FRAMEWORK FOR MOTION PREDICTION OF VEHICLES IN A SIMULATION ENVIRONMENT

Juan Felipe Medina Lee
Centre for Automation and Robotics (CAR), CSIC-UPM, juan.medina@csic.es

Vinicius Trentin
Centre for Automation and Robotics (CAR), CSIC-UPM, vinicius.trentin@csic.es

Jorge Villagra
Centre for Automation and Robotics (CAR), CSIC-UPM, jorge.villagra@csic.es

Abstract

Efficient testing and validation of software components for highly automate vehicles is one of the key challenges to be solved for their massive deployment. The number of driving situation and environment variables makes validation almost intractable with real vehicles in open roads, and the testing reproducibility can only be achieved via simulation. This manuscript presents a framework and preliminary results for motion prediction of vehicles in a simulation environment that is being currently developed by the AUTOPIA Program.

Keywords: Driving corridors, Simulation environment, Autonomous vehicle, LCM, Motion Prediction, Probabilistic Reachable Sets.

1 INTRODUCTION

Autonomous vehicles need a lot of hours of testing before going into the market. For example, Google self driving cars drive 10.000 - 15.000 miles a week to collect data for testing their software [19]. These requirements are very difficult to meet with the legal restrictions of autonomous vehicles in public roads [21]. A robust simulation framework would allow to test the control algorithms of autonomous vehicles safely and to improve wrong behaviors when special conditions are met. In [7], the authors propose a simulation tool of cooperative maneuvers among autonomous vehicles in which virtual and real vehicles can conjunctively interact. Another work found in literature use real data collected from autonomous vehicles for creating new simulation scenarios and improve driving behaviors [10].

One of the main expectations regarding autonomous vehicles is to reduce the number of accidents. In order to to that, it is necessary to predict the motion of the surrounding vehicles and people for a safe navigation of the ego-vehicle. To do this verification, the set of states reachable within a finite or infinite time interval needs to be computed.

Reachable sets are the union of all possible states a system can reach when starting within a bounded set of initial states and subject to a set of possible input and parameters values [3].

Except for a few cases, the exact computation of the reachable sets is impossible to perform. For this reason, the computation must be done using abstraction methods or computing an approximation of the reachable sets of the system [11].

Reachable sets have been used for motion prediction, in a deterministic way, in [5, 13]. The former used abstractions to model the reachable states of other vehicles and to verify if the planned trajectory of the ego vehicle is safe. The later used Hamilton-Jacobi reachability analysis for the determination of prediction sets for human driven vehicles in a lane changing scenario.

The probabilistic reachability analysis in the same context appeared, for example, in [2] where the authors used Markov chains to predict the future behavior of traffic participants, considering their dynamics, their interactions with one another and also the limitation of driving maneuvers due to the road geometry resulting in crash probabilities for the possible paths of the ego vehicle.

In this work, we propose a way to compute the probabilistic reachable sets for motion prediction using probabilistic zonotopes and a linear system model. We also propose a simulation framework for testing and validation, which bring us the possibility of creating new scenarios that would be very difficult to obtain with an autonomous vehicle on the street or with pre-recorded logs on a human-driven vehicle.

The remainder of the paper will be as follows:

Section 2 describes the modules of the developed framework, including: the simulation environment, map creation, the design of a testing scenario for validation purposes and the navigable lanes generation. Section 3 presents a review of the probabilistic zonotopes, the system model and how the reachable sets are generated. Section 4 presents the results for the developed framework and section 5 concludes.

2 FRAMEWORK

2.1 SIMULATION ENVIRONMENT

This chapter describes the software architecture implemented to facilitate the interaction between the high-level software for autonomous vehicles and SCANeR Studio simulator [20].

2.1.1 Vehicle software architecture

The autonomous vehicles from the AUTOPIA Program [6] use a multi-process architecture for handling all the input/output modules, by sharing messages using Lightweight Communications and Marshalling (LCM) API [15]. This API provides a set of libraries and tools for inter-process communication in real-time systems, which is based on a publish-subscribe message passing model using UDP multicast as underlying transport layer [14].

Fig. 1 shows all modules used in the instrumented vehicles from the AUTOPIA Program, interconnected with the LCM API [9].

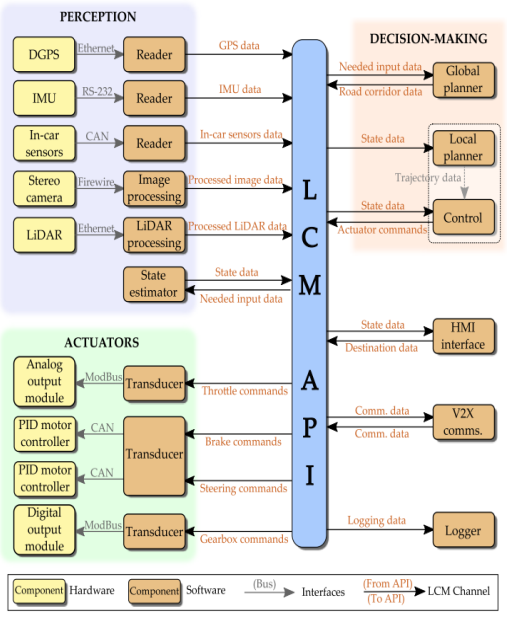


Figure 1: Architecture of autonomous vehicles of AUTOPIA Program

2.1.2 Simulator description

SCANeR studio is a software suite for automotive simulation, addressing both testing and driving for autonomous vehicles, human machine interactions or driving assistance systems. This tool provides all the necessary modules to build a realistic virtual world: road environment, vehicle dynamics, traffic, sensors, real or virtual drivers, headlights, weather conditions and scenario scripting [20].

This simulator provides an API which allows users to customize their interaction with it, by developing their own specific modules that may access and modify data from a running simulation. This API also offers the possibility of controlling vehicles and pedestrians, modify the scenario's conditions or access sensory data. In order to handle all this information, SCANeR Studio uses a network protocol that allows the user to be informed of state changes in the simulation and to send messages to specific modules to control them. One of the advantages of this methodology is that the user can develop his own modules in any language that can load the provided dynamic libraries, which will trigger the transmission and reception of SCANeR Studio Ethernet network messages.

2.1.3 Interaction with the simulator

A software module was implemented to handle the simulated vehicle as if it were the real one, so we could use the same decision-making software to control both. This was possible through the implementation of two independent software processes: the first for handling the perception and the second for handling the actuators of the simulated vehicle. This architecture is shown in Fig. 2.

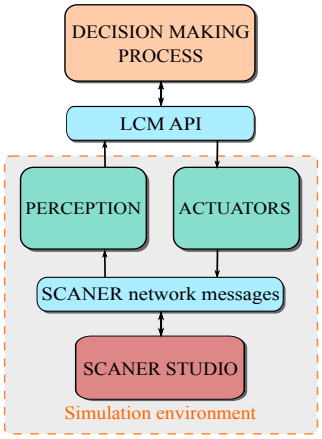


Figure 2: Software architecture to interact with the simulator

The perception process receives messages from

SCANeR Studio and transmits them through LCM periodically, with the same frequency as the modules in the real vehicle. This module transmits information about the vehicle dynamics (speed and acceleration), the steering wheel state, the GPS location and the bounding boxes of the near obstacles. Fig. 3 displays the perception data from the vehicle in the middle of a simulation.

Figure 3: Reading information from simulated vehicle

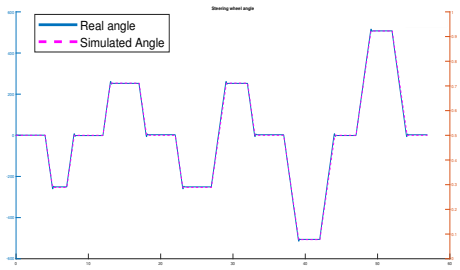
The actuator process, on the other hand, receives messages from the LCM API and sends them to SCANeR Studio network in order to control the vehicle. This module also emulates the behavior of the actuators implemented in the real vehicle for moving the steering wheel and the breaking pedal. The throttle pedal does not have this problem because it is an electric signal in the real vehicle. Fig. 4 shows the comparison of the real and simulated actuators in different scenarios.

Fig. 4a shows the steering wheel angle for different control inputs with the car standing still. The continuous blue line is the behavior of the real vehicle, and the dotted magenta line is the behavior of the simulated car. Fig. 4b shows the longitudinal speed (blue) and acceleration (orange) for a 30% throttle pedal during 100m and then stopping with 0.3 breaking command. Continuous lines are for the real vehicle and dotted lines are for simulated vehicle.

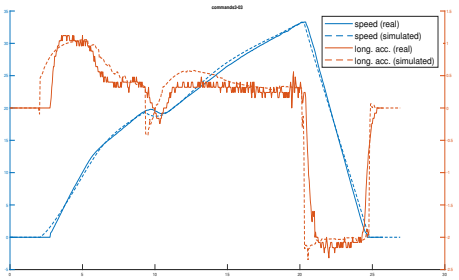
2.2 MAP CREATION

Maps have become a very important component of autonomous driving systems [18]. High definition maps provide a lot of information about the surrounding environment of the vehicle, helping in some vital processes such global route calculation and local path planning. They also contain data about regions that cannot be observed by the sensors, allowing to compensate inadequacies in the sensory data and to have a more reliable behavior in autonomous vehicles.

We choose to use the *Lanelet2* framework [18] for map creation. It is an open-source map framework implemented in C++ which has different advan-



(a)



(b)

Figure 4: Comparison of the steering wheel angles for real and simulated vehicle (a) and comparison of longitudinal behavior of real and simulated vehicle (b)

tages such as interactions between lanes and regions, information on different areas, implications of traffic rules and software modularity. At the moment, AUTOPIA Program uses Open Streets Map data for generating driving corridors automatically [12]. This technique expands the original map representation, replacing polylines by polynomial-based roads, whose sections are defined using cubic Bezier curves. This approach was adopted after performing an exhaustive comparison with different curve primitives [8]. The representation of the map using Bezier curves provides a better fit of the road shape, but it is only implemented for single-lane roads. *Lanelet2* framework eases the handling of multi-lane roads, as seen in section 2.4.

Fig. 5 shows an example of a simple map represented using lanelets standard. The left side of the figure presents the *physical layer* of the map, which contains observable information such as geographic location, border-lines of the road, or the center-lines of the lanes. The right side of the figure shows the *relational layer* of the map in which the elements of the physical layer are connected and converted to lanes, roads or areas.

The combination of both, physical and relational layers, will be used for generating the navigable corridors for the autonomous vehicle.

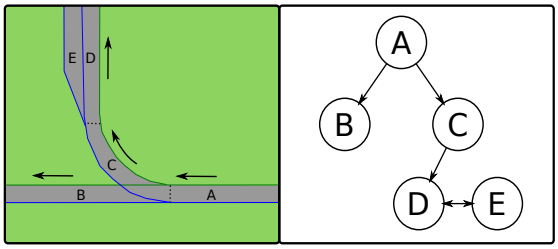
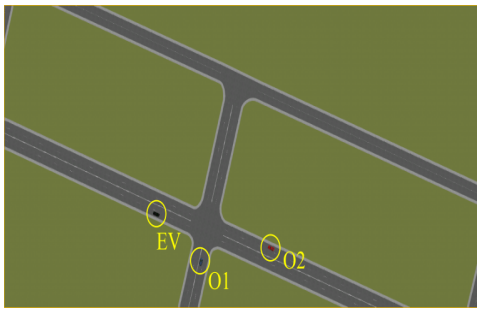


Figure 5: Map representation of a simple road bifurcation using lanelets

2.3 SCENARIO DESIGN FOR TESTING

The scenario for validating our framework consists in a four double-lane intersection located in Arganda del Rey, Spain. We placed 3 vehicles in the scene, all approaching to the main intersection from different directions. This scenario is shown in Fig. 6.

(a)



(b)

Figure 6: Urban scenario representation using lanelets (a) and SCANeR Studio testing Scenario (b)

Fig. 6a shows the scenario map created using lanelets and Fig. 6b shows the same scenario created in SCANeR Studio. Both environments have the same Geo-location. As seen in Fig. 6b, the vehicle coming from the left is the ego vehicle; and the vehicles coming from the bottom and from the left are the dynamic obstacles. Table I shows the initial state configured in the simulator for each vehicle in the scene.

Table 1: Initial states of vehicles in the scenario

ID	Dist. to intersection	Init. speed
EV	30m	N/A
$O1$	50m	50km/h
$O2$	15m	20km/h

The proposed experiment consists in obtain the possible navigable corridors for each obstacle, compute the reachable sets for a 3 second time-horizon and compare the final position of the simulated vehicles with the prediction using reachable sets. Each simulation is going to be run 3 times with different accelerations: $-0.5\ m/s^2$, $0\ m/s^2$ *and* $1\ m/s^2$.

Next sections will be based on the scenario described in this chapter.

2.4 DYNAMIC GENERATION OF NAVIGABLE CORRIDORS

There are different techniques for representing the navigable space of an autonomous vehicle. In [16], the authors compare some of them. Autonomous vehicles from the AUTOPIA Program use navigable corridors to represent the near surroundings and to calculate the optimum path to follow [6].

We propose to use the relational and physical layer of the lanelets map in order to obtain all the navigable corridors for the vehicles in the scene. The length of these corridors is equal to the maximum distance that the car can reach with its current speed in a specified time, assuming a constant maximum acceleration.

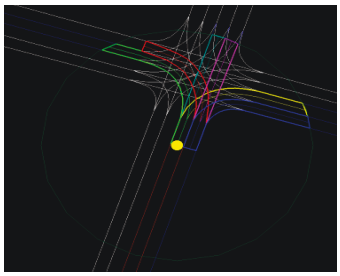
First, we obtain the current lanelet(s) where the vehicle is located, comparing the position and the orientation in the physical layer. Next, we perform a graph search for surrounding lanelets starting from the vehicle lanelet(s) to create a lanelet-sequence for each corridor. Each lanelet-sequence must comply three conditions:

- It cannot be longer than the search horizon;
- It cannot contain any loops;
- It cannot intersect the vehicle-lanelet(s);

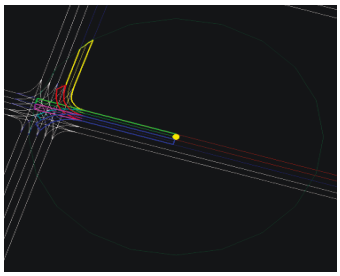
Fig. 7a and 7b show the navigable corridors for the obstacles using a 3.0 s horizon and a maximum acceleration of 3 m/s^2. In the case of the obstacle 1, the length of the corridors is 30.16m. For obstacle 2, the length of the corridors is 55.16m.

We implemented a C++ application that computes the navigable corridors. It obtains the state

of the vehicles (location, orientation, and speed) through LCM messages, and uses *Lanelet2* framework for generating the corridors, their centerlines and their boundaries in less than 40 ms.



(a)



(b)

Figure 7: Navigable corridors for bottom Obstacle (a) and Navigable corridors for right Obstacle (b)

3 REACHABLE SETS AND SYSTEM MODEL

In this section we present the methodology that we implemented to generate the reachable sets.

3.1 PROBABILISTIC ZONOTOPES

In this work, we use probabilistic zonotopes for the motion prediction. Probabilistic zonotopes were proposed in [4] and they can be defined as a mixed list of generators of a zonotope and a Gaussian zonotope [3].

Zonotopes are a special case of convex polytopes and can be interpreted as the Minkowski sum of finite line segments [11]. A zonotope Z is a set

$$Z = \{x \in \mathbb{R}^n | x = c + \sum_{i=1}^{p} \beta_i g_i, -1 \leq \beta_i \geq 1\}$$

with c being the center and $g_1 ... \ g_p$ the generators of the zonotope.

Gaussian zonotopes (G-zonotopes with certain mean) are zonotopes that have the intervals $\beta_i \in [-1, 1]$ replaced by independent Gaussian distributed random variables $\mathbf{N}_i(0, 1)$ [3], and can be characterized by:

$$\mathbf{Z} = c + \sum_{i=1}^{p} \mathbf{N}_i(0, 1) g_i$$

Probabilistic zonotopes (G-zonotopes with uncertain mean) are defined as a G-zonotope $\mathbf{Z}$, where the center is uncertain an can have any value within a zonotope Z [4]:

$$\mathscr{Z} = Z \boxplus \mathbf{Z}$$

The operator $\boxplus$ combines the Gaussian zonotope $\mathbf{Z}$ with the zonotope Z. The resulting probabilistic zonotope $\mathscr{Z}$ is neither a set nor a random vector, hence there does not exist a probability density function describing $\mathscr{Z}$ [4]. However, $\mathscr{Z}$ can be represented by an enclosing probabilistic hull (Definition 4.3 from [3]).

3.2 SYSTEM MOTION MODEL

The linear system model used in this paper is defined by the stochastic differential equation (1) [3].

$$\begin{gathered} \dot{\mathbf{x}} = A\mathbf{x}(t) + Bu(t) + C\xi \\ \mathbf{x}(0) \in \mathbb{R}^n, \ u(t) \in U \subset \mathbb{R}^n, \ \xi \in \mathbb{R}^m \end{gathered} \tag{1}$$

This model includes a stochastic input ξ, represented by a white Gaussian noise that is mixed with the uncertain input u.

The state variable $\mathbf{x}$ is a probabilistic zonotope that contains the position (s_x, s_y) and the velocity v_x. It is represented by enclosing hulls which include all possible probability density functions for the time interval [4].

The matrices A, B and C for the simulated model are described below.

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \ B = \begin{bmatrix} 1 \end{bmatrix}, \ C = \begin{bmatrix} 0.2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0.2 \end{bmatrix}$$

The input $u(t)$ of the system is the acceleration and it is contained in the interval below

$$u(t) \in \begin{bmatrix} 0, \ 0 \\ 0, \ 0 \\ -0.5, \ 3.0 \end{bmatrix}$$

which means that a minimum and maximum acceleration of -0.5 and 3.0 m/s^2, respectively, can be applied to the vehicles.

https://doi.org/10.17979/spudc.9788497497169.520

3.3 GENERATION OF THE REACHABLE SETS

For every simulated obstacle, we find all the possible navigable corridors it may follow and for each of these corridors, a road-based grid is created. The reachable sets are then calculated based on the model from (1) using a combination of the libraries *SPOT* [17] and *CORA* [1], both implemented in Matlab. Fig. 8a shows the enclosing probabilistic hull (EPH) for one time interval. This probabilistic hull is then projected into a grid (Fig 8b) that is applied to every road-based grid previously created (Fig. 9a).

This reshaping is necessary because the current model does not take into account the paths the vehicle can follow, considering motion only in the x-axis.

Besides the model from (1), the *Acceleration-Based Occupancy* from [5], already implemented in *SPOT*, is intersected with the solution of each time interval. This abstraction considers that an over-approximated reachability for a time interval can be described by a polygon which includes the circles, with center and radius calculated by (2), for two consecutive time intervals. The polygon and the resulting reachable set for a time interval can be seem in Fig 9b.

$$c(t) = \begin{bmatrix} s_x(0) \\ s_y(0) \end{bmatrix} + \begin{bmatrix} v_x(0) \\ v_y(0) \end{bmatrix} t,\ r(t) = \frac{1}{2} a_{max} t^2 \quad (2)$$

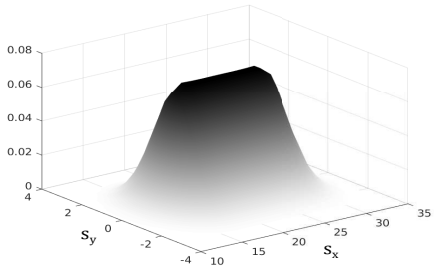
4 RESULTS

The model from Section 3 was applied to the vehicles described in Table 1 in the testing scenario from Section 2.3.

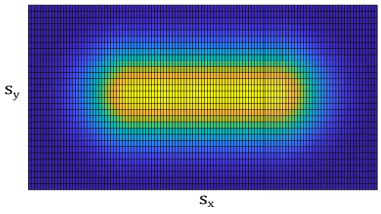
The resulting reachable sets for the 2.9-3.0 s time interval are shown in Fig. 10 for both obstacles.

In order to have a way to validate the results, some simulations were made in SCANeR Studio. The dots in Fig. 10 represent the results from these simulations varying the final velocity and the path the vehicles could take. The different colors represent the possible lanes from Fig. 7.

It can be seem in Fig. 10b that one of the red dots is outside of the reachable sets computed for the time interval. The reason for this is that the simulated vehicle was not able to perform the turn for the dynamic parameters of the scenario. Fig. 11 shows the acceleration profile for this specific simulation. For the other points, Table 2 presents the average probability of the simulated car landing in one of the most probable cells at 3 seconds.



(a)



(b)

Figure 8: Enclosing Probabilistic Hull (EPH) for a time interval (a) and its projection on a 2D grid (b)

Table 2: Average probability of landing in the most probable cells

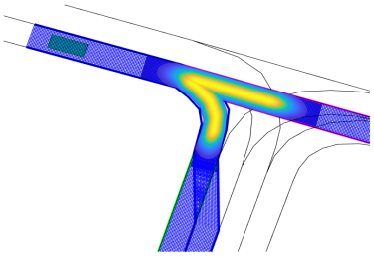
Obstacle	Average probability
Obstacle 1	88.44% ± 17.41
Obstacle 2	89.84% ± 17.50

The computed reachable sets from the obstacles are necessary for a safe navigation of the ego-vehicle, so it is necessary to create a grid from its the point of view that includes all this information. Instead of being a road-based grid, it is a vehicle-based grid, which does not take into account the lanes and includes all the reachable sets from the other vehicles. The resulting interpolation for the 2.9-3.0 s time interval is displayed in Fig. 12.

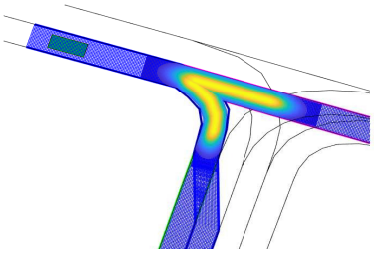
5 CONCLUSION AND FUTURE WORK

The developed framework allowed the communication between a control software and the SCANeR Studio simulator through the LCM middleware. It uses a lanelet-based map structure for analyzing information in relational and physical layers, in order to generate navigable corridors for the autonomous vehicle.

This work also served to introduce the frame-

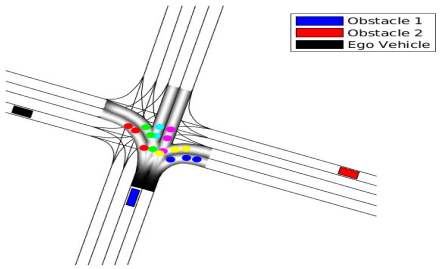


(a)

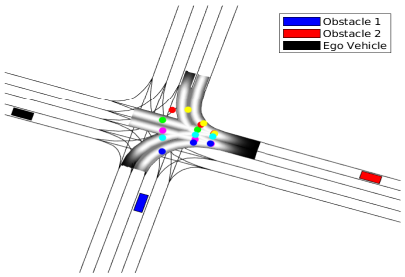


(b)

Figure 9: Reshape of the EPH projection from Fig. 8b into the possible lanes (a) and the resultion reachable sets with the intersection with the polygon (b)



(a)



(b)

Figure 10: Probabilistic reachable sets for the obstacle 1 (a) and obstacle 2 (b) in the time interval 2.9 - 3 s

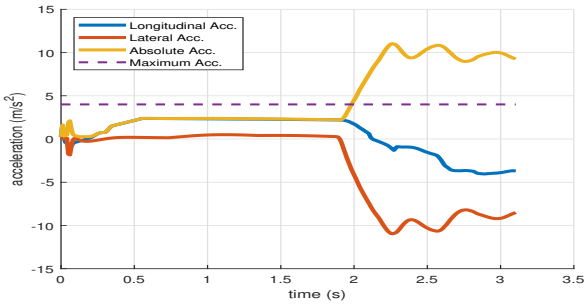


Figure 11: Acceleration profile

work being used by the AUTOPIA Program for the implementation of the probabilistic reachability analysis for the motion prediction of vehicles. Using simulations, we could validate the results obtained so far, proving the system is correct.

In the future, the simple linear motion model will be changed for a more complex hybrid one, where we can take into account velocity constraints and also consider the interaction between different vehicles/obstacles.

Acknowledgement

This work has been partially funded by the Spanish Ministry of Science, Innovation and Universities with National Project COGDRIVE (DPI2017-86915-C3-1-R), the Community of Madrid through SEGVAUTO 4.0-CM (S2018-EMT-4362) Programme, and by the European Commission through the Projects PRYSTINE (ECSEL-783190-2) and SECREDAS (ECSEL-783119-2).

References

[1] M. Althoff. "An introduction to CORA 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, pp. 120-151, 2015.

[2] M. Althoff, O. Stursberg and M. Buss, "Model-Based Probabilistic Collision Detection in Autonomous Driving." In: *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 2, pp. 299–310, 2009.

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

[4] M. Althoff, O. Stursberg, and M. Buss, "Safety assessment for stochastic linear sys-

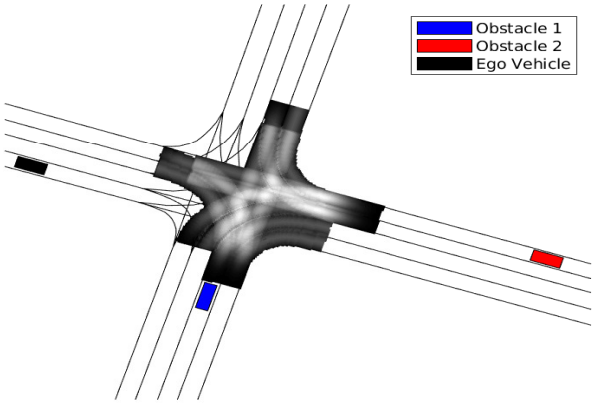


Figure 12: Occupancy grid for the ego vehicle in the time interval 2.9 - 3 s

tems using enclosing hulls of probability density functions," In: *Proc. of the European Control Conference*, pp. 625-630, 2009.

[5] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," In: *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187-202, 2016.

[6] A. Artunedo, J. Godoy, and J. Villagra, "A decision-making architecture for automated driving without detailed prior maps," In: *2019 IEEE Intell. Veh. Symp.*, 2019.

[7] A. Artunedo, J. Godoy, R. Haber, J. Villagra, and R. M. D. Toro, "Advanced Co-simulation Framework for Cooperative Maneuvers among Vehicles, "In: *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2015-October, pp. 1436-1441, 2015.

[8] A. Artunedo, J. Godoy, and J. Villagra, "A Primitive Comparison for Traffic-Free Path Planning," IEEE Access, vol. 6, pp. 28801-28817, 2018.

[9] A. Artunedo, "Decision-Making Strategies for Automated Driving in Urban Environments, Ph.D. Thesis, Universidad Politécnica de Madrid, Madrid, 2019.

[10] M. Da Lio et al., "Exploiting dream-like simulation mechanisms to develop safer agents for automated driving: The 'Dreams4Cars' EU research and innovation action,"In: *IEEE Conf. Intell. Transp. Syst. Proceedings, ITSC*, vol. 2018-March, pp. 1-6, 2018.

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

[12] J. Godoy, A. Artuñedo, and J. Villagra, "Self-Generated OSM-Based Driving Corridors," *IEEE Access*, vol. 7, pp. 20113–20125, 2019.

[13] V. Govindarajan and R. Bajcsy, "Human Modeling for Autonomous Vehicles: Reachability Analysis, Online Learning, and Driver Monitoring for Behavior Prediction," Master Thesis, EECS Department, University of California, Berkeley, 2017.

[14] A. Huang, E. Olson, and D. Moore. "LCM: Lightweight Communications and Mar- shalling." In: *International Conference on Intelligent Robots and Systems*, pp. 4057–4062, 2010.

[15] A. S. Huang, E. Olson, and D. Moore. "Lightweight Communications and Marshalling for Low Latency Interprocess Communication." In: *Tech. rep. Massachusetts Institute of Technology*, 2009.

[16] C. Katrakazas, M. Quddus, W. H. Chen, and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," In: *Transp. Res. Part C Emerg. Technol.*, vol. 60, pp. 416–442, 2015.

[17] M. Koschi and M. Althoff, "SPOT: A tool for set-based prediction of traffic participants," In: *Proc. of the IEEE Intelligent Vehicles Symposium*, pp. 1679-1686, 2017.

[18] F. Poggenhans et al., "Lanelet2: A high-definition map framework for the future of automated driving." In: *IEEE Conf. Intell. Transp. Syst. Proceedings*, ITSC, vol. 2018-November, pp. 1672–1679, 2018.

[19] "Google software testing" [Online]. Available: http://www.theverge.com/2016/2/1/10892020/google-self-driving-simulator-3-million-miles

[20] "Scaner Studio" [Online]. Available: https://www.avsimulation.fr/solutions/#studio

[21] "The state of autonomous legislation in Europe." [Online]. Available: https://autovistagroup.com/news-and-insights/state-autonomous-legislation-europe