Hindawi
Journal of Advanced Transportation
Volume 2018, Article ID 2586520, 32 pages
https://doi.org/10.1155/2018/2586520

Research Article

A Rapid Prototyping Environment for Cooperative Advanced Driver Assistance Systems

Kay Massow[1] **and Ilja Radusch**[2]

[1]*Daimler Center for Automotive IT Innovations, Technical University of Berlin, Berlin, Germany*
[2]*Fraunhofer Institute for Open Communication Systems (FOKUS), Berlin, Germany*

Correspondence should be addressed to Kay Massow; kay.massow@fokus.fraunhofer.de

Received 26 August 2017; Accepted 11 December 2017; Published 20 March 2018

Academic Editor: Yuchuan Du

Advanced Driver Assistance Systems (ADAS) were strong innovation drivers in recent years, towards the enhancement of traffic safety and efficiency. Today's ADAS adopt an autonomous approach with all instrumentation and intelligence on board of one vehicle. However, to further enhance their benefit, ADAS need to cooperate in the future, using communication technologies. The resulting combination of vehicle automation and cooperation, for instance, enables solving hazardous situations by a coordinated safety intervention on multiple vehicles at the same point in time. Since the complexity of such cooperative ADAS grows with each vehicle involved, very large parameter spaces need to be regarded during their development, which necessitate novel development approaches. In this paper, we present an environment for rapidly prototyping cooperative ADAS based on vehicle simulation. Its underlying approach is either to bring ideas for cooperative ADAS through the prototyping stage towards plausible candidates for further development or to discard them as quickly as possible. This is enabled by an iterative process of refining and assessment. We reconcile the aspects of automation and cooperation in simulation by a tradeoff between precision and scalability. Reducing precise mapping of vehicle dynamics below the limits of driving dynamics enables simulating multiple vehicles at the same time. In order to validate this precision, we also present a method to validate the vehicle dynamics in simulation against real world vehicles.

1. Introduction

Advanced Driver Assistance Systems (ADAS) are integrated functions of road vehicles, designed to support the driving process. ADAS can replace or complement decisions and actions of human drivers by precise machine tasks. This allows eliminating driver errors which may lead to many problems, like accidents, congestions, or pollution. These problems gain more and more importance caused by growing number of vehicles, which push the road traffic systems over their capacity limits. For this reason, ADAS, such as the functions Mercedes-Benz combined in their INTELLIGENT DRIVE [1] concept, are currently strong innovation drivers for the automotive industry.

Today's ADAS are realized through an autonomous approach with all instrumentation and intelligence on board of one vehicle. However, in order to assemble more of these functions to reach fully autonomous driving in a complex road network, very expensive sensors and complex machine intelligence are required [1]. Thus, to further enhance the area of application for ADAS with reasonable implementation effort for sensors and intelligence, ADAS need to cooperate in the future [2]. Such cooperative ADAS will be enabled by communication between ADAS deployed on different vehicles and on road infrastructures, for example, using V2X communication [2]. In case of Cooperative Adaptive Cruise Control (CACC) [3] information is shared among vehicles in a platoon aiming at a harmonized cruising speed.

As ADAS can directly intervene into vehicle control, their design and implementation are highly safety-critical. Hence, comprehensive evaluation methodologies are of vital importance for the ADAS development process, as, for example, described in [4]. For the development of cooperative ADAS, however, new evaluation methods are necessary. Due to the complexity of the addressed traffic scenarios, employing real world vehicles would require a tremendous effort. Thus, especially for the early phases of prototyping, simulations will become increasingly important. In the first stage of

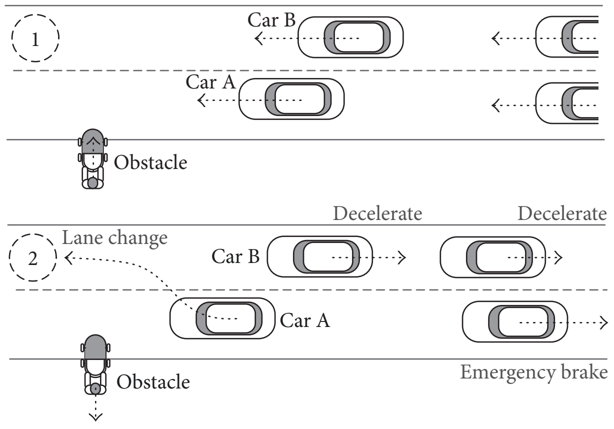


Figure 1: Emergency Lane Changing Situation [5].

prototyping, a novel idea for an application needs to prove its feasibility, such as solving a hazardous situation by a new cooperative ADAS. The cooperative aspect makes the number of parameters to be considered while prototyping very large. Therefore, developers need to handle these large parameters' space to come to such a verdict about feasibility. This handling includes finding and tuning parameters in a time-consuming trial and error manner.

The prototyping environment presented in this paper is designed to support developers in the first stage of prototyping, when an idea for a new cooperative ADAS is tested for feasibility. For this purpose, our prototyping environment dedicatedly supports handling large parameter spaces inherent in cooperative ADAS. For further clarification of the described problem of finding and tuning these parameters we introduce the following example. Let us consider the idea "Cooperative Emergency Lane Changing" (CELC), which imposes a large parameter space to be handled by the developer. A hazardous scenario to be solved by CELC is depicted in Figure 1.

Two cars drive side by side on a two-lane road and Car A is ahead about half a car length. Unforeseeable, an obstacle appears on the left lane in front of Car A. The distance to the obstacle is so close that the driver's response time is too long for an emergency stop. Furthermore, an abrupt stop involves the danger of rear end collisions by following vehicles. The right lane is blocked by Car B. The velocity of the obstacle entering the road is assumed to be slow enough so that a lane change maneuver would solve the situation. A lane change maneuver of Car A would require an immediate braking reaction of Car B to avoid a crash of both vehicles. In order to allow Car A to change the lane, Car B does not need to brake to standstill, which is less likely to provoke a rear end accident than a stop to standstill of Car A. The CELC System, deployed on both vehicles, recognizes the hazardous situation. Car A informs Car B about its intent to change the lane. As a result, negotiation in a fraction of a second results in Car B and its followers braking autonomously, allowing Car A to change the lane autonomously, while the followers of Car A are informed to do an emergency brake. The whole process is done before the drivers of both vehicles even realize the situation.

For an automated system, such a situation is very complex to deal with. Normally, even if a crash is unavoidable, from the perspective of the system, braking would be the safer option than evading by steering [5, 6]. This is due to the fact that preferring evading over braking requires the correct recognition of the situation with a confidence level which is hard to reach. This problem is described in [6] as the dilemma between braking and steering and a theoretical resolution of the dilemma is given, assuming all relevant problem parameters are known. However, if a vulnerable road user is involved, a collision, even with lower speed, is an inacceptable option and so the situation gets even more complex. Additionally, with each additional vehicle needed to be involved in solving the situation by cooperation, the parameter space to be considered by developers of such systems grows.

To give an impression of how fast such a parameter space tends to explode, we will illustrate this with the example of CELC. The parameters influencing the stopping distance of Car A depend on car specific properties like its mass, speed, current acceleration, brake efficiency, and the tire profile condition. Additionally, environment specific parameters like road slope, friction, depending on the road surface, and weather condition need to be considered. These parameters can be taken into account to determine the most probable minimal stopping distance for Car A and a related value of confidence. Now, situation specific parameters like the confidence about the following facts need to be considered. The obstacle will enter the road, the obstacle is vulnerable, there is a lane which can be used to evade, there is a car on that lane blocking Car A, that car is Car B and communication with Car B is possible, and there is no car on the left lane following Car A so close that rear end accidents are likely to happen. Nearly the same number of parameters need to be considered by Car B for itself and maybe by other cars following behind. With every further car involved, such as further cars following Car B on the right lane, the number of relevant permutations of parameters explodes with an exponential rate (see [7] for a specific example). Finally the confidences calculated need to be combined cooperatively, to determine if an evasion maneuver should be performed or not. Different parameter constellations result in different minimum distances of Car A to the obstacle, which can be handled by the application. This is again a variable of the related research and can give an answer to the question if the CELC application makes sense to be developed and to assess its impact to traffic safety.

In order to deal with such a big parameter space and to come to a decision if a cooperative ADAS makes sense to be developed for certain situations, developers need a prototyping environment that is able to map the these parameters and to create relevant situations. This includes mapping physics and an iterative prototyping process supporting the mentioned trial and error manner to create, refine, and assess cooperative ADAS. In this way, a candidate cooperative ADAS can be studied and either incrementally be brought through the stage of prototyping, or be discarded. For this purpose, we present a prototyping environment using vehicle simulation, which is capable of supporting the described process, while being able to map physics up the required level of precision for multiple vehicles at the same time. To ensure the transferability of gained simulation results to the real world, we present a validation method which is able to validate the simulation models against real world vehicles within the boundaries of the scope of our simulator.

The rest of this paper is organized as follows. In Section 2, we derive the requirements for our prototyping environment. Section 3 summarizes the relevant related work and its suitability to meet these requirements. In Section 4, we design our prototyping environment based on vehicle simulations and, in Section 5, we propose a method to validate its vehicle dynamics.

2. Requirements

As described in the previous section, a prototyping environment is required to assess cooperative ADAS at an early stage in development. Before we derive the requirements such a prototyping environment imposes, we first classify more precisely the class of cooperative applications we are focusing on. Subsequently we define the scope of our prototyping environment along with a set of reference applications of cooperative ADAS, which cover all aspects of cooperative ADAS within this scope. These applications will help to define the technical requirements of our simulator and guide its validation.

2.1. Cooperative ADAS Definition. The concept of Intelligent Transportation Systems (ITS) is considered as a way to enhance the transport system using information and communication technologies for the planning, management, and operation. The generic term ITS is commonly used for integrated application of communications, control, and information processing in this context [8]. These applications bring users, vehicles, and infrastructure in a relationship to exchange information concerning the actual task of transportation. For this purpose, ITS applications are of particular interest for dynamic traffic management, as they can help to estimate and predict the traffic flow more accurately and thus enable to control vehicles on the road more effectively.

Among many application domains of ITS, the Automated Highway System (AHS) is one of the most important items [9]. AHS assumes that vehicles are guided along highway lanes autonomously rather than by the driver, using sensors and communication devices, which link the road and the vehicles. This concept implies that at least one dedicated lane on existing highway infrastructure is reserved only for fully automated vehicles with intelligent technology and aims at reducing driver error and thereby increasing safety and traffic throughput considerably [9, 10].

Along the way of automating parts of all the driving tasks, Advanced Driver Assistance Systems (ADAS) gained popularity and most of the related researches in the field of ITS/AHS focus on the ADAS design. For a better understanding of the terms ITS/AHS/ADAS, Figure 2 depicts the schematic relation between them [11]. In the 1990s, vast studies on ADAS control design and assessment of their impact on traffic flow have been done in the context of the PATH[1] program in the USA and the programs PROMETHEUS[2] and DRIVE[3] in Europe. ADAS entered the series production of vehicles starting with ABS (Antilock Braking System) and ESP (Electronic Stability Program) through today's ACC (Adaptive Cruise Control) and LKA (Lane Keeping Assist) which potentially would already enable full automation.

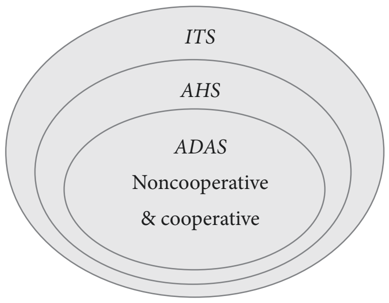


Figure 2: Schematic relations of ITS, AHS, and ADAS [11].

In the 2000s, with the upcoming V2X [12] communication technologies, cooperative systems that link vehicles and road infrastructure received more and more attention. Research interests began to turn to cooperative systems, with large funded initiatives, among many others, the IVI[4] initiative in the USA and projects CVIS[5], DRIVE-C2X[6], and SAFESPOT[7] in Europe. This research resulted in new development of cooperative ADAS applications, algorithms, and control concepts [13]. Finally, current research initiatives like the EU funded project Autonet2030[8] and the German research project IMAGinE[9] aim at closing the gap between automation and cooperation, which includes interaction control among cooperative vehicles, including both automated and manually driven vehicles. As deployment horizon, 2020–2030, is expected, predicting that in 2030 50% of sold passenger cars could be highly autonomous [14, 15].

As indicated by Figure 3, ADAS can be characterized by their level of automation [16] and cooperation [2]. In order to elaborate this characterization, Figure 3 orders different ADAS applications according to their specific degree of cooperation and automation [17]. The prototyping environment described in this work is more useful to prototype applications, which require a high degree of both dimensions, automation, and cooperation, at the same time. These cooperative ADAS are arranged close to the diagonal in the figure. Thus, the scope of the prototype environment, indicated by the arrow in the figure covers the area around the diagonal and grows with the degree dimensions from the bottom left to the top right.

2.2. Scope. The scope of our prototyping environment describes the class of applications, its constraints on driving dynamics, and the goals of prototyping. We define this scope to cover a subset of cooperative ADAS for the following reason.

As outlined in the previous section, cooperative ADAS define a certain class of applications for vehicles, which include active intervention by control of vehicle actors and cooperation enabled by V2X communication. In addition to

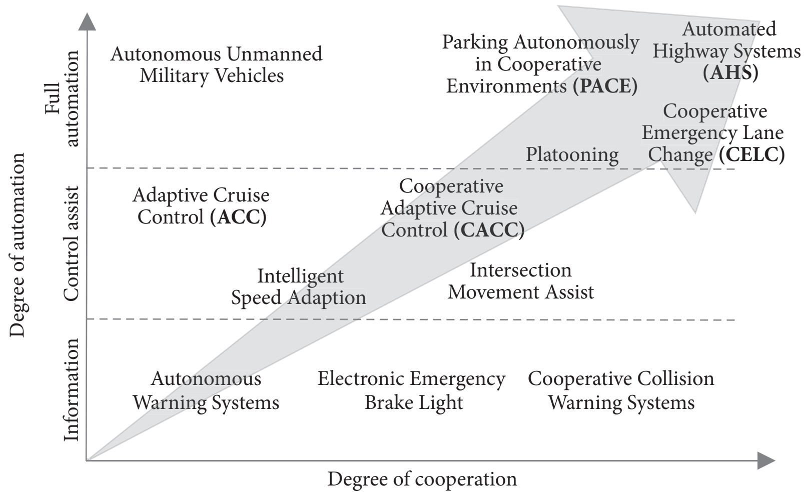

Autonomous Unmanned Military Vehicles
Parking Autonomously in Cooperative Environments (PACE)
Automated Highway Systems (AHS)
Cooperative Emergency Lane Change (CELC)
Platooning
Adaptive Cruise Control (ACC)
Cooperative Adaptive Cruise Control (CACC)
Intelligent Speed Adaption
Intersection Movement Assist
Autonomous Warning Systems
Electronic Emergency Brake Light
Cooperative Collision Warning Systems
Full automation
Control assist
Information
Degree of automation
Degree of cooperation

Figure 3: Usefulness of the prototyping environment for developing application, depending on their level of automation and cooperation.

the ability to perceive the vehicle's environment and driving conditions it might also be necessary to be able to control the vehicle at its dynamic limits. This is, for example, required to perform safe accident avoidance maneuvers. However, mapping driving dynamics in simulations in a physically realistic way at or beyond the limits of driving dynamics requires complex and highly nonlinear simulation models. Such models grow in complexity and computational demand with their precision at the limits of driving dynamics. This characteristic makes vehicle dynamics simulation to be a conflicting requirement with simulating multiple vehicles at the same time, which is necessary for the cooperative aspect. Thus, we need a tradeoff between the number of vehicles to be simulated at the same time and the precision of mapping physics realistically at the limits of driving dynamics. For the prototyping environment developed in this work, we solve this tradeoff with the following assumptions.

We define the scope of our prototyping environment to be restricted to the class of applications which are meant to prevent accidents, not to mitigate them. Therefore, we can assume a certain safety margin which should always be regarded by the applications. That excludes very close drive-by maneuvers at high speeds or with very high acceleration forces, among others. For our simulator it is thus sufficient to map a smaller range of dynamics, as its purpose is to match real vehicle dynamics up to the limits of driving dynamics only.

Once we have defined the class of applications of cooperative ADAS and the scope of our simulator, we can define a set of reference applications for our simulator. These reference applications cover all aspects of the class of cooperative ADAS within the defined scope of our simulator. Thus, this set of reference applications can be used to derive the requirements of the simulator and for validation later. We can summarize the following aspects to be relevant for the applications to be developed with our simulator:

(i) Vehicle dynamics are relevant for the correct function of the application.

(ii) A cooperative aspect involving multiple vehicles is a vital prerequisite for the application to work.

(iii) The correct evolution of the absolute position of vehicle involved in cooperative maneuvers needs to be predictable by the application precisely, which gets relevant at lower speeds where the safety margin allows small distances to other objects

2.2.1. CELC: Cooperative Emergency Lane Change. The CELC application described earlier in Section 1 was initially proposed in [5]. The number of involved vehicles ranges from two to tens of vehicles including the followers on both lanes which need to brake in a coordinated way. We choose CELC as one reference application to especially cover the lateral vehicle dynamics aspect of cooperative ADAS. The lateral dynamics are foremost relevant for the application deployed on the evading vehicle, to predict its evasion trajectory. This trajectory must not cut the trajectory of the braking vehicle and avoid collision with the vulnerable obstacle as well. Initial speeds of around 50 km/h or higher are assumed, as for lower speeds evading hardly makes sense to be preferred over full braking. As mentioned earlier, assuming speeds in this range, the application should always regard a certain safety margin. Thus, the precision of the evolution of the vehicle position is less important than the lateral dynamics which determine the shape of the evasion trajectory.

2.2.2. CACC: Cooperative Adaptive Cruise Control. The CACC application is an extension of the adaptive cruise control system. Communication in vehicle platoons is used to enhance the automated longitudinal control in a way that string stability in the platoon is achievable [3]. For CACC, the control loop involves accelerator and brakes of multiple vehicles in a platoon. In such platoons, ranging from two to hundreds of vehicles, small changes of the speed of one vehicle can be amplified in the upstream direction considerably, which requires very precise control algorithms to avoid. This makes CACC extremely sensitive to longitudinal vehicle dynamics and thus its requirements to a simulation include realistic mapping of longitudinal vehicle dynamics. With CACC as reference application, we cover the longitudinal dynamics aspects of cooperative ADAS. A precise evolution of absolute positions is not necessarily required, if the relative distances between vehicles in platoons are mapped realistically.

2.2.3. PACE: Parking Autonomously in Cooperative Environments. The PACE application allows drivers to leave their vehicles at the entry of a parking garage and let them drive and park to free spots autonomously using series sensors, as presented in [18]. This is enabled by cooperation between the vehicles and the infrastructure of the parking garage [19]. Cooperative aspects are, for instance, camera-based off-board vehicle localization, parking spot allocation, maneuver planning (e.g., to avoid deadlocks), or sharing sensor information among vehicles to recognize moving objects like pedestrians. In contrast to CACC and CELC the driving speeds are comparatively low. With lower speeds, the impact of driving dynamics on the vehicles motion gets less in relation to the kinematics, due to the lower forces and slip. However, the accuracy of the evolution of the absolute vehicle position gets more important, since driving in narrow spaces requires driving close to other objects. The number of involved vehicles for PACE ranges from one interacting with the infrastructure of the parking garage up to tens moving on a parking deck.

2.3. Technical Requirements. Active controlling of a vehicle by an ADAS requires consideration of vehicle dynamics, which includes a great number of parameters regarding the vehicle and its environment [20–22]. As outlined in Section 1, an ADAS deployed on a vehicle additionally needs to regard many application-specific parameters, like those related to situation awareness [23, 24]. Depending on the specific application, this parameter space might grow in a linear manner with each additional vehicle involved in a situation. A simulator for such ADAS has to be able to map this parameter space in simulation. For cooperative ADAS deployed on multiple vehicles (such as CELC, CACC, and PACE), the number of parameters to be considered by the simulator might even grow exponentially. Depending on the specific application, multiple instance of a cooperative ADAS deployed on different vehicles involved in the simulation might need to interact with each other. Thus, each of them also needs to consider the relevant parameters of multiple other vehicles. This makes the parameter space to be handled by the simulator very large. For the technical requirements of our simulator, we address the following two aspects concerning this large parameter space:

(i) In order to enable our simulator to map the full parameter space as described, we need to create the tradeoff mentioned in (Section 2.2) between mapping vehicle dynamics precisely and simulating multiple vehicles at the same time. While the number of vehicles needs to be as big as required by cooperative ADAS, the precision of vehicle dynamics needs to be validable within our scope (Section 2.2).

(ii) The application-specific part of this parameter space is the object of analysis handled by the developer using the simulator, in order to create, enhance, and assess applications. For this purpose, a big number of application-specific parameters may be introduced, tuned, or dropped by developers. Therefore, we create an iterative process for prototyping cooperative ADAS, supported by our simulator, to enable developers to handle the application-specific part of the parameter space.

Towards the named two aspects, in the following, we define the high level requirements for our simulator, described on the basis of the three reference applications (ELC, CACC, and PACE). For this purpose, we first identify the originators of the requirements. These are the cooperative ADAS *application* to be deployed in our simulator, the *developer* using our simulator, and the *scope* of our simulator. The first three main entities of our simulator defined as clusters of requirements are derived from the DVE model [25], which models the loop of *driver*, *vehicle*, and *environment* while driving which will be observed by the *application*. These three are complemented with the *communication*, which addresses the cooperation aspect, and finally the *architecture* of the simulator which needs to fit our *scope*. In Table 1, the originators are arranged at the column headers and the requirement clusters at the row headers. The table cells describe the concrete requirement of an originator to a requirement cluster. In the following, we describe these requirements in more detail. Starting the named five cluster, we decompose them to derive the final list of requirements for our simulator. This list does not raise a claim of completeness but rather serves as a guideline to design our simulator in Section 4.

2.3.1. Vehicle

Chassis. The most important requirement of vehicle simulation model is issued by the simulator scope. Here we need to realize the tradeoff between mapping physics realistically and reducing the computational effort to gain performance. This can achieved in the following way:

(i) Disregard effects like torsion of the chassis which have minimal impact on the vehicle dynamics within our scope.

(ii) Disregard deformation of the chassis due to collisions, which allows us to model the vehicles as simple rigid bodies.

(iii) Vehicle kinematic needs to be modelled more precisely, as required by the PACE application for the correct evolution of the absolute vehicle position.

Sensors and Actuators. For the interaction of the applications with the vehicle model, actuators for acceleration, brake, and steering are needed. Sensors are necessary to enable the applications to sense the state of the vehicle and its environment. Modelling sensors can be extremely complex and computationally expensive. However, within our scope,

Table 1: Requirements.

	Application	Developer	Scope
Vehicle	Actuators, sensors	Parameterize models and sensors, validation	Tradeoff between precision and computational effort
Environment	Perception by sensors	Scenario definition (quick)	Simple models to address computational effort
Driver	Interaction by actuators	Define behavior	-
Communication	Short range V2X	Parameterize	Simple models to address computational effort
Architecture	Deployment	Open interfaces, iterative prototyping process, visualization, time, repeatability	Distribution to increase performance

the following set of few sensors having simple models with few parameters should be sufficient for the majority of cooperative ADAS applications, while offering developers the possibility of hooking up further custom sensors.

(i) Frontal sensor like radar [1, 3] as needed by CACC

(ii) Side sensors like a blind spot detection system [1] as needed by CELC

(iii) Lane detection sensor like a stereo camera [1] as needed by PACE for precise navigation

Parametrization. All relevant parameters of the vehicle model regarding dynamics and kinematic, as well as the parameters defining the sensor models, need to be adaptable by developers. Changing these parameters during prototyping cooperative ADAS is a highly recurrent, iterative process and needs to be designed in a way to enable very short cycles thus. For this purpose, at this point we identify the definition of such an *iterative prototyping process* as an additional separate requirement. Parametrizing vehicle models regarding vehicle dynamics requires validation, for example, to match simulation to a certain real world vehicle. Here we identify another separate requirement, a *vehicle dynamics validation method* needed to validate the vehicle model and its parameterization against real physics.

2.3.2. Environment. The environment of the vehicles in our simulator needs to contain objects that can be perceived by sensors and infrastructural elements that the vehicles can interact with. This includes static objects like street furniture (such as traffic lights) and moving objects (like road users), as, for example, needed by CELC. For the majority of research done on cooperative ADAS within our scope, the following environmental features should be sufficient:

(i) Simulation road infrastructure for CACC and CELC can be generated from simple multilane road segments, while applications like PACE require complex street grids including at least the mapping of basic traffic rules and obstructing objects like pillars and walls.

(ii) Environmental objects should be representable by low computationally demanding simple geometrics, as already motivated in the context of sensor models.

(iii) Properties of objects and infrastructure influencing vehicle dynamics and perception need to be parameterizable, for example, the friction of road surface or weather conditions.

(iv) Location and time-bound triggers are needed; for instance, to simulate CELC an obstacle has to be moved in front of the vehicle very closely.

(v) Scriptable procedures are required, like braking events in certain situations for CACC.

Generating and iteratively modifying scenarios including complex road infrastructures as described, as well as its parametrization and scripting, are a time-consuming job for developers. Thus, at this point, we identify the need for a *scenario definition* for our simulator as an additional separate requirement, which enables rapid prototyping in a very time effective way.

2.3.3. Driver. A driver moving a vehicle in our simulator is needed first as input for a cooperative ADAS application deployed on this vehicle and second to generate surrounding traffic for sensor perception. In that way, certain situations can be created as, for instance, required for simulating CELC. In order to imitate driver behavior, the basic tool fitting our scope is speed-annotated routes [5] to be defined by developers. Defining such routes, however, may become more and more time-consuming with a growing number of vehicles and larger simulation scenarios as, for example, needed for PACE. In order to reduce the effort for developers, in addition to speed annotations, a set of basic driver behaviors and related features should be provided by our simulator, such as the following:

(i) Autonomous, microscopic driver behaviors regarding traffic [26], including, stopping in front of red traffic lights, giving way, regarding speed limits, and avoiding collisions with other vehicles and objects

(ii) Scripted maneuvers defined by developers as part of the *scenario definition* like speed changes and braking maneuvers that can be triggered by the environment, as required by CACC

(iii) Parameterization of the maneuvers that need to be varied while prototyping as the driver reaction time and randomized behavior (e.g., swaying in the lane)

(iv) Different yet reproducible random seeds to guarantee repeatability

2.3.4. Communication. As an integral aspect of cooperative ADAS, communication between vehicles and infrastructure needs to be mapped by our simulator. The applications within our scope (vehicle control and cooperation) naturally focus on short range real-time communication, like required by CELC. Communication with such focus is enabled by single-hop V2X communication which allows us to disregard complex multihop propagation models [12, 27]. Additionally, cellular communication might be required to integrate remote information source. This could, for instance, be required to include external traffic information in CACC, like positions of occurring traffic jams or constructions sites ahead.

Similar to the approach of modelling sensors, our simulator should provide simple set models for V2X and cellular communication [27] including the following parameters for developers to vary: delay, packet lost, range, and bandwidth. Again we provide the ability to hook up further custom models. To provide an example, researching CACC for very large platoons could be required, which might result in the demand for mapping multihop communication or packet collision due to many communication nodes being involved.

2.3.5. Architecture. In order to maximize the number of vehicles that can be part of a simulation of cooperative ADAS, the simulation models are to be designed to reduce their demand on processing power. The most important requirement to the architecture of our simulator is to maximize the processing power to be available for simulation. This can be achieved by designing the architecture to be able to scale the simulation over multiple instances running different machines. From the perspective of developers, the architecture should further enable the following:

(i) Visualization of the running simulation

(ii) Providing an open interface to hook up the applications instead of forcing a certain technology to this end, for example, MATLAB®/Simulink®

(iii) Supporting the usage of the iterative process mentioned as separate requirement and in this context a time effective parameterization of the simulation

2.3.6. Scenario. The scenario definition should contain all relevant parameters of the simulation model (vehicle, sensors, and communication), the parameter setting of the applications, and the definition of the environment and the definition of driver input for a simulation. Thus, the scenario should contain all the information that developers need to specify and vary with regard to the iterative prototyping process which we already identified as a separate requirement.

2.3.7. Process. As already outlined in the preceding requirements, for prototyping cooperative ADAS, a process is needed to support developers handling the application-specific part of the related parameter space. The design of all aspects of our simulator should be aligned with such a process, which brings an idea of a cooperative ADAS through an iterative process of refining and assessment towards a plausible candidate for implementation. Thus, prior to designing the simulator, we first need to define this iterative process.

2.3.8. Vehicle Dynamics Validation Method. As we identified while defining the requirements of the vehicle simulation model, parametrizing vehicle models in terms of vehicle dynamics requires a method for validation. Such a method is also required to validate the vehicle model of our simulator and its parameterization done by developers against real physics. Aligned with the paradigm of our iterative process and the scope of our simulator, this validation method should focus on balancing the degree of validity and the time developers needed to invest for validation.

3. Related Work

In order to realize a prototyping environment for cooperative ADAS meeting our requirements derived in the previous chapter, we have reviewed existing simulators in the field of ADAS research. In the following, we give an overview of the work in this field and describe the most relevant simulators with respect to their suitability to meet our requirements. The simulation tools relevant for cooperative ADAS research can be divided into the following four groups.

3.1. Traffic Simulators. First, there are traffic simulation tools. Among these traffic simulators, which can be distinguished between macro-, meso-, and microscopic modelling approaches [26], SUMO [26] and VISSIM [28] are two popular representatives of the microscopic approach. The aim of such microscopic traffic simulators is to model traffic flows with a focus on the impact of chances of the infrastructure or of traffic influencing applications to the traffic. Each vehicle taking part in simulations is modelled as an individual object, while its movement in the traffic is mapped by no collision vehicle following models like Krauss, Wiedemann, or IDM [26]. Although each vehicle is modelled, vehicle following models are oriented on mapping the traffic flow as accurately as possible. For this purpose, a high number of vehicles need to be simulated, which implies for performance reasons to almost disregard vehicle dynamics and apply a collision-free movement by always keeping a minimum safety distance. For this reason, microscopic traffic simulators cannot meet our requirements and we therefore do not go into a more detailed description.

3.2. Driving Simulators. The second group of simulators is dedicated to the involvement of human drivers into the simulation. For this group, OpenDS [29], OSS [30], and SILAB [31] are popular examples. Such driving simulators are used for testing or researching ADAS from a perspective of interaction with human drivers, which makes realistic scene visualization important. Although most driving simulators are built upon submicroscopic vehicle models, the modelling is mostly focused on a driving behavior that feels realistic from the perspective of the driver instead of mapping realistic physics. This is due to the missing inertia effect feedback for the driver in a pure software simulation. There are also driving simulators which do emulate such inertia effects by utilizing motion platforms, like the NHTSA [32]. For this simulator, mapping real physics has been implemented; however the necessary hardware installation makes it quite expensive and so it does not scale for more than one vehicle. Expensive hardware brings it out of scope for our requirements and leads to the next group of simulators.

3.3. Real World Vehicle Simulations. The third group include real vehicles in simulation. This group of vehicles in the loop simulators belongs to testing systems that leverage robot controlled automated driving of series production vehicles for testing ADAS in the last phase of their development cycle [4], especially active safety systems. In this way, test and verification procedures, which involve dangerous situations, can be performed without endangering human drivers and with a high precision in repeatability. Thus, maneuvers can be tested precisely at the threshold between “system must react” and “system should not react,” endless times. Such testing systems are very expensive, as in addition to the vehicles to be tested, a large overhead of equipment, testing areal, and testing personnel is needed. This overhead grows with each more vehicle taking part in the tests. A middle way here is the VEHIL simulator [33], which involves a real world ego vehicle deployed on a fixed location and dummy vehicles surrounding the ego vehicle, emulating traffic participants. This group of simulators mainly belongs to the final development phase of ADAS and does not fit our requirements for a prototyping environment, in terms of cost and scale.

3.4. Vehicle Dynamics Simulators. The fourth group of simulation tools refers to the vehicle simulators. These simulators are most relevant group for realizing the requirements of our prototyping environment. Similar to the driving simulators, they are built upon submicroscopic simulation models; however the focus here is on vehicle dynamics and realistic mapping of physics. When designing our prototyping environment, many simulators have been reviewed. In the following, we give a more detailed look into some representatives of this group, which we consider as the closest candidates to fulfill our requirements.

3.4.1. SiVIC. SiVIC [34] is a simulation software developed at INRETS, commercialized by the CIVITEC sarl. SiVIC has physics based vehicle model which addresses realistic mapping of vehicle dynamics [34]. However, it is hard to find information if the vehicle dynamics model has been validated against real world vehicles. The focus is put on sensor models and on simulating the behavior of the vehicle and its embedded sensors to reproduce the reality of a situation from the perspective of an ADAS [34]. A lot of work has been done on developing very detailed sensor models like radar, lidar, and camera [35]. The latter is a reason for the very accurate rendering engine, which dedicatedly aims at accurate shadowing and lighting computation and a very detailed visual mapping of the simulation. One of the dedicated assets of SiVIC is its library of sensor models. As interface between the simulation and applications, RTMaps is used, through which other development software like MATLAB/Simulink can be coupled. SiVIC is built upon a distributed architecture, which allows for deployment of the simulator over multiple computers and by providing V2X sensors simulation, cooperative ADAS development is also addressed [34]. The initial objective of the SiVIC platform was to prototype local perception applications. Later, several extensions have been made to this sensor simulation platform to allow the virtual prototyping of new hardware in the loop and software-in-the-loop applications. This makes SiVIC an unattractive candidate for our prototyping environment, as for our requirements, the focus is not on sensor models or hardware-in-the-loop simulations, but on combining all prerequisites modelled on a level which enables tuning large application related parameter spaces for development of cooperative ADAS. And we consider an implementation basis aiming at the final goal parameter tuning to be crucial in order to meet the performance needs issued by the requirement of involving as many vehicles as possible in the simulation. Although sensor input is of course of importance for cooperative ADAS, our prototyping environment is meant to help developers more on designing cooperative patterns instead of interpreting sensor input. For this reason, we will rely on basic sensor models and provide open interfaces to integrate existing models widely available, finally, since SiVIC is a commercial product, which is necessary to finance the development of the named features that are not important for our requirements.

3.4.2. PreScan. PreScan is a simulation platform initially developed by TNO and commercialized by TASSInternational, which is used in the automotive industry for development of ADAS based on sensor and V2X communication technologies as well as autonomous driving applications. PreScan claims to be based on real physics [33] and can be used for model-based controller design (MIL) to real-time tests with software-in-the-loop (SIL) and hardware-in-the-loop (HIL) systems, the latter in combination with the earlier mentioned VEHIL. In contrast to SiVIC, description of features and interfaces is at least on an informational level available on a public website. According to [34], for SiVIC, PreScan is the only comparative sensors simulation platform available, but it mainly provides simple sensor models for an easy traffic management and ADAS prototyping but not enough realistic for control/command applications. However, PreScan also provides sensor models for radar, lidar, camera, and so forth and there is even a cooperation with LIDAR manufacturer Ibeo in the field of automated scenario generation based on laser scanner technology. PreScan’s sensor models and its vehicle dynamic model are exchangeable, so models already validated with other simulators can be linked in. Interfaces exist for MATLAB/Simulink, Java, CarSim, veDYNA, dSPACE, and National Instruments. Cooperation between ADAS is also addressed, by providing V2X communication sensors.

Using PreScan is a process divided into four steps [34]: build scenario, model sensors, add control system, and run experiment. Once this is done, typically, 1000 scenarios can be run within a couple of hours in an automated way. This whole process is on a first glance a promising approach to meet the requirements of our prototyping environment. However, this process of using PreScan regarding MIL, SIL, HIL, and its orientation on the V-Model [33] indicates that, in the end, the focus is on test and verification, not on an iterative agile approach as we need it regarding tuning large application related parameter spaces. This is underlined by the fact that TASSInternational has announced a cooperation with CETECOM to provide a comprehensive validation service of cooperative driving and connected vehicle technologies on functional and communication level. The orientation of PreScan might be well suited for in-depth analysis of existing ADAS but is very extensive tool which makes it too time-consuming and computational expansive for the prototyping environment we are aiming at.

3.4.3. Torcs. In contrast to the commercial simulators already described, there are some open source simulators (among them RACER [36], TORCS [37], and VDrift [38]) which we have considered to be the basis for our prototyping environment. Most of them do not differ much from each other regarding our requirements. However, for us the most relevant among them is TORCS, as it is most widely used in research and the best documented one. Thus, in the following we describe TORCS in more detail.

TORCS is a free, open source car racing simulation implemented in C++, available for Linux, FreeBSD, Mac OS X, and Microsoft Windows. TORCS is designed for the development of racing bots, human players can also control vehicles with a steering wheel and pedals. It has been used before for similar use cases like our prototyping environment, for instance, as input source of vehicle dynamics data for a test bed for ECU testing and verification [39]. The vehicle dynamics model has not been validated against real world vehicles. However, the underlying engine models tire dynamics are based on the Pacejka tire model [22], accessible engine related factors, for example, torque, rpm, and the suspension, gearbox, and variable road conditions. Thus, the models cover most relevant aspects and can adapted as required, so prerequisites for achieving realistic driving dynamics are present. Sensor models need to be added; a basic implementation of ideal and noisy front sensors for objects and tracks is available. V2X communication needs to be added and distribution among multiple computers exists very rudimentary by a system that was actually designed for online competitions. The scenario creation is oriented on designing racetracks but provides a solid basis for our requirements.

Although SiVIC and PreScan are powerful and mature simulation tools and despite a lot of missing features of TORCS as described, after thorough investigation we considered TORCS as the most promising candidate as basis for our prototyping environment. The most relevant fact is that TORCS is open source and can be extended freely, which has more weight than, for example, existing sensor models and nice looking visualization engines. Nevertheless, this approach of course implies that a decision to implement a new simulator from scratch instead of using an existing one has already been made.

3.5. Conclusion. Existing simulators in the field of ADAS research can be divided into four groups, while the group of vehicle simulators regarding vehicle dynamic are the only relevant group to be considered as basis for our prototyping environment. The commercial tools among them, as described with the examples SiVIC and PreScan, bring a lot of powerful features, but regarding the requirements derived in the previous section, these tools bring also a lot of overhead of features, like sophisticated sensor models. This entails a great complexity, which negatively affects the goal of having a lightweight tool for rapid prototyping the class of applications in the field of cooperative ADAS, as described earlier. Besides that, adding features to these commercial tools, which necessary from our perspective, is difficult due to the closed source code and limited interfaces. Indeed, among the open source simulators, TORCS is the most promising candidate as a basis for our prototyping environment. Although a lot of requirements need to be added, it brings a decent physics and vehicle dynamics implementation where we could start from. However, after a deep analysis of the implementation work needed to be done on TORCS, we eventually decided setting up a new implementation and porting the relevant aspects from TORCS. This new implementation, analyzed as a whole, saves implementation work for future work and gives us the opportunity to create a new dedicated architecture with a Java instead of C++, which is better suited for addressing our requirement distribution on different platforms.

4. The Simulator

In this section, we present our simulator, the prototyping framework for cooperative ADAS as derived from the scope defined in Section 2.2. As outlined in the scope, our simulator does not intend to compete with specific detailed vehicle dynamic simulators. The specific needs for prototyping cooperative ADAS are addressed instead. In order not to restrict developers too much in case of a specific need for a higher level of precision of modelling arises, the architecture of our simulator is designed to allow extending and exchanging each model.

In the following we present the design of the different aspects of the simulator according to the technical requirements defined in Section 2.3. In the context of this paper, we focus on the key design aspects to handle large parameter spaces, that is, the tradeoff between modelling precision and executing performance, and the iterative prototyping process. We briefly address essentials of all other aspects in a summarizing manner. We start with the design of the iterative prototyping process for developers to be supported by the simulator, since all other aspects of the simulator are aligned with this process. Subsequently, we describe the vehicle models and the simulator architecture in detail while environment, driver, scenario definition, and the simulator reference implementation are described briefly. The novelties here are not the vehicle models and their related equations

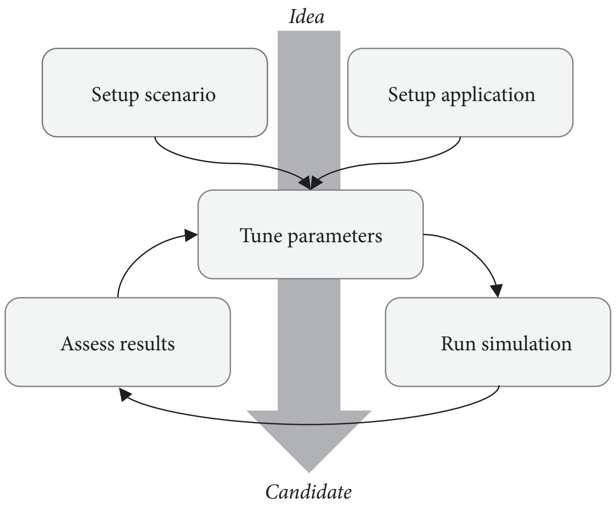


Figure 4: Iterative prototyping process.

themselves, but their combination aiming at the tradeoff between opposing requirements as described in Section 2. We devote a separate section to the vehicle dynamics validation method defined as the final requirement.

4.1. Prototyping Process. We propose the following process to bring an idea of a cooperative ADAS through an iterative process of refining and assessment towards a plausible candidate for further development. Aligned with this process, developers can use our simulator in a trial and error manner to create, refine, and assess. In this way, the idea of a cooperative ADAS can either be brought incrementally through the stage of prototyping or be discarded as quickly as possible.

The process is depicted in Figure 4. Starting with the implementation of the cooperative ADAS application to be prototyped, developers define a set of working parameters of the application. An initial set of these parameters needs to be given by developers in the first step. The same applies to the simulation scenario and its parameters. Subsequently, the process iteratively traverses an arbitrary number of cycles including the three steps, running the simulations, assessing its results, and tuning the parameters of scenario and application. This cycle is completed as soon as the simulation results suggest that the cooperative ADAS application under research is realizable and has an effect as expected. Otherwise, the cycle is to be terminated after a number of cycles high enough without any progress on the expected results, so the idea needs to be either reconsidered or discarded.

In order to reach either of both verdicts as fast as possible, the cycle time needs to be minimized. Accordingly, the goal of the simulator design is to minimize the execution time of one cycle. The execution time of the simulation run is significantly depending on the preferably low complexity of the simulation models (vehicle, environment, and communication) and the scalability of the simulator architecture. Tuning the parameters of application and the scenario by developers is highly depending on a pragmatic scenario description of the simulator. These aspects will be object of the following subsections.

4.2. Architecture. Figure 5 depicts a high level view of the architecture of our simulator consisting of two parts, the *Developer Implementation* containing the components developers need to define and implement and the *Simulation Framework*. The latter contains multiple *Simulation Instances*, which can be distributed over multiple machines, and the *Simulation Main Instance*, which is coupled to the *Visualizer*. This architecture, its components, and its underlying paradigms are described in the following.

As defined in the Section 2.3, the main requirement of the architecture of our simulator is to maximize the computational power that can be used for execution. For this purpose, the simulation should be able to scale-out [40] rather than scale-up, which requires the architecture to decouple the simulation in space. The requirements of the architecture issued by developers rather aim at extending/exchanging simulation models, on visualization, and on open interfaces to hook up applications and models. This requires the architecture to decouple the simulation from theses aspects, which we will further refer to as decoupling in complexity. To support the iterative prototyping process, the architecture also needs to address the time aspect, which we will further refer to as decoupling in time. In the following, before designing the architecture, we will elaborate the three dimensions of decoupling in more detail.

4.2.1. Decoupling Simulation in Time. The time aspect to be regarded by the architecture is issued by the requirements, which derive from the iterative prototyping process. At some cycles of the process, the developer will face the need to observe the simulations online very detailed in slow motion or with a high resolution of single frames, either chart-based or using a rendered visualization of the simulation scenario. For this purpose, an arbitrary small simulation step time is required to enhance the precision of the simulation and its observable output. At other cycles in the prototyping process, the level of precision and online observation might be less relevant than the number of parameterizations and simulation runs. This case rather requires to increase the simulation step time, to run simulations faster than real time, automated, and without visualization. These considerations lead to the conclusion, to design our architecture to support both a fixed simulation step size that results in variable execution times and a fixed simulation time that results in a variable simulation step size. In both cases, the variable part depends on the complexity of the simulation and the available performance of the executing machine.

4.2.2. Decoupling Simulation in Space. Decoupling in space refers to the idea of splitting the simulation scenario area in different cohesive subareas to enable distributing the execution of simulation models at different *simulation instances* (see Figure 5), that is, on multiple machines. Such splitting is required to cluster the vehicles in a scenario according to their potential interaction radius. Vehicles that interact physically

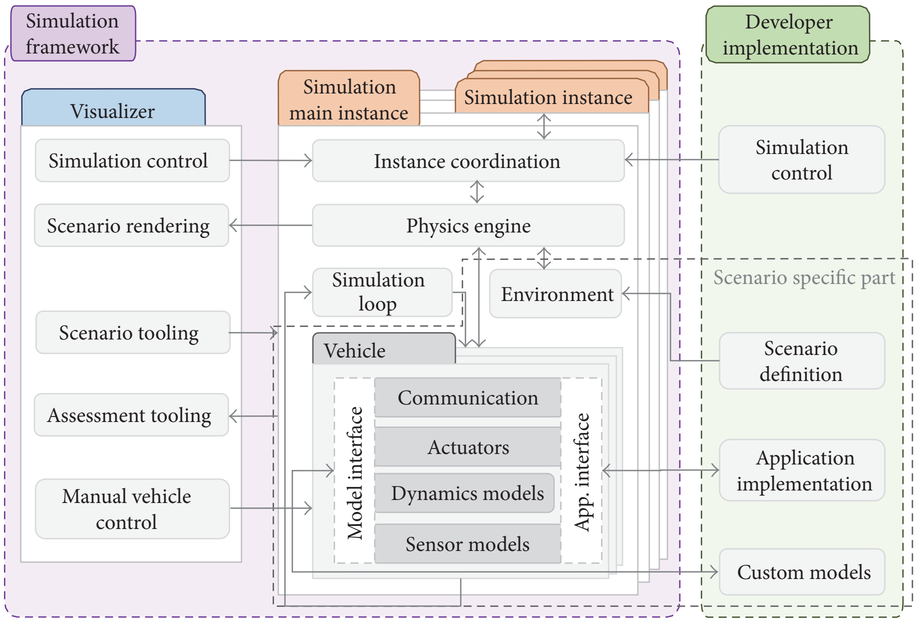


Figure 5: Architecture.

should be simulated on the same machine, that is, if they are, for instance, recognizably mutually by sensors or can even collide with each other. Otherwise the synchronization effort for the underlying physics engine would be too high, as the full physics state of the vehicles had to be synchronized between different machines with each simulation step, by the *instance coordination* component (see Figure 5).

For decoupling in space we propose a fixed discretization the area of simulation scenario using a Discrete Global Grid System (DGGS) as pursued by the DGGS Standards Working Group [41]. For our architecture, the minimum cell size of the grid needs to span at least a multiple of the mentioned physical interaction radius (PIR) of the vehicles. Cohesive areas covered by the cells need to be allocated to different machines for simulation, to cover the whole area of the scenario as exemplary depicted in Figure 6. In this example a hexagonal DGGS is split in three areas (orange, green, and blue) allocated to three different machines. At all cells at the border between two areas, the physics simulation is done by both machines synchronously. For these overlapping cells, one machine is the master (indicated by the background color) which determines the state of the vehicle actuators for all vehicles in one cell. The master executes the vehicle model and thereby determines the forces to be applied to the vehicle body. These forces are synchronized with the other machine, the slave (indicated by the border color), which only does the physics simulation using the position and the forces. The scenario of Figure 6 accordingly requires one cell to have two slaves, as this cell in the center of the scenario borders three cohesive areas. When a vehicle in a slave cell gets within the PIR to a master cell at the same machine, the former master needs to hand over the full state of all sensors, actors, and vehicle submodels to the new master. In this way the communicated information for synchronization can be reduced to three vectors, position, orientation, and forces applied to the vehicle.

4.2.3. Decoupling Simulation in Complexity. Decoupling in complexity refers to designing our architecture to run several parts of the simulation remotely and allow custom implementation of these parts by developers by providing open interfaces. Implementing these interfaces the respective part can either be deployed to the simulator or run remotely using an execution environment of developer's choice. Issued by the requirements, these parts include the following:

(i) Cooperative ADAS application: the interfaces to be implemented are sensor input and actuator output (Figure 5).

(ii) Simulation models (sensors, vehicle, and communication): depending on the complexity of a simulation model, developers can decide whether the computational effort for execution or the communication overhead for synchronization of the simulator with the model state remotely is more relevant in each case (Figure 5).

(iii) Visualization: decoupling visualization from the simulation is necessary to enable developers to run headless simulations on a remote server and visualize it locally. However, a complete decoupling would require to synchronize all visually relevant object information from all simulation instances (see Section 4.2.2) to the visualizing component. This approach demands a high load of network traffic (assuming a frame rate of above 25 Hz for a smooth object transition). We reduce this traffic by applying the decoupling in space approach (see Section 4.2.2). The visualization is coupled to one local *simulation*

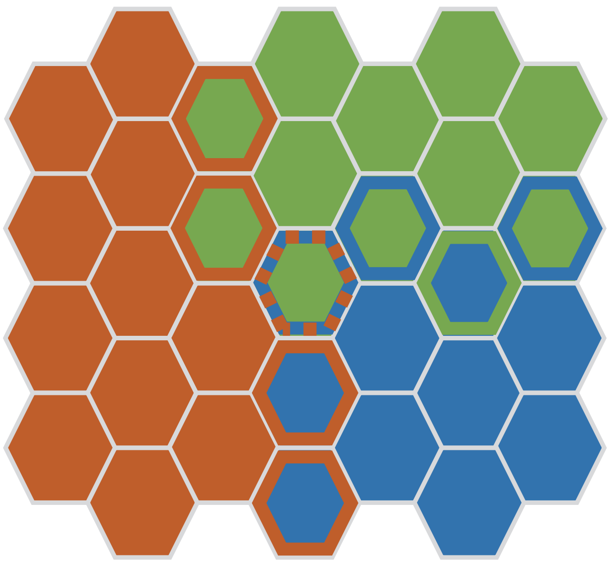


Figure 6: Physics simulation scale-out enabled by DGGS.

instance, which runs a dynamic area of cohesive cells depending on the current view port of the visualization. In that way, the synchronization rate with other instances can be reduced to the simulation step size while the visualization is interpolated by the *physics engine* of the local *simulation instance* (Figure 5).

The vehicle model describes the evolution of the vehicle in the simulation influenced by physics. The vehicle body in the simulation receives forces by interaction with the environment (other objects, gravity, or the air) and by its powertrain, beginning at the engine with the generation of power and finally delivering it to the road surface. The simulation of physics can be arbitrarily complex depending on the envisioned level of accuracy. Due to this complexity, and the related computational effort, the vehicle model is the most important part for the design of our simulator. The vehicle model is the crucial point in the simulator design for balancing the two requirements, mapping real physics, and scaling the number of vehicles in the simulation, since both have an impact on the computational expense. For the sake of this balance (see the scope in Section 2.2), we have chosen to design the vehicle body as a solid without deformation, interacting with its environment based on rigid body dynamics [42], which is sufficient for us as earlier described in the requirements in Section 2.3. In the following, we describe in detail the aspects of the vehicle model used for our simulator. The decisions we made on each aspect were an iterative process on finding the balance of keeping complexity as low as possible and still meeting our requirements, which was finally validated as described in Section 5.

4.3. Vehicle Body. The vehicle body receives external forces and moments following the three axes (longitudinal, lateral, and vertical) coming from interaction of the wheels with the road, interaction with the environment, and from the vehicle power train. In our simulator, the vehicles are considered as solids concerning the interaction of objects, that is, terms of collisions determination between objects. The evolution of the vehicle objects is described in accordance with the dual track model. The dual track model described in [20] offers the following features:

Table 2: Degrees of freedom of the dual track model.

Name in [20]	Name in this work	Description
x_v, y_v, z_v	$p(t)$	Global coordinates of the center of mass
$\psi_v, \theta_v, \varphi_v$	$r(t)$	Vehicle orientation in the global coordinates
$\rho_{Ri}, i = 1 \cdots 4$	$\omega_{w,j}$	Rotation of wheel $i(j)$ around its rotation axes
z_{Ri} $i = 1 \cdots 4$	x_j	Vertical movement of wheel $i(j)$
δ_H	$\delta_{i,H}, \delta_{o,H}$	Steer angle (H), inner (i, H) and outer (o, H)

(i) The vehicle body is modelled as a single rigid body with uniform mass distribution.

(ii) Each wheel has an individual position relative to the vehicle's center of mass.

(iii) Wheel/road interaction is modelled with a separate wheel model.

(iv) The vehicle suspension is simplified: the suspension for every wheel has only a single degree of freedom in vertical direction relative to the vehicle body.

(v) The steer angles of the front wheels are assumed to equal, which is for our model replaced with the Ackerman steer angle [20].

In total the dual track model describes 15 degrees of freedom, as shown in Figure 7 and listed in Table 2. The forces applied to the vehicle according to these degrees of freedom are generated by the subsystems of the vehicle model (suspension, braking system, power train, and aerodynamics). These subsystems and the resulting forces to be applied are described in the following subsections.

4.4. Wheel Model. For the driving wheels (the wheel at the driven axle), the dynamic equation for the wheel rotational dynamics can be described by (1). The inertia $J_{w,j}$ of the wheel x is driven by the wheel torque $T_{w,j}$ provided by the power train. For the driven wheels, the inertia of the transmission and the engine increases $J_{w,j}$. $T_{w,j}$ is reduced by the brake torque $T_{b,j}$ and the tractive force $F_{t,j}$ acting on the wheel radius r. In that way, the rotational wheel speed $\omega_{w,j}$ is changed:

$$J_{w,j}\dot{\omega}_{w,j} = T_{w,j} - T_{b,j} - F_{t,j} \cdot r. \tag{1}$$

The tractive force at each wheel is determined by the TNO MF-Tyre 5.2 tire model [43] introduced by Hans B. Pacejka in [22], an empirical model that simulates the wheel friction based on a large set of parameters measured for a set of

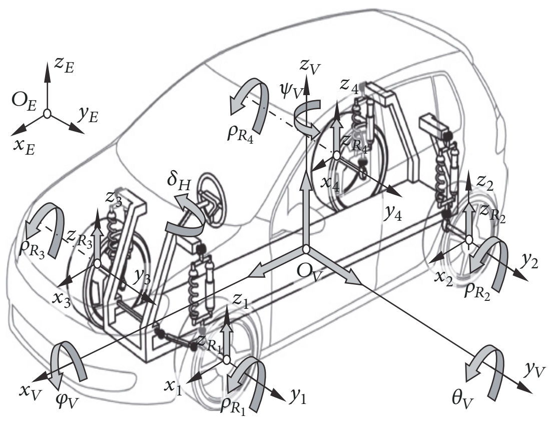


Figure 7: Degrees of freedom of the dual track model [20].

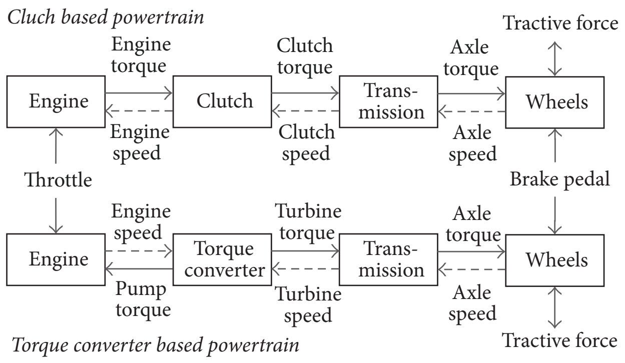


Figure 8: Powertrain.

real tires. The default parameters are taken from [44] and adapted to specific tires using a suitable tool (ADAMS/Tire in our case), using the method proposed in [44]. The tire model is of high importance in terms of physically realistic simulation, as it models the contact between the vehicle and the road surface. It thereby turns all forces generated by the vehicle into motion with regard to the external forces acting on its body. Thus, the tire model cannot be simplified too much. Pacejka's model fits our needs at this point, as it provides a decent tradeoff between realism and complexity. The brush model [22] and the Burckhardt model [45] are two alternatives for tire modelling we have reviewed and discarded, as we were not able to find parameter settings which lead to a sufficient level of validity for our desired span of driving dynamics (see Section 5).

As proposed in [46] the brake torque $T_{b,j}$ on each wheel j is model according to (2) as a function of the main brake cylinder pressure P_m with proportional distribution factor k_p between rear axle ($T_{b,r}$) and front axle ($T_{b,f}$) and a constant brake gain k_b. The pressure is function (3) of the brake actuator position $P_a[0\cdots 1]$ with a first-order system plus a constant time delay τ_d, which models the actuator delay. According to the method described in [46] the parameters K, T, and τ_d can be obtained from experimental data.

$$\begin{aligned} T_{b,r} &= P_m k_b k_p, \\ T_{b,f} &= P_m k_b \left(1 - k_p\right), \end{aligned} \tag{2}$$

$$K \cdot P_a\left(t - \tau_d\right) = T \cdot \dot{P}_m(t) + P_m(t). \tag{3}$$

4.5. Suspension. The suspension model we use for our simulator induces a simple spring damper system with one degree of freedom along the vertical axis of each wheel (see also Section 4.3). This approach is similar to the full car suspension model in [21], except for the separate modelling of the wheel stiffness. Suspension systems found in real vehicles offer a more complex nonlinear damping behavior and more degrees of freedom as variable camber angles [20]. The main purpose of the vehicle suspension is to maintain the contact between wheel and road for varying road conditions including holes and bumps. For our simulator we observed through validation (see Section 5) that such a simple spring damper system is sufficient to meet our requirements. The wheel load F_z (the force acting perpendicular to the road) is modelled by a simple spring damper system, defined by two forces: the oscillatory force F_s and the damping force F_{damp}.

$$F_z = F_s - F_{\text{damp}}, \quad F_s = -kx, \; F_{\text{damp}} = -c\dot{x}. \tag{4}$$

F_s changes the length of the spring x, in our case the distance between the attachment point of the wheel suspension at the vehicle body and the wheel axle, about a certain amount. The damping force acts against any change of the spring length. The system is defined by two the coefficients, stiffness (spring constant) k and the damping coefficient c. If the vehicle stands still, the system will rest in an equilibrium, where F_{damp} is zero and F_s equals 0.25 times the gravity force (for four wheel vehicles). For an attached mass m (in case of a vehicle suspension this mass is the wheel), the damping characteristic ζ of the system is given by

$$\zeta = \frac{c}{2\sqrt{mk}}. \tag{5}$$

The damping characteristic can be divided into three categories: overdamped ($\zeta > 1$), underdamped ($\zeta < 1$), and critically damped ($\zeta = 1$).

4.6. Powertrain. The powertrain is a cascade of components beginning at the engine with the generation of torque, delivering it to the wheels, and describing the changes of torque to rotation speed. For our simulator we describe the model of the powertrain in accordance with [21] as a composition of four the submodels engine, torque converter, transmission, and wheel model. Since we also consider vehicle models with semiautomatic transmissions, the torque converter model in our simulator can be replaced with an automated clutch model. Figure 8 depicts the interaction between these four submodels and the brake.

4.6.1. Torque Converter/Clutch. In vehicles with automatic transmissions, the torque converter transfers the rotating power generated by the engine to the transmission. The torque converter normally is a fluid coupling, which decouples the engine from the load of the vehicle. This enables the engine to run in idle speed when the vehicle is stopped, while, at high speed difference between transmission and engine, the torque converter multiplies the torque of the engine transmitted to the transmission due to this decoupling. In vehicles with semiautomatic transmissions, the coupling between engine and transmission is realized with a clutch, which is automatically operated by actuators and therefore removes the need for a clutch pedal that is normally operated by drivers in manually shifted vehicles. Both options, torque converter and clutch, are modelled in our simulator.

The torque converter is modelled according to the static model of Kotwicki [21] because of its simplicity. We have observed a sufficient agreement with experimental data for the range of operating conditions relevant for the focus of our simulator. The model is a quadratic regression from a simple experiment of measuring the input and output speeds and torques of the torque converter obtaining the coefficients a_1, a_2, a_3, b_1, b_2, b_3. The torque at the engine side (pump torque) T_p and the torque at the transmission input side (turbine torque) T_t as functions of the engine speed ω_e and the transmission input speed ω_t are written as

$$\begin{aligned} T_p &= a_0 {\omega_e}^2 + a_1 \omega_e \omega_t + a_2 {\omega_t}^2 \\ T_t &= b_0 {\omega_e}^2 + b_1 \omega_e \omega_t + b_2 {\omega_t}^2. \end{aligned} \tag{6}$$

The clutch model also decouples speed and torque on the transmission site (connected to the clutch disk) from the engine site (connected to the flywheel disk). There are three states of the clutch, open, slipping, and closed, which can be modelled independently. For our simulator, the open and closed are modelled simply by transferring full and no torque; however the slipping state needs a more precise modelling. As described in [47], the slipping state, to be accurate, should be modelled regarding nonlinear characteristics of the clutch including changing friction coefficients depending on temperature, pressure of the disks, and the dynamics of the clutch actuator. For the requirements of our simulator, a sufficient accurate model for the clutch in slipping state that can be written as

$$J_e \dot{\omega}_e = T_e - T_t (xc) . \tag{7}$$

The derivative of the angular speed of the engine is $\dot{\omega}_e$. The engine inertia J_e is driven by the engine torque T_e on the engine side of the clutch and is loaded by the clutch torque T_t on the transmission side of the clutch. The clutch torque is modelled as a function of the clutch actuator xc, the so-called clutch transmissibility curve. The nonlinear characteristics of this curve are identified from the experimental tests. When the clutch is open, T_t is zero; when the clutch is closed, T_t is the full torque load of the vehicle. For our model, we were able to achieve satisfactory results (see Section 5) using a linear curve of $0 \cdots 1$, for fixed time intervals (measured from experiments) of clutching and declutching in each gear.

4.6.2. Transmission. The transmission translates the output torque of the clutch/torque converter according to the current gear. The transmission output is forwarded to the wheel model, where the updated wheel speed is fed back through transmission and clutch/torque converter to the engine model to update the engine speed. Additionally, the transmission lowers the output torque by introducing a torque loss, which depends on the modelled vehicle and the external forces acting on the vehicle, proportioned by the ratio of the current selected gear. In our model, the transmission also covers the differential, which distributes the transmission output torque to the drive shafts of each driven wheel and the propeller shaft (in case of rear-wheel drive). The translation of the torque T_t and angular speed ω_t on the clutch/torque converter side of the transmission to the side of the driven wheels T_w and ω_w is defined by (8) considering the ration R, the transmission inertia J_g of the gear i, and the differential ratio R_d. The torque is split to drive the left wheel l and the right wheel r according to (9), while their resulting angular speed is averaged to determine the angular speed of the driven axle.

$$\begin{aligned} T_w &= \frac{1}{R_i * R_d} T_t - T_g \mid T_g = J_{gi} \dot{\omega}_w, \\ \omega_t &= \frac{1}{R_i * R_d} \omega_w \end{aligned} \tag{8}$$

$$\begin{aligned} T_{w,r} = T_{w,l} &= \frac{T_w}{2}, \\ \frac{\omega_{w,r} + \omega_{w,l}}{2} &= \omega_w. \end{aligned} \tag{9}$$

The resulting wheel torque is forwarded to the wheel model to drive the vehicle. For positive wheel torques the vehicle accelerates and the wheel speed increases and the updated angular wheel speed is fed back to the gearbox which translates it to the updated engine speed through the clutch/torque converter.

The transmission model includes an automatic gear selection mode, which is oriented on the model for operating characteristics of automatic transmissions described in [48]. The determination of shifting gears up and down is done using a gear shifting schedule as depicted in Figure 9. The map is a diagram of engine speed in 1/s and accelerator pedal in percent. There is a separate curve for each gear to shift up and down through the diagram. The data describing these curves is determined from experiments. The same applies for the operation of the clutch actuator. For our model we determined the clutch actuator dynamics to be sufficient represented by a fixed time for shifting from and to each gear, and startup respectively.

4.6.3. Engine. The engine model in general translates a given throttle position into a drive torque generated to drive the vehicle. We consider this engine torque T_e to be limited by a maximum torque depending on the engine rotation speed ω_e. The throttle position maps the engine torque request $T_{e,r}$ on the range between minimum and maximum torque, depending on the current rotation speed. This mapping is realized using a vehicle-specific steady-state engine map as described in [46, 49]. The dynamics of the actual engine torque T_e are modelled as a function of $T_{e,r}$ using a first-order

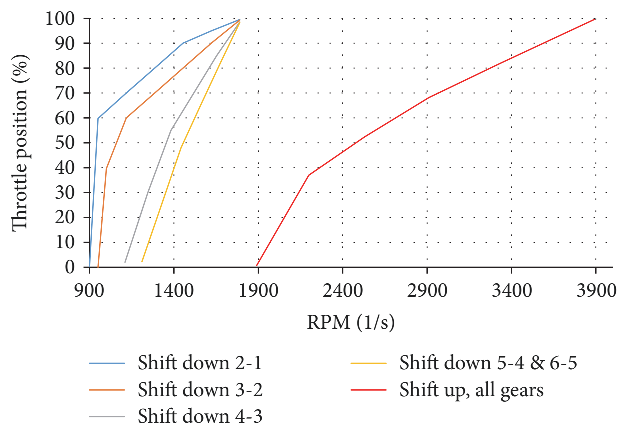


Figure 9: Gear shifting schedule.

system plus a constant time delay, following the approach of modelling the brake pressure as described in Section 4.4. This approach is inspired by the throttle actuator model in [46]. The difference here is that our approach allows enabling applications to interact with the vehicle model using the torque request as direct input. The engine inertia J_e is driven by T_e and loaded by the transmission torque T_t. The parameters K, T, and τ_d can be obtained from experimental data [46]. In case of a connected torque converter or an open clutch, the engine dynamics are described by (10). Otherwise, for a closed clutch, the engine speed is calculated as a multiple of the wheel speed.

$$J_e \dot{\omega}_e = T_e - T_p, \tag{10}$$

$$K \cdot T_{e,r}(t - \tau_d) = T \cdot \dot{T}_e(t) + T_e(t). \tag{11}$$

4.7. Aerodynamics. The drag force applied to the vehicle model by air resistance is determined by the aerodynamics model. The drag is directly velocity-dependent, in contrast to the rolling resistance of the wheels. As a result of a study on different air resistance models in automotive simulation and their impact on our vehicle model within our scope, we came to the conclusion that a simple air resistance model as used in [21] is sufficient. The model is described by (12), where ρ is the mass density of air, 1.293 kg/m^3, v is the speed of the vehicle, c_w is the drag coefficient of the vehicle, and A its frontal area.

$$F_d = \frac{1}{2} \rho v^2 c_w A. \tag{12}$$

4.8. Rigid Body Dynamics. Derived from the scope (see Section 2.2) the results of collisions between objects are not of relevance for our simulator. Hence, the only relevant information is if a collision occurs or not, which allows us to disregard the simulation results after the point in time when a collision has occurred. Assuming this, the objects being part of the simulation and their interaction can be model employing rigid body dynamics. This approach requires assuming the objects in the simulation to be rigid; that is, objects collide elastically. Thus, their motion can be determined using Newton's second law, which is less computationally expensive than regarding the deformation of objects. An introduction to the simulation of rigid bodies including collision response is given in [42]. The vehicle body is, hence, modelled as a cuboid object. The sum of all forces determined by the submodels described above and by collision with other objects is applied to the vehicle body to change its speed. According to (13), the vehicles acceleration a is proportional to the force F acting on its mass m, while its angular acceleration $\dot{\omega}$ results from force that acts on the vehicle body inertia at a point p_a different from the vehicle body center of mass p_{cm}. The current position $p(t)$ of the vehicle body and its orientation $r(t)$ are then calculated according to (14) and (15). F is the sum of the forces acting on the vehicle body (Sections 4.3–4.7):

$$\dot{\omega} = \frac{\tau}{I} \mid \tau = (p_a - p_{\text{cm}}) \times F \mid a = \frac{F}{m}, \tag{13}$$

$$p(t) = \int_0^t v(t)\, dt \mid v(t) = \int_0^t a\, dt, \tag{14}$$

$$r(t) = \int_0^t \omega(t)\, dt \mid \omega(t) = \int_0^t \dot{\omega}\, dt. \tag{15}$$

4.9. Sensors and Communication Models. The state of the vehicle and its environment is perceived using sensors. The specific quality of perception in real world is much influenced by the specific sensor. The same applies for the specific communication hardware in terms of communication. As motivated in Sections 2.2 and 2.3, from the perspective of the application to be prototyped, these specific properties are reflected in the precision of information about the vehicle and its environment. For our simulator, we consider this precision to be representable by a set of sensor and communication models and their parameters, which can be extended by developers. The most important aspect for this approach is to assure a sufficiently precise ground truth for these models to work. This ground truth is created by the vehicle and environment models, by representing the position and motion of all sensible objects and their state in the simulation. Table 3 summarizes all sensor and communication models with their parameters. For all of them, a Gaussian randomized noise, processing delay, and variation are provided.

4.10. Environment, Driver, Scenario. As mentioned earlier, in the context of this paper, we do not go into detail about the modelling of environment and driver behavior of our simulator. However, for the sake of completeness with respect the requirements derived in Section 2.3, we briefly describe the essential aspects of our design decisions towards environment and driver behavior to address simulation performance and developers interaction. The key features of the environment are as follows:

(i) The simplicity of all objects other than vehicles is modelled by simple geometrics (polygonal planes, cuboids, and tetrahedrons) for safe computational effort and human effort for design. Coping without textures to the greatest extent, our visualization still allows for an appealing puristic scene rendering using simple Phong shading techniques [50] (see Figure 10).

Table 3: Sensor and communication models.

Sensor/measurands	Property	Character	Parameters
GPS: position [lat, lng]	Atmospheric error	Changes over time	Magnitude
	Shadowing error	Depends on distance to buildings	Magnitude, distance weight
	Noise	Changes each sample	Magnitude
	Delay	Emulates signal processing time	Average time, variation
Frontal object sensor: distance [m]	Aperture	Sensing on vehicle frontal axis plus angel	Magnitude, steering offset
	Noise	Changes each sample, grows with distance	Magnitude, distance weight
	Delay	Emulates signal processing time	Average time, variation
Lane marking sensor: distance [m, m]	Noise	Changes each sample	Magnitude
	Delay	Emulates signal processing time	Average time, variation
Lateral object sensor: Rel. position [m, m], length [m]	Noise	Changes each sample	Magnitude
	Delay	Emulates signal processing time	Average time, variation
	Aperture	Fixed angles (front/rear)	Magnitude
Vehicle state: all parameters (Sections 4.3–4.7)	Noise	Changes each sample	Magnitude
	Delay	Emulates signal processing time	Average time, variation
V2X ad hoc communication: short range, 802.11p	Delay	Emulates signal travel and processing time	Average time, variation, distance weight
	Packet loss	Loss rate per distance	Loss rate, distance
Cellular communication: same as V2X + region, bandwidth	Region	Specific coverage areas	Circle or polygon
	Max. bandwidth	Reached maximum causes addition delay	Limit in byte per second

(ii) Automatic generation of traffic infrastructure and static objects from existing map material is in order to speed up scenario generation by developers. The plain generated infrastructure can be modified and complemented by developers. Event triggers and scripted procedures are related to the environment; for example, time-bound occurrence road geometry changes (see constructions site in Figure 10) can be added.

The key features of the driver input are as follows:

(i) Definition of driver behavior using speed-annotated routes along the road map links (see Figure 10): in addition to the target speed, developers can attach time or event triggered, scripted maneuvers to these routes to create specific situations.

(ii) Automatic generation of driver behavior in traffic is realized by an extendable driving controller hierarchy of various speed and steering controllers.

The description of a simulation scenario defined by developers contains all scenario specific information about environment, driver input, simulation models, and their parameterization. For all position related elements of a scenario, our simulator provides visual tool support, for example, to place objects and vehicles and define routes. All scalar elements are defined in a set of configuration files. This refers, for instance, to parameters of vehicle models, sensors, and randomized behavior on routes like swaying or driver reaction time. The *scenario definition* bundles all that information and distributes it to all *simulation instances* (see Section 4.2) by the *instance coordination*. In order to enable a high repeatability of a simulation, the *scenario definition* optionally includes a seed set for all randomized parameters of the simulation models.

Examples of scenario created for our simulator can be found in [51, 52]. Figure 10 gives an impression of an example scenario displayed by the *Visualizer* (see Figure 5). The scenario is near Ernst-Reuter-Platz in Berlin, Germany, that has been generated automatically from Open Street Map (OSM) [53]. A route defined by the developer is depicted as a blue line. In this example scenario the route has no annotated speeds. The target speed is taken from the imported OSM map speed limit. Three speed controllers provided by the simulator work together in this example scenario. The target speed is realized by the mission-controller, which is overridden by the traffic-light-controller if the vehicle needs to stop at a red traffic light. Both of them are overridden by the follow-vehicle-controller in case a slower vehicle is in front.

4.11. Implementation. The implementation of our simulator design has been used in research before ([51, 52, 54–56]) under the name PHABMACS. Although the focus of this work is not on implementation, we give a brief overview of its key aspects. We have chosen Java as programming language. Although C++ as a native language may provide a

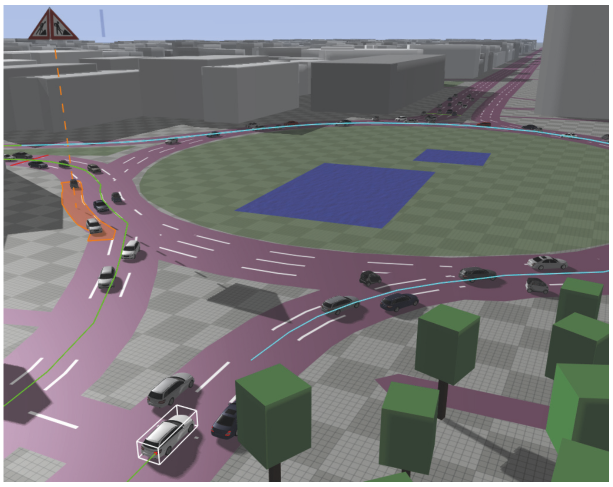


Figure 10: Simulator reference implementation, example scenario: construction site at round about, Ernst-Reuter-Platz, Berlin, Germany.

higher execution performance, we consider the performance of recent Java Virtual Machines (VM) as sufficient. The VM approach of Java addresses distribution over heterogeneous systems (see Section 4.2.2 decoupling in space) better than C++. Aspects like built-in memory management, garbage collection, and modern language features like lambda expression make Java the better candidate with regard to rapid prototyping.

4.11.1. Interfaces. Application code and additional sensor models can either be created in Java and deployed to the PHABMACS framework or be interfaced from MATLAB/Simulink. For interaction with further technology of developer's choice, we have integrated PHABMACS with VSimRTI [57], a runtime infrastructure which enables coupling with an arbitrary number of different simulators of the various aspects, such as network simulators. In combination with our CAN-Bus hardware interface, PHABMACS can also be hooked up with in vehicle controller setup and replace real world vehicles for testing.

4.11.2. Visualization. The implementation of the *Visualizer* (see Figure 5) for the PHABMACS visualization engine is a custom implementation based on LWJGL [58], a Java based library to access OpenGL [58] functions on the target system. This engine was initially introduced in [55]. The screen shot in Figure 10 was rendered by this visualization engine.

4.11.3. Vehicle Dynamics. The implementation of the vehicle dynamics is realized in PHABMACS as follows. The *dynamics models* of the *vehicle* (see Figure 5) as described in Section 4.2 are implemented in plain Java. For performance reasons, all differential equations are solved using Euler's method [59], which is sufficient for the required precision, as shown in Section 5. In one simulation loop, all the forces acting on each vehicle are calculated and fed to the *physics engine*, which applies these forces to the objects in the simulation and calculates their position for the next loop. Figure 11 depicts the calculation flow for one simulation step using equations (1), (2)–(4), and (6)–(15). Beginning with the *simulation loop*, the *simulation instance* triggers the *vehicle* to apply the external forces of *aerodynamics* and *suspension* to its *rigid body*. Subsequently, the actuator positions are updated on the *wheel* model and the *power train*, which result in updated internal forces to be applied to the *rigid body*. Finally, all forces are combined and transmitted to the *physics engine*, which determines the position update of the vehicle body in a simulation step. The minimum frequency of simulation steps is configurable (default value is 200 Hz). If the simulation is running in real-time mode, the simulation speed is lowered if the execution time of one simulation step exceeds the minimum frequency for performance reasons.

The *physics engine* implementation is built on top JBullet[10], a Java port of the Bullet Physics Library[11]. Using an existing implementation safes some very time-consuming work regarding the execution of rigid body dynamics. We have reviewed alternatives like libGDX[12] and Ode4j[13] which did not meet our requirements concerning real-time behavior, matureness, and clean, object oriented APIs. Although the native version of JBullet offers multicore support and even GPU accelerated physics computation, the Java version is single-threaded. For this reason, we are currently developing a custom physics engine. The results described in Section 5, however, were generated using JBullet.

5. Vehicle Dynamics Validation

Bevor the simulator described in Section 4 can be used to analyze cooperative ADAS applications; its ability to map vehicle dynamics sufficiently realistically needs to be validated. This validation assures that the accuracy of the simulation's representation of real vehicles is sufficient to apply simulation results of an applications to its behavior in the real world. As earlier pointed out for our simulator, mapping vehicle dynamics is supposed to be realistic below the highly nonlinarites near the limits of driving dynamics. Derived from the simulator scope in Section 2.2, this mapping needs to be evaluated within the specific span of driving dynamics for which a specific application is supposed to work. For validating our simulator, we propose a validation method that is able to testify validity for such a certain span of driving dynamics. This method is aligned with the requirements to our simulator issued by the three applications (CELC, CACC, PACE). Moreover, the method aims to enable developers to match the simulation models to specific real world vehicles of their choice in reasonable time. This time aspect is addressed by minimizing the number of required driving maneuvers, while maximizing the number of validated driving dynamics parameters in the defined span.

5.1. State of the Art in Validating Vehicle Dynamics.

To determine if a vehicle dynamics simulation model is valid over the complete domain of its applicability would be very costly and time-consuming, especially when extremely high model

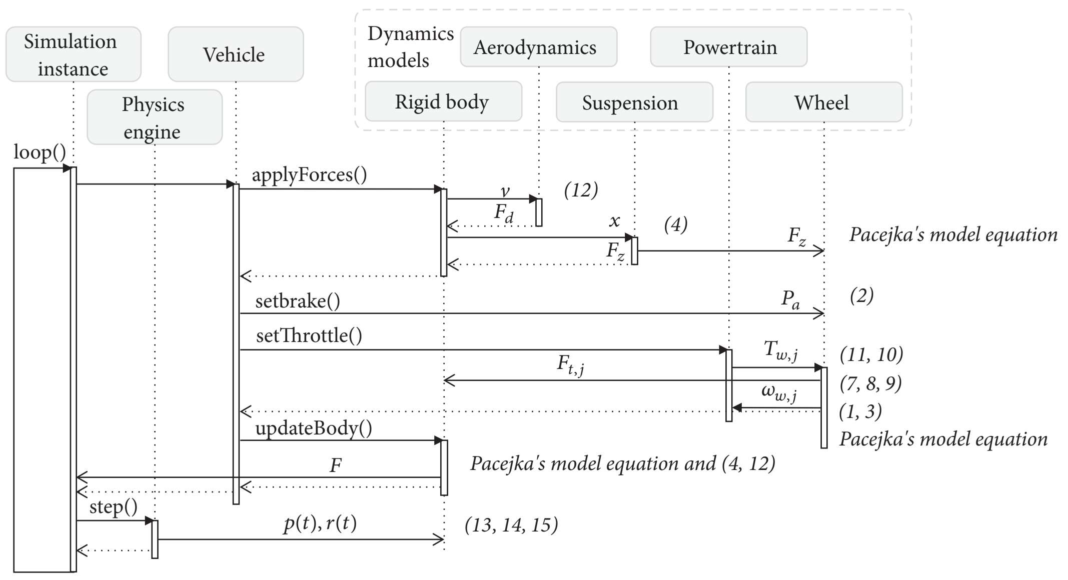


Figure 11: Calculation sequence of simulation models.

confidence is required. In [60], the relationship between the effort to gain model confidence and the value of this confidence to the user is described as depicted in Figure 12. For our validation method, we consider the maximum value threshold of the required model confidence to be imposed by the application range of the respective ADAS to be simulated. Examples for such thresholds will be given in the remainder of this section. This restriction is the key aspect of our method, which enables reduction of effort and time for validation.

The standard for validating vehicle dynamics simulations is to compare results of vehicle simulations with real world measurements [61]. An in-depth literature review on this topic and on the question of how to verify vehicle dynamics simulation can be found in [61]. Most works in this field use a set of standardized maneuvers, which were created to assess the performance of vehicle dynamics of real world vehicles. A list of such maneuvers and their ability to expose driving dynamics behavior can be found in [61]. These maneuvers can be classified in two groups. First, fundamental, artificial maneuvers (like ISO 7401) are created to determine main dynamical characteristics of vehicles and second purpose-dependent maneuvers to approximate real world driving (like ISO 3888-2). The relevant response type of the maneuvers can be identified as transient or steady-state. The system behavior of transient responses is observed between an initial and a final equilibrium state (e.g., in ISO 3888-2), while in a steady-state response the variables of interest do not change with the time. Kutluay and Winner [61] recommend combining maneuvers fundamental and purpose-dependent maneuvers with transient and steady-state responses for validation and analyze them in the time and the frequency domain, to cover as much aspects of driving dynamics as possible.

Comparing the outputs of the simulation with real world measurements is mostly done by graphical comparison of charts and subjective judgement about the quality of the matching as done in [62]. Heydinger et al. [63] improved this method by obtaining different metrics from time and frequency domain of the measured outputs of simulations and real world experiments. This includes a statistical analysis of several repetitions of different maneuvers by deriving the 95% confidence interval from the experiments and a check if the simulation stays within this interval. Kutluay and Winner [64, 65] improved this approach by introducing further metrics and a method to handle, split, and align the data of simulation and experiment. This enables to determine where the simulation performs weakly and to gain more confidence on the validated simulation models. Unfortunately, this method was created for validating lateral dynamics only.

5.2. Proposed Vehicle Dynamics Validation Method. Our proposed validation methodology uses six selected maneuvers and is inspired by the work of [63, 64]. Both works aim at validation of the full spectrum of dynamics, so that, for instance, regarding lateral dynamics, a rollover situation of a vehicle can be predicted. As explained above, our method does not aim at finding the limits of validity, but to validate the simulation within a certain range of driving dynamics, issued by the scope of our simulator and the specific application to be simulated. This allows us to define a validation method with fewer parameters to be checked, using fewer repetitions of maneuvers to be driven by a human driver in public traffic, without the need for driving robots or a dedicated test area, as well as a simple set of tools for the measurement. This makes our method applicable for validating our simulator against an arbitrary real world vehicle in a reasonable time frame. The validation method consists of four steps:

(1) The simulation models are preparameterized according to the specification of the real world vehicle to be matched.

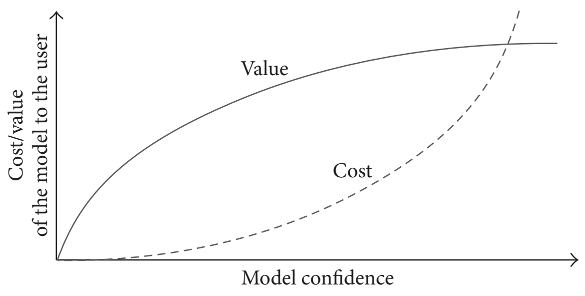


Figure 12: Model confidence.

(2) The lateral dynamics are considered isolated from the longitudinal dynamics. The lateral dynamics are matched and validated inspired the requirements imposed by the CELC application (Section 2.2). Beginning with a fundamental maneuver, the J-Turn, lateral acceleration and yaw rate are analyzed in a linear region. Subsequently, a purpose-dependent maneuver, the double lane change, is used analyze behavior of the lateral vehicle dynamics comprehensively.

(3) The longitudinal dynamics are matched and validated inspired by the requirements of the CACC application (Section 2.2). Beginning with a coasting maneuver, inertia, rolling resistance, and drag of the vehicle are determined isolated from the power train of the vehicle. Followed by a straight line braking and a straight line deceleration maneuver, the braking characteristics and the drag torque of the power train are analyzed isolated from each other. Subsequently, a straight line acceleration maneuver reveals the matching of the power train characteristics while accelerating.

(4) The isolated lateral and longitudinal dynamics analyzation of steps two and three are combined. Inspired by the requirements of the PACE application (Section 2.2) an oval driving maneuver is conducted including acceleration, deceleration, and braking elements under observation by an extern positioning measurement system.

For each maneuver of step two to four, a set of experimental data is chosen for calibrating the simulation model by modifying its parameters. The validation is done using a set of experimental different from the one used for calibration once the calibration is done for all steps. This process is necessary to separate calibration from validation, since otherwise the resulting method would aim at a best possible reproduction of the experimental data by the parameterizing the simulation model, instead of validation. In the following, we will describe our proposed validation method by example for these three applications CELC, CACC, and PACE. For this purpose, we calibrate and validate the vehicle simulation model of a Smart for two (W451) series vehicle with a 75 Kw engine.

5.3. Step I: Preparation. To begin the calibration with an initial parameterization of the vehicle, we obtain some parameters that can be taken from specifications or can be easily measured or estimated with sufficient accuracy see Table 4. The vehicle body dimensions, wheelbase, track width, and vehicle mass were taken from the vehicle datasheet. The wheel radius was directly measured at the vehicle. The maximum steer angle as well as the steering ratio is calculated from the turning circle diameters driven with different steering wheel angles. The wheel inertia is calculated from the wheel mass and radius. The capturing of the vehicle data is done via CAN-Bus access. No additional sensors were used, that is, measurement of acceleration is based on the output of the ESC (Electronic Stability Control) system sensors of the vehicle.

5.4. Step II: Lateral Dynamics. Considering lateral dynamics, the most relevant input parameter for the simulation model and real world vehicle is the steering wheel angle. The most relevant output parameters for comparing both are yaw rate and lateral acceleration. The basic approach of the validation is to capture all three parameters in real world experiments rather than feeding the steering wheel angle for each experiment to the simulation model and compare the output parameters of the simulation model with the ones of the real world experiment. The span of lateral dynamics for which our Smart vehicle has been validated is derived from the CELC application. The experiments were performed at two different speeds, 50 km/h and 25 km/h with a target steering wheel angle of 100 degrees. For our Smart vehicle, these numbers result in a maximum lateral acceleration of around 0,4 g, which is roughly the limit to drive the Smart without the Electronic Stability Control to intervene. This defines the span of driving dynamics for which the application can be simulated with validated simulation models later.

5.4.1. Maneuver Lat-I. For the first maneuver to calibrate and validate the lateral dynamics of the simulation model, we propose the J-Turn. This maneuver, defined in ISO 7401.2003, provides information about the lateral direction attributes of the vehicle [64]. It begins with the vehicle going straight at a certain speed. The driver then turns the steering wheel to a certain angle quickly, holds it fixed for some seconds and keeps the vehicle speed as constant as possible. In this way, a step input for the steering angle is emulated, which generates the step response and the steady-state response of the vehicle. While the former pertains to driving the vehicle near the limits of driving dynamics, the latter is the more relevant for our validation method. According to the method of [65] we capture and handle data from this maneuver, with a rise time of below 0.15 s between 10% and 90% of the final (steady-state) steering wheel angle. To improve repeatability, the driver had a set of training runs and from the final captured data, a set of runs were selected, which had most similar characteristics of the steering input.

(a) Data Handling. A set of 11 experiments for each speed was selected for validation. The experiments driven at 50 km/h

Table 4: Vehicle properties.

Parameter	Value
Body dimensions (l/w/h)	2.50 m/1.52 m/1.55 m
Wheelbase/wheel radius/Track width	1.81 m/0.242 m/1.35 m
Mass of vehicle/test crew/wheel	730 kg/130 kg/20 kg
Engine inertia	0.16 kg/m^2
Suspension stiffness (spring constant)	26.5 kN/m
Transmission inertias (N/1/2/3/4/5/6/R)	0,0/37/0.34/0.42/0.4/0.4/0.4/0.37 kg/m^2
Frontal area, drag coefficient	1.93 m^2, 0.37
Steering ratio	2.31
Clutch curve engaging (R/N/1 ⋯ 6)	0.26/0.0/2.618/0.667/0.6313/0.5/1.15/1.20 s
Clutch curve disengaging (R/N/1 ⋯ 6)	0.6/0.0/0.60/0.617/0.57/0.30/0.3/0.3 s
Clutch curve open (R/N/1 ⋯ 6)	0.367/0.0/0.367/0.367/0.317/0.317/0.317/0.317 s

(±2 km/h) are described exemplary in the following. The reference point at which the steering angle reaches 50% of its steady-state value in each experiment are calculated and aligned in time. As the 10 experiments can be considered as samples drawn from a full population, we employ Student's t-distribution to calculate experimental data zone (EDZ) by the 95% confidence interval around the mean values [65] of lateral acceleration and yaw rate for visual graphical comparison. This EDZ can be used to check if the time histories of the simulation outputs remain inside them, which has been originally proposed as a validity criterion by [63]. The upper and lower bounds U and L of the confidence band are calculated according to (16) [65]. The calculation is done using the MATLAB implementation of Student's t inverse cumulative distribution function "t inv", and the standard deviation "std" for σ, where v is the degree of freedom (the number of experiments) and μ is the mean value of data.

$$U, L = \left\{\mu \mp C\frac{\sigma}{\sqrt{N}}\right\}, \qquad C = t\,\text{inv}\,(0.95, v), \quad N = \frac{v}{2} - 1. \tag{16}$$

(b) Metrics. According to [65], in addition to lateral acceleration and yaw rate, this process is also done for the following further metrics: steady-state gains, rise times, peak times, maximum magnitudes, and maximum overshoot ratios. As earlier mentioned, for the part of the J-Turn in our validation method, it is sufficient to reach validity for the steady-state gain and to check the EDZ visually. The other metrics are omitted for the presentation of the following results. However, although less relevant, they are still valuable to gain information about the behavior of simulation model at the limits of driving dynamics.

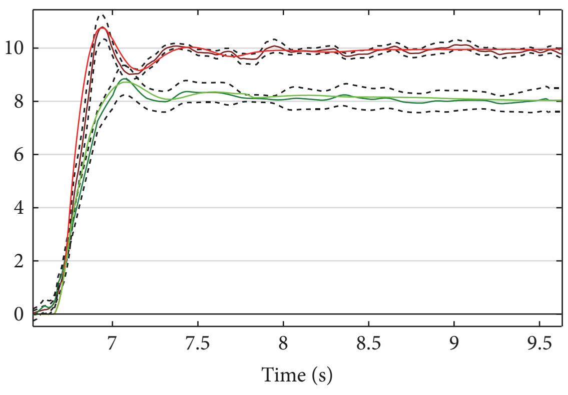
(c) Results. The results of the steady-state gains validation are shown in Table 5. The steady-state interval for yaw rate and lateral acceleration of the experiment data is determined [65], the gain values are averaged *("average of experiments")*, and the confidence interval around them is calculated *("confidence interval")*. The same procedure is applied for the simulation data *("average of simulations")* and error between both averages is calculated ("average of simulations error" absolute/percentage). Finally, the input values of the experiment data, the steering wheel input, is averaged and fed to one simulation run. For the resulting gains *("averaged simulation")*, the error to the experiment is calculated. The first validity criterion is fulfilled as the average of simulations stays within the 95% confidence interval. For the averaged simulation, there are no other samples to generate a confidence interval. Therefore, a 5% subjective error allowance is introduced as the second validity criterion [65], which is also fulfilled as shown on Table 5. The third validation criterion is to the check the EDZ of the steady-state part visually, which begins at 7.95 s. As depicted in Figure 13, both simulation graphs (yaw rate and lateral acceleration) stay within the confidence band for the steady-state part; that is, we consider maneuver Lat-I to be valid. A look at the transient part before second 7.95 reveals that both simulation graphs rise too quickly and settle too late in comparison with the experiment. However, as earlier mentioned this transient part relates to driving the vehicle near the limits of driving dynamics and is therefore not a necessary validity criterion for our validation method.

Table 5: Validation results: Maneuver Lat-I.

Steady state gains	Lat. acceleration [m/s^2]	Yaw rate [°/s]
Average of experiments	−40.24	19.82
Confidence interval	−42.07–−38.41	19.52–20.12
Average of simulations	−40.64	19.85
Average of simulations Err.	**0.4** (0.99%)	**0.03** (0.15%)
Averaged simulation	−40.93	19.91
Averaged simulation Err.	0.69 (**1.71**%)	0.19 (**0.96**%)
Verdict	Pass	Pass

5.4.2. *Maneuver Lat-II.*

For the second maneuver to calibrate and validate the lateral dynamics of the simulation model, we propose the double lane change, a purpose-dependent maneuver. Once we have validated the lateral steady-state behavior of the vehicle as a subset of the lateral dynamics by



—	Yaw rate simulation	(deg/s) × 0.5
—	Yaw rate experiment	(deg/s) × 0.5
—	Lateral acceleration simulation	(g) × −20
—	Lateral acceleration experiment	(g) × −20
- - -	95% confidence intervals	

Figure 13

the J-Turn, we extend the examination to a maneuver which exhibits comprehensive characteristics of lateral dynamics in a real world maneuver. The double lane change, defined in ISO 3888/1, approximates the behavior of the vehicle in an emergency maneuver situation like CELC, where the driver needs to switch from one lane to the other and back. Heavy understeering, oversteering or even a rollover situation can occur during such a maneuver. In addition to the steady-state gains already validated, with the double lane change maneuver we gain information about the response of the vehicle on suddenly changing steering wheel inputs in real world situation. This information is gained according to the method of [64] mainly by comparing the peaks of the yaw rate and the lateral acceleration off simulation and experiment, as well as their phase shift to the steering wheel input.

(a) Data Handling. For the validation of the Smart vehicle simulation model, a set of 6 experiments were selected, again each of two different speeds as defined for maneuver Lat-I. The experiments driven at 50 km/h (±2 km/h) are described in the following for explanation of the general method. To improve the alignment of the experimental data, the first and the second half wave of each maneuver are split depending on the middle portion of the steering wheel angle between them. The middle portion either reaches one or multiple local extrema, or a steady-state part. The split points are either the first and the last extremum, or the beginning and the end of the steady-state part. If there is only one local extremum, this the data is split at this point [64]. The first and the second half waves each of all experiments are then aligned by their reference points, which are selected to be at 50% of the steering angle value of the first maximum of each half wave [64].

(b) Metrics. Subsequently, the metrics for the yaw rate and the lateral acceleration of the double lane change are calculated. According to [65], these metrics are as follows:

(i) The time lags to the steering wheel angle are considered to be valid for errors below 0.05 s. An absolute acceptance band is better applicable then percentage, since the absolute values of the experiments are very small.

(ii) The temporal coordinates of the four extrema in both half waves are valid within the confidence interval and if their peak time error is smaller than 0.05 s for the average of simulations and smaller than 0.1 s for the averaged simulation. Using a percentage validity would be impractically, as the cumulative character of the time value would cause the percentage error band to grow along the time axis.

(iii) The spatial coordinates of the four extrema in both half waves are valid within the confidence interval and if the gain error of the averaged simulation is smaller than 5%.

(iv) Finally, the EDZ is checked visually for the yaw rate and the lateral acceleration.

(c) Results. The metrics are calculated similar to the procedure for the J-Turn maneuver (Lat-I Section 5.4.1). As the time lag results in Table 6 show that the average of simulations error and the averaged simulation error stay below 0.05 s for the lateral acceleration. The same applies for the yaw rate, except for the average of simulations at the second half wave, which exceeds the validity limed by 0.0017 s. The spatial and temporal coordinates of the yaw rate are presented in Table 8. The yaw rate temporal coordinates errors stay within the confidence interval, below 0.05 s for the average of simulations and below 0.1 s for the averaged simulation. All yaw rate spatial coordinates errors stay below 5%. The same applies for the lateral acceleration presented Table 7, except for the *average of simulations error* at the first and the fourth extremum, which exceed the 5% validity limit. Finally, checking the yaw rate and lateral acceleration charts of the simulation visually presented in Figure 14 reveals that both remain within the EDZ of the experiment completely. From the result of the validity analysis, we can observe that the metrics we derived for double lane change maneuver fulfill all objective criteria except for three specific points. These points reveal potential weaknesses of the simulation model, which we need to check individually in the following.

(1) The yaw rate time lag of the second half wave exceeds the validity limit by (0,0017 s), small enough that we can subjectively rate that point as valid.

(2) The fourth lateral acceleration gain extremum exceeds the 5% validity limit by 0.82%. The experiments lateral acceleration graph at this point shows an unusual shape with a dent and a small peak, instead of the parabolic shape as expected. We figured that this shape is due to the eccentric position of the ESP acceleration sensor in our test vehicle, which causes the measurement of lateral acceleration to be influenced by high changes of the roll angle. We were able to reproduce a comparable shape by

Table 6: Validation results Maneuver Lat-II: time lags.

Time lag to steering wheel angle [s]	1st half Lat. Accel.	2nd half Yaw Rate	1st half Lat. Accel.	2nd half Yaw Rate
Average of experiments	0.1368	0.1300	0.1204	0.1314
Confidence interval	0.1121–0.1616	0.119–0.1409	0.1107–0.1301	0.1208–0.142
Average of simulations	0.0879	0.0878	0.0800	0.0797
Average of simulations error	**0.049** (35.79%)	**0.0422** (32.46%)	**0.0404** (33.56%)	*0.0517* (39.35%)
Averaged simulation	0.0912	0.0866	0.0789	0.0813
Averaged simulation error	**0.0456** (33.32%)	**0.0434** (33.39%)	**0.0415** (34.48%)	*0.0501* (38.13%)
Verdict	Pass	Pass	Pass	*Fail > 0,05*

Table 7: Validation results Maneuver Lat-II: lateral acceleration extrema.

Lat. acceleration extrema	First		Second		Third		Fourth	
	Gain in m/s	Time in s	Gain in m/s	Time in s	Gain in m/s	Time in s	Gain in m/s	Time in s
Average of experiments	−38.84	13.15	36.36	14.32	36.06	16	−36.23	17.11
Confidence interval	−46.49–−33.56	13.12–13.18	34.28–40.03	14.32–14.18	31.51–42.05	16.14–16.01	−40.63–−31.85	17.1–17.11
Average of simulations	**−34.3**	13.18	**36.13**	14.35	**37.41**	15.96	**−34.09**	17.08
Average of simulations error	4.54 (11.7%)	**0.03** (0.19%)	0.22 (**0.62%**)	**0.03** (0.21%)	1.35 (**3.73%**)	**0.05** (0.28%)	2.13 (5.88%)	**0.03** (0.2%)
Averaged simulation	−35.55	13.13	35.89	14.22	37.71	15.91	−34.12	17.01
Averaged simulation error	3.29 (*8.47%*)	**0.02** (0.15%)	0.47 (**1.29**%)	**0.1** (0.7%)	1.65 (**4.58%**)	**0.09** (0.56%)	2.11 (*5.82%*)	**0.1** (0.58%)
Verdict	*Fail > 5%*	Pass	Pass	Pass	Pass	Pass	*Fail > 5%*	Pass

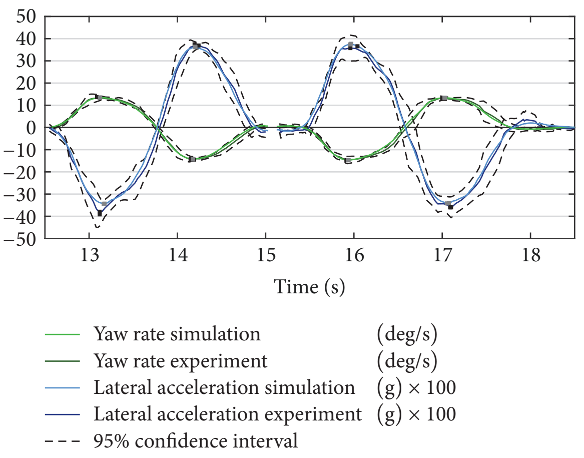


Figure 14: Validation Maneuver Lat-II: experimental data zones.

moving the virtual sensor in our simulated vehicle from the vehicle center to the same eccentric position as in the test vehicle. This gave us the confidence to rate the point as valid, without repeating the validation with a dedicated sensor equipment.

(3) The first lateral acceleration gain extremum exceeds the 5% validity limit by 3.47%, which is considerably high. The experimental data graph (see Figure 14) at this point shows a nonparabolic, almost triangular shape for the upper third of the curve, which reveals the reason. We observed that the steepness of the first amplitude in the maneuver shows a high variance, as the test driver fails to reach a higher repeatability in this phase. This fact, in addition to the eccentric sensor position, results in a weak matching of the aligned raw data for averaging. Since the lateral acceleration at the extremum two to four, as well as the other metrics, indicates sufficient validity, we consider the missing validity at the first extremum as a measuring error.

Therefore we consider the double lane change maneuver as valid. Again, for the described three points, we override the objective verdict derived from the metrics, by a subjective verdict. If a higher and more objective confidence is desired, a repetition of the experiment with a higher driving repeatability and a dedicated sensor setup would be needed.

5.5. Step III: Longitudinal Dynamics. Once the lateral dynamics are validated, we proceed with validating the longitudinal dynamics isolated from the lateral dynamic accordingly. Our validation method currently covers simulation models using the clutch model. The torque converter is disregarded so far and will be part of an extension of our method in future work. Considering longitudinal dynamics, the input parameter for the simulation model and real world vehicle are the throttle and the brake pedal. The output parameters for comparison are the vehicle speed and longitudinal acceleration. The basic approach of the validation is analogue to lateral dynamics, to capture all four parameters in real world experiments rather than feeding the input parameters for each experiment to the

Table 8: Validation results Maneuver Lat-II: yaw rate extrema.

Yaw rate extrema	First		Second		Third		Fourth	
	Gain in °/s	Time in s	Gain in °/s	Time in s	Gain in °/s	Time in s	Gain in °/s	Time in s
Average of experiments	13.45	13.15	−14.26	14.15	−14.76	15.93	13.27	17.03
Confidence interval	12.51–15.2	13.17–13.05	−15.56–−13.57	14.15–14.26	−16.48–−13.25	15.92–15.9	12.35–14.26	17.03–16.98
Average of simulations	**13.42**	13.13	**−14.16**	14.18	**−14.67**	15.94	**13.34**	17.03
Average of simulations error	0.03 (0.24%)	**0.03** (0.19%)	0.1 (0.71%)	**0.03** (0.18%)	0.09 (0.58%)	**0.01** (0.06%)	0.07 (0.54%)	**0** (0%)
Averaged simulation	13.72	13.18	−14.08	14.09	−14.53	15.9	13.17	17.03
Averaged simulation error	0.27 (**2.01**%)	**0.03** (0.23%)	0.18 (**1.26**%)	**0.06** (0.42%)	0.23 (**1.56**%)	**0.03** (0.19%)	0.1 (**0.75**%)	**0** (0%)
Verdict	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

simulation model and compare the output parameters of the simulation model with the ones of the real world experiment.

The longitudinal dynamics of the simulation are predominantly influenced by the powertrain submodels, the brake model, and the aerodynamics model. In order to get confidence about the validity or nonvalidity for each of these specific parts, each part should be validated separately. However, this approach would require an immense effort. Our proposed validation method aims at reducing the validation effort to driving five separate maneuvers. Using these maneuvers, aerodynamics, brake and power train can be considered isolated from each other. In that way, the power train submodels can be validated in combination without a dedicated validation for each mode separately. This is further enabled by introducing the engine output torque and rounds per minute (rpm) as intermediate input/output parameters.

The span of longitudinal dynamics for which our Smart vehicle has been validated is derived from the CACC application (see Section 2.2). Analogue to the lateral dynamics validated for the constraints of the CELC application, this longitudinal dynamics span defines the limits for which the CACC application can be simulated with validated simulation models later. The span is defined by a maximum speed of 45 km/h (displayed as 50 km/h on tachometer), a maximum deceleration of −0,4 g, and a maximum acceleration of a span between 0,4 g and 0,2 g mapped on the speed interval between 0 km/h and 45 km/h.

5.5.1. Maneuver Long-Ia Coasting. For the first maneuver we propose the straight line coasting as used in [66] to calibrate the aerodynamics of the simulation model. The vehicle is accelerated up to a certain initial speed on a track which needs to have no slope. Once this initial speed is reached, the driver switches the transmission to neutral and the vehicles rolls to standstill. In that way, the decelerating forces, air drag, and rolling resistance are isolated. The reference point for alignment we define to be the point in time when the vehicle stops, since this is the best identifiable point in each run at which the vehicle reaches the same state. The main metric for validation is the longitudinal acceleration. The vehicle speed is less suitable for quantitative validation due to its cumulative character, which causes the validation error to grow with the time progress during the maneuver. Besides that, measuring the speed using the series on board sensors is typically quite error-prone. Thus, we use the vehicle speed as a qualitative error indicator only. Analogue to the lateral dynamics validation, the maneuver is valid if the longitudinal acceleration of the *averaged simulations* has an error of below 5% and the *average of simulation* stays within the 95% *confidence interval* of the *averaged experimental* data. For the validation of our Smart vehicle, we have chosen 50 km/h as initial speed. The results of the maneuver were used to calibrate the drag coefficient and the resistance between wheels and road surface.

5.5.2. Maneuver Long-Ib Braking. For the second maneuver to calibrate the brake torque, we propose to extend maneuver Long-Ia (straight line coasting) by applying a constant brake force after switching the transmission to neutral. The driver needs to apply the same brake pedal position for each run to create a high repeatability of the brake torque. In that way, the brake force can be analyzed as isolated quantity, as is it added to the deceleration forces which already have determined by maneuver Long-Ia. The input parameter for the simulation model of the brake is the brake torque captured from the experiments. The reference point for data alignment is again the time when the vehicle stops. The brake system of our Smart vehicle has been validated with an initial speed of 50 km/h using different brake torques. The results of the maneuver were used to calibrate the brake torque of the simulation model.

5.5.3. Maneuver Long-Ic Decelerating. For the third maneuver, we propose another variation of the straight line coasting maneuver (Long-Ia und Long-Ib) to validate the drag torque of the power train. Instead of applying the brake, the driver does not switch to neutral, so that the engine drag torque decelerates the vehicle to standstill. In that way, the drag torque of the whole power train can be analyzed as an isolated quantity.

(a) Data Handling. In contrast to the maneuvers Long-Ia und Long-Ib, we need a different data splitting and alignment method in order to expose the power train dynamics. Using the standstill time again would result in a good matching of the average vehicle speed; however the flanks of the

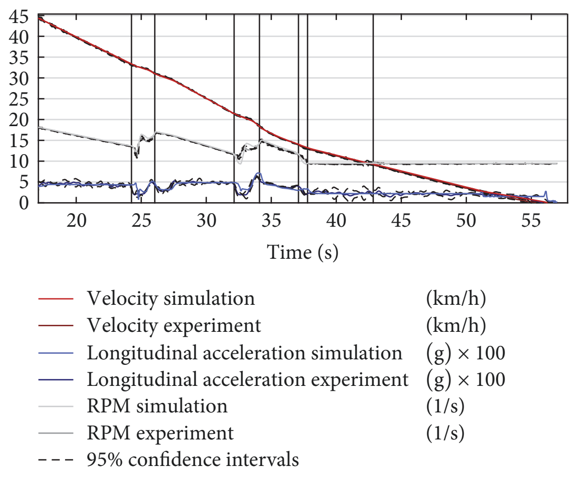


Figure 15: Validation Maneuver Long-Ic: experimental data.

longitudinal acceleration and the rpm curvature while clutch operating would get unsharp. Thus, we introduce the points in time when the clutch is opened and closed as reference points for splitting and aligning the data. In this way, the state transition between engaged gears and gear switching operations is sharpened and clearly exposed for analyzation. Although, this method causes the confidence interval around the speed graph of the experimental data to grow more with the time, but sharpens the average graph of the longitudinal acceleration, which is the more relevant metric for validation.

The powertrain of our Smart vehicle has been validated with an initial speed of 45 km/h with set of 6 experiments, described in the following for explanation of the general method. Additional maneuver Long-Ia und Long-Ib, the desired engine torque, the clutch times, and the current gear are taken from the experiments as further input parameter for the simulation. Two aspects of the power train simulation model are therefore not part of the validation, the gear shifting schedule and the engine map which calculates the desired engine torque from the pedal position. Both models are very specific so that it would take a lot of time to determine their parameters and there is little benefit for the validation of the longitudinal dynamics. Therefore, both of them are isolated by feeding the desired engine torque of the experiments to the engine model and the gear shift times to the transmission model and the clutch. Similar to the process used for the J-Turn maneuver (Lat-I), the experimental data is split and aligned and the metrics are calculated. In Figure 15, the average of the engine rpm, the vehicle speed and the longitudinal acceleration is presented. The vertical lines mark the points in time when the clutch is opened and closed.

(b) Metrics. The metrics to be evaluated here are the averaged longitudinal acceleration gains against the 95% confidence interval and a 5% error interval while the clutch is engaged for each gear separately (presented in Table 9), as well as the visual comparison of the graphs while clutching. The reason for validating the averaged longitudinal acceleration gains instead of applying a more complex shape comparison approach, results from the following thoughts. While the clutch is closed, the negative torque request forcing the engine drag torque to decelerate the vehicle is constant as the minimum engine torque is applied by the motor management. Thus, the negative acceleration is almost constant most of the time. There are some exceptions in this consistency issued by the motor management as for example in our case during the first second of the third gear and during the second gear. However these exceptions are not due to the test driver's behavior and therefore constant with each experiment. Consequently, if the cumulated deceleration brings the simulation model down to the same speed as in the experiment at the end of each gear, the averaged acceleration within this gear is a sufficient validation criteria. Further confidence in this assessment is reached by checking the EDZ of the longitudinal acceleration.

(c) Results. As the results in Table 1 show, our Smart vehicle model is valid for the longitudinal acceleration gains in each gear. The vehicle speed and the engine rpm are analyzed by graphical comparison of the EDZ. The EDZ in this case is very narrow while experimental driving a high reproducibility could be achieved. The vehicle speed graph matches the EDZ during the whole maneuver, while the rpm and the longitudinal acceleration leave the EDZ while clutching. This is an indicator that our clutch simulation model is in line with our expectations, too simple to precisely map the highly nonlinear processes of a clutch (see Section 4.6.1). The shape of the longitudinal acceleration graph seems to match qualitatively, however the peaks exceed the EDZ up to 0.02 m/s. This is probably caused by a delay of about 0.2–0.3 s at the flanks of the simulation model while declutching. Although the absolute error is comparatively low, it exceeds the confidence interval. This fact needs to be considered by application developers when using our simulator as a potential limitation of the simulation model.

5.5.4. Maneuver Long-II Combined Braking Decelerating. The fourth maneuver combines all decelerating longitudinal forces we have isolated and validated before. For that purpose, we combine maneuvers Long-Ib and Long-Ic, which causes the vehicle to be decelerated by the power train drag torque and a constant brake torque at the same time. In this way, all forces applied during a regular braking maneuver are included in the validation. (see straight line braking maneuver [63, 66]).

(a) Data Handling and Metrics. Our Smart vehicle has been validated for this maneuver with an initial speed of 45 km/h using different brake torques. A set of 6 experiments done with 40 Nm target brake torque are described in the following for explanation of the general method. Similar to maneuvers Long-Ib and Long-Ic, the experimental data is aligned and the metrics are calculated. The reference points for alignment are defined by the times when the clutch opens and closes, as well as the time when the vehicle stops. The main metric for validation is again the steady-state gain of the longitudinal acceleration while braking and the vehicle speed and rpm are qualitative indicators. The steady-state gain of

Table 9: Validation results Maneuver Long Ic: long acceleration gains.

Long. acceleration gains [m/s^2]	Gear 3	Gear 2	Gear 1	Neutral
Average of experiments	4.553	4.227	3.935	2.026
Confidence interval	4.151–4.955	3.832–4.622	3.778–4.092	1.773–2.279
Average of simulations	4.361	4.332	4.085	1.994
Average of simulations error	0.192	0.105	0.150	0.032
Averaged simulation	4.680	4.362	4.126	1.966
Averaged simulation error	0.127 (**2.79**%)	0.135 (**3.19**%)	0.191 (**4.85**%)	0.06 (**2.96**%)
Verdict	Pass	Pass	Pass	Pass

Table 10: Validation results Maneuver Long II: steady-state gains.

Steady state gains	Long. acceleration [g] ∗ 100
Average of experiments	21.168
Confidence interval	19.395–22.941
Average of simulations	20.950
Average of simulations error	0.218
Averaged simulation	20.214
Averaged simulation error	0.95 (**4.51**%)
Verdict	Pass

the longitudinal acceleration is calculated form the point in time when the average brake torque of all experiments is applied until the vehicle stops.

(b) Results. The results of this validation are presented in Table 10. The validity criterion if fulfilled, as the averaged simulation error does not exceed 5% and the average of simulations remains within the confidence interval. Finally the EDZ is checked visually (Figure 16). The average of simulations graph for the longitudinal acceleration remains within the confidence interval, except for the last second before the vehicle stops. Different from the simulation graph, the experiment graph rises with a little dent in that last second, although the brake torque does not rise. This behavior probably originates from a nonlinearity of the disc brake system at low rotation speeds, which we have not modelled in our brake model. However, since the brake torque in sum brings the vehicle and the simulation model to stand still in the same time (the rising and the falling flanks of the longitudinal acceleration match the confidence interval) we can rate the behavior of our simulation model at this point as valid. Checking the speed EDZ reveals as small mismatch between 17 km/h and standstill. As we can observe the speed difference in experiment and simulation between initial speed and standstill in the same time period, we can rate this mismatch as a sensor measuring error of the vehicle.

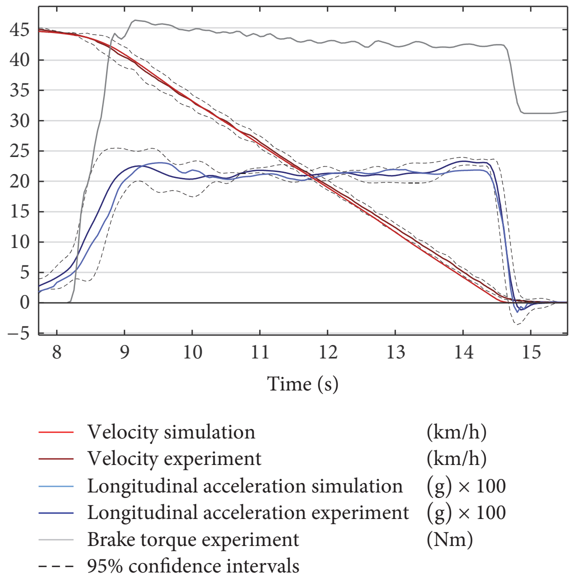
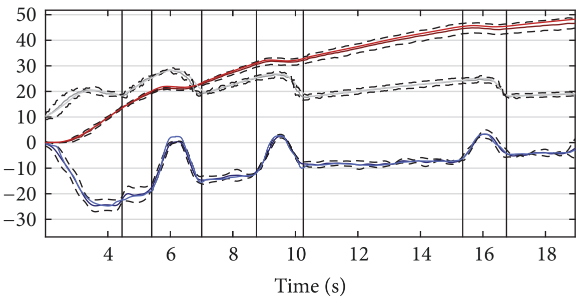


Figure 16: Validation Maneuver Long-II: experimental data zones.

5.5.5. Maneuver Long-III Accelerating. The final maneuver to validate the longitudinal dynamics aims at studying the accelerating forces of the power train. For that purpose, we propose a straight line acceleration maneuver [66]. This maneuver requires a test track with no slope and the driver needs to apply the accelerator pedal with a constant angle for each experiment. The power train of our Smart vehicle has been validated up to 50 km/h with different acceleration of 30%, 50%, and 70% pedal position. A set of 6 experiments done at 50% are described in the following for explanation of the general validation method. The experimental data is fed to the simulation using the desired engine torque, the current gear, and the times when the clutch opens and closes, as motivated for maneuver Long-Ic.

(a) Data Handling and Metrics. The data handling is also done in accordance with maneuver Long-Ic. For the metrics, we propose to apply a regression analysis of the longitudinal acceleration instead of averaging as done for maneuver Long-Ic for the following reason. For our Smart vehicle, in contrast to the drag torque of the power train while deceleration, its drive torque while accelerating is not kept constant by the motor management while the clutch is closed. While acceleration, a constant pedal position leads to a slightly falling desired torque. For this reason, it is not possible for the driver to produce a constant desired torque when driving an experiment. Assuming the test driver would be able to apply a precisely constant pedal position, this would still lead to the same highly reproducible behavior as in maneuver Long-Ic, and therefore the same metric of averaging the longitudinal acceleration could be applied. However, due



—	Velocity simulation	(km/h)
—	Velocity experiment	(km/h)
—	Longitudinal acceleration simulation	(g) × 100
—	Longitudinal acceleration experiment	(g) × 100
—	RPM simulation	(1/s)
—	RPM experiment	(1/s)
- - -	95% confidence intervals	

Figure 17: Validation Maneuver Long-III: experimental data zones.

to the falling torque, we need an indicator which exhibits possible uncertainties of the test driver's pedal input that result in a varying slope of the desired torque.

For this purpose, we propose to apply a simple linear regression of the longitudinal acceleration for each section between the clutch operations. The calculation is done using least square matching method [67]. We use the MATLAB function "*polyfit*" to calculate the slope and the intercept [67] from the resulting regression line. Since the intercept of a line is not a suitable value for validation, we calculate the midpoint of the regression line from the slope and the intercept at the mid time value between the split points of each gear. This is done for the averaged experiment data to calculate the average slope and the average center of the graph and for each experiment separately to calculate the 95% confidence interval in accordance with the maneuver Long-Ic.

(b) Results. Figure 17 depicts the resulting graphs and Table 11 presents the validation results, which pass the validation criteria similar to maneuver Long-Ic by fitting all confidence intervals and an overall error below 5%. The longitudinal acceleration graph shows negative acceleration peak of 2.3 cm/s^2 for the simulation during the gear switch from gear 1 to gear 2, which is 1.5 cm/s^2 different from the experiment. This fact needs to be considered by application developers. Finally we check the EDZ of vehicle speed and engine rpm, which remain within the 95% confidence. The latter tells us that our clutch simulation model does a better job for shifting up than for shifting down (see maneuver Long-Ib). The speed EDZ grows with the progress of the experiment due to cumulative error of the pedal position over time, as described earlier for maneuver Long-Ic. Although the simulation graph does not match the experiment, it still fits the EDZ. In general this deviation could be issued by the speed error of the simulation model cumulated over time. And in general it could be considered as a valid deviation as far as the cumulative model error does not exceed the cumulative error of the pedal position which broadens the

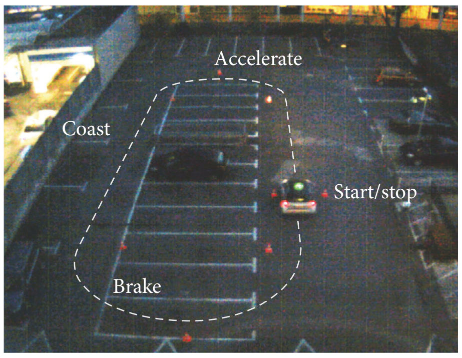


Figure 18: Oval maneuver combined dynamics.

speed EDZ. However in our case, the deviation of both graphs is most likely due to a measuring error of the vehicle speed in the experiment. We came to that conclusion since the integrated acceleration of the experiment leads to a higher final speed than measured. This means that there is no way to match the speed and the acceleration graph at the same time in simulation. In this case, the acceleration graph represents the more relevant value, as it has no cumulative error.

5.6. Step IV: Combined Dynamics.

After calibrating and validating the longitudinal and lateral driving dynamics isolated from each other, in the final step of validation we combine lateral and longitudinal dynamics. In this way, we uncover potentially disregarded interdependencies between both. At the same time we aim at validation of the requirement on correct evolution of the vehicle position in our simulator, motivated by the requirements to our simulator issued by the PACE application (Section 2.2.3). Additionally, with the combined dynamics validation, we complement the maneuvers for longitudinal validation (Long-I–Long-III), which are all steady-state maneuvers, by a transient maneuver. In order achieve these goals, we need a set of maneuvers, which cover braking and acceleration on a straight and while turning, as well as some load changes for the transient part of the longitudinal dynamics. Towards these goals, we propose an oval drive maneuver that covers all these aspects in one single maneuver.

The maneuver is depicted in Figure 18. The vehicle starts at a fixed position with a straight line acceleration followed by an acceleration in a U-turn [66]. At the end of the U-turn, the test driver continues with a deceleration on a straight line. The next U-turn is driven with the brake applied constantly until the start point on the straight is reached again (see brake in a turn maneuver [66]). Having the start point and the end point at the same position turn the maneuver in a closed loop driving maneuver. Moreover, it creates the same equilibrium state for the start and the end of the transient response of the vehicle, which also serves as splitting point for data alignment. Both aspects, the open loop character and the equilibrium state, help to enhance repeatability of the experiments. The maneuver is driven in both directions in order to cover right and left turns. Acceleration and braking need to be applied as constant as possible by the test driver

TABLE 11: Validation results Maneuver Long II: steady-state gains.

	Gear 1		Gear 2		Gear 3		Gear 4	
	Slope [cm/s^3]	Midpoint gain [cm/s^3]	Slope [cm/s^3]	Midpoint gain [cm/s^3]	Slope [cm/s^3]	Midpoint gain [cm/s^3]	Slope [cm/s^3]	Midpoint gain [cm/s^3]
Average of experiments	−2.518	20.05	−1.389	13.45	−0.39	7.94	−0.403	4.38
Confidence interval	−2.74–−2.29	17.79–22.3	−1.442–−1.336	12.61–14.29	−0.403–−0.376	7.59–8.29	−0.443–−0.362	2.95–5.8
Average of simulations	**−2.431**	**20.6**	**−1.394**	**13.55**	**−0.402**	**7.94**	**−0.388**	**4.15**
Average of simulations error	0.087	0.55	0.005	0.1	0.012	0	0.015	0.23
Averaged simulation	−2.621	19.98	−1.345	12.9	−0.375	7.591	−0.419	4.288
Averaged simulation error	0.1 (**4.09%**)	0.07 (**0.35%**)	0.04 (**3.17%**)	0.55 (**4.09%**)	0.01 (**3.85%**)	0.34 (**4.4%**)	0.016 (**3.97%**)	0.092 (**2.1%**)
Verdict	Pass	Pass	Pass	Pass	Pass	Pass	Pass	Pass

in each run in order to achieve a high reproducibility of the resulting response.

5.6.1. Data Handling. All experiments are driven in a row, to reach best possible repeatability of the position and speed. With each oval driven in a row, the test driver's skill to drive the same trajectory of position and speed will rise. In our case, we dropped the first 10 of 40 experiments (ovals driven) as training sets. From the remaining 30 experiments 21 were selected for validation in each direction. We use a set of 21 experiments driven counterclockwise to explain the validation exemplary. In our example, we aim at precise driving in narrow spaces with low speeds, as motivated by the PACE application. We therefore have validated the combined dynamics with a maximum speed of around 25 km/h in an oval of 24 × 11.5 meter. For position tracking, we used the camera-based localization system described in [68]. Due to the size of the area to be covered for the oval maneuver, the camera was positioned at a height of 21.6 meters at the south side of the area. The lateral viewing angle and height lead to a calculated precision between 6.6 cm and 4.8 cm depending on the distance between the tracked object and the camera [68]. The experimental data is split at the points in time when the vehicle starts and stops at the equilibrium point. Each oval drive is simulated separately, while the start position of the simulated vehicle its set to the start position of the experiment.

5.6.2. Metrics. The main aspects for the metrics of this maneuver are to validate first if the final position of the simulation is close to the position of the experiment and second if it travelled the on similar trajectories to this position. Towards these aspects, we introduce the *position-speed-domain* and the *travelled-distance-domain*. In contrast to the maneuvers for lateral and longitudinal dynamics validation, which have been analyzed in the time domain, we employ the position domain. This enables both to reduce the cumulative position error rising with the time and to validate of the correct evolution of the position. For this purpose, we consider the speed of the vehicle over its 2-dimensional position, which results in a 3-dimensional state space to be validated (the *position-speed-domain*). The position domain has been used for model validation in [69] however the speed was disregarded. In addition to the position domain, we introduce the *travelled-distance-domain*, which enables reducing the 2-dimensional position domain to a scalar value.

We further introduce the deviation between two positions in both domains to be validated as metric for the following three validity criteria:

(1) The spatial deviation between the final position of the experiment and the simulation is considered to be valid below 5% error of the travelled final distance and within the 95% confidence interval bounds.

(2) The *travelled-distance-domain* EDZ of the spatial deviation between each position of the experiment and the simulation for which the travelled distance is equal is checked visually.

(3) The *position-speed-domain* EDZ is visually checked in 3-dimensional state space span by the position and the speed.

The first step to calculate the metrics is to calculate the average position trajectory for the experiments A and the simulations $\widetilde{A}$. For this purpose, one trajectory g is selected as base and for each position (x_i, y_i) in this base trajectory, the closest position (x, y) in each of the m remaining trajectories, the family f_s with $s = 1, \ldots, m$ and the family $h_t = \{f_s, g\}$ with $t = 1, \ldots, m+1$ is determined according to (17). The resulting sets A and $\widetilde{A}$ of n positions form the averaged position trajectories according to (18).

$$P_i = \left\{(x, y) \mid (x, y) \in f_s;\ \left\|(x, y) - (x_i, y_i)\right\|_{\min}\right\}, \quad \left\{(x_i, y_i) \mid i = 1, \ldots, n;\ (x_i, y_i) \in g\right\}, \tag{17}$$

$$\begin{aligned} A_i &= \overline{(x, y)}_i \\ &= \frac{1}{m+1}\left((x_i, y_i) + \sum_{j=1,\ldots,m} (x_j, y_j)\right) \mid (x_j, y_j) \\ &\in P_i. \end{aligned} \tag{18}$$

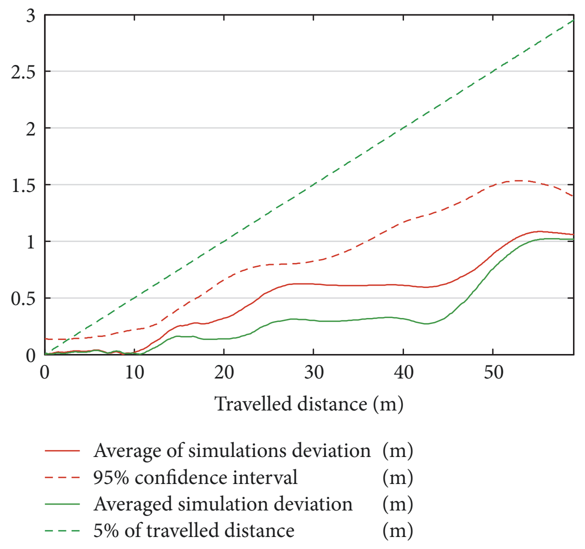


Figure 19: Validation combined dynamics: travelled-distance domain.

(a) Travelled-Distance-Domain. For each position A_i in the averaged experiment, the spatial distance to the preceding position A_{i-1} is summed up to the travelled distance d_i. The deviation D_i between averaged simulation $\widetilde{A}$ and averaged experiment A is determined by finding the positions in both graphs for which the travelled distance is minimal and calculating spatial distance between them. The same method is applied to determine the confidence interval using the spatial distances between all experiments and the averaged experiment according to (19). The upper bound of the confidence interval UD is calculated according to (20), while the lower bound is not relevant this time.

$$D_i = \left\{ \left\| A_i - \widetilde{A}_j \right\| \, \middle| i, j = 1 \cdots n \middle| \left| d_i - \widetilde{d}_j \right|_{\min} \right\}, \quad d_i = \sum_{k=1\cdots i} \left\| A_k - A_{k+1} \right\|, \; i = 1 \cdots n-1, \tag{19}$$

$$UD_i = \left\{ \frac{e_i}{m+1} + C \frac{\sigma_{ii}}{\sqrt{N}} \right\}, \quad \sigma_{ii} = \operatorname{std}(D_i), \; e_i = \sum_{k \in D_i} k. \tag{20}$$

The resulting graphs are depicted in Figure 19. Checking this graph visually validates the second validation criterion. The spatial deviation between the average of simulations and average of experiments is valid within the confidence interval (between the upper bound and zero). The spatial deviation between the averaged simulation and the averaged experiment is valid below 5% of the travelled distance.

(b) Position-Speed-Domain. First, we calculate the position EDZ around the averaged position trajectory of the experiments A. For each position A_i in A (see (18)), we determine the distance to each of the trajectories h_t (see (17)) by intersecting the normal r_i to q_i (A_i to its successor position) as set M according to (21). From M_i we can calculate the position EDZ at A_i consisting of the two points L_i and U_i by determining the confidence interval of all distances in M_i and multiplying it with the orthogonal r_i according to (22).

$$M_i = \left\{ \left| \overline{(x,y)}_i - (x,y) \right| \mid (x,y) \in h_t \wedge (x,y) \in r_i \right\}, \quad r_i \perp q_i \mid q_i = \overrightarrow{A_i A_{i+1}}, \tag{21}$$

$$L_i, U_i = \left\{ \overline{(x,y)}_i \mp \frac{r_i}{|r_i|} \cdot C \frac{\sigma_i}{\sqrt{N}} \right\}, \quad N = \frac{m+1}{2}, \; \sigma_i = \operatorname{std}(M_i), \; C = t\operatorname{inv}(0.95, N-1). \tag{22}$$

To calculate the speed EDZ, for each position in A_i, the measured speed of the vehicle is used to calculate its average and its confidence interval accordingly, which are finally plotted as the third dimension on the 2-dimensional position. The same process is done for the simulation. The resulting graphs are depicted in Figure 20. The thus obtained *position-speed-domain* EDZ has two dimensions to be checked, the position and the speed. As the graph shows, in contrast to maneuver Long-III, the confidence intervals do not grow with the time progress of the experiment and shrink to about zero at the start/stop position. In this way, the deviation of the position between averaged simulation and averaged experiment can be checked visually regarding the question if the cumulative position error of the simulation stays within the EDZ of the experiments. Hereby, the third criterion can be validated. The drawback here is the fact that since the simulated data does accumulate the error of its deviation from the real world model, it is unlikely that the averaged simulation graph will stay inside the confidence intervals for the whole maneuver. Accordingly, for the third validation criterion, the location where the EDZ is left is checked and rated by a subjective verdict.

Table 12: Validation results combined dynamics.

	Final travelled distance (FTD) [m]	Position error at FTD [m]
Average of experiments	59.66	-
Confidence interval	58.51–60.81	−1.26–1.26
Average of simulations	60.24	-
Average of simulations error	0.58	**1.06**
Averaged simulation	60.02	-
Averaged simulation error	0.36 (0.6%)	1.02 (**1.71%**)

5.6.3. Results. The results related to the first validation criterion are presented in Table 12. At the final travelled distance of the *average of experiment* at 59.66 m, the position error of the *average of simulations* is 1.06 m (within the confidence interval) and the position error of the *averaged simulation* is 1.71%. Thus, the first validation criterion is passed.

Figure 19 reveals that in the *travelled-distance-domain* the *average of simulations* graph stays below the *confidence interval* graph and the *averaged simulation* graph stays below

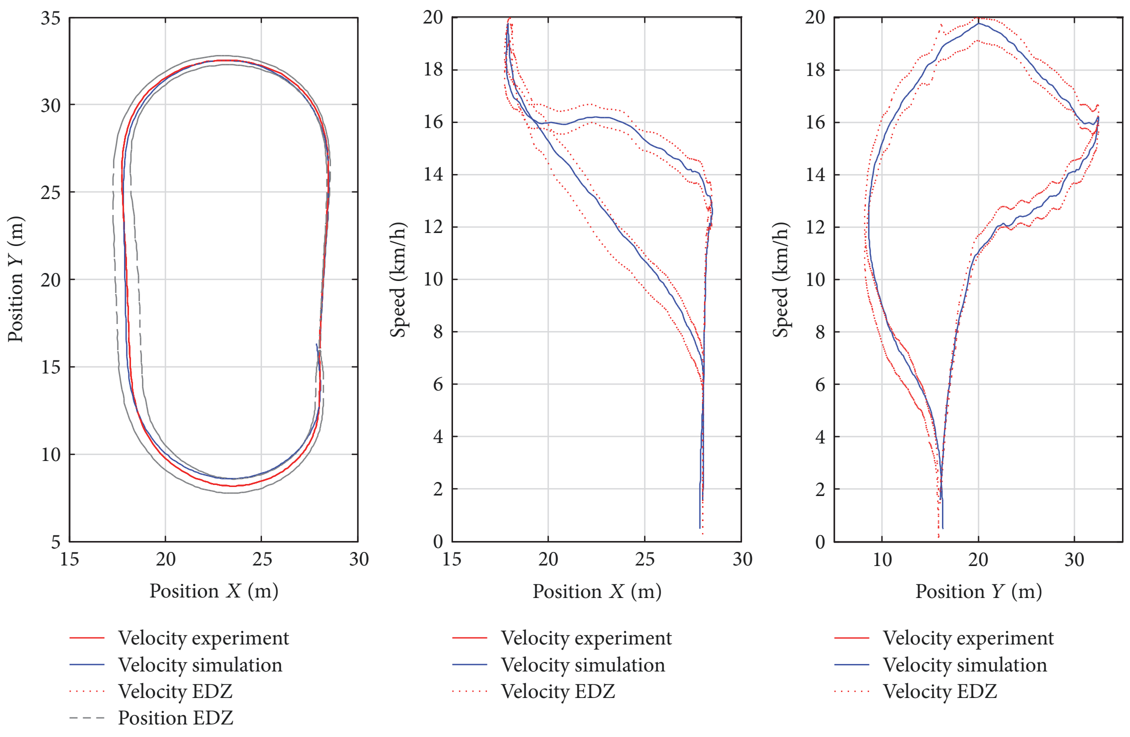


Figure 20: Validation combined dynamics: position-speed domain.

5% for the whole maneuver. Thus, the second validation criterion is passed.

Concerning the third validation criterion, the *position-speed-domain* graph (Figure 20) shows that the final position of the simulation model value overshoots experimental one by 0,443 m. Hence, the average speed of the simulation model leaves the EDZ 9 cm away from the averaged stopping position of the experiment at 4.02 km/h as the subjective verdict is valid in this case. As the EDZ is left very close to the final position while braking, a potential inaccuracy of the brake model can hardly be distinguished from measuring inaccuracies of the experiment. This refers to measuring the speed at very low speeds as well as the measured brake torque that is fed to the simulation mode. In order to cope with this subjective aspect, we have rated the first and second criterion validated objectively in the *travelled-distance-domain*.

(a) Discussion. The motivation to apply both domains for validation can be summarized as follows. In both domains the speed is regarded decoupled from position, which isolates the cumulative speed error over the time. The *position-speed-domain* however disregards the travelled distance. The simulation graph could vermiculate very close to experiment and still be apparently valid if it stays within the confidence interval. Besides that, the confidence interval of the *position-speed-domain* will be reduced to zero at the final position; that is, the simulation will eventually leave the confidence interval due to its cumulative position error. For this reason, a subjective verdict of the user is needed for validation. In the travelled-distance-domain the experiment cumulates the deviation error and so provides a magnitude for objective validation. However, valid magnitudes bounds can reach relatively high deviations, in our case about 1.5 meters after 50 m travelled distance. If this deviation had an orthogonal direction, which would be unacceptably high as the simulation model would drive that 1.5 meters off the track. For this reason, we need to consider both domains complementary. From the *travelled-distance-domain* we can derive a necessary but not sufficient condition for the simulation model to be valid. As the same applies for the *position-speed-domain*, only both domains considered together can deliver the sufficient condition for the simulation model to be valid.

6. Conclusion

Very large parameter spaces need to be regarded during the development of cooperative ADAS, as their complexity grows with each vehicle involved in an automated and cooperative maneuver. Thus, novel development approaches are required to bring an idea for a cooperative ADAS through the prototyping stage towards a plausible candidate for further development. In this paper we propose such an approach with our rapid prototyping environment based on vehicle simulation, aligned with an iterative prototyping process of application refining and assessment. This environment addresses the opposing requirements of automation and cooperation by a tradeoff between simulating multiple vehicles at the same time, while mapping their vehicle dynamics as precisely as required.

With this compromise we describe simulation models which reduce computational effort and we propose an architecture which provides scalable computational power to execute these models. The novelty here is not the models and their related equations themselves, but their combination aim at the named tradeoff between opposing requirements. Therefore, we focus on mapping vehicle dynamics precisely and computational efficiently to this end, as vehicle dynamics are the key feature to realize the aforementioned compromise. For this purpose, we define a scope of application for our prototyping environment, which excludes simulation at the limits of driving dynamics. Within this scope, we can validate realistic mapping of vehicle dynamics. This validation assures that the representation of real vehicles in simulation is sufficient to apply simulation results of an application to its behavior in the real world. To this end, we propose a vehicle dynamics validation method, which is based on existing validation approaches and fitted to our scope. This method combines isolated and combined consideration of longitudinal and lateral vehicle dynamics. Its key enabler to address rapid prototyping is reducing the required validity to a specific span of vehicle dynamics within the boundaries of the scope of our prototyping environment, instead of finding the limits of validity. In this way, minimizing the number of required driving maneuvers while maximizing the number of validated vehicle dynamics parameters in the defined span is realized.

As a proof of concept we have implemented the proposed prototyping environment in Java. We have validated this implementation against a real world vehicle (Smart W451 series) in terms of vehicle dynamics, using the proposed validation method. We have demonstrated this method for the specific span of driving dynamics issued by three example applications CELC (Cooperative Emergency Lane Chance), CACC (Cooperative Adaptive Cruise Control), and PACE (Parking Autonomously in Cooperative Environment).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors would like to acknowledge the comments of Prof. Dr. Peter Pepper. They would also like to thank Dipl. Ing. Fabian M. Thiele for his great implementation work on the PHABMACS simulator and M.S. Sebastian Bunk and Dipl. Inf. (FH) Niko Pfeifer for their indispensable assistance with the experiments. The authors also acknowledge the support by the German Research Foundation and the Open Access Publication Funds of Technische Universität Berlin.

Endnotes

1. http://www.path.berkeley.edu/.
2. PROgraMme for a European Traffic of Highest Efficiency and Unprecedented Safety, 1986–1994.
3. Dedicated Road Infrastructure for Vehicle safety in Europe, 1989–1995.
4. Intelligent Vehicle Initiative 1998–2005.
5. www.cvisproject.org.
6. http://www.drive-c2x.eu/.
7. http://www.safespot-eu.org/.
8. http://www.autonet2030.eu/.
9. http://www.imagine-online.de.
10. JBullet, http://jbullet.advel.cz/.
11. Bullet Physics Library, http://bulletphysics.org/.
12. libGDX, https://libgdx.badlogicgames.com/.
13. Ode4j, http://ode4j.sourceforge.net/.

References

[1] J. Ziegler, P. Bender, M. Schreiber et al., "Making bertha drive-an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[2] O. Sawade and I. Radusch, "A selection process for next generation cooperative driver assistance systems," in *Proceedings of the 20th Intelligent Transport Systems World Congress (ITS '13)*, pp. 1–9, October 2013.

[3] S. E. Shladover, C. Nowakowski, X.-Y. Lu, and R. Ferlis, "Cooperative adaptive cruise control: Definitions and operating concepts," *Transportation Research Record*, vol. 2489, pp. 145–152, 2015.

[4] H.-P. Schoener, S. Neads, and N. Schretter, "Testing and Verification of Active Safety Systems with Coordinated Automated Driving," in *Proceedings of the 21st (ESV) International Technical Conference on the Enhanced Safety of Vehicles*, Stuttgart, Germany, 2009.

[5] K. Massow, *A Real-World-Testbed for Cooperative Intervening Driving Assistance Systems*, TU-Berlin, Berlin, Germany, 2008.

[6] F. Leimbach, U. Schmortte, and H. Kiebach, "Global harmonisation of test procedures for driver assistance systems," in *Proceedings of the 23rd International Technical Conference on the Enhanced Safety of Vehicles (ESV '13)*, no. 13-0234, 2013.

[7] M. During and K. Lemmer, "Cooperative maneuver planning for cooperative driving," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 3, pp. 8–22, 2016.

[8] K. Chen and J. C. Miles, "ITS handbook 2004: Recommendations from the world road association (PIARC)," 2004.

[9] N. Congress, "Automated highway system: an idea whose time has come," *Public Roads*, vol. 58, no. 1, pp. 1–8, 1994.

[10] S. Cheon, *An Overview of Automated Highway Systems (AHS) And The Social And Institutional Challenges They Face*, University of California Transportation Center, 2003.

[11] M. Wang, "Generic model predictive control framework for advanced driver assistance systems," 2014.

[12] B. Schünemann, K. Massow, and I. Radusch, "A novel approach for realistic emulation of Vehicle-2-X communication applications," in *Proceedings of the 2008 IEEE 67th Vehicular Technology Conference-Spring (VTC '08)*, pp. 2709–2713, May 2008.

[13] S. E. Shladover, "Recent International Activity in Cooperative VehicleHighway Automation Systems," No. FHWA-HRT-12-033. 2012.

[14] A. De La Fortelle, X. Qian, S. Diemer, J. Grégoire, F. Moutarde et al., *Network of Automated Vehicles: The Autonet 2030 Vision*, ITS World Congress, Detroit, United States, 2014.

[15] D. Mohr, H.-W. Kaas, P. Gao, D. Wee, and T. Moller, "Automotive revolution–perspective towards 2030: how the convergence of disruptive technology-driven trends could transform the auto industry," Tech. Rep., McKinsey & Company, 2016.

[16] SAE International Standard J3016, "Taxonomy and Definitions for Terms related to On-Road Motor Vehicle Automated Driving Systems, SAE International," 2014.

[17] S. E. Shladover, "Cooperative (rather than autonomous) vehicle-highway automation systems," *IEEE Intelligent Transportation Systems Magazine*, vol. 1, no. 1, pp. 10–19, 2009.

[18] P. Jeevan, F. Harchut, B. Mueller-Bessler, and B. Huhnke, "Realizing autonomous valet parking with automotive grade sensors," in *Proceedings of the 23rd IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems (IROS '10)*, pp. 3824–3829, Taipei, Taiwan, October 2010.

[19] J. Einsiedler, O. Sawade, B. Schäufele, M. Witzke, and I. Radusch, "Indoor micro navigation utilizing local infrastructure-based positioning," in *Proceedings of the 2012 IEEE Intelligent Vehicles Symposium (IV '12)*, pp. 993–998, Alcala de Henares, June 2012.

[20] D. Schramm, M. Hiller, and R. Bardini, *Modellbildung und Simulation der Dynamik von Kraftfahrzeugen*, vol. 124, Springer, Berlin, Germany, 2010.

[21] R. Rajamani, *Vehicle Dynamics and Control*, Springer Science & Business Media, New York, NY, USA, 2012.

[22] H. B. Pacejka, *Tire and Vehicle Dynamics*, Elsevier, 2005.

[23] Z. Papp, "Situational awareness in intelligent vehicles," in *Handbook of Intelligent Vehicles*, A. Eskandarian, Ed., pp. 62–78, Springer, 2012.

[24] H. Winner, S. Hakuli, and G. Wolf, *Handbuch Fahrerassistenzsysteme*, Vieweg+Teubner, Wiesbaden, 2009.

[25] A. Eskandarian, "Fundamentals of driver assistance," in *Handbook of Intelligent Vehicles*, A. Eskandarian, Ed., pp. 493–524, Springer, 2012.

[26] D. Krajzewicz et al., "Recent development and applications of SUMO-Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 4, 2012.

[27] R. Protzmann, K. Massow, and I. Radusch, "An evaluation environment and methodology for automotive media streaming applications," in *Proceedings of the 8th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS '14)*, pp. 297–304, Birmingham, UK, July 2014.

[28] PTV AG, "Vissim homepage," 2016, http://vision-traffic.ptvgroup.com/en-uk/products/ptv-vissim/.

[29] R. Math, "OpenDS: A new open-source driving simulator for research," GMM-Fachbericht-AmE 2013, 2013.

[30] OSS, "Driving Simulator," 2016, http://www.transportation.technologyventures.com/simwiki.

[31] SILAB, "Driving Simulation and SILAB," 2016, http://www.wivw.de/en/silab/.

[32] G. J. Heydinger, M. K. Salaani, W. R. Garrott, and P. A. Grygier, "Vehicle dynamics modelling for the National Advanced Driving Simulator," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 216, no. 4, pp. 307–318, 2002.

[33] F. Hendriks, M. Tideman, R. Pelders, R. Bours, and X. Liu, "Development tools for active safety systems: Prescan and VeHIL," in *Proceedings of the 2010 IEEE International Conference on Vehicular Electronics and Safety (ICVES '10)*, pp. 54–58, July 2010.

[34] D. Gruyer, S. Choi, C. Boussard, and B. D'Andrea-Novel, "From virtual to reality, how to prototype, test and evaluate new ADAS: application to automatic car parking," in *Proceedings of the 25th IEEE Intelligent Vehicles Symposium (IV '14)*, pp. 261–267, June 2014.

[35] D. Gruyer, O. Orfila, V. Judalet, S. Pechberti, B. Lusetti, and S. Glaser, "Proposal of a virtual and immersive 3D architecture dedicated for prototyping, test and evaluation of eco-driving applications," in *Proceedings of the 2013 IEEE Intelligent Vehicles Symposium (IEEE IV '13)*, pp. 511–518, June 2013.

[36] M. Kankaanranta and P. Neittaanmäki, "RACER: a non-commercial driving game which became a serious tool in the research of driver fatigue," in *Design and Use of Serious Games*, pp. 171–184, Springer, 2009.

[37] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner, "TORCS: The Open Racing Car Simulator," v1.3.6, 2014.

[38] F. Kehrle, J. V. Frasch, C. Kirches, and S. Sager, "Optimal control of formula 1 race cars in a VDrift based virtual environment," in *Proceedings of the 18th IFAC World Congress*, pp. 11907–11912, September 2011.

[39] U. Drolia, Z. Wang, Y. Pant, and R. Mangharam, "AutoPlug: an automotive test-bed for electronic controller unit testing and verification," in *Proceedings of the 14th IEEE International Intelligent Transportation Systems Conference (ITSC '11)*, pp. 1187–1192, October 2011.

[40] A. Vahdat, M. Al-Fares, N. Farrington, R. N. Mysore, G. Porter, and S. Radhakrishnan, "Scale-Out networking in the data center," *IEEE Micro*, vol. 30, no. 4, pp. 29–41, 2010.

[41] M. Purss et al., "Discrete Global Grid Systems SWG," 2014, http://www.opengeospatial.org/projects/groups/dggsswg.

[42] R. Featherstone, *Rigid Body Dynamics Algorithms*, Springer, New York, NY, USA, 2014.

[43] TNO, "MF-Tyre User Manual Version 5.2," 2001.

[44] A. Ortiz, J. A. Cabrera, A. J. Guerra, and A. Simon, "An easy procedure to determine Magic Formula parameters: A comparative study between the starting value optimization technique and the IMMa optimization algorithm," *Vehicle System Dynamics*, vol. 44, no. 9, pp. 689–718, 2006.

[45] M. Burckhardt, *Fahrwerktechnik: Radschlupf-Regelsysteme*, vol. 16, Vogel-Verlag, 1993.

[46] K. Hedrick, D. Godbole, R. Rajamani, and P. Seiler, "Stop and go cruise control," Tech. Rep., PATH, California, Calif, USA, 1999.

[47] L. Glielmo, L. Iannelli, V. Vacca, and F. Vasca, "Gearshift control for automated manual transmissions," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 1, pp. 17–26, 2006.

[48] H. Braess and U. Seiffert, *Vieweg Handbuch Kraftfahrzeugtechnik*, Springer, Fachmedien Wiesbaden, Germany, 2013.

[49] B. Chretien, L. Nouvelière, N. A. Oufroukh et al., "A vehicle simulator for an efficient electronic and electrical architecture design," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1967–1982, 2013.

[50] R. Hall, "A characterization of illumination models and shading techniques," *The Visual Computer*, vol. 2, no. 5, pp. 268–277, 1986.

[51] O. Sawade, B. Schaufele, and I. Radusch, "Collaboration over IEEE 802.11p to enable an intelligent traffic light function for emergency vehicles," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '16)*, February 2016.

[52] P. Dellacqua, F. Bellotti, R. Berta et al., "Safe drive map concept for road curve monitoring," in *Proceedings of the 18th Euromicro Conference on Digital System Design (DSD '15)*, pp. 293–296, August 2015.

[53] M. Haklay and P. Weber, "Openstreetmap: user-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.

[54] A. Amditis, P. Lytrivis, I. Karaseitanidis, M. Prandtstetter, and I. Radusch, "Tomorrow's transport infrastructure: From static to elastic mobility," in *Proceedings of the 20th Intelligent Transport Systems World Congress (ITS '13)*, October 2013.

[55] D. Becker, A. Munjere, J. Einsiedler, K. Massow, F. Thiele, and I. Radusch, "Blurring the border between real and virtual parking environments," in *Proceedings of the 2016 IEEE Intelligent Vehicles Symposium (IV '16)*, pp. 1205–1210, June 2016.

[56] K. Massow, B. Kwella, N. Pfeifer et al., "Deriving HD maps for highly automated driving from vehicular probe data," in *Proceedings of the 19th IEEE International Conference on Intelligent Transportation Systems (ITSC '16)*, pp. 1745–1752, Rio de Janeiro, Brazil, November 2016.

[57] B. Schünemann, "V2X simulation runtime infrastructure VSimRTI: an assessment tool to design smart traffic management systems," *Computer Networks*, vol. 55, no. 14, pp. 3189–3198, 2011.

[58] "Lightweight Java Game Library," http://www.lwjgl.org/.

[59] L. Euler, "Institutionum calculi integralis," Vol. 1. imp. Acad. imp. Saènt., 1768.

[60] R. G. Sargent, "An introductory tutorial on verification and validation of simulation models," in *Proceedings of the Winter Simulation Conference (WSC '15)*, pp. 1729–1740, December 2015.

[61] E. Kutluay and H. Winner, "Validation of vehicle dynamics simulation models—a review," *Vehicle System Dynamics*, vol. 52, no. 2, pp. 186–200, 2014.

[62] S. Sulaiman, "Modeling and validation of 7-DOF ride model for heavy vehicle," *Proceedings of the ICAMME*, 2012.

[63] G. J. Heydinger, W. R. Garrott, J. P. Chrstos, and D. A. Guenther, "A methodology for validating vehicle dynamics simulations," SAE Technical Papers 900128, 1990.

[64] E. Kutluay and H. Winner, "Assessment methodology for validation of vehicle dynamics simulations using double lane change maneuver," in *Proceedings of the 2012 Winter Simulation Conference (WSC '12)*, December 2012.

[65] E. Kutluay, *Development and Demonstration of a Validation Methodology for Vehicle Lateral Dynamics Simulation Models*, VDI-Verlag, Düsseldorf, Germany, 2013.

[66] J. P. Chrstos and P. A. Grygier, "Experimental testing of a 1994 Ford Taurus for NADSdyna validation," SAE Technical Papers 970563, 1997.

[67] D. R. Wittink, *The Application of Regression Analysis*, Allyn and Bacon, London, UK, 1988.

[68] D. Becker, F. Thiele, O. Sawade, and I. Radusch, "Cost-effective camera based ground truth for indoor localization," in *Proceedings of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM '15)*, pp. 885–890, July 2015.

[69] C. Patrick Norman, *A Method for Modeling And Prediction of Ground Vehicle Dynamics And Stability in Autonomous Systems*, Diss. Virginia Polytechnic Institute and State University, 2011.