electronics

MDPI

Article

Virtual Validation of an Automated Lane-Keeping System with an Extended Operational Design Domain

Patrick Weissensteiner [1,*], Georg Stettinger [1], Johannes Rumetshofer [1,2] and Daniel Watzenig [1,2]

[1] Virtual Vehicle Research GmbH, 8010 Graz, Austria; Georg.Stettinger@v2c2.at (G.S.); Johannes.Rumetshofer@v2c2.at (J.R.); daniel.watzenig@tugraz.at (D.W.)
[2] Institute of Automation and Control, Graz University of Technology, 8010 Graz, Austria
* Correspondence: Patrick.Weissensteiner@v2c2.at

Abstract: Virtual testing using simulation will play a significant role in future safety validation procedures for automated driving systems, as it provides the needed scalability for executing a scenario-based assessment approach. This article combines multiple essential aspects that are necessary for the virtual validation of such systems. First, a general framework that contains the vital subsystems needed for virtual validation is introduced. Secondly, the interfaces between the subsystems are explored. Additionally, the concept of model fidelities is presented and extended towards all relevant subsystems. For an automated lane-keeping system with two different definitions of an operational design domain, all relevant subsystems are defined and integrated into an overall simulation framework. The resulting difference between both operational design domains is the occurrence of lateral manoeuvres, leading to greater demands of the fidelity of the vehicle dynamics model. The simulation results support the initial assumption that by extending the operation domain, the requirements for all subsystems are subject to adaption. As an essential aspect of harmonising virtual validation frameworks, the article identifies four separate layers and their corresponding parameters. In particular, the tool-specific co-simulation capability layer is critical, as it enables model exchange through consistently defined interfaces and reduces the integration effort. The introduction of this layered architecture for virtual validation frameworks enables further cross-domain collaboration.

Keywords: virtual validation; ALKS; ODD; ADS; co-simulation; virtual environment; scenario engine; simulation framework; safety validation; self-driving cars

Citation: Weissensteiner, P.; Stettinger, G.; Rumetshofer, J.; Watzenig, D. Virtual Validation of an Automated Lane-Keeping System with an Extended Operational Design Domain. *Electronics* **2022**, *11*, 72. https://doi.org/10.3390/electronics11010072

Academic Editors: Zita Vale, John Ball and Mattia Ricco

Received: 18 November 2021
Accepted: 22 December 2021
Published: 27 December 2021

1. Introduction

Based on information from the World Health Organization, the number of traffic deaths occurring annually is rising steadily, reaching 1.35 million in 2016. However, the rate of deaths relative to the global population, which is constantly growing, has stagnated in recent years. Taking the increasing motorisation in large parts of the world into account suggests that existing safety measures have proven their effectiveness [1]. The shift towards advanced driving assistance systems (ADAS) taking over the responsibilities of the driver has played a vital role in increasing safety for vehicle passengers and traffic participants, especially for vulnerable road users such as pedestrians and cyclists [2].

The Society of Automotive Engineers (SAE) introduced a taxonomy for automation levels of ADAS and ADS. The six SAE levels define the degree of automation concerning a vehicles' driving task. SAE levels 0–2 are defined as partial automation, which means systems (e.g., various ADAS available on the market) take over part of the dynamic driving task (DDT) but the driver has to monitor the environment all the time and intervene if necessary. The DDT incorporates all real-time operational and tactical functions required for on-road vehicle operation. For SAE Levels 3+, an automated driving system (ADS), which incorporates the necessary hard- and software, is introduced. In this article, this is also referred to as automated driving (AD) function. The ADS should be capable of performing the entire DDT over a sustainable time period. This includes monitoring

the driving environment, which involves the detection of objects and particular events (e.g., the braking of other detected traffic participants) and giving a response. This is referred to as the object and event detection and response (OEDR) subtask of the DDT. Essentially, this combines the task of monitoring the entire relevant driving environment and detecting potential elements that the ADS needs to react to. In the case of a SAE Level 3 ADS, the human driver acts as a fallback. For the operation of ADS, the concept of the operational design domain (ODD) is introduced. The ODD describes a set of conditions under which the ADS can be activated and operate in [3–5]. Apart from the safety aspects, ADS can also help to reduce emissions and congestion, increase the driving comfort of vehicle passengers, and enable new business models [6].

In Figure 1, an overview of the relation between the concepts introduced in this chapter is given. ADS are seen as potential solutions for reducing the amount of traffic deaths. For the validation of such systems, a scenario-based safety assessment is a promising approach. Such approaches are enabled using appropriate simulation frameworks. In the following subsections (Sections 1.1 and 1.2), the safety assessment of ADS as well as approaches for virtual testing are introduced.

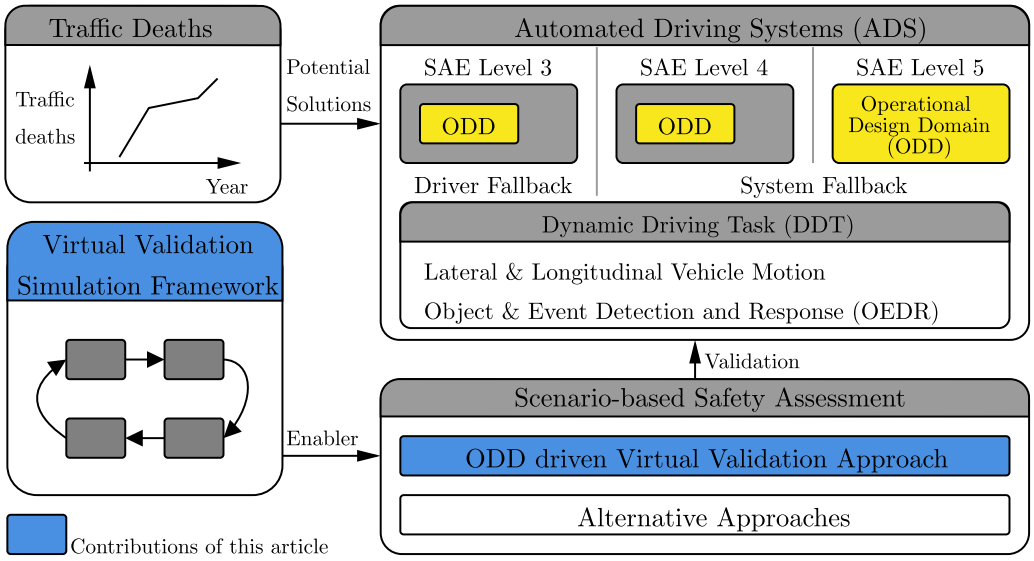


Figure 1. Overview of the relation between the concepts introduced in Section 1.

1.1. Safety Assessment for ADS

Due to the open parameter space of the ODD (e.g., road traffic), an infinite number of traffic situations can occur, which makes absolute proof for the safety of ADS impossible. The RAND Cooperation states that an ADS would have to be driven hundreds of million miles on public roads to demonstrate its safety reliability compared to human drivers [2]. In the case of a software update or functional extension of the ADS, this process would have to be repeated, which is unfeasible from a time and cost perspective. Currently, this is one of the major factors hindering the introduction of ADS in public roads. However, various safety assessment approaches for ADS [7–9] have emerged. Most prominent is the scenario-based approach, which uses the definition of a scenario that was originally introduced by [10]. Essentially, a scenario is a sequence of actions and events triggered by defined traffic participants. The scenario-based approach should reduce the otherwise infinite test scope to a reasonable number of scenarios. It only considers scenarios of interest that are defined and omits the parts with no significant actions. A six-layer concept expands the scenario concept (see [11], originally introduced as a five-layer model [12]), which classifies all relevant parameters to describe the necessary elements of a scenario in different layers.

Various research projects and initiatives concerning the safety validation of ADS were initiated around the globe. In Japan [13,14] and in Europe [15–17], projects emerged with a strong focus on scenario-based assessment approaches towards a safety validation of ADS. In the US, there are various NHTSA reports [5,18] and other ongoing actions dealing with the described topics [19]. Other initiatives include releasing national standards as guidance for the safe development and rollout of automated vehicles [20,21]. Currently, there is only one regulation from the United Nations (UN) that deals with ADS; this is a uniform provision concerning the approval of automated lane-keeping systems (ALKS) [22]. The ALKS is defined as an ADS that can be activated if a set of conditions is met, including highways and a maximum operational speed of 60 km/h. The regulation is intended for passenger cars. An implementation of an ALKS will be used as an ADS for the simulation study presented in this article.

Even if the number of concrete tests for the safety validation of ADS can be reduced by introducing a scenario-based assessment, the execution of all these scenarios in the real world is not feasible. Therefore, testing methods with a certain degree of virtualisation using X-in-the-loop (XIL) are introduced into the validation procedures. This article focuses on an entirely virtual test setup (virtual testing) as part of the safety validation of ADS, which is referred to as virtual validation. This means that all aspects of the ADS software stack are tested in a simulation environment [5].

1.2. Virtual Testing of ADS for Safety Assessment

Current literature suggests that virtual testing using simulation plays a dominant role in future safety validation processes for ADS, as it provides scalability. However, major challenges are associated with the increasing complexity of integrating all necessary subsystems into a complete simulation framework. Furthermore, such a virtual test environment depends on the simulation's scope and goals and the tested system. In general, a generic ADS can be seen as concatenation of the four subsystems *sensing*, *perception*, *planning* and *control*. Those four subsystems perceive the environment with respective sensors (*sensing*), interpret it (*perception*), and output a perceived environment to the *planning* subsystem. This perceived environment can consist of an object list (dynamic and static objects in the environment) and other types of information (e.g., lane markings). Based on this, a certain target actuation is defined by the *control* subsystem using a trajectory as an input [5]. Hence, it is also possible to use a simulation to only test certain subsystems—e.g., the *perception* based on pre-recorded sensor data. However, for a general safety assessment of ADS, the complete chain of ADS subsystems must be in effect. Therefore, the real environment is exchanged with a virtual one. As this serves as an input to the *sensing* subsystem, the virtual environment needs to be able to represent the necessary elements affecting the respective sensors. As a replacement for the actual sensor hardware, sensor models are introduced. They simulate actual sensor behaviour based on the input from the environment simulation. Sensor models either output raw data (e.g., point clouds) as an input for the *perception* subsystem or object lists. In the case of the object list output, certain tasks of the *perception* subsystem are already accomplished inside the respective sensor model. This could be the case if specific sensors are modelled, where the real hardware also directly outputs object lists instead of raw data streams. An overview of the current state of the art regarding sensor models can be found in [23]. Finally, the simulation loop is closed by integrating the vehicle dynamics; it then updates the position of the ego vehicle in the virtual environment. This environment simulation is usually controlled by a scenario engine using scenario descriptions as inputs.

Apart from the research gaps in sensor modelling described in [23], providing the respective input to these sensor models is another primary task. Specific tools have emerged that enable the creation of a virtual environment, often including the simulation of accurate sensor models directly inside the environment. Additionally, these tools need to provide ways to interpret scenario descriptions and execute respective test cases. For external sensor models that are not directly part of the environment simulation, it is essential that all required information from the virtual environment can be extracted as

ground truth during the simulation runtime. Examples of such tools are Vires VTD [24], aiSim [25], and IPG CarMaker [26]. An open-source solution that initially emerged as a way to generate synthetic data for the training and testing of perception algorithms is CARLA [27]. Currently, this is extended to provide means for use in safety assessment procedures for ADS and is also used for the proposed simulation framework presented in this article.

Since simulation models are generally simplified versions of real-world interactions, certain errors are expected to be introduced into the overall simulation. Therefore, means to deal with this uncertainty quantification and model verification and validation are of utmost importance, especially if ADS-equipped vehicles' safety is validated using virtual testing [28]. In general, simulating the behaviour of automated vehicles, especially under challenging operating conditions, is a task in which different models and tools from different domains (e.g., vehicle dynamics and sensor modelling) need to be combined. For this cross-domain challenge, a co-simulation approach is needed [5,29]. Such an approach is presented in Section 3 of this article.

1.3. Scope of Work

This article deals with a validation approach for an ALKS using virtual testing. For the ALKS, two different versions of potential ODD definitions are looked upon and the respective simulation frameworks are designed. This includes deriving the required model fidelities for all relevant subsystems of the proposed virtual validation framework and their implementation in an overall simulation architecture. The simulation results are compared for both ODD versions and their respective frameworks. In the end, a multiple-layer approach for a general framework for virtual validation is proposed. The layers themselves are explained in detail, and potential future research areas are discussed.

Essentially, the two main contributions of this article are the following (also marked in blue in Figure 1):

- An ODD-driven virtual validation approach;
- A virtual validation simulation framework.

A more detailed explanation of the concrete contributions of this article is given in Section 6.

1.4. Structure of the Article

Section 2 introduces the approach for virtual validation regarding an ALKS in detail. This also includes an overview of the general simulation framework for virtual validation, including all relevant subsystems. Furthermore, both ODD versions for the ALKS are explained. In Section 3, both frameworks for the standard and an extended ODD are presented. Section 4 evaluates the results obtained for both ODD versions and frameworks. Section 5 introduces the concept of the layered simulation framework as a consequence of the two different ODDs that were analysed in Section 3. Furthermore, the interfaces are discussed in detail, including relevant parameters. Section 6 provides a discussion of the simulation results and the major contributions of this article. Additionally, future research areas are presented. An overview of the different chapters of this article can be seen in Figure 2.

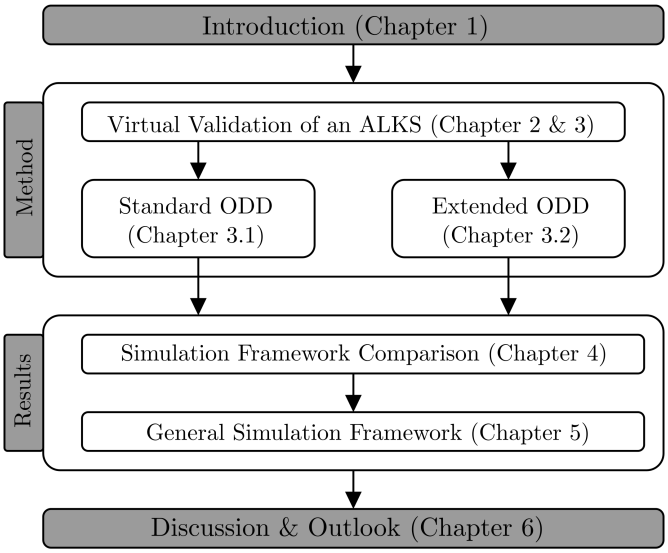


Figure 2. Graphical representation of the different chapters in this article.

2. Virtual Validation of an ALKS

Virtual validation is an essential part of every ADS safety validation procedure. This chapter aims to explain the relevant definitions and implemented versions of an ALKS. A general overview of the necessary simulation framework for the virtual validation of these ADS and the required subsystems, including the appropriate interfaces, is given.

The concept of an ODD for ADS was first introduced by the SAE together with the taxonomy on automation levels as a definition under which conditions an ADS has to take over the DDT [3]. Although certain literature on defining the ODD for particular aspects (e.g., best practices, definitions for specific studies) exists [30–33], currently there is only one specification for an ODD taxonomy available [34]. The taxonomy uses the scenery (e.g., drivable area), the environmental conditions (e.g., weather), and the dynamic elements (e.g., traffic) as the main categories. Furthermore, it is used as a basis for developing an international standard regarding an ODD taxonomy [35]. The proposed ODD structures, including categories and subcategories, can be seen in Figure 3.

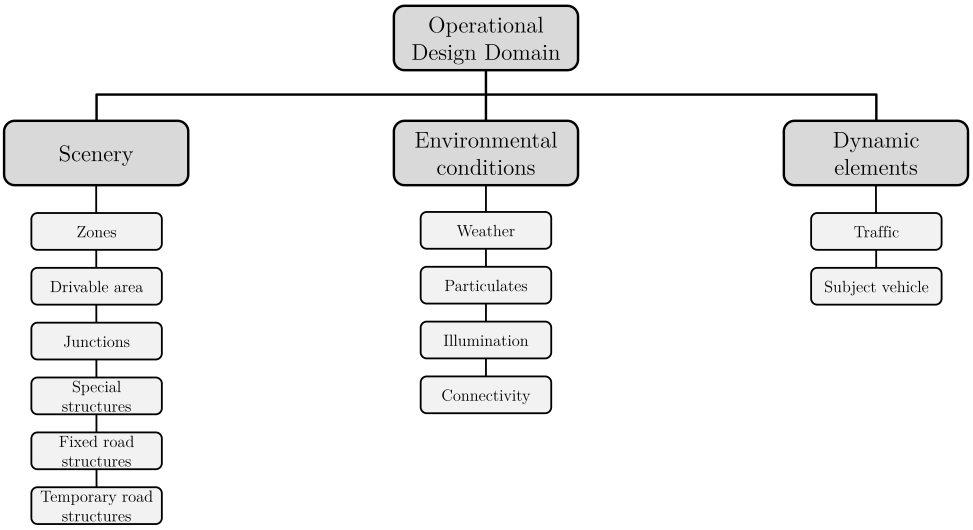


Figure 3. ODD categories and subcategories as defined in [34].

Most of the required information for the ODD definition of an ALKS can be directly extracted from the respective UN regulation [22]. A graphical representation of this information can be seen in the green area in Figure 4. The ego vehicle, equipped with an ALKS,

needs at least 46 m of forwarding detection range and a lateral detection range of one lane to each side of the ego vehicle. In Figure 4, this is represented as a yellow area, with vehicles inside (green vehicles) and outside the sensing area (grey vehicles). Furthermore, the ego vehicles' velocity is restricted to a maximum of 60 km/h. Based on the respective UN regulation definition, an ALKS can only be activated on roads where pedestrians and cyclists are prohibited. A physical separation that divides the traffic moving in the opposite direction must be present. Based on the ODD taxonomy defined in [34], certain aspects of the ODD are not stated directly in the UN regulation. This mainly concerns the weather conditions, as these are only indirectly covered by defining a minimum forward detection range. Therefore, the allowed weather conditions depend on the sensor setup of the ego vehicle and the respective capabilities of the implemented ALKS to deal with such adverse conditions. For this article, a general description of an ALKS ODD is used to derive the required simulation framework. Secondly, the ODD is extended to include construction zone typical lane offsets, compared to the original lane guidance. This will be further explained in Section 3. Various other potential ODD extensions are, however, beyond the scope of this article. This includes external knowledge from communications such as vehicle-to-X (V2X), the consideration of adverse weather conditions, and raising the allowed speed limit for the ego vehicle (red area in Figure 4).

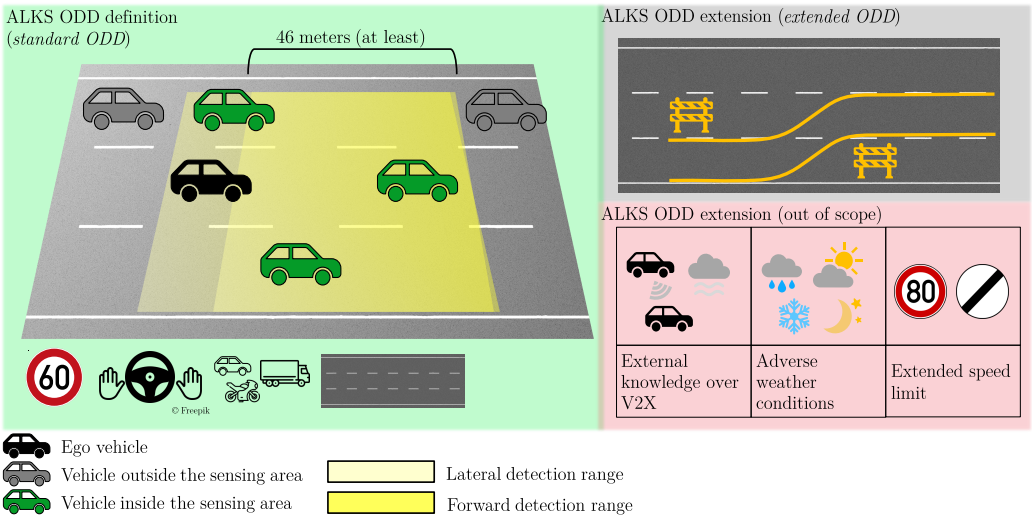


Figure 4. Overview of the required ODD for an ALKS, including potential extensions and out-of-scope elements.

Evaluating a specific implementation of an ALKS is out of the scope of this article. Nevertheless, it is mandatory to use such a function to showcase the dependence of the virtual validation framework on the specified ODD of the ALKS function. Therefore, two different options for an ALKS were implemented; see Table 1. The function acts as a combination of lane-keeping and adaptive cruise control (ACC) for both options. The first option, map-based ALKS, implements a lane-keeping functionality based on a pre-captured map of the motorway. The considered motorway is the A2 near Graz (Austria), which was digitalised as part of an official test route for the testing of highly automated driving [36]. The digital version of the motorway is available as ASAM OpenDRIVE® map [37].

The second option, the perception-based ALKS, adds a perception task to the function by performing lane-keeping based on lane marking information gathered during runtime. Concretely, a sensor model that represents the behaviour and output of an Intel Mobileye® [38] camera-based system is implemented. The perception-based ALKS, therefore, needs to reconstruct relevant lane markings purely based on the sensor input. These two options provide the needed flexibility during the development of the simulation framework, especially for the ODD extension described in Section 3.

Table 1. Implemented options of the utilised ALKS-based ADS.

ALKS Options	Details
Map-based ALKS	Lane-keeping based on a pre-captured map of the motorway
Perception-based ALKS	Lane-keeping based on lane marking information gathered during runtime

Using virtual testing as one of the main test methods for the safety validation of ADS requires an adequate simulation framework that incorporates all relevant subsystems. In general, there are already many different simulation frameworks, each of which usually has a specific purpose. Therefore, combining the various aspects to a more complete framework, including the interfaces, makes sense. This framework can be seen in Figure 5 and is an extension of the architecture provided in [39,40] for an automated shuttle use case.

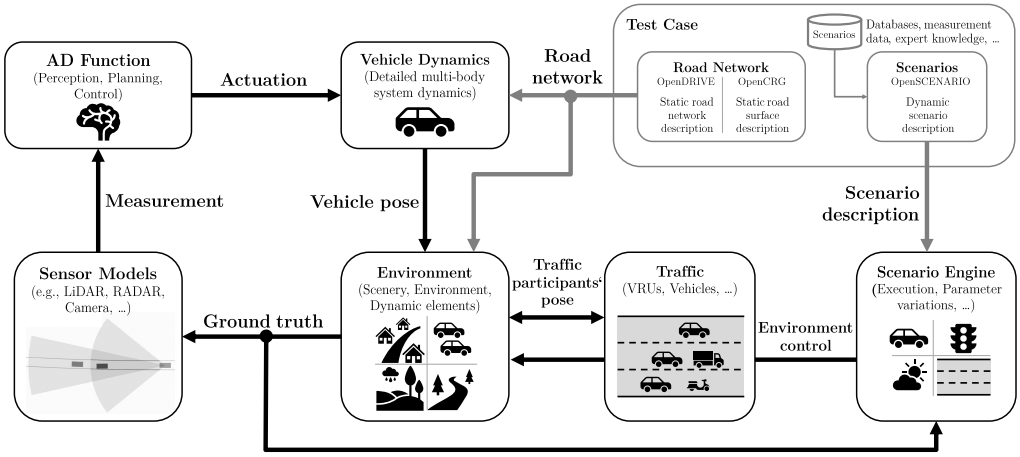


Figure 5. Overview of the necessary subsystems for a virtual validation framework for ADS.

The four main blocks, which are essential and represent the core of every architecture for the virtual validation of ADS, are the ADS itself (AD function), the vehicle dynamics of the ego vehicle, the virtual environment (representing the defined ODD), and the sensor models. Sensor models use the ideal ground truth input from the virtual environment and calculate the output based on the modelled behaviour of actual sensors. The output is either data that have already been processed (e.g., object lists for detected traffic participants, lane markings) or raw sensor data. An overview of the state of the art regarding sensor models can be found in [23]. A detailed explanation of the exact interaction of sensor models with the rest of a virtual validation framework can be found in [41]. A more detailed explanation of those four blocks can be found in [39]. Generally, these four main blocks do not need to be executed in a purely virtual manner. Certain combinations of real-world and virtual elements lead to different X-in-the-loop testing architectures, such as vehicle-in-the-loop, as presented in [42,43], or hybrid testing (see [44]). For the scope of this article, the focus is purely on virtual testing.

For the evaluation of cooperative functions and for the generation of required test scenarios, including a traffic simulation (as shown in [45–49]), a solution is possible. This can either be an open-source solution (e.g., [50]) or a traffic block representing a proprietary software, as seen in Figure 5. The last block that is active during runtime (all relevant black blocks in Figure 5) is the scenario engine. This uses the scenario description provided as part of the executed test case to control the virtual environment. This includes the traffic participants (TP), the weather, and various other aspects of the environment. Depending on the exact implementation of the virtual validation framework, the scenario engine is either part of the virtual environment (e.g., [51]) or a separate library (e.g., [27]).

The greyed-out block represents the test case and is not active during runtime, but rather provides the necessary input to other blocks before executing a simulation run. For a test case, the scenario description (e.g., ASAM OpenSCENARIO® [52]) and the respective road network (e.g., ASAM OpenDRIVE®) need to be defined. Additional

parameters, such as the coupling stepsize of the co-simulation, are also part of the test case description. Another possible file format that is part of the test case definition could be ASAM OpenCRG® [53]. This file format defines a detailed description of the road surface. Therefore, it can be seen as an additional layer of fidelity for the general modelling of the road, as a static part of the virtual environment.

Having defined the necessary subsystems for a virtual validation framework leads to considering the respective interfaces, especially when bearing a concrete implementation of such a framework in mind. An overview of the interfaces, their respective signals, and applicable standards for the definition of the interface can be seen given in Table 2. It can be seen that no applicable standard is available for the actuation signals (output of the AD function and input for the ego vehicle dynamics) and the pose signals of certain traffic participants in the simulation (including the ego vehicle). This also applies to environment control, which contains various signals used to control relevant traffic participants and the weather. However, for the road network and the scenario description, as already mentioned in the explanation of Figure 5, the ASAM standards OpenDRIVE® and OpenSCENARIO® can be applied. The relevant interfaces concerning the sensor models, input and output, can be defined using the open simulation interface as presented in [54] and implemented in [55]. This is currently being further developed as the ASAM OSI® standard [56]. This interface was initially designed for the exchange between the environment simulation and the sensor models. However, it is also possible to further extend the concept to include various other interface definitions between relevant subsystems of a virtual validation simulation framework.

Table 2. Interfaces of the virtual validation framework.

Interface	Signals	Applicable Standard
Actuation	Throttle, brake, steering angle	N/A
Measurement	Sensor measurement	ASAM OSI®
Ground truth	Environment ground truth	ASAM OSI®
Vehicle pose	Global pose of ego vehicle	N/A
Road network	Defined road network	ASAM OpenDRIVE®
TP pose	Global pose of TP	N/A
Environment control	Signals for TP & weather control	N/A
Scenario description	Scenario description	ASAM OpenSCENARIO®

Different model fidelities are possible for three of the four main blocks (excluding the AD function, as this is the system that needs to be validated). The model fidelity is categorised into low, medium, and high fidelity. The high-fidelity model gives the most accurate representation of the respective subsystem but is usually associated with the greatest computational effort. This categorisation mainly stems from the sensor model domain (see [23]), but can also be mapped to the other subsystems. Table 3 lists examples for each category and subsystem. In the case of the sensor model subsystem, low fidelity refers to ideal models that generate an environment model (e.g., object lists for detected traffic participants) based on ground truth data from the simulation environment, only taking occlusion into account (e.g., [57]). A medium-fidelity implementation of sensor models is usually referred to as a probabilistic model, as it adds statistical failure rates and therefore modifies the object list entries (e.g., [58,59]). Physical sensor models have a high model fidelity, as they are directly based on physical principles of the respective sensor type (e.g., [60]). For the environment model, a low model fidelity is characterised by placing objects and updating their respective pose. However, it could be sufficient only to represent this in 2D. In the medium-fidelity case, the representation of objects is then extended to 3D. An actual physics-based rendering engine (e.g., enabling ray tracing) is included for the high-fidelity model. For the vehicle dynamics subsystem, the lowest possible model fidelity is represented by a point mass model. For the next stage, the medium-fidelity single-track or double-track model (either including or excluding a dedicated tyre model) fits into this category, as it is possible to display specific characteristics of the cars' driving behaviour. Multibody vehicle models (including tyre models) with many

degrees of freedom are categorised as high fidelity. A recent comparison on the impact of different fidelity for vehicle dynamics models is given in [61]. A more detailed taxonomy for the standardisation of vehicle dynamics simulation models for passenger cars and their requirements in specific driving manoeuvres, down to the component level, is currently given in the ISO standard [62]. An overview of vehicle dynamics model fidelities is presented in [5].

Table 3. Different model fidelities for the three relevant subsystems.

Model Fidelity	Low	Medium	High
Sensor model	Object list-based. Based on ground truth data from simulation environment in the FOV (e.g., [57,63])	Based on ideal models adding statistical failure rates, modified object list entries (e.g., [58,59])	Based on the physical principles of the respective sensor type (e.g., [60])
Environment model	Able to place objects and update their pose, 2D representation of the scene	3D representation of objects, no physics-rendering engine	Physics-based rendering engine enabling ray tracing
Vehicle dynamics model	Point mass model	Single track vehicle model, double track vehicle model (excluding or including a dedicated tyre model)	Multibody vehicle model (including a tyre model)

Before the actual implementation of the simulation framework for the virtual validation of an ALKS, as presented in Section 3, the concrete requirements for each subsystem are derived from the defined ODD. This includes the standard ODD (green area of Figure 4) as well as the extension towards a lane offset (grey area of Figure 4). The requirements for the four main subsystems, as presented in Figure 5, are displayed in Table 4. For the AD function, the mentioned subcategories are based on [18]. The tactical and operational manoeuvres and the relevant event and response pairs of the object and event detection and response (OEDR) (which is a subtask of the DDT; see [3]) are crucial not only as requirements for the function itself but also for the other subsystems. The possible manoeuvres of the function have a direct influence on the fidelity of the vehicle dynamics model. In the case of the standard ODD, the implemented vehicle dynamics require a dynamic model that takes the friction between the road and tires into account. For the extended ODD, as more challenging lateral manoeuvres are possible, a model displaying the car's lateral behaviour is required. In contrast, the ability to detect and react to vehicles in front leads to necessary capabilities for the sensor models. Based on the UN regulation for ALKS presented in [22], very concrete requirements for the virtual environment are derived. First, lane markings are required, as they define the feasible space for the ADS. As a direct consequence, this means that roads, including respective lanes, need to be included in the virtual environment. Secondly, as other traffic participants are part of the ODD, they need to be represented virtually. Additionally, the roads included in the virtual environment need to be georeferenced if they are derived directly from the real world and not generated generically. This means the road sections that are part of the virtual environment have the exact same longitudinal and lateral coordinates as their respective real-world sections. Therefore, real-world measurements from this section can be directly used in simulations—e.g., to perform cross-validation checks. Most importantly, the sensor model subsystem needs to respect the potential occlusion of traffic participants (induced either by other traffic participants or the static scenery) and needs to have a 3D field of view (FOV) in which the respective traffic participants are detected. These requirements are a direct consequence of the OEDR capabilities. They lead to specific traffic scenarios (e.g., cut in or cut out) where both mentioned requirements are essential.

Table 4. Requirements for the different subsystems of the virtual validation framework for the ALKS with a standard and extended ODD.

Subsystem	Requirements
AD function	Tactical and operational maneuvers: maintain speed, car following, lane-keeping. OEDR: relevant event/response pairs from [18]. Minimum risk manoeuvre: based on [22].
Vehicle dynamics	Dynamic model taking the friction between the road and tires into account (standard ODD). Medium-/high-fidelity model used to display the correct lateral behaviour of the car (extended ODD).
Environment (virtual)	Lane markings (therefore also road, lanes, lane width, road curvature, elevation, lateral profile). Other traffic participants (cars, trucks, motorcycles) GPS -> Georeference (based on the needed environment information for ADS).
Sensor models	Occlusion behavior, 3D-FOV (based on the necessary traffic scenarios).

3. Virtual Validation of an ALKS with Varying ODD

After introducing the model fidelity concept and the virtual validation framework, including the relevant subsystems, the concrete implementation of such a framework for the ALKS under consideration is presented in this chapter. The reduced framework can be seen in Figure 6 and contains the vehicle dynamics (of the ego vehicle), the environment simulation (containing the virtual environment and the scenario engine), the sensor models, and the AD function (implementend as ALKS based on Table 1). Figure 7 shows the mapping of the different model fidelities to the respective subsystems. In the first step, the chosen models and tools for the two different ODDs are elaborated. In the second step, the framework for the extended ODD is explained in a more detailed manner. The simulated scenarios and the individual results are then presented in Section 4.

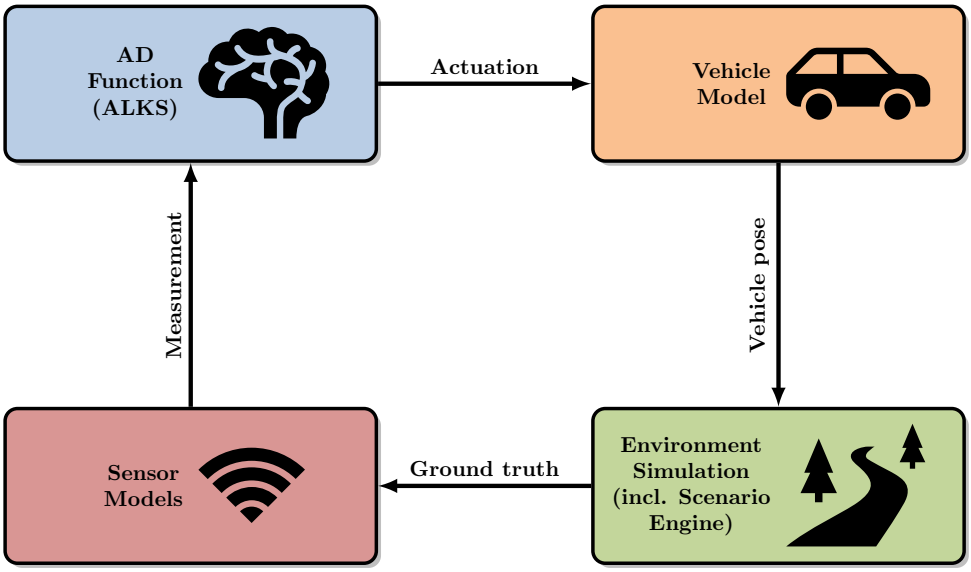


Figure 6. Reduced virtual validation framework for the ALKS with the respective ODDs.

3.1. Virtual Validation Framework for the ALKS with a Standard ODD

In the following subchapters, each implemented subsystem is briefly explained. Further details on the implementation are provided in Section 3.2.

3.1.1. AD Function

The implemented AD function fulfils all the necessary requirements defined in Table 4. It is purely implemented in Python and has a modular structure that incorporates all

relevant blocks necessary for the ALKS function, enabling the rapid prototyping and development of AD functions, with a strong focus on the subtasks of path planning and control. Furthermore, it can be used to benchmark various algorithms in the planning and control domain, making use of generic interfaces between the mentioned elements [64]. Concretely, this AD development framework generally consists of the actual AD function and a respective simulation plant. The AD function, as mentioned, has a block-based structure consisting of the most important subtask of such a function (this includes perception, state estimation, decision making, trajectory planning, and tracking). The simulation plant is essentially a low-fidelity Python-based implementation of the necessary blocks (vehicle dynamics, virtual environment, and sensor models) in order to gain a corresponding closed-loop architecture for efficient development. Only the AD function without the simulation plant was used for the virtual validation framework discussed in this article. Both ALKS options, as presented in Table 1, were implemented; however, for the final simulations, only the perception-based ALKS was used. This function uses the environment information from the sensor models (traffic participants and lane markings) to calculate the longitudinal and lateral actuation of the ego vehicle. The implementation of the AD function corresponds to the medium-fidelity category, as not all elements stated in [22] (e.g., the minimum risk manoeuvre) are considered.

3.1.2. Sensor Models

For the sensor models necessary for object detection, a low-fidelity implementation was used that is based on [63]. The implementation was carried out in Python. Only the calculation of the occlusion between traffic participants and other parts of the virtual environment was developed in C++, as this is associated with a high computational effort. This sensor model takes a ground truth object list as an input and outputs the respective detected objects inside the 3D-FOV. The FOV can be adjusted for its range and opening angle (in two dimensions). For the lane markings, an Intel Mobileye®-like sensor was implemented, as briefly explained for the perception-based ALKS version in Section 2. For that, points along the lane markings of the dedicated lane in the virtual environment are extracted during runtime and provided as input to the sensor model. Using this input, the model calculates the coefficients and domains of cubic polynomials, modelling the left and right borders. for the left and right borders. The perception block of the AD function uses this information to reconstruct the drivable area for the vehicle in consecutive time steps.

3.1.3. Virtual Environment

For the virtual environment, the baseline was the ASAM OpenDRIVE® map from the A2 motorway near Graz, Austria (see [36]). The map was imported into the tool *RoadRunner* from Mathworks®; see Figure 8. The tool provides an interactive editor and the ability to design 3D scenes for ADS testing. Most importantly, it enables the generation of a 3D environment using an ASAM OpenDRIVE® map as a basis. The *RoadRunner* tool has an exporting option specifically for the open urban driving simulator CARLA [27], which was utilised, as CARLA was the tool of choice for the virtual environment. Furthermore, the Python-based library *scenario_runner* [65] was used as well, as this is the implementation of a scenario engine capable of interpreting ASAM OpenSCENARIO® files and executing the respective scenario directly in CARLA. The library was slightly adapted to enable the correct interpretation and execution of the scenarios described in Section 4. The exported 3D model of the virtual environment based on the provided ASAM OpenDRIVE® map was then imported using CARLA's pipeline to generate virtual environments. Thereby, both the 3D model and the ASAM OpenDRIVE® are needed to create a usable CARLA map, as the navigation of traffic agents in CARLA (and various other pieces of information such as the road and lane identifiers) are directly extracted from the underlying road network description. Generally, this virtual environment is much more detailed than would have been necessary based on the requirements of the ALKS (compare the required model fidelity in Figure 7), since the implemented sensor models are all operating on an object list level.

However, this can be seen as an intermediate step. Once medium- or high-fidelity sensor models are introduced into the framework, more detailed information from the virtual environment during runtime is necessary.

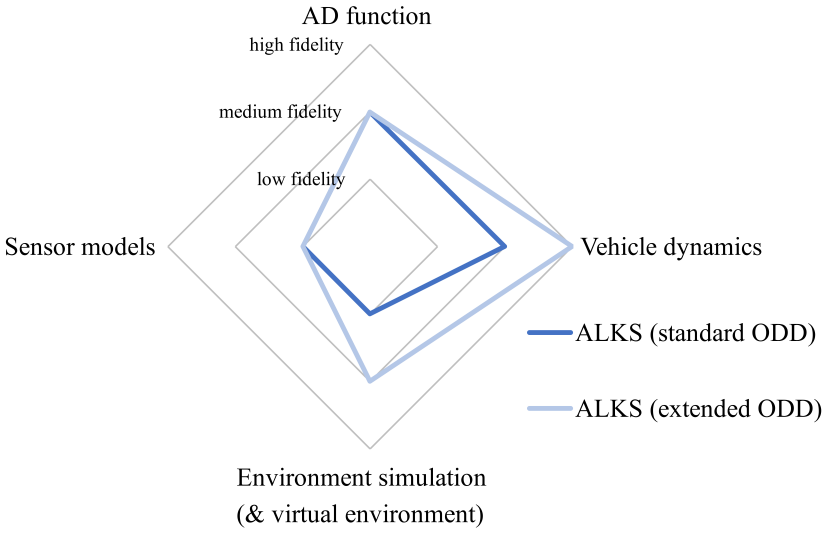


Figure 7. Visual representation of the required model fidelities for the different subsystems of the virtual validation framework.

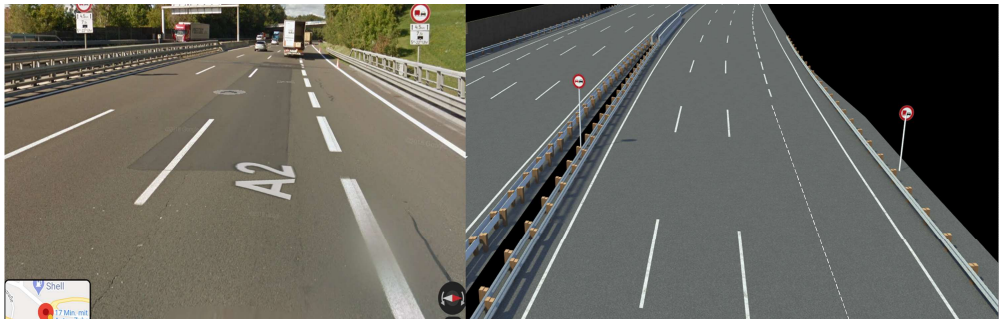


Figure 8. Comparison between the real (**left**) and virtual (**right**) environment on a motorway section near Graz, Austria.

3.1.4. Vehicle Dynamics

For the ALKS with the standard ODD, the requirements of the vehicle dynamics are manageable. The manoeuvres of the ego vehicle are mostly focussed on the longitudinal motion (e.g., braking) with some lateral motion due to the lane-keeping, since rapid steering movements because of narrow curves or other influences from the ODD are excluded. Therefore, the implemented vehicle dynamics model is a single-track model with a dedicated tyre model, which is part of the simulation plant of the AD function development framework mentioned in Section 3.1.1. This is purely implemented in Python and uses many of the parameters from [66].

3.2. Virtual Validation Framework for the ALKS with an Extended ODD

Here, the virtual validation framework to simulate the ALKS with the extended ODD is discussed. Compared to the standard ODD, lane offsets are included, which leads to additional requirements for the model fidelity of the subsystems.

3.2.1. AD Function & Sensor Models

The AD function is the same for the framework with the extended ODD. This assumption is coherent, as the ALKS has no added functionality per se, as it is technically

still a combination of lane-keeping and ACC. However, the lane-keeping task is certainly more demanding for the added construction zone situation. Once again, the implemented sensor models are equivalent because there are no added requirements for the sensor model subsystem from the ODD extension.

3.2.2. Virtual Environment

The virtual environment is alternated compared to the one used for the ALKS with the standard ODD, which directly reflects a real-world motorway section. For the extension of the ODD, the virtual environment was adapted using the Mathworks®-*RoadRunner* tool. The tool enables introducing a lane offset, similar to a typical situation presented by construction work on highways, which influenced not only the 3D model but also the underlying road network (ASAM OpenDRIVE®). The altered virtual environment can be seen in Figure 9. As this requires the adaption of the original ASAM OpenDRIVE®, ideally using a relevant tool, the needed virtual environment for this extended ODD is classified as being medium-fidelity.

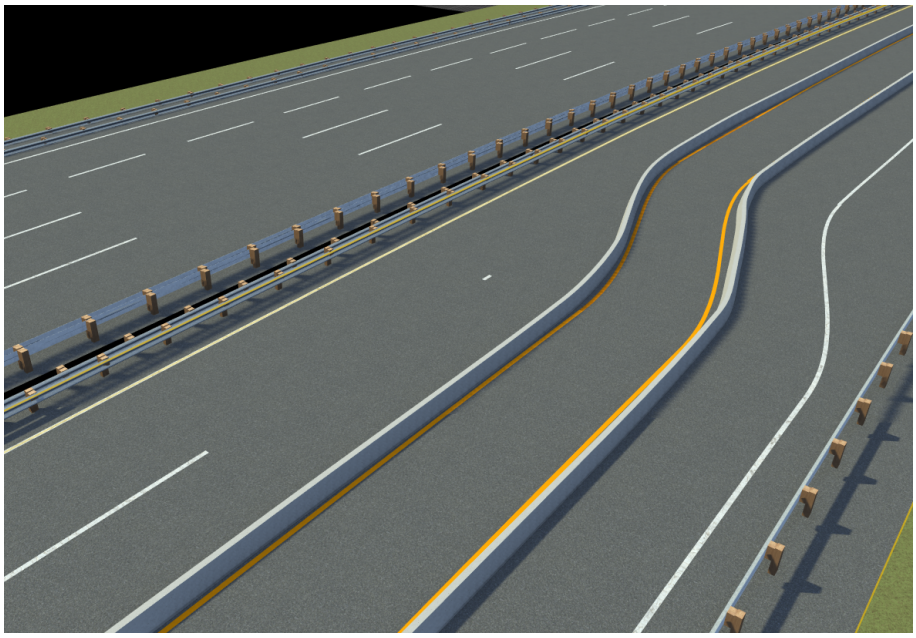


Figure 9. Altered virtual environment, which includes a construction zone typical lane offset.

3.2.3. Vehicle Dynamics

Considering that the ego vehicle has to go through much more lateral motion for the extended ODD considering the already described lane offsets, only a high-fidelity vehicle dynamics model can fulfil these requirements. Therefore, the software IPG CarMaker, which provides realistic behaviour to the limits of vehicle dynamics, was implemented in the framework for the extended ALKS ODD. It has an efficient implementation of a multibody system, which is non-linear and real-time capable [26]. Furthermore, it gives access to various pre-defined models for passenger vehicles, suitable for simulation studies as conducted for this article. Additionally, it is possible to import ASAM OpenDRIVE® maps for the road layout. This is an essential requirement, as it allows for the same road layout in both the vehicle dynamics tool and the virtual environment.

3.2.4. Implementation Details

For the concrete implementation, the case of the extended ODD for the ALKS is explained in detail. All the discussed aspects are also valid for the framework with the standard ODD, as the only difference between both setups is the vehicle dynamics model used. For the virtual environment, which in principle also differs between the two framework versions, nothing changes from the perspective of the interfaces or the actual integration in

the framework itself. In general, coupling all the previously discussed main subsystems into an overall framework to simulate them is called co-simulation. This is understood as the distributed modelling and simulation of multiple subsystems, which form a more extensive, coupled system [5]. Each subsystem is thereby defined as a dynamic system. Such a system is characterised by an internal state (state variables) and a notion of how this state evolves. It is furthermore possible to have dynamic systems that only have discrete states (e.g., traffic lights) [29]. Approaching the implementation of the simulation framework following co-simulation methods leads to multiple exploitable advantages. One concrete example would be the possibility for the reliable distribution of one or numerous subsystems to other locations and computers (remote setup). Ultimately, this leads to the necessary deterministic behaviour of the simulation framework. This characteristic is essential, as it enables the generation of reliable test results using virtual testing. The software AVL Model.CONNECT^{TM} was used for the co-simulation setup [67]. Internally, it uses model-based coupling approaches to enable an efficient co-simulation with discrete time steps of the various dynamic systems with minimum induced coupling errors [68,69]. Using the independent co-simulation platform (ICOS), it is possible to integrate the AD function, the sensor models, and CARLA (combining the virtual environment and the scenario engine in one block) with the respective Python API [70]. With AVL Model.CONNECT^{TM} it becomes possible to integrate various domain-specific tools into a co-simulation setup. This is exploited in the integration of the vehicle dynamics tool IPG CarMaker. The structure of the complete co-simulation setup for the ALKS with extended ODD can be seen in Figure 10. The function block before the IPG CarMaker block distinguishes between the various internal CarMaker states regarding the driving mode of the ego vehicle (e.g., initial calculation or drive mode). Therefore, a direct feedback loop from the IPG CarMaker block to the function block was introduced. This feedback loop enables to efficiently operate the complete framework and even interchange the vehicle dynamics block with the already explained Python-based approach introduced in Section 3.1 as the interfaces for the input (the actuation) and the output (the vehicle states) are the same. The interfaces between the other subsystems are implemented as already explained for the general structure shown in Figure 5. The scenario engine from CARLA was extended to allow it to interact and exchange information during runtime directly. Another relevant aspect is the subsystem scheduling, which defines the order of each timestep calculation between the different subsystems of the co-simulation setup. In this case, a sequential order, starting with the vehicle dynamics, is chosen. This scheduling also leads to minimised coupling errors between the subsystems, as only the input of the first subsystem needs to be extrapolated. The chosen parameters for the co-simulation are presented in Section 4.

Another perspective of the co-simulation setup, where the sequences between the different subsystems during runtime are displayed, can be seen in Figure 11.

The co-simulation instance from AVL Model.CONNECT^{TM} acts as the simulation master generating a co-simulation test case. That instance sends the scenario description to the vehicle dynamics subsystem and to the environment simulation. In both cases, it is necessary to know the initial values of the ego vehicle (e.g., starting position and velocity) that are stated in the scenario. The other information in the scenario (e.g., trajectories of other traffic participants) is vital for the environment simulation. The environment simulation then creates the scenario engine process based on the given scenario description. Furthermore, the configuration of the ground truth extraction is defined. This includes details of the lane marking extraction and if the extraction mechanism should consist of static objects. The sensor setup specified in the test case is used for the sensor model instance, which includes the concrete sensor specification (e.g., FOV and range). The AD function is instantiated with the information on the ego vehicle's target velocity and end location. This is also defined in the test case with respect to the defined ODD. For all subsystems, the calculation step size of each subsystem is defined and used for generating the respective instances.

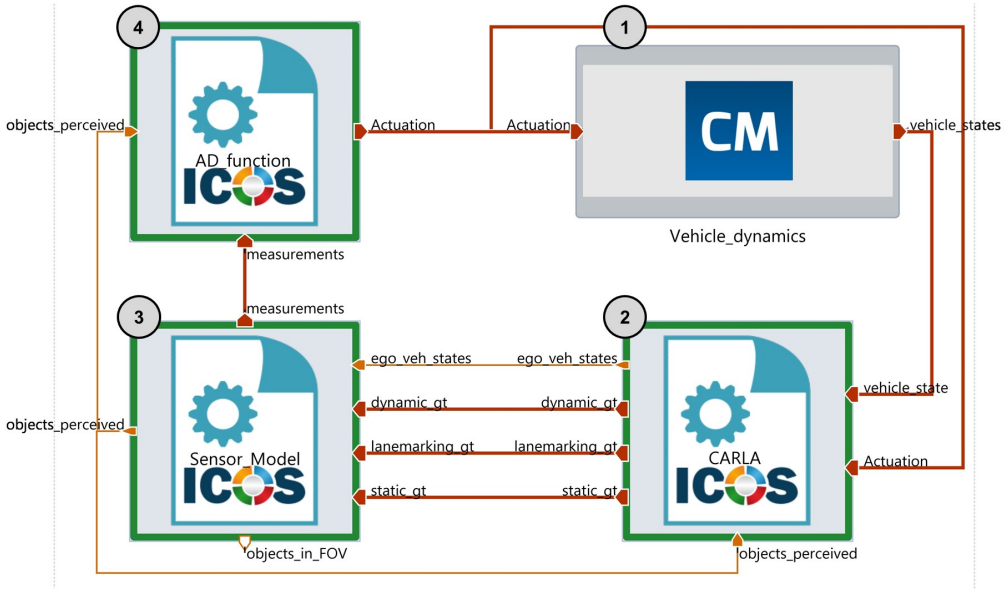


Figure 10. Overview of the implemented co-simulation architecture using AVL Model.CONNECT^{TM}.

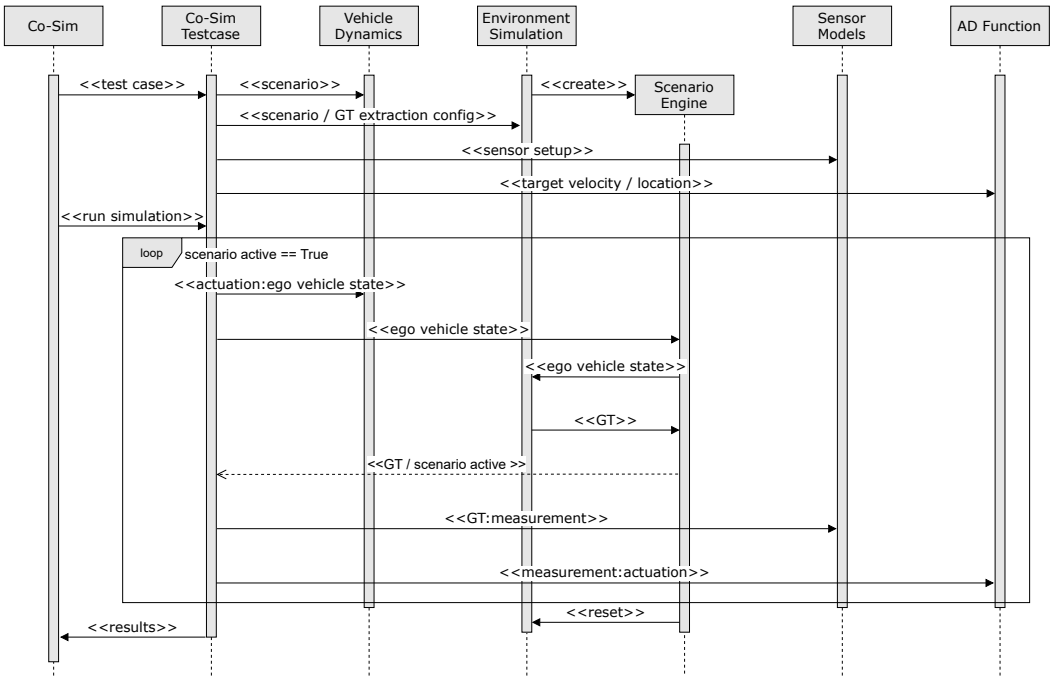


Figure 11. Sequence diagram of the implemented co-simulation framework for the virtual validation of ADS.

After the initialisation phase, the simulation of the concrete test case is started (*run_simulation*). As long as the scenario is active (which is checked constantly by the co-simulation test case instance based on the information from the scenario engine), the calculation sequence starts with the determination of the ego vehicle states in the vehicle dynamics module. This information is transferred to the environment simulation using the scenario engine as an intermediate subsystem. It controls all traffic participants in the virtual environment based on the scenario description. The ground truth from the environment simulation is then extracted by the scenario engine and used as input for the sensor models. Next, the output of the sensor models (*measurement*) is used as input to the AD function, which calculates the actuation, which is the input for the vehicle dynamics model. After that, another calculation loop starts, as long the scenario is evaluated as

active (by the scenario engine based on pre-defined stopping criteria either in the scenario description or the test case). The scenario engine resets the whole environment simulation once the scenario, and therefore the loop, is finished. Another test case can be simulated right after without having to restart the entire simulation. As the last step, the results from the test case are gathered and stored respectively.

In addition to the sequence diagram presented in Figure 11, a short pseudo-code for the simulation of one test case is provided in the Algorithm 1. At first, all required subsystems are initialised using the provided information from the respective test case description. Secondly, these subsystems are calculated stepwise for as long as the scenario is active, which means no trigger to end the scenario (e.g., simulation time reaches the set value) is activated. In the end, the environment simulation is reset for the next test case to be simulated. All the required information for evaluating the simulation run is stored in separate log files for each subsystem.

Algorithm 1: Simulation of a test case with the proposed co-simulation framework.

```
Result: Log files from all relevant subsystems
initialise vehicle dynamics with scenario description from test case;
initialise environment simulation with scenario description from test case;
initialise sensor models with sensor setup from test case;
initialise ad function with target velocity and initial location from test case;
while Scenario is active do
    calculate vehicle dynamics with actuation from ad function;
    update ego vehicle state in environment simulation;
    calculate sensor models with ground truth from environment simulation;
    calculate actuation of ad function with sensor model output;
end
reset environment simulation;
```

4. Simulation Framework Comparison for the ALKS with Varying ODD

In this section, the results of the simulated scenario for both ODD versions (standard and extended) are presented. An overview of both simulation frameworks, for the standard and the extended ODD, is given in Table 5. For the extended ODD, the virtual environment of the environment simulation is adapted and the vehicle dynamics of the ego vehicle are calculated using IPG CarMaker.

Table 5. Overview of the concrete subsystems for both simulation frameworks (for the standard and the extended ODD).

Subsystem	Standard ODD	Extended ODD
AD function:	ALKS	ALKS
Sensor models:	Low-fidelity object sensor and lane marking sensor	Low-fidelity object sensor and lane marking sensor
Environment simulation:	A2 motorway as virtual environment	Modified A2 motorway with added contruction zone
Vehicle dynamics:	Python-based single track model	IPG CarMaker

Table 6 shows the complete test case description, including all relevant parameters, that were chosen for the simulations. The co-simulation is parametrised with the calculation step sizes of all subsystems and the coupling step size in between models. Furthermore, zero-order hold (ZOH) is chosen as a simple coupling algorithm for extrapolating the input from the last to the first subsystem (in terms of execution order). The execution order of the subsystems is displayed in Figure 10 as numbers in the top left corner of the respective model blocks. The sequence starts with the vehicle dynamics subsystems and then calculates sequentially in clockwise order. For the sensor model subsystem, all relevant parameters of the two implemented sensors are stated, including orientation,

range and FOV. The vehicle dynamics are defined stating the used software (and version). Additionally, the used vehicle configuration, which is part of the standard software package, is displayed. In the Python-based vehicle dynamics, the equivalent configurations of all applicable parameters are chosen. Furthermore, the target velocity is defined as an AD function parameter for both ODD cases.

Table 6. Parameters for the two virtual validation frameworks for the ALKS.

Subsystem	Parameter	Value	Unit
Co-simulation: Test case	Coupling step size	0.02	s
	Coupling algorithm	ZOH	-
	Vehicle dynamics step size	0.02	s
	Environment model step size	0.02	s
	Sensor model step size	0.02	s
	AD function step size	0.02	s
	Execution order	Sequential	-
	Scenario description	adapted from [71]	-
Sensor models	Number of sensors	2	#
Type of sensor: Low-fidelity object sensor	Sensor orientation	front-facing	-
	Sensor range	55	m
	Sensor vertical FOV	25	deg
	Sensor horizontal FOV	60	deg
Type of sensor: Lane marking sensor	Sensor orientation	front-facing	-
	Sensor range	30	m
	Sensor horizontal FOV	78	deg
Vehicle dynamics	Software	IPG CarMaker	-
	Version	8	-
	Vehicle specification (standard vehicle)	Tesla Model S	-
AD function: ALKS parameter	Target velocity (standard ODD)	16	m/s
	Target velocity (extended ODD)	19	m/s

For the ALKS with the standard ODD, a typical motorway scenario, adapted from [71], was chosen. An overview of the scenario is shown in Figure 12. The ego vehicle is driving behind two other traffic participants when one of those vehicles makes a lane change to the left. The occluded vehicle appears from the perspective of the ego vehicle, which requires a respective reaction. The ego vehicle should reduce its speed so that a collision, or even an undercutting of a critical distance between the two vehicles, is avoided. The simulation results are shown in Figure 13. It contains the results of the framework originally designed for the standard ODD (with the single-track vehicle dynamics model) displayed as a dashed grey line in Figure 13, while the extended framework is represented as a solid black line. The vehicle dynamics are operating under the defined conditions. Therefore, a high correspondence between the results of both frameworks can be observed. This similarity applies to the lateral error (distance between the vehicles' position and the virtual centerline of the lane) and the steering actuation. These signals can be observed in the first and second subplot of Figure 13, respectively. Only for the ego vehicles' velocity can a small difference between both signals over an extended time during the scenario be observed. This can be seen in the third subplot of Figure 13 and is due to the implemented longitudinal controller used in the AD function.

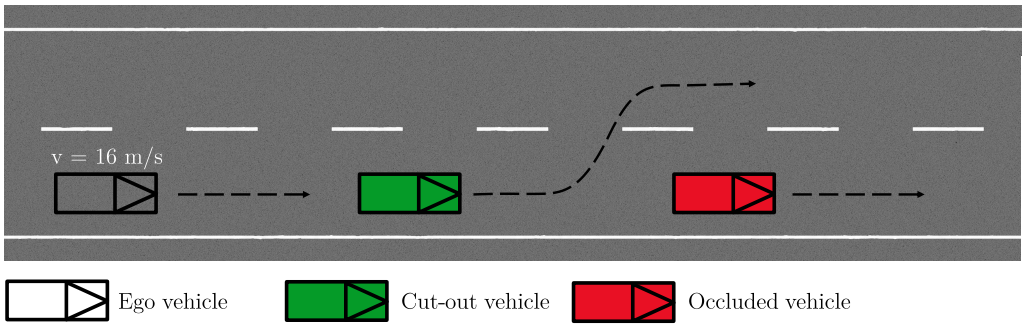


Figure 12. Overview of the original motorway scenario for the standard ODD of the ALKS (adapted cut-in scenario from [71]).

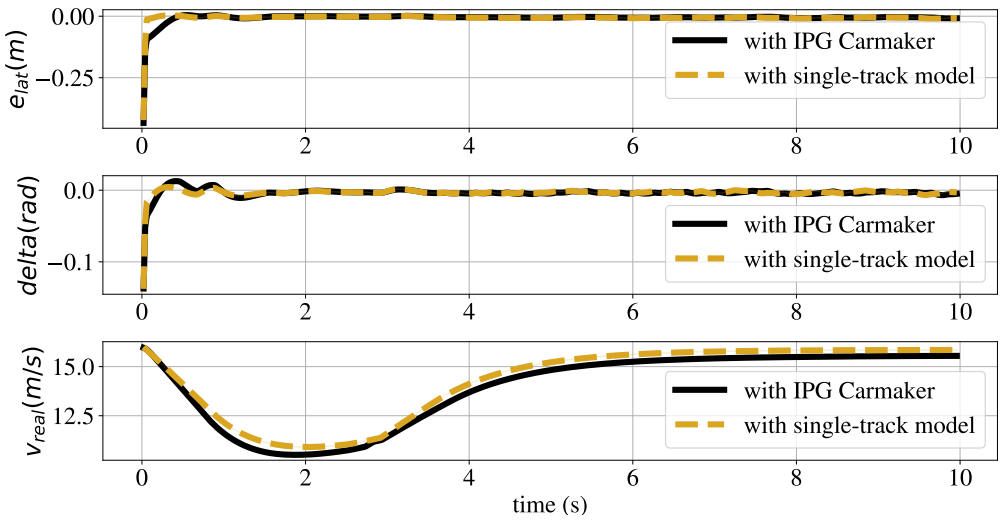


Figure 13. Comparison of the lateral error, the steering angle and the velocity of the ego vehicle for the two different simulation architectures in the original motorway scenario (standard ODD).

For the extended ODD, the chosen scenario can be seen in Figure 14. The ego vehicle, which has implemented the ALKS as an AD function, should follow its lane in this simple setup. It is forced to perform a quasi lane change, since the lane is offset because the motorway has fewer available lanes. The yellow lane markings characterise this area. Once again, not only the dedicated framework for this extended ODD, as explained in Section 3.2, is simulated but also the other one, to evaluate the differences in the results. Figure 15 shows the results in the lateral error, the steering actuation and the ego vehicles' velocity for both frameworks in the three respective subplots. The ego vehicles' AD function adapts the velocity during the lane offset based on the input from the lane marking sensor. The resulting difference in velocity between both framework simulations is similar to the original motorway scenario for the standard ODD and has the same reasoning. However, a much bigger divergence can be seen in the steering actuation. Being a closed-loop simulation, this means that the internal lateral controller has to output a much higher actuation to keep the lateral error as small as possible. This can be observed in the first subplot of Figure 15, where the difference in lateral error is at most 10 cm. If the AD function did not adapt its velocity based on the input from the lane marking sensor, the resulting lateral error difference between both frameworks would be much greater. That is because the Python-based single-track model of the framework for the standard ODD cannot accurately represent such demanding lateral manoeuvres.

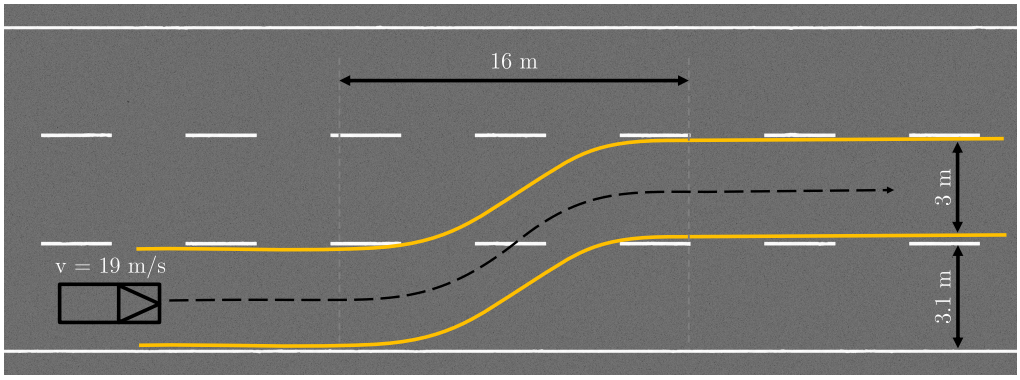


Figure 14. Overview of the construction scenario due to the ODD extension.

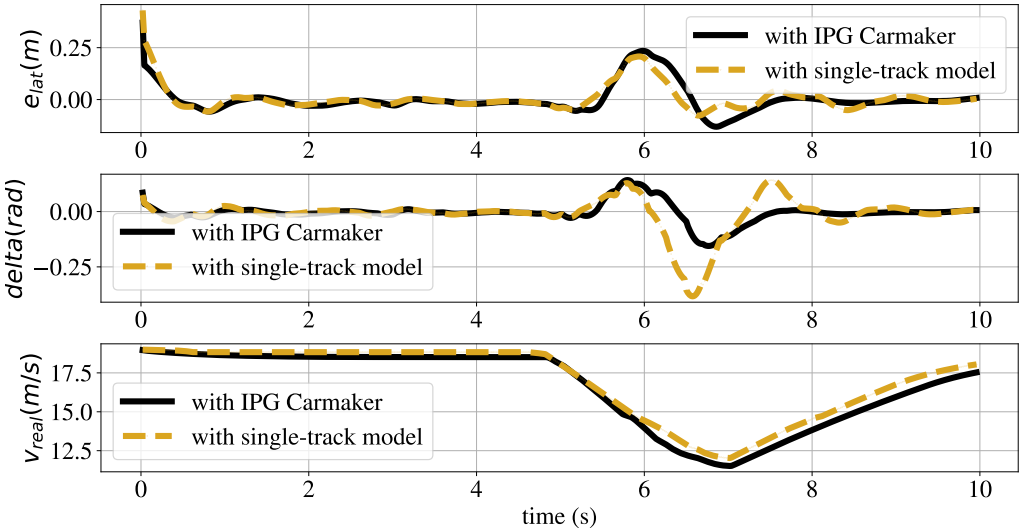
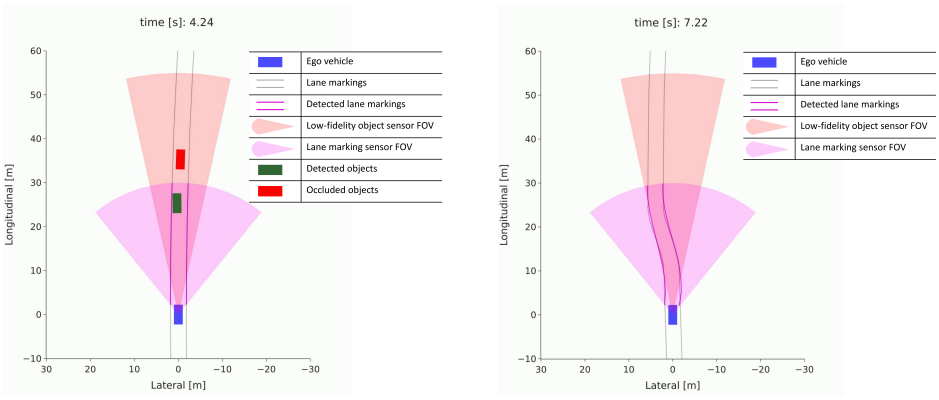


Figure 15. Comparison of the lateral error, the steering angle and the velocity of the ego vehicle for the two different simulation architectures in the construction scenario (extended ODD).

The visualisation of both scenarios, for the standard and the extended ODD, can be seen in Figure 16a,b, respectively. The FOV of both implemented sensors (the object and the lane marking sensor) and the lane markings and objects (ground truth and detected) from the environment simulation are displayed.



(**a**) (**b**)

Figure 16. Visualisation of both ALKS simulation setups from the sensor model point of view. (**a**) Cut-out scenario (ALKS with standard ODD). (**b**) Construction scenario (ALKS with extended ODD).

The simulations show that in the case of the standard ODD, the results of both framework versions (with medium- or high-fidelity vehicle dynamics) are similar. Hence, the framework with the medium-fidelity vehicle dynamics, as initially planned for the ALKS with the standard ODD, is suitable. In the case of the scenario for the extended ODD, the behaviour of the two simulated frameworks is quite different, especially for the steering actuation. This indicates a difference in behaviour between both models, whose effect is only minimised due to the closed-loop architecture of the implemented virtual validation framework. Therefore, this leads us to the conclusion that the medium-fidelity vehicle dynamics model is not suitable for the extended ODD, which supports our original assumption during the design phase of both framework versions. Further extending the ODD for more demanding weather conditions (e.g., rain) would further amplify the need for high-fidelity vehicle dynamics. However, the environment simulation and, therefore, the sensor model subsystem would also be affected by such an extension.

5. General Simulation Framework for Virtual Validation

Comparing both frameworks, the intended framework for the standard and the extended ODD show that only one subsystem (the vehicle dynamics) is different. Combining the Table 6 with the structure given by the general virtual validation framework (see Figure 5) leads to the conclusion that such a framework naturally has multiple layers, which are all necessary and need to be parameterised correctly. These layers are depicted in Figure 17. At the very bottom is the co-simulation layer (red layer in Figure 17). This is also the layer that is the most agnostic to the exact use case or even domain. However, choosing the right execution order, coupling step size, and ideal coupling algorithm is still critical in order to reduce coupling errors. On top of this, a tool-specific co-simulation capability comes into play (green layer in Figure 17). Compared to the basic co-simulation layer, this layer is particular for the domain of virtual validation for the ADS and should define the exact signals exchanged between subsystems. This layer should be an implementation of the interfaces from the general virtual validation framework given in Table 2. For the remaining two layers, the concrete subsystems are examined. In the third layer (blue layer in Figure 17), concrete simulation tools are considered. Based on the overview of the co-simulation structure provided in Figure 10, this layer can either be an actual tool (in terms of software—e.g., IPG CarMaker) or an API providing the interface for the last layer. The final layer (grey layer in Figure 17) describes the actual simulation models. This could be a vehicle dynamics model of a concrete passenger car (potentially including various simulation models of sub-modules, such as a steering model) or any other type of model included in the simulation framework and operating during the simulation runtime. Table 7 combines all the necessary parameters of the identified layers in an overview. The most important aspects of each layer are all listed.

Table 7. Different framework layers and their respective parameters and formats.

Framework layer	Parameters	Formats	Description	Unit
Co-simulation	Coupling step size	Co-simulation setting specification format (e.g., XML)	Step size between simulation tools/models	s
	Simulation tool/model step size		Time between calculation step of individual tool/model	s
	Execution order		Order in which the individual models are calculated	#
	Extrapolation method		Used method for extrapolation (e.g., ZOH)	-
Tool-specific co-simulation capability	Physical interface	ASAM OSI®	Defines the exchanged signals between tools/models	-

Table 7. *Cont.*

Framework layer	Parameters	Formats	Description	Unit
Simulation tools	Parameters for first 5 layers of the 6 layer model [11]	ASAM OpenSCENARIO®	Scenario description	-
	Parameters describing the road network	ASAM OpenDRIVE®	Road network description	-
	Parameters describing the road surface	ASAM OpenCRG®	Road surface description	-
Simulation models	Model specific parametrisation	Various	Necessary parameters to define the respective models	-

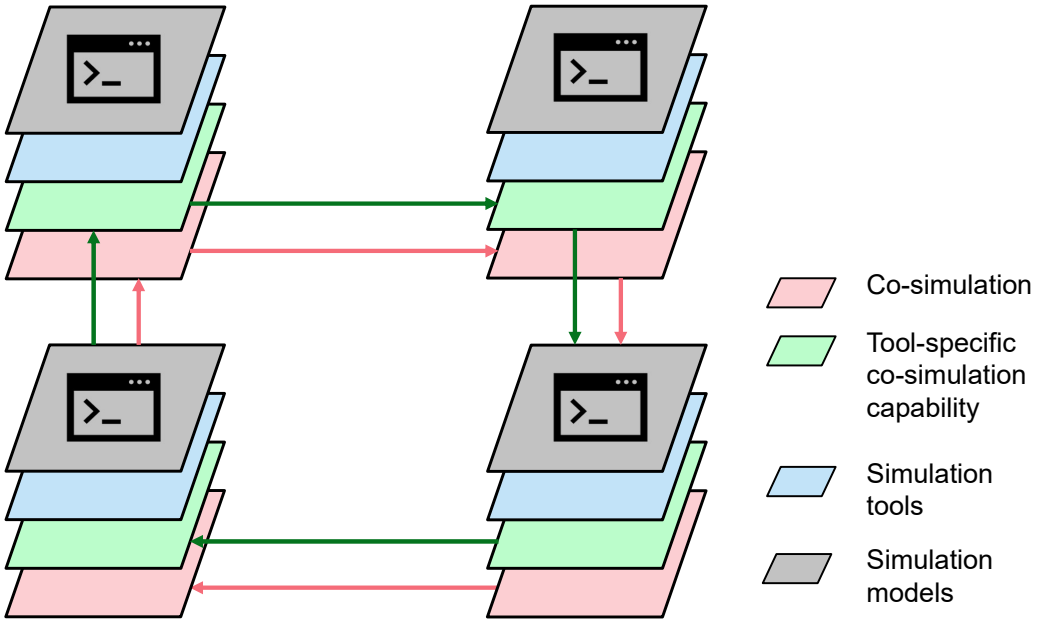


Figure 17. Overview of a proposal for the conceptual enhancement of simulation framework architectures for virtual validation.

6. Discussion and Outlook

This article combines multiple essential aspects necessary for the virtual validation of ADS. On the one hand, a general framework that contains the vital subsystems needed for virtual validation is introduced. Furthermore, the interfaces between the subsystems are defined, partially relying on the ASAM OSI® standard. Additionally, the concept of model fidelities is presented and extended towards all relevant subsystems. Previously, model fidelity was only discussed regarding the sensor models and in the vehicle dynamics model domain; however, this was in a different context. Moreover, the ODD defined from the respective ADS is linked to the required model fidelities of the subsystems. On the other hand, the structure of a virtual validation framework was analysed in detail. An essential aspect for the harmonisation of such virtual validation frameworks is the identification of the framework layers (see Figure 17), including the respective parameters. In particular, the second layer, the tool-specific co-simulation capability, is essential, as it enables two critical aspects of virtual validation frameworks. First, the simulation tool configuration and simulation model exchange are enabled through consistently defined interfaces. This directly leads to the second aspect, which is the reduction in integration effort. Furthermore, this allows other aspects to function, such as the execution of complete subsystems remotely in a distributed co-simulation setup, which is often preferred when considering the intellectual property in specific models. Using a dedicated co-simulation approach also provides the possibility for further optimisation in the case of many potential test runs, as the calculation sequence can be changed towards a parallel approach to reduce the overall calculation time. For the sensor model and the environment simulation

subsystem, ASAM OSI® provides the detailed interface definition needed. However, for a typical scenario engine (e.g., the *scenario_runner* module used for both frameworks in this article), the interfaces are not defined consistently, which makes integration into the overall framework costly in terms of time. Therefore, this is a potential area for the extension of ASAM OSI® or the development of new standards. An extensive summary of the major contributions this article provides to the topic of ADS safety validation is provided in Figure 18.

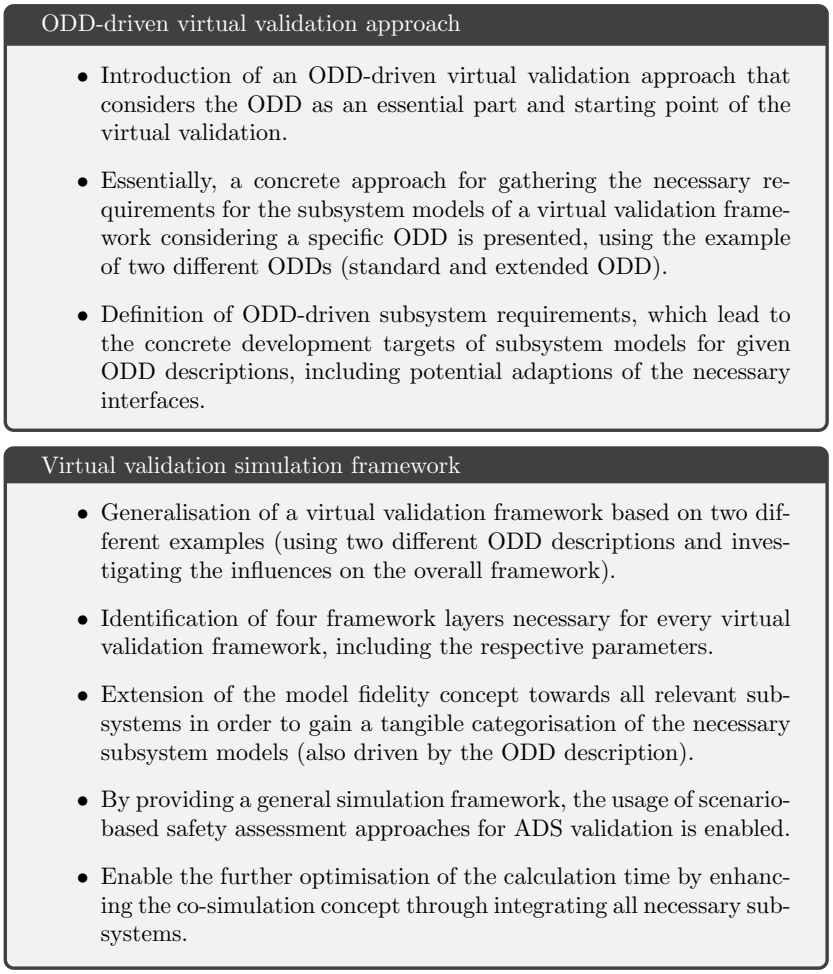


Figure 18. Main contributions of this article to the virtual validation of ADS.

Direcetions for further research include possible methods and processes that could be used to efficiently derive the required model fidelity of virtual validation subsystems. Combining a structured definition of such a framework with a consistently defined ODD would also lead to a more traceable safety case for ADS validation. The continuous extension of the ODD for a specific ADS would then also be anchored in a process. Hence, such a virtual validation framework enables various safety assessment approaches to be used. A concrete example is the usage of a framework in the effectiveness assessment of safety measures in automated vehicles, as defined in [72]. Another usage concerns achieving the coverage of a given ODD-utilising simulation or for executing sensitivity analysis and therefore optimising scenario parameters for later usage in validation approaches [73–75]. Starting with an ADS with a given target ODD in which it needs to operate safely, the parameters of the ODD (using specifications such as those defined in [34]), as well as potential disturbances inside this domain, are described. A comprehensive overview and an explanation of possible disturbances categorised for the different subsystems of an ADS are given in [13]. These disturbances, in combination with the ODD

parameters, are used to define concrete scenarios. Once the concrete scenarios are derived, a credible simulation framework must be made available, including the specific subsystems verified for the given ODD. In combination with verified subsystems (especially sensor models and environment simulations) for a particular ODD, which are a prerequisite to be used in the overall simulation framework, the extensive use of virtual testing for the safety assessment of ADS in a given ODD becomes possible. This framework is then used to simulate these scenarios. At the same time, an evaluation of the results is carried out with various specific key performance indicators (KPIs) to determine if a particular scenario succeeded or failed. This concept will be explored in future publications using the findings presented in this article—most prominently, the virtual validation framework—as a basis. To conclude, the contributions provided in this article—most prominently, the virtual validation framework and the introduced approach for ODD-driven ADS validation—offer multiple practical applications for future usage. Using these as a basis for a scalable testing method (such as virtual testing) enables the validation of ADS in ever more complex ODDs. This approach can be exploited during the development and testing of ADS, but, more importantly, during such systems' homologation and certification phase. Such use cases will also be explored in future research.

Author Contributions: Conceptualisation, P.W., G.S., J.R. and D.W.; software, P.W. and J.R.; validation, P.W. and G.S.; writing—original draft preparation, P.W.; writing—review and editing, P.W., G.S., J.R. and D.W.; visualisation, P.W.; supervision, D.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received financial support within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Styria (Dept. 12), and the Styrian Business Promotion Agency (SFG). The work presented in this paper is part of the HEADSTART project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 824309. Content reflects only the authors' view and the European Commission is not responsible for any use that may be made of the information it contains.

Acknowledgments: The publication was written at Virtual Vehicle Research GmbH in Graz, Austria. The authors would like to acknowledge the financial support within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Digital and Economic Affairs (BMDW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ALKS	Automated lane-keeping system
AD	Automated driving
ADS	Automated driving system
ACC	Adaptive cruise control
TP	Traffic participants
OEDR	Object and event detection and response
ODD	Operational design domain
FOV	Field of view
DOF	Degree of freedom
UN	United Nations
V2X	Vehicle-to-X (e.g., infrastructure)
ICOS	Independent co-simulation
XIL	X-in-the-loop
KPI	Key performance indicators

ZOH	Zero-order hold
SAE	Society of Automotive Engineers

References

1. World Health Organization. *Global Status Report on Road Safety 2018: Summary*; World Health Organization: Geneva, Switzerland, 2018. Available online: http://apps.who.int/iris/bitstream/handle/10665/277370/WHO-NMH-NVI-18.20-eng.pdf?ua=1 (accessed on 10 November 2021).
2. Blumenthal, M.S.; Fraade-Blanar, L.; Best, R.; Irwin, J.L. *Safe Enough: Approaches to Assessing Acceptable Safety for Automated Vehicles*; Technical Report; RAND Corporation: Santa Monica, CA, USA, 2020.
3. *SAE J 3016—Taxonomy and Definitions for Terms Related to On-Road Motor Vehicle Automated Driving Systems*; SAE International, On-Road Automated Driving (ORAD) Committee: Warrendale, PA, USA, 2021. [CrossRef]
4. SaFAD. Safety First for Automated Driving. 2019. Available online: https://connectedautomateddriving.eu/mediaroom/framework-for-safe-automated-driving-systems/ (accessed on 17 November 2021).
5. Schnelle, S.; Salaani, K.; Rao, S.J.; Barickman, F.S.; Elsasser, D. *Review of Simulation Frameworks and Standards Related to Driving Scenarios*; Number: DOT HS 812 815; National Highway Traffic Safety Administration: Washington, DC, USA, 2019.
6. Watzenig, D.; Horn, M. (Eds.) *Automated Driving: Safer and More Efficient Future Driving*; Springer International Publishing: Cham, Switzerland, 2017. [CrossRef]
7. Riedmaier, S.; Ponn, T.; Ludwig, D.; Schick, B.; Diermeyer, F. Survey on Scenario-Based Safety Assessment of Automated Vehicles. *IEEE Access* **2020**, *8*, 87456–87477.
8. Junietz, P.; Bonakdar, F.; Klamann, B.; Winner, H. Criticality Metric for the Safety Validation of Automated Driving using Model Predictive Trajectory Optimization. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; IEEE: Maui, HI, USA, 2018; pp. 60–65. [CrossRef]
9. Junietz, P.; Steininger, U.; Winner, H. Macroscopic Safety Requirements for Highly Automated Driving. *Transp. Res. Rec. J. Transp. Res. Board* **2019**, *2673*, 036119811982791. [CrossRef]
10. Ulbrich, S.; Menzel, T.; Reschka, A.; Schuldt, F.; Maurer, M. Defining and Substantiating the Terms Scene, Situation, and Scenario for Automated Driving. In Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems, Gran Canaria, Spain, 15–18 September 2015; IEEE: Gran Canaria, Spain, 2015; pp. 982–988. [CrossRef]
11. Scholtes, M.; Westhofen, L.; Turner, L.R.; Lotto, K.; Schuldes, M.; Weber, H.; Wagener, N.; Neurohr, C.; Bollmann, M.; Hiller, J.; et al. 6-Layer Model for a Structured Description and Categorization of Urban Traffic and Environment. *IEEE Access* **2021**, *9*, 59131–59147. [CrossRef]
12. Bagschik, G.; Menzel, T.; Maurer, M. Ontology based Scene Creation for the Development of Automated Vehicles. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, China, 26–30 June 2018; pp. 1813–1820. ISSN: 1931-0587. [CrossRef]
13. JAMA; SAKURA. *Automated Driving Safety Evaluation Framework Ver. 1.0-Guidelines for Safety Evaluation of Automated Driving Technology*; Technical Report, Sakura Research Project: Kawaguchi, Japan, 2020.
14. Antona-Makoshi, J.; Uchida, N.; Kitahara, E.; Ozawa, K. *A Safety Assurance Process for Automated Driving Systems*; ITS World Congress 2019: Singapore, 2019.
15. PEGASUS. PEGASUS Method—An Overview. 2019. Available online: https://www.pegasusprojekt.de (accessed on 15 November 2021).
16. Wagner, N.; Weissensteiner, P.; Coget, J.B.; Eckstein, L.; Bracquemond, A. *Common Methodology for Data-Driven Scenario-Based Safety Assurance in the HEADSTART Project*; ITS European Congress 2020: Los Angeles, CA, USA, 2020.
17. ENABLE-S3. *Testing and Validation of Highly Automated Systems—Summary of Results*; 2019. Available online: https://enable-s3.eu (accessed on 13 November 2021).
18. Thorn, E.; Kimmel, S.; Chaka, M. *A Framework for Automated Driving System Testable Cases and Scenarios*; Technical Report DOT HS 812 623; National Highway Traffic Safety Administration: Washington, DC, USA, 2018.
19. U.S. Department of Transportation. *Virtual Open Innovation Collaborative Environment for Safety*; White Paper; Working Draft Version A; U.S. Department of Transportation: Washington, DC, USA, 2021.
20. UK Law Commission and Scottish Law Commission. Automated Vehicles: Summary of Consultation Paper 3—A Regulatory Framework for Automated Vehicles. 2020. Available online: https://www.lawcom.gov.uk/project/automated-vehicles/ (accessed on 16 November 2021).
21. Enterprise Singapore. *Technical Reference for Autonomous Vehicles*; Technical Report Part 1—4; Enterprise Singapore: Singapore, 2019.
22. United Nations. UN Regulation on Automated Lane Keeping Systems is Milestone for Safe Introduction of Automated Vehicles in Traffic. Library Catalog. 2020. Available online: www.unece.org (accessed on 10 October 2021).
23. Schlager, B.; Muckenhuber, S.; Schmidt, S.; Holzer, H.; Rott, R.; Maier, F.M.; Saad, K.; Kirchengast, M.; Stettinger, G.; Watzenig, D.; et al. State-of-the-Art Sensor Models for Virtual Testing of Advanced Driver Assistance Systems/Autonomous Driving Functions. *SAE Int. J. Connect. Autom. Veh.* **2020**, *3*, 233–261. [CrossRef]
24. Hexagon. Virtual Test Drive—Enabling Safety Validation in Autonomous Driving and ADAS System Simulation. Available online: https://hexagon.com/ (accessed on 22 December 2021).

25. aiMotive. aiSim 3.0—The World's First ISO26262 ASIL-D Certified Simulator Tool. 2021. Available online: https://aimotive.com/aisim-3.0 (accessed on 22 December 2021).
26. GmbH, I.A. *CarMaker: Virtual Testing of Automobiles and Light-Duty Vehicles*; IPG Automotive GmbH: Karlsruhe, Germany, 2020.
27. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. *arXiv* **2017**, arXiv:1711.03938.
28. Riedmaier, S.; Danquah, B.; Schick, B.; Diermeyer, F. Unified Framework and Survey for Model Verification, Validation and Uncertainty Quantification. *Arch. Comput. Methods Eng.* **2021**, *28*, 2655–2688. [CrossRef]
29. Gomes, C.; Thule, C.; Broman, D.; Larsen, P.G.; Vangheluwe, H. Co-simulation: State of the art. *arXiv* **2017**, arXiv:1702.00686.
30. Farah, H.; Bhusari, S.; van Gent, P.; Mullakkal Babu, F.A.; Morsink, P.; Happee, R.; van Arem, B. An Empirical Analysis to Assess the Operational Design Domain of Lane Keeping System Equipped Vehicles Combining Objective and Subjective Risk Measures. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 2589–2598. [CrossRef]
31. Automated Vehicle Safety Consortium. *AVSC Best Practice for Describing an Operational Design Domain: Conceptual Framework and Lexicon*; Technical Report; SAE International: Warrendale, PA, USA, 2020. Available online: https://avsc.sae-itc.org/ (accessed on 22 December 2021).
32. Gyllenhammar, M.; Johansson, R.; Warg, F.; Chen, D.; Heyn, H.M.; Sanfridson, M.; Söderberg, J.; Thorsén, A.; Ursing, S. Towards an Operational Design Domain That Supports the Safety Argumentation of an Automated Driving System. In Proceedings of the 10th European Congress on Embedded Real Time Software and Systems, Toulouse, France, 29–31 January 2020.
33. Lee, C.W.; Nayeer, N.; Garcia, D.E.; Agrawal, A.; Liu, B. Identifying the Operational Design Domain for an Automated Driving System through Assessed Risk. In Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV), Las Vegas, NV, USA, 19 October–13 November 2020; pp. 1317–1322. ISSN: 2642-7214. [CrossRef]
34. *PAS 1883:2020*; BSI Standards Limited: London, UK, 2020.
35. *ISO/AWI 34503—Road Vehicles—Taxonomy for Operational Design Domain for Automated Driving Systems*; International Organization for Standardization: Geneva, Switzerland. Available online: https://www.iso.org/standard/78952.html (accessed on 14 November 2021).
36. Ladstädter, R.; Luley, P.; Ladstätter, S.; Mayer, H. *UHD Mapping von Teststrecken für Automatisiertes Fahren*; Dreiländertagung der DGPF, der OVG und der SGPF: Vienna, Austria, 2019.
37. ASAM e.V. ASAM OpenDRIVE—Open Dynamic Road Information for Vehicle Environment. Available online: https://www.asam.net/standards/detail/opendrive/ (accessed on 22 December 2021).
38. Mobileye. Mobileye—An Intel Company. Available online: https://www.mobileye.com/ (accessed on 22 December 2021).
39. Weissensteiner, P.; Stettinger, G.; Tieber, K.; Rehrl, K. Virtual Risk Assessment for the Deployment of Autonomous Shuttles. *Transp. Res. Rec.* **2021**, *2675*, 131–140.[CrossRef]
40. Weissensteiner, P.; Stettinger, G.; Tieber, K.; Watzenig, D.; Rehrl, K. Risk minimisation for autonomous shuttles in suburban environments based on virtual validation. In Proceedings of the ITS World Congress 2021, Hamburg, Germany, 11–15 October 2021. [CrossRef]
41. Schmidt, S.; Schlager, B.; Muckenhuber, S.; Stark, R. Configurable Sensor Model Architecture for the Development of Automated Driving Systems. *Sensors* **2021**, *21*, 4687. [CrossRef] [PubMed]
42. Siegl, S.; Ratz, S.; Düser, T.; Hettel, R. Vehicle-in-the-Loop am Prüfstand zur Validierung von ADAS/AD. *ATZelektronik* **2021**, *16*, 64–69. [CrossRef]
43. Solmaz, S.; Holzinger, F. A Novel Testbench for Development, Calibration and Functional Testing of ADAS/AD Functions. In Proceedings of the 2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE), Graz, Austria, 4–8 November 2019; IEEE: Graz, Austria, 2019; pp. 1–8. [CrossRef]
44. Solmaz, S.; Holzinger, F.; Mischinger, M.; Rudigier, M.; Reckenzaun, J. Novel Hybrid-Testing Paradigms for Automated Vehicle and ADAS Function Development. In *Towards Connected and Autonomous Vehicle Highways*; Hamid, U.Z.A., Al-Turjman, F., Eds.; Series Title: EAI/Springer Innovations in Communication and Computing; Springer International Publishing: Cham, Switzerland, 2021; pp. 193–228. [CrossRef]
45. Hallerbach, S. *Simulation-Based Testing of Cooperative and Automated Vehicles*; University of Oldenburg, Department of Computer Science: Oldenburg, Germany, 2020.
46. Nalic, D.; Pandurevic, A.; Eichberger, A.; Rogic, B. Design and Implementation of a Co-Simulation Framework for Testing of Automated Driving Systems. *Sustainability* **2020**, *12*, 10476. [CrossRef]
47. Nalic, D.; Eichberger, A.; Hanzl, G.; Fellendorf, M.; Rogic, B. Development of a Co-Simulation Framework for Systematic Generation of Scenarios for Testing and Validation of Automated Driving Systems*. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 1895–1901. [CrossRef]
48. Nalic, D.; Pandurevic, A.; Eichberger, A.; Rogic, B. Software Framework for Testing of Automated Driving Systems in a Dynamic Traffic Environment. *arXiv* **2020**, arXiv:2011.05798.
49. Nalic, D.; Li, H.; Eichberger, A.; Wellershaus, C.; Pandurevic, A.; Rogic, B. Stress Testing Method for Scenario-Based Testing of Automated Driving Systems. *IEEE Access* **2020**, *8*, 224974–224984. [CrossRef]
50. Lopez, P.A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; Wießner, E. Microscopic traffic simulation using SUMO. In Proceedings of the 21st IEEE International Conference on Intelligent Transportation Systems, Maui, HI, USA, 4–7 November 2018; IEEE: New York City, NY, USA, 2018. [CrossRef]

51. von Neumann-Cosel, K. *Virtual Test Drive–Simulation umfeldbasierter Fahrzeugfunktionen*; Technical University of Munich, Faculty of Computer Science: Munich, Germany, 2014.
52. ASAM e.V. *ASAM OpenSCENARIO*; Volume 1.1.0.; ASAM e.V.: Hoehenkirchen, Germany, 2021, Available online: https://www.asam.net/standards/detail/openscenario/ (accessed on 22 December 2021).
53. ASAM e.V. *ASAM OpenCRG—Open Curved Regular Grid*; Volume 1.2.0.; ASAM e.V.: Hoehenkirchen, Germany, 2021, Available online: https://www.asam.net/standards/detail/opencrg/ (accessed on 22 December 2021).
54. van Driesten, C.; Schaller, T. Overall Approach to Standardize AD Sensor Interfaces: Simulation and Real Vehicle. In *Fahrerassistenzsysteme 2018*; Bertram, T., Ed.; Springer Fachmedien Wiesbaden: Wiesbaden, Germnay, 2019; pp. 47–55. [CrossRef]
55. Marko, N.; Ruebsam, J.; Biehn, A.; Schneider, H. Scenario-based Testing of ADAS—Integration of the Open Simulation Interface into Co-simulation for Function Validation. In Proceedings of the 9th International Conference on Simulation and Modeling Methodologies, Technologies and Applications, SIMULTECH 2019, Prague, Czech Republic, 29–31 July 2019; SCITEPRESS—Science and Technology Publications, LDA.: Setubal, Portugal, 2019; pp. 255–262. [CrossRef]
56. ASAM e.V. *ASAM OSI—Open Simulation Interface*; Volume 3.3.0.; ASAM e.V.: Hoehenkirchen, Germany, 2021. Available online: https://www.asam.net/standards/detail/osi/ (accessed on 22 December 2021).
57. Stolz, M.; Nestlinger, G. Fast generic sensor models for testing highly automated vehicles in simulation. *Elektrotechnik Informationstechnik* **2018**, *135*, 365–369. [CrossRef]
58. Genser, S.; Muckenhuber, S.; Solmaz, S.; Reckenzaun, J. Development and Experimental Validation of an Intelligent Camera Model for Automated Driving. *Sensors* **2021**, *21*, 7583. [CrossRef]
59. Hirsenkorn, N.; Hanke, T.; Rauch, A.; Dehlink, B.; Rasshofer, R.; Biebl, E. A non-parametric approach for modeling sensor behavior. In Proceedings of the 2015 16th International Radar Symposium (IRS), Dresden, Germany, 24–26 June 2015; IEEE: Dresden, Germany, 2015; pp. 131–136. [CrossRef]
60. Hirsenkorn, N.; Subkowski, P.; Hanke, T.; Schaermann, A.; Rauch, A.; Rasshofer, R.; Biebl, E. A ray launching approach for modeling an FMCW radar system. In Proceedings of the 2017 18th International Radar Symposium (IRS), Prague, Czech Republic, 28–30 June 2017; IEEE: Prague, Czech Republic, 2017; pp. 1–10. [CrossRef]
61. Kinne, D. Impact of Vehicle Dynamics Model Fidelity in the Development of ADAS. In *NAFEMS World Congress 2019*; NAFEMS: Quebec City, QC, Canada, 2019.
62. *ISO/DIS 11010-1 Passenger Cars—Simulation Model Classification—Part 1: Vehicle Dynamics*; International Organization for Standardization: Geneva, Switzerland, 2021.
63. Muckenhuber, S.; Holzer, H.; Ruebsam, J.; Stettinger, G. Object-Based Sensor Model for Virtual Testing of ADAS/AD Functions. In Proceedings of the International Conference on Connected Vehicles and Expo (ICCVE), Graz, Austria, 4–8 November 2019. [CrossRef]
64. Rumetshofer, J.; Stolz, M.; Watzenig, D. A Generic Interface Enabling Combinations of State-of-the-Art Path Planning and Tracking Algorithms. *Electronics* **2021**, *10*, 788. [CrossRef]
65. CARLA. ScenarioRunner for CARLA; 2018. Available online: https://github.com/carla-simulator/scenario_runner/ (accessed on 22 December 2021).
66. Althoff, M.; Koschi, M.; Manzinger, S. CommonRoad: Composable benchmarks for motion planning on roads. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; IEEE: Los Angeles, CA, USA, 2017; pp. 719–726. [CrossRef]
67. Model.CONNECT™—AVL's Open Model Integration and Co-Simulation Platform; AVL List GmbH. Available online: https://www.avl.com/model.connect- (accessed on 22 December 2021).
68. Benedikt, M.; Watzenig, D.; Hofer, A. Modelling and analysis of the non-iterative coupling process for co-simulation. *Math. Comput. Model. Dyn. Syst.* **2013**, *19*, 451–470. [CrossRef]
69. Stettinger, G.; Horn, M.; Benedikt, M.; Zehetner, J. Model-based coupling approach for non-iterative real-time co-simulation. In Proceedings of the 2014 European Control Conference (ECC), Strasbourg, France, 24–27 June 2014; pp. 2084–2089. [CrossRef]
70. Virtual Vehicle Research GmbH. Cross-Domain Co-Simulation. 2018. Available online: https://www.v2c2.at/icos-hl/ (accessed on 22 December 2021).
71. Tenbrock, A.; König, A.; Keutgens, T.; Bock, J.; Weber, H.; Krajewski, R.; Zlocki, A. The ConScenD Dataset: Concrete Scenarios from the highD Dataset According to ALKS Regulation UNECE R157 in OpenX. *arXiv* **2021**, arXiv:2103.09772.
72. *ISO/TR 21934-1:2021 Road Vehicles—Prospective Safety Performance Assessment of Pre-Crash Technology by Virtual Simulation—Part 1: State-of-The-art and General Method Overview*; International Organization for Standardization: Geneva, Switzerland, 2021.
73. Batsch, F.; Daneshkhah, A.; Cheah, M.; Kanarachos, S.; Baxendale, A. Performance Boundary Identification for the Evaluation of Automated Vehicles using Gaussian Process Classification. In Proceedings of the 2019 IEEE Intelligent Transportation Systems Conference (ITSC), Auckland, New Zealand, 27–30 October 2019; pp. 419–424. [CrossRef]
74. Batsch, F.; Daneshkhah, A.; Palade, V.; Cheah, M. Scenario Optimisation and Sensitivity Analysis for Safe Automated Driving Using Gaussian Processes. *Appl. Sci.* **2021**, *11*, 775. [CrossRef]
75. Batsch, F.; Kanarachos, S.; Cheah, M.; Ponticelli, R.; Blundell, M. A taxonomy of validation strategies to ensure the safe operation of highly automated vehicles. *J. Intell. Transp. Syst.* **2020**, 1–20. [CrossRef]