to appear in the Proceedings of the 29th IEEE Intelligent Vehicles Symposium, 2018

CoInCar-Sim: An Open-Source Simulation Framework for Cooperatively Interacting Automobiles

Maximilian Naumann[1], Fabian Poggenhans[1], Martin Lauer[2] and Christoph Stiller[1,2]

***Abstract*— While motion planning techniques for automated vehicles in a reactive and anticipatory manner have already been widely presented, cooperative motion planning has only been addressed recently. For the latter, interaction between traffic participants is crucial. Consequently, simulations where other traffic participants follow simple behavioral rules can no longer be used for development and evaluation. To close this gap, we present a multi vehicle simulation framework. Conventional simulation agents, using a simple, rule-based behavior, are replaced by multiple instances of sophisticated behavior generation algorithms. Thus, development, test and simulative evaluation of cooperative planning approaches is facilitated. The framework is implemented using the Robot Operating System (ROS) and its code will be released open source.**

***Keywords*— Simulation, Cooperative Automated Vehicles, Motion Planning, Behavior Generation, Trajectory Planning.**

I. Introduction

In the last decades, tremendous progress has been made in the field of advanced driver assistance systems [1]. Current projects of research facilities and industry are aiming at SAE level 4 or 5, i.e. full automation [2]. Consequently, the algorithms must be able to perceive, understand and act in various, complex environments.

In the development of perception algorithms for automated driving, datasets such as KITTI [3] and Cityscapes [4] play a key role. They do not only contribute by allowing comparisons through several benchmarks, but also by providing preprocessed input data and desired output, so called ground truth. Thus, researchers can focus on the development of new algorithms rather than spending their time for cumbersome collection and labeling of experimental data.

In the field of motion planning, datasets are less well suited. While static or quasi-static scenarios can be provided as input, the desired output or optimal solution is highly dependent on the goal. The latter can be objective, such as time-optimality, or subjective, such as passenger comfort. Consequently, a ground truth for application-oriented goals is hard to determine.

Considering interactions, as necessary in cooperative motion planning, even the comprehensive definition of scenarios is not possible anymore, as the behavior of other traffic participants largely depends on the action of the ego vehicle.

[1]FZI Research Center for Information Technology, Mobile Perception Systems, 76131 Karlsruhe, Germany {naumann,poggenhans}@fzi.de

[2]Karlsruhe Institute of Technology (KIT), Institute of Measurement and Control Systems, 76131 Karlsruhe, Germany {martin.lauer,stiller}@kit.edu

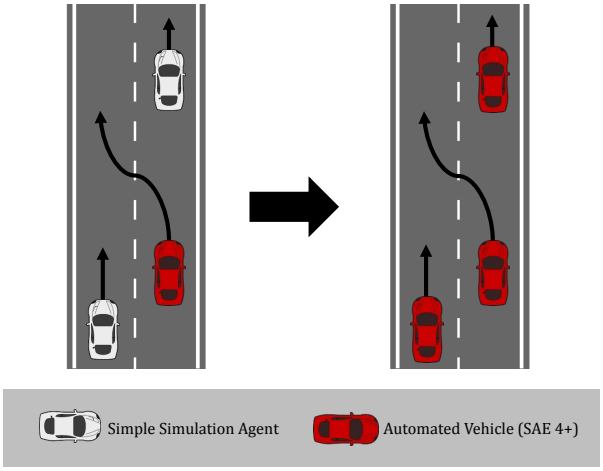


Fig. 1: **CoInCar-Sim**: In contrast to classical vehicle-in-the-loop approaches (*left*), CoInCar-Sim (*right*) facilitates the simultaneous simulation of several instances of sophisticated behavior generation algorithms instead of using simple driver models.

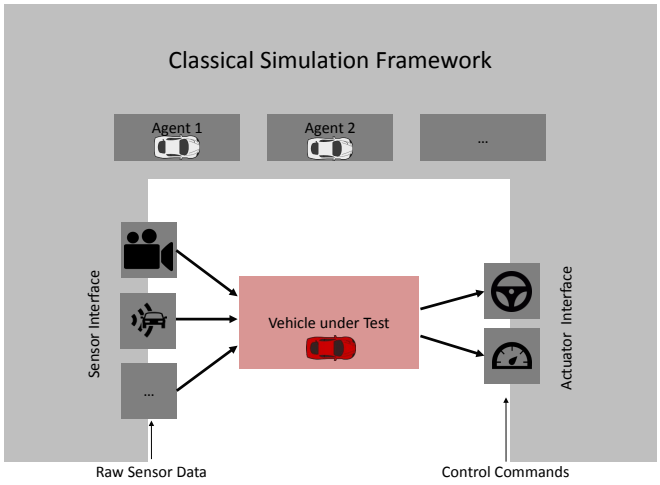
In a merging scenario for example, traffic participants in a lane will (only) open a gap, if they recognize a desired lane change onto their lane.

Still, developing and testing algorithms solely on the road is not an option, due to high costs, time effort and security reasons. Moreover, approaches that are not yet real-time capable cannot be tested on the road. Furthermore, repeatability is limited in real road experiments. Consequently, an evaluation on the road is inappropriate for research purposes.

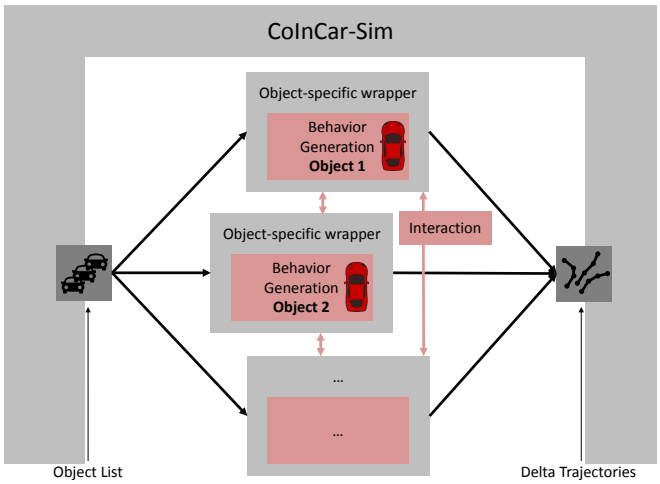
As the evaluation on the road is not an option, using a simulation framework suggests itself. However, existing simulation frameworks, as explained further in the following section, typically do not focus on the interaction of traffic participants, but rather simulate a single vehicle very comprehensively.

Contribution

This work presents concepts and implementation of an open-source simulation framework, which allows to simulate scenarios arising from potentially cooperative situations (cf. Fig. 1). The framework focuses on the interaction of traffic participants by enabling the simultaneous simulation of several instances of one or more automated vehicles. Research topics that are already covered by existing simulation frameworks or datasets, such as sensor data processing and



(a) Classical simulation frameworks commonly focus on the full simulation of a single vehicle, including both the perception and the control algorithms. Other traffic participants, often called "agents", are provided by the framework and use simple driver models.



(b) CoInCar-Sim: The multi vehicle simulation framework facilitates the use of a sophisticated behavior generation algorithm per object. Different objects can use different algorithms or different instances of the same algorithm. Perception and control algorithms are not in-the-loop, but their effects on the vehicles' behavior can be evaluated, allowing a different configuration in each wrapper.

Fig. 2: Comparison of CoInCar-Sim to classical simulation frameworks.

low level control, are not focused on in the framework. However, the effects of uncertain perception and control on the behavior generation can of course be modeled (cf. Fig. 2). In the presented framework, behavior and trajectory planning algorithms that incorporate interaction between traffic participants can be developed, tested and compared in various cooperative scenarios.

Alongside with the simulation framework, we propose a common, basic interface for behavior planners.

Outline

The remainder of this paper is structured as follows: Section II introduces the related work. Subsequently, the required features for a simulation framework for cooperative behavior generation are derived. After that, the concept and implementation of the framework is presented, followed by examples for its application. Finally, the work is concluded in Section VI.

II. Related Work

Simulation frameworks have supported the development and validation of advanced driver assistant systems from the beginning. Those mostly commercial simulation frameworks, such as *IPG CarMaker*, *TASS PreScan* [5], *VIRES VirtualTestDrive*, *dspace ASM*, *SiVIC* [6] and many more, provide emulation of raw sensor data and simulate vehicle dynamics, based on detailed physics models. These detailed environment models facilitate the test of the complete vehicle, including the hardware such as the electronic control units. However, they typically focus on the simulation of a single vehicle, and provide simple driver models for other traffic participants in the scene. These driver models are for example provided by the traffic flow simulation *SUMO* [7]. Here, the focus is often on longitudinal behavior and traffic participants can only follow fixed paths, such that smooth lane changes are replaced by teleportation from lane to lane, as Zofka et al. describe in their simulation approach [8]. This is of course a strong restriction and makes those approaches unsuitable for the desired purpose.

Existing open-source multi-car simulation platforms such as *TORCS* [9] often are designed for the purpose of a racing game. Consequently, they lack pedestrians, intersections, traffic rules and other characteristics of public road traffic. The recent open-source simulation platform *CARLA* [10] presents the full complexity of urban road traffic. It is, however, built upon the proprietary *Unreal Engine 4 (UE4)*, focusing on a detailed physics modeling for approaches such as end-to-end learning. Receiving abstract object information and providing a desired trajectory, which is state of the art in modular approaches, is not intended.

The goal of comparability and reproducibility in the field of motion planning is addressed by *CommonRoad* [11]. In order to benchmark motion planning approaches, they provide both recorded and hand-crafted scenarios alongside with vehicle models and cost functions. However, in contrast to this work, they distinguish between one or more *agents* that are controlled by a single planner, and *obstacles* with a predefined motion. While this approach is helpful for exact and deterministic benchmarking, it does not model interaction between obstacles and agents. Thus, it is not intended for motion planning approaches focusing on mixed traffic[1]. Rather, the CommonRoad scenarios can be integrated into the proposed framework, allowing to replace the obstacles by agents with a different planning approach.

[1]Human drivers and automated vehicles sharing the road.

III. Required Features

The goal of this simulation framework is to support researchers in the development, testing and comparison of cooperative behavior generation algorithms. The requirements for this purpose are derived in the following.

Multi Vehicle Simulation: The symbiosis of multiple instances of the same behavior generation algorithm alongside each other is a basic requirement. However, in real traffic, vehicles running different algorithms and human drivers will share the road. In order to test such situations, different implementations have to run simultaneously in the same framework. This necessitates *common interfaces*, including the frequency with which they provide information[2], and the provision of a simulation time. These interfaces must allow for *communication*. Both implicit communication, e.g. via the driven trajectory, as well as explicit communication, such as V2X-messages is necessary. Furthermore, enabling *human drivers* to control a vehicle in the simulation framework allows for the evaluation of human-machine interaction in a safe environment. For testing confidential algorithms, the interconnection of multiple vehicles over *network* (e.g. TCP/IP) should be facilitated.

Visualization: For better notion and first analysis, the framework should provide a vivid visualization, yet allowing to use own visualization.

Replicability: The co-simulation of different, interacting approaches under the influence of noisy perception is per se not deterministic. Hence, the simulation should be recordable in order to allow downstream in-depth analysis of corner cases.

Usability: The framework should be open-source and free for commercial use, as there is much contribution from industry in the field of automated driving. Obviously, the framework therefore should not be based on or contain any closed-source or limited-use libraries.

IV. Concept

In this section, the architecture of the framework, along with its underlying concept is presented. As the Robot Operating System (ROS) [12] already meets some of the required features natively, and it is well-suited for automated driving in general [13], we decided to use ROS as underlying middleware. To be precise, ROS provides an interface definition language, it allows for communication via network, and it supports recording all data sent via the interfaces. Furthermore, it provides many tools to improve the usability and facilitate visualization. Readers that are new to ROS are referred to [12] and [13] for a better understanding of the remainder of this work. In the following, we will outline our environment model along with the purpose and definition of its key interfaces. After that, the wrapper, connecting the algorithms to be tested to the environment model, as well as the scenario definition are explained.

[2]The frequency does not need to be equal for all algorithms, but a controller proven to be stable at 100Hz should not rely on a 20Hz visual odometry, for example.

A. Environment Model: Time and Localization Management

The simulation framework must model all relevant interaction between the system under test and its environment. This functionality is covered by the *environment model*. In order to both maximize the performance and minimize the maintenance effort, the environment model is kept as lean as possible: Only those parts of the environment that are key to the interaction of traffic participants are modeled. In real road traffic, participants interact when they compete for a certain space at the same time. Motivated by this, the environment model manages only the time and the location of all objects. Other environmental circumstances such as the road condition, the weather and many more certainly have an effect on the interaction, but their effect differs from vehicle to vehicle[3].

The *time management* provides the simulation time, which can be real time, slower or faster. The *localization management* provides an array of object states, each containing the current position of the object. Motion is reported to the localization management by providing a desired motion in form of a delta trajectory. The delta trajectory contains poses at certain times, with reference to the current pose of the object at the current time. It will be explained in more detail in the next subsection. In-between the poses, the localization management interpolates linearly. The motion model, as it also largely depends on the object dynamics, is provided by the object itself in the wrapper. This outsourcing facilitates a lean core of the simulation framework, yet allowing to integrate detailed motion models. The wrapper will also be explained in-depth in the following.

B. Interfaces

As the framework uses ROS as middleware, ROS messages as interfaces to the environment model are an obvious choice. The two major messages of the localization management are the `ObjectState`, respectively an array of those, and the `DeltaTrajectory`. Since their source code in the ROS interface definition language is available on github[4], they are only discussed briefly: The object state consists of an object identifier, an object classification and the current motion state of the object. Furthermore, the object state contains the object's shape. When used by the localization management in the presented simulation framework, all parts of of the motion state except for the pose are marked invalid, as the localization management does not prescribe any underlying motion model.

The delta trajectory depicts the future motion of an object. It consists of delta poses that each describe the delta between the current pose and the desired pose, along with the time difference in which the desired pose is to be reached. Thus, for a physically feasible motion, it should start with pose

[3]While camera-based perception is weaker at night, LIDAR-based perception is weaker in rain, for example.

[4]`https://github.com/fzi-forschungszentrum-informatik/automated_driving_msgs/`

zero[5] at time difference zero, meaning that it does not jump at the start.

In addition, since sending messages is a core ROS functionality, the framework natively supports V2X-Communication. We defined an additional message called `AdditionalHeaderForCommunication`. It contains information about who sent the message, and from which pose the message was sent. With this information, effects of message transmission can also be modeled, as further explained in the following.

C. Wrapper

The algorithms under test should not be modified to avoid discrepancies between their behavior in practice and in simulation. Since their interface does not necessarily meet the interface of the simulation framework, we make use of a wrapper. Further, sensor and actuator characteristics are modeled in the wrapper, which allows to simulate the effects of those characteristics on the system under test. The characteristics that can be modeled include for example uncertainty and sensor range, but also reduced friction in the motion model (cf. Fig. 3). As well, ghost objects can be inserted in the wrapper. When behavior generation algorithms are developed within the framework, we encourage the use of the above described interfaces also inside the wrapper. A module that computes velocity and acceleration of objects by building finite differences, assuming a rigid body motion, can then for example serve as a minimal wrapper.

Regarding V2X-communication, phenomena like message drop or delays can be modeled in the wrapper, and can also be tuned per object.

Thus, approaches in an early stage can access exact and comprehensive data, while being combined in the same simulation environment with more advanced approaches that access realistic data, or even model sensor failure.

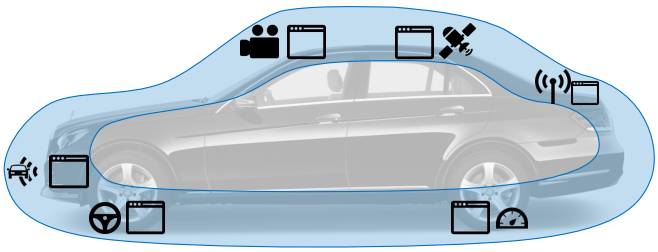


Fig. 3: Example of a wrapper for an object in the simulation framework: As input for the algorithms under test, the output of sensor processing algorithms is modeled from the information of the localization management. The output of the algorithms under test are fed to a controller with a simulated actuator, which then passes the desired motion back to the localization management.

[5]zero translational difference and zero rotational difference

D. Scenario Definition

As stated previously, cooperative scenarios cannot be comprehensively defined, since they are largely affected by the objects' interaction. Rather, we can define the initial scene along with goals for the objects in the scene that lead to a cooperative scenario. For this, it is recommended to first define a place and map for the simulation[6]. The authors encourage the use of real places in the simulation, allowing every researcher to build an own map with the desired information. The use of lanelet maps [14] for behavior generation and planning is recommended but not mandatory. However, as the localization management is based on Cartesian coordinates, but real places are described on the surface of the Earth, a common map projection must be utilized when using real places and maps of them. We decided to use Universal Transverse Mercator (UTM).

Once the place is set, we define objects with an ID, a classification (car, truck, ...), a shape and an initial pose along with a delta trajectory. Each object can then be operated by a behavior generation algorithm, along with a suitable wrapper and a goal, or a human driver with a manual control, e.g. through a steering wheel connected to the simulation framework. The goal can be defined in different levels: It can be collision avoidance, the minimization of a certain cost functional, or reaching a certain space at a certain time. If an object is not operated, it just follows its initial delta trajectory.

E. Usage

ROS facilitates launching multiple ROS nodes with the tool `roslaunch`. To use the latter, the previously described, abstract scenario definition must be translated into ROS launchfiles. While a more detailed manual is provided with the source code, we now briefly introduce the launchfile usage.

First, a wrapper for the behavior generation algorithm to be tested must be written. Again, when algorithms are developed within the framework, the use of the presented interfaces inside the wrapper is recommended. By doing so, modules that are provided with the framework can be used as a wrapper, without the need to implement a wrapper from scratch. Then, a launchfile is created, launching the algorithm to be tested wrapped in its wrapper. This launchfile can be made configurable, for example regarding the modeled sensor setup. It is called *object launchfile* in the following, since it operates an object in the simulation framework.

Next, the initial trajectory has to be defined. Here, the Java OpenStreetMap Editor (JOSM)[7] can be used to determine a path. An initialization module, provided with the framework, transforms this path and a desired initial velocity into a trajectory.

Subsequently, the overall launchfile, also provided with the framework, has to be modified: Parameters such as the

[6]Technically, the simulation can also take place in empty space, but this complicates the goal definition.

[7]`https://josm.openstreetmap.de/`

simulation speed have to be set. Objects have to be defined as a combination of an initialization with a trajectory, and an object launchfile. Of course, several objects can be defined with the same behavior generation algorithm, as requested in Section III. This is done by using the same object launchfile with a different initialization.

Finally, this overall launchfile contains the scenario definition. It can be launched via `roslaunch`.

F. Visualization

For visualizing the scenarios, we make use of the 3D visualizer `RViz` that is provided with ROS. RViz can be extended by plugins, visualizing custom ROS messages. To illustrate the environment, we provide plugins for the lanelet map, the object states, as well as the desired motion, depicted by the delta trajectory. For the latter, we use a color code for the absolute time, as in the Bertha Benz Memorial Drive [15], to facilitate the estimation of potential collisions: Overlapping circles of the same color denote a future collision. A sample view of the plugins is given in Fig. 4.

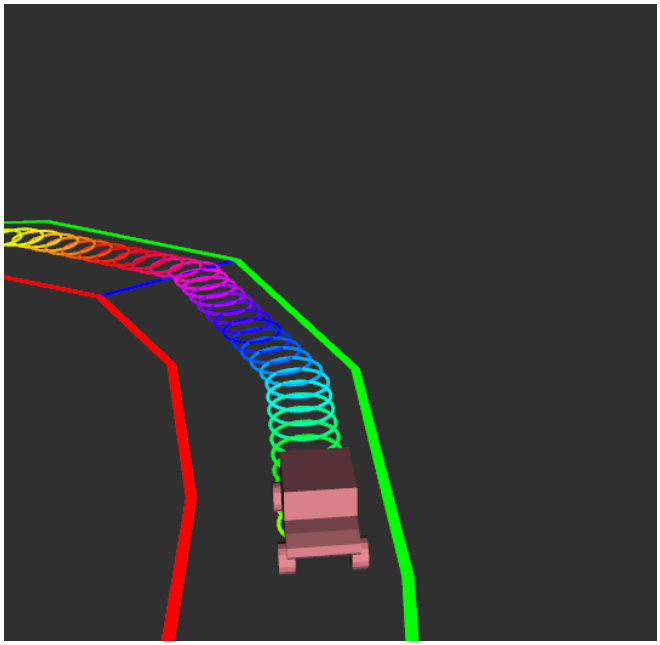


Fig. 4: RViz Plugins: The vehicle is visualized by the ObjectStateArray-Plugin. The lanelet map (right bound: green, left bound: red) is visualized by the Lanelet-Plugin. The current plan for the future trajectory is visualized by the DesiredMotion-Plugin. The time is color-coded as further explained in the text.

V. Use Case

In order to show the benefits of our framework, we have wrapped our previously introduced prototype implementation of a cooperative motion planner [16] and created an object launchfile. The motion planner generates cooperative trajectories for a single vehicle in mixed traffic, without explicit communication. The idea behind the approach is to find the globally optimal solution for all participants in a situation. Then, the ego vehicle acts according to this optimal solution. The implementation is based on path velocity decomposition: Assuming a fixed path for every traffic participant, velocity profiles are sampled. The optimal solution is chosen as the combination of velocity profiles, that yields the lowest value of a predefined cost functional. For further details on the approach and its implementation, it is referred to [16].

As this planner is a prototype, it is implemented in python and not yet real-time capable. Consequently, we make use of the ability to slow down the simulation time. Furthermore, measurement uncertainties are not yet considered in the approach, so we do not use simulated perception output, but access the ground-truth data, provided by the localization management. Also, for this prototype testing, we assume perfect control. That is, the desired trajectory, determined by the planner, is passed straight to the localization management, without emulating a controller and an actuator.

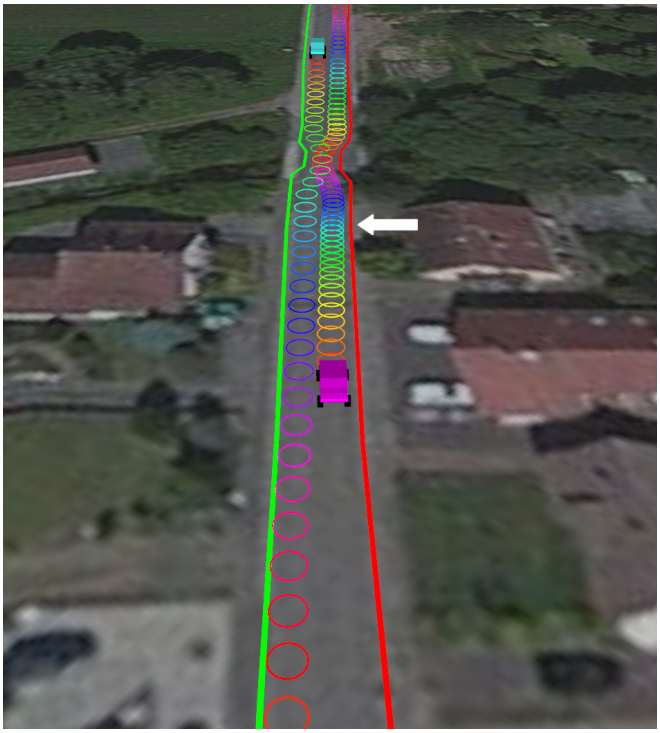


Fig. 5: Testing a cooperative planning approach in the simulation framework: The two vehicles (magenta and cyan) are approaching a narrowing. Looking at the color-coded future trajectories, we see that the vehicles will pass each other in front of the narrowing, indicated with the white arrow.

As a first test for the cooperative planner, we have initialized two instances of the same planner, approaching a narrowing (cf. Fig. 5). In order to do so, we just had to launch the respective object launchfile twice, providing a different ID and initial trajectory for each vehicle. In Fig. 5 we can see that both vehicles agree on the globally optimal solution, even without explicit communication: the cyan vehicle drives first, and the magenta vehicle drives second though the

narrowing.

VI. Conclusions and Future Work

We have presented CoInCar-Sim, an open-source simulation framework for cooperative interacting automobiles. In contrast to existing frameworks, this work focuses on the interaction of traffic participants. It facilitates to instantiate several objects with SAE4+ capable behavior generation algorithms, allowing to replace widely-used reactive driver models that do not allow for development and evaluation of cooperative planning algorithms. Different objects can either use different behavior generation algorithms or different instances of the same algorithm. Perception and control algorithms as well as V2X communication algorithms are not in-the-loop, but their effects on the vehicles' behavior can be evaluated. As each object has an individual wrapper, different perception, control and communication setups can be tested simultaneously. This allows to test early approaches, accessing ground truth data and assuming perfect control, along with more advanced approaches that access realistic data, or even model sensor failure.

In the future, the authors intend to extend the framework by a benchmark for cooperative behavior generation. The simulation framework will be released open-source at `https://github.com/coincar-sim`.

Acknowledgements

This work was supported by the Tech Center a-drive and by the Deutsche Forschungsgemeinschaft (German Research Foundation) within the Priority Programme "SPP 1835 Cooperative Interacting Automobiles". Furthermore, the authors like to thank Tobias Strauss for fruitful discussions and his valuable feedback and their student assistants who largely contributed to the implementation.

References

[1] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, "Three decades of driver assistance systems: Review and future perspectives," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 4, pp. 6–22, winter 2014.

[2] SAE, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles - J3016_201609," 2016.

[3] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets Robotics: The KITTI Dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[5] M. Tideman and M. v. Noort, "A simulation tool suite for developing connected vehicle systems," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 713–718.

[6] D. Gruyer, S. Choi, C. Boussard, and B. d'Andréa Novel, "From virtual to reality, how to prototype, test and evaluate new ADAS: Application to automatic car parking," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 261–267.

[7] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[8] M. R. Zofka, S. Klemm, F. Kuhnt, T. Schamm, and J. M. Zöllner, "Testing and validating high level components for automated driving: simulation framework for traffic scenarios," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 144–150.

[9] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS, The Open Racing Car Simulator," http://www.torcs.org, 2014.

[10] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, 2017, pp. 1–16.

[11] M. Althoff, M. Koschi, and S. Manzinger, "CommonRoad: Composable benchmarks for motion planning on roads," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 719–726.

[12] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, 2009, p. 5.

[13] A. M. Hellmund, S. Wirges, Ö. Ş. Taş, C. Bandera, and N. O. Salscheider, "Robot operating system: A modular software framework for automated driving," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Nov 2016, pp. 1564–1570.

[14] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, June 2014, pp. 420–425.

[15] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb, "Making Bertha Drive - An Autonomous Journey on a Historic Route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, Summer 2014.

[16] M. Naumann and C. Stiller, "Towards Cooperative Motion Planning for Automated Vehicles in Mixed Traffic," in *IEEE/RSJ Intl. Conf. Intelligent Robots and Systems Workshops*, Sep 2017. [Online]. Available: https://arxiv.org/abs/1708.06962