, which equals to 100 time steps. This results in 8.64 million pose combinations that need to be evaluated per CO. In total, this results in 25.93 million pose combinations for S1 and 86.44 million pose combinations for S2. Using only 5 acceleration profiles reduces the number of pose combinations to 18.00 million for S1 and 60.01 million for S2, respectively. The execution time for scenario S1 and S2 is shown in Table I. On the dedicated (GP106) / embedded (GP10B) GPU, it takes 10/15 ms to evaluate the proposed algorithm for S1 and 21/49 ms for S2. Considering only 5 acceleration profiles reduces the execution time to 8 /11 ms for S1 and 14/35 ms for S2. More than two thirds of the total execution time is spent for collision recognition. The GPU execution is more than two magnitudes faster than CPU execution, benefiting from hardware-accelerated trigonometrical functions. The communication overhead of the client/server architecture of Thrift adds 0.5 ms on top of the algorithm execution time.

TABLE I

PERFORMANCE RESULTS FOR THE PROPOSED ALGORITHM. SHOWN IS THE NUMBER OF POSE COMBINATIONS EVALUATED AS WELL AS THE MEDIAN EXECUTION TIME IN MS (LOWER IS BETTER) ON THE CPU, DEDICATED GPU (GP106), AND EMBEDDED GPU (GP10B).

Scenario	# pose combinations	CPU	GPU GP106	GPU GP10B
S1 (3 CO)	25.93 million	1800 ms	10 ms	15 ms
	18.00 million	1600 ms	8 ms	11 ms
S2 (10 CO)	86.44 million	11600 ms	21 ms	49 ms
	60.01 million	9600 ms	14 ms	35 ms

VII. CONCLUSIONS

In this work, a fully model-based multi-modal parallelizable algorithm is presented. This algorithm is able to estimate upcoming criticality of complex traffic scenarios at very early stages. The architecture of the algorithm allows the further inclusion of road infrastructure and mobile objects. This architecture also allows the algorithm to be ported to different GPUs. The implementation on vehicle-compatible hardware proves in a prototypical manner its feasibility to function in production vehicles. Short execution times, deterministic results, and the absence of false-positives, prove the adequacy of the algorithm for passive and active vehicle safety systems.

REFERENCES

[1] J. Sousanis, "World vehicle population tops 1 billion units," *Wards Auto*, vol. 15, 2011.

[2] M. Müller, P. Nadarajan, M. Botsch, W. Utschick, D. Böhmländer, and S. Katzenbogen, "A statistical learning approach for estimating the reliability of crash severity predictions," in *19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 2199–2206.

[3] World Health Organization, *Global Status Report on Road Safety 2015*. World Health Organization, 2015.

[4] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[5] "From machine learning to artificial intelligence," https://www.mcafee.com/enterprise/en-us/solutions/machine-learning.html, accessed: 2019-01-29.

[6] F. Reway, W. Huber, and E. P. Ribeiro, "Test methodology for vision-based adas algorithms with an automotive camera-in-the-loop," in *2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, Sept. 2018, pp. 1–7.

[7] A. Broadhurst, S. Baker, and T. Kanade, "Monte carlo road safety reasoning," in *2005 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, June 2005, pp. 319–324.

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

[9] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—a local, continuous method," in *2014 IEEE Intelligent Vehicles Symposium Proceedings (IV)*. IEEE, June 2014, pp. 450–457.

[10] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, no. 11-12, pp. 939–960, 2008.

[11] R. Ansorge and H. J. Oberle, *Mathematik für Ingenieure: Band 1: Lineare Algebra und analytische Geometrie, Differential- und Integralrechnung einer Variablen*. Wiley-VCH Verlag GmbH & Co, 2000.

[12] FGSV, *Richtlinien für die Anlage von Autobahnen (RAA)*. Forschungsgesellschaft für Straßen- und Verkehrswesen e.V., 2008.

[13] M. Mitschke and H. Wallentowitz, *Dynamik der Kraftfahrzeuge*. Springer, 1972, vol. 4.

[14] D. Schramm, M. Hiller, and R. Bardini, *Vehicle Dynamics: Modeling and Simulation*. Springer, 2014.

[15] Insurance Institute for Highway Safety, "Special issue: Car size, weight and safety," *Status Report*, vol. 44, no. 4, 2009.

[16] K. Van Miert, *Case No COMP/M.1406-HYUNDAI/KIA*. Office for Official Publications of the European Communities, 1999.

[17] G. Heydinger, R. Bixel, W. R. Garrott, M. Pyne, J. G. Howe, and D. A. Guenther, *Measured Vehicle Inertial Parameters-NHTSAs Data Through November 1998*. Society of Automotive Engineers, 1999.

[18] R. Isermann, *Fahrdynamik-Regelung: Modellbildung, Fahrerassistenzsysteme, Mechatronik*. Vieweg & Sohn Verlag, 2006.

[19] G. J. Forkenbrock and D. Elsasser, *An Assessment of Human Driver Steering Capability*. National Highway Traffic Safety Administration, 2005.

[20] B. A. Schlake, "Mathematical models for pedestrian motion," Diplomarbeit, Westfälische Wilhelms-Universität Münster, 2008.

[21] J. Zębala, P. Ciępka, and A. RezA, "Pedestrian acceleration and speeds," *Problems of Forensic Sciences*, vol. 91, pp. 227–234, 2012.

[22] M. Shimrat, "Algorithm 112: Position of point relative to polygon," *Communications of the ACM*, vol. 5, no. 8, p. 434, 1962.

[23] E. Haines, "Point in polygon strategies," *Graphics Gems IV*, vol. 994, pp. 24–26, 1994.

[24] R. Leißa, K. Boesche, S. Hack, A. Pérard-Gayot, R. Membarth, P. Slusallek, A. Müller, and B. Schmidt, "AnyDSL: A partial evaluation framework for programming high-performance libraries," *Proceedings of the ACM on Programming Languages (PACMPL)*, vol. 2, no. OOPSLA, pp. 119:1–119:30, Nov. 2018.

[25] R. Leißa, K. Boesche, S. Hack, R. Membarth, and P. Slusallek, "Shallow embedding of DSLs via online partial evaluation," in *Proceedings of the International Conference on Generative Programming: Concepts & Experiences (GPCE)*. ACM, 2015, pp. 11–20.

[26] Insurance Institute for Highway Safety, "Evidence mounts that reducing force of airbag inflation lowers risk," *Status Report*, vol. 39, no. 7, 2004.

[27] Insurance Institute for Highway Safety, "Head-protecting side airbags reduce driver fatality risk by 45 percent," *Status Report*, vol. 38, no. 8, 2003.