Sequential lidar sensor system simulation: a modular approach for simulation-based safety validation of automated driving

Philipp Rosenberger¹ · Martin Friedrich Holder¹ · Nicodemo Cianciaruso¹ · Philip Aust¹ · Jonas Franz Tamm-Morschel¹ · Clemens Linnhoff¹ · Hermann Winner¹

Received: 3 April 2020 / Accepted: 6 August 2020 / Published online: 23 September 2020
© The Author(s) 2020

Abstract

Validating safety is an unsolved challenge before autonomous driving on public roads is possible. Since only the use of simulation-based test procedures can lead to an economically viable solution for safety validation, computationally efficient simulation models with validated fidelity are demanded. A central part of the overall simulation tool chain is the simulation of the perception components. In this work, a sequential modular approach for simulation of active perception sensor systems is presented on the example of lidar. It enables the required level of fidelity of synthetic object list data for safety validation using beforehand simulated point clouds. The elaborated framework around the sequential modules provides standardized interfaces packaging for co-simulation such as Open Simulation Interface (OSI) and Functional Mockup Interface (FMI), while providing a new level of modularity, testability, interchangeability, and distributability. The fidelity of the sequential approach is demonstrated on an everyday scenario at an intersection that is performed in reality at first and reproduced in simulation afterwards. The synthetic point cloud is generated by a sensor model with high fidelity and processed by a tracking model afterwards, which, therefore, outputs bounding boxes and trajectories that are close to reality.

Keywords Lidar sensor simulation · Safety validation · Automated driving

1 Introduction

The scenario-based approach for safety validation of highly automated driving (HAD) was presented as a result of the recently finished research projects *PEGASUS* and

ENABLE-S3. The follow-up project *SET Level 4to5* continues this approach with a special focus on simulation-based testing. As perception plays a central role for HAD, simulation-based testing requires synthetic sensor data of validated fidelity at different processing states or interfaces.

In this regard, it is differentiated between sensor front-end and sensor system to distinguish signal transmitting, perception, and pre-processing (the front-end) from data processing [1], as can be seen in Fig. 1. While the former is dominated by hardware components, the latter mainly consists of processing algorithms. Therefore, a sensor model is technically a replacement of the real, physical sensor by a simulation model, which describes the measuring principle of the sensor and generates outputs defined as detections [2], p. 2].

The modules themselves are found through functional decomposition, as shown in the previous work of the authors [1]: Typical automotive lidar sensors output detections as a point cloud (PCL), which is an ID-ordered distance and intensity tuple at **IF1** that is also the output of the sensor model. **IF2** is part of the data processing unit before, e.g., object tracking and classification takes place, as

✉ Philipp Rosenberger
rosenberger@fzd.tu-darmstadt.de

Martin Friedrich Holder
holder@fzd.tu-darmstadt.de

Nicodemo Cianciaruso
cianciaruso@fzd.tu-darmstadt.de

Philip Aust
philip.aust@aunama.de

Jonas Franz Tamm-Morschel
jonas.tamm@web.de

Clemens Linnhoff
linnhoff@fzd.tu-darmstadt.de

Hermann Winner
winner@fzd.tu-darmstadt.de

¹ Institute of Automotive Engineering, Technische Universität Darmstadt, Darmstadt, Germany

Fig. 1 A generic perception sensor system for object detection along with the interface (IF) definition adapted from [1]. After signal processing, the data from one or more sensors can be merged on IF1 by the alignment and fusion module. By observing measurements over time, the tracking module generates an object list whose entries are then classified

it provides features in Cartesian coordinates in the vehicle coordinate system [[2], p. 2], e.g., the output of a sensor fusion unit that combines two lidar sensors into one PCL. Consequently, because of being able to provide a so-called object list (OL) at IF3, as well, the sequential simulation approach discussed in this work is called a sensor system simulation. OL generation is chosen in this work as an exemplary data processing unit, as it serves as a basis for HAD functions or as input for sensor fusion schemes.

To guarantee a broad applicability, the output is generated on different interfaces of the processing pipeline via the standards Open Simulation Interface [3] (OSI) and Functional Mock-up Interface [4] (FMI) in the sequential procedure, as proposed by van Driesten and Schaller [5]. The sequential structure provides a high level of modularity, testability, interchangeability, and distributability. Common simulation models of lidar sensors provide either PCLs or OLs. The sequential approach presented in this work calculates OLs based on previously generated PCLs. As explained in Sect. 2.1, this is expected to increase the fidelity of the OLs as the information from the PCL is processed. This is especially important for occlusions, as will be shown later. The modularity allows the exchange of individual components via the standardized interfaces. The basic idea of the presented simulation approach is to use all information available from ground truth (GT) in such a way that faithful simulation is possible. First, a PCL is generated and merged with the OL available from GT. This allows mimicking the behavior of a real tracking algorithm when generating the OL.

The further sections are organized as follows: In Sect. 2, the sequential approach is proposed as consequence of the state of the art limitations and the usage of the relatively new OSI data structure within the FMI interface is shown in detail. The individual modules used with the sequential approach are described in Sect. 3. First results from comparison of simulation output with real data from a measurement are presented in Sect. 4. Finally, a short conclusion is stated and an outlook towards further steps is proposed.

2 Sequential approach for lidar sensor system simulation

Before the new sequential approach for perception sensor system simulation and its framework are described in the following section, a brief overview of the state of the art is given.

2.1 State of the art of perception sensor system simulation

There are multiple tools available (commercial or open-source) that already provide parameterizable models to generate either synthetic PCLs or OLs with different levels of fidelity [6]. Actual methods for lidar sensor simulation to generate a synthetic PCL are either ray casting / tracing methods or projection methods like "z-buffer" as described and compared in the previous work of the authors [1]. For simulation of lidar-typical OLs, it is common sense that a stochastic or phenomenological approach is sufficient to generate desired uncertainties of existence (e.g., FP- / FN-rates), states (e.g., bounding box dimensions / location), and classes, as they are described by Dietmayer [7]. Still, there are simulation purposes, where stochastic models are fine to use, e.g., in early development phases of HAD planning functions.

Although, when it comes to faithful reproduction of sensor behavior, as, e.g., shown by Aust [[8], pp. 17–22], stochastic models are easy to falsify. A possible situation would be an intersection, where the ego car carrying a lidar sensor system approaches an already waiting car from behind. The actual sensor will gain reflections only from the rear of the car in front and the data processing will initialize the corresponding object with either almost no length or a default value for it. Without the availability of a PCL or at least the knowledge about its generation within the sensor, a stochastic approach will most likely fail in generating the length of the car in front with required fidelity for simulation based safety validation of following functions of HAD. In consequence, having this simple and frequent example of actual traffic in mind, it is not advisory to use stochastic models for scenario-based testing in simulation. Therefore, a sequential approach with modules derived from functional decomposition is necessary (but not sufficient) to achieve faithful perception sensor system simulation.

2.2 Sequential approach to tackle issues of the state of the art

Besides the already described limitation in fidelity, actual stochastic single-solution OL simulation lacks further

abilities to be used for simulation-based safety validation of HAD. The first issue of a single-solution OL generation is the obvious lack of modularity for the user. It is not possible to replace parts of it with modules of higher or lower fidelity or even the actually used algorithms within the real system.

Without standardized interfaces like FMI and a commonly agreed data structure like OSI, there is no interchangeability of the models and it is not possible to generate input for them with different environment simulation tools. Besides, it should be ensured that the interfaces of the simulation models reflect the real sensors' interfaces, what is ensured for the OSI structure [9] and ISO 23150 [2], as described by van Driesten and Schaller [5]. Additionally, it is not possible to distribute or scale the generation of synthetic sensor data, which is the most computationally expensive part of the overall simulation and therefore must be distributed for sensible testing. Finally, state-of-the-art OL generation cannot be validated at all, when it is not possible to evaluate each functional part itself, found by functional decomposition, as a module test.

Consequently, the sequential approach consisting of functionally decomposed modules tackles the existing limitations in modularity, interchangeability, distributability, and testability of actual single-solution simulations.

2.3 Standard-compliant framework around the sequential approach

Not only the consideration of primitives like modularity, interchangeability, and testability is required for choosing data structures and communication interfaces, but also the system architecture itself should be designed to promote the aforementioned objectives. The framework's design objective is to separate the actual simulation model development from establishing communication and data exchange protocols. On the one hand, this concerns the questions which and how many interfaces have to be considered for the so-called business logic, i.e., the actual algorithms for problem solving. On the other hand, it concerns the verification of the result. Especially, components that are deeply embedded in a simulation tool can only be tested in a cumbersome way, but errors or non-negligible behavioral deviations should be detected as early as possible and not only during the integration test, but ideally already at unit level.

To realize the described progress to the state of the art, the standard-compliant framework for the modular approach is being implemented on two levels to separate two objectives: communicating with a co-simulation master and effectively processing sensor data.

2.3.1 Communicating with a co-simulation master

FMI for co-simulation is an established specification and enjoys the confidence of numerous research partners and tool manufacturers, which is why it is chosen as communication interface. Implementations—so-called slaves—behave like a state machine controlled by a master. The module implementing the FMI is called Functional Mock-up Unit [[4], p. 6] (FMU). In addition to the actual application in the form of a software library, it contains a model description that specifies the inputs and outputs of the unit. The only shortcoming is the missing possibility of the current FMI 2.0 standard to directly transfer complex data structures—such as OSI. Therefore, a packaging specification has been published along with OSI, which also defines a convention for exchanging binary data and, thus, circumventing the aforementioned limit [10]. The data are serialized into memory, and only the position and size of this memory area are transferred as pointers via FMI.

The framework described in this work implements a *sensor model* as described in the packaging specification [[10], Kinds of Models]. It can be used with every (co-)simulation tool that is implementing the interface specifications and filling the necessary fields of the data structure. For correct sensor data processing, it is important to ensure that the co-simulation master is calling all its slaves, which are part of the sensor data processing sequence, synchronously. Therefore, an equal clock frequency is recommended. Figure 2 gives an overview of a typical co-simulation master that calls the **SensorModelFMU** first to obtain sensor data from a sensor view and then passes that data to a **LogicalModelFMU** for further processing.

2.3.2 Efficiently processing sensor data

Besides the data transfer from co-simulation master to FMU, the minimization of development effort posed at the beginning is tackled by the framework. Each subtask of sensor modeling could be implemented as an FMU, but would require the developer to serialize and deserialize the data structure for processing besides considering the operating state of the FMU. It is obviously not computationally efficient to perform these tasks repeatedly, when not necessary because of, e.g., distributed simulation on different physical machines. At the same time, the various algorithms should be easily interchangeable. For this reason, a further abstraction layer, the inner layer, is included to facilitate the development of the actual business logic. The so-called strategy design pattern [[11], pp. 315ff] is used to ensure that both levels are decoupled from each other. An adjustment was made in the process: Efficient strategies concentrated on one task can be connected in series to solve tasks that are

Fig. 2 Interaction diagram showing the communication between a co-simulation master and its slaves. The **SensorModelFMU** generates sensor data to be processed by a **LogicalModelFMU** (for example, a HAD function)

Fig. 3 Class diagram to illustrate the associations of the inner layer, showing the **Strategy** interface and two exemplary implementations that would be called sequentially by the FMU. Class / interface operations have been left out (except one) for the reason of clarity

more complex. Figure 3 gives an overview of the inner layer exemplary implemented in this work.

When calculating a so-called step (the call frequency is specified by the co-simulation master), the strategy implementation depends only on the presence of two objects at runtime: The OSI data structure and execution parameters.

The latter are available to allow the parameterization of the business logic from the outside. Together with the outer layer, configuration parameters, which may be defined in the model description of the FMU, are passed on. The interface is kept general to ensure that there is no

Fig. 4 Component diagram outlining the FMU's internal structure

dependency of the inner and outer layer. Figure 4 outlines the FMU's internal structure.

2.3.3 Reusability and testability

Two major goals are achieved by the presented framework: The first is reusability. At the outer level, the FMU as a whole is easily reusable with any compliant co-simulation master. The business logic, represented by a strategy, is reusable even outside the FMU. The second goal is testability. As the primary use cases are research and development, a business logic is likely subject to change very often. To ease debugging if a modified strategy does not comply with one's

self-proclaimed goals, a dedicated strategy library can easily be debugged using a preferred tool by recording the input passed to the strategies into a data stream and replay a simulation by feeding the strategy under test with the previously recorded data.

3 Modules of the sequential lidar sensor system simulation

Figure 5 shows the sequential structure of the lidar system simulation approach using functional blocks and interfaces as presented in earlier work of the authors [1].

Fig. 5 Modular approach for lidar sensor system simulation, on the example of a closed loop simulation for HAD function test

When generating an OL, the environment simulation provides global GT data, often in a world-coordinate frame. This is transformed into the vehicle or sensor coordinate frame and the ideal OL is available keeping only data within an assumed (and often idealized) sensor's field of view. Besides, the ideal PCL is rendered by means of a ray tracing engine that generates ideal raw scans for one or multiple sensors. Until here, no distortions that effectively occur in real-world measurements [12] have been taken into account. Therefore, adversarial effects such as noise must be applied to the PCL and OL in post processing. If more than one sensor is simulated, multiple PCLs are transformed and merged into a common coordinate frame of the ego vehicle. All processing steps from PCLs into OL in real sensors that involve a number of complex algorithms are bypassed in this modeling approach: The simulated data processing model uses the ideal OL and the realistic PCL to generate realistic, lidar-typical OLs without actually running these algorithms, technically by re-compiling information available from GT.

3.1 Simple ray casting for PCL generation

Each point in a PCL reflects a distance measurement from the sensor to the hit point, which translates to depth measurement in the computer graphics domain. An efficient algorithm is given by the depth map in a rasterized image, but the so-called "z-buffer" has the drawback of limited resolution at larger distance. The ray tracing algorithm in contrast offers a higher accuracy but comes to the cost of larger complexity.

In a simplified formulation, called ray casting, rays are shot into the virtual scenery and the distance from the emitter to the closest intersection point is reported. This model abstracts a lidar beam as an infinitesimally thin ray and the lidar beam alignment of the simulated sensor is configured from the increments between adjacent rays, which results in hit points as visualized in Fig. 6. It neglects beam divergence but results

Fig. 6 Beam pattern at 20-m distance of Ibeo LUX 2010 [13] with hit points from simple ray casting

Fig. 7 Super-sampling of the beam and beam pattern application for PCL calculation

in only two parameters, namely the horizontal and vertical spacing of rays.

3.2 Sensor model for faithful PCL generation

To achieve a higher fidelity of the sensor model, the additional effects of the actual beam pattern caused by diode and lens including its divergence, the different reflectivity of different materials, the attenuation of the signal's intensity, the thresholding within the sensor after reception, and the sensor's counter resolution and noise behavior are considered during PCL calculation.

At first, the rendering module, as seen in Fig. 5 is augmented. When a single ray hits anything, a bidirectional reflectance distribution function (BRDF) is applied that considers distance and direction of the ray, as well as the material reflection properties at the hit point [14, 15]. As the rendering module is parameterized by the sensor model via the **SensorViewConfiguration**, it is possible to parameterize the origin and alignment of the rays to shoot.

To model beam divergence with infinitesimally thin rays, the concept of super-sampling is applied, as visualized in Fig. 7. Rather than shooting only a single ray per beam ($\otimes$) of the overall beam alignment, beam divergence is modeled by shooting multiple rays for each beam ($\circ$). The chosen corner-case demonstrates the need for beam divergence within the sensor model. When the lidar beam hits an edge, the simple ray casting produces only a single echo from the center of the beam, with the distance to this exact spot. The real sensor instead will receive reflections from the whole area within the beam and therefore produces two peaks in the signal after analog–digital converter (ADC). Super-sampling now solves this lack of fidelity. Subsequently and beyond state of the art, the beam pattern of the simulated real sensor is applied over the intensity calculation for each beam, including the light emission characteristic of the diode and the beam shape and focus formed by the lens.

With this simulation approach, as shown by Tamm-Morschel [16], it becomes possible to approximate the signal as available after the ADC within the sensor. Here, an additional noise floor can be added to the intensity over distance depending on the ambient light. The super-sampled PCL as collected from rendering, beam pattern application, and noise models for distance and intensity is ordered by radial distance, as visualized with green dots in Fig. 8. Next, the counter resolutions of the sensor's intensity ΔI and distance calculation Δr are applied, which results in the black signal in Fig. 8. Then, the dynamic threshold is set depending on the signal–noise ratio

Fig. 8 Intensity calculation, counter resolution, and dynamic thresholding within the sensor model

(SNR) and the peaks are determined with their corresponding distance values to be set for the output echoes. Additionally, either the echo-pulse width or intensity is derived from thresholding, depending on the configuration of the real sensor that is simulated.

3.3 Tracking simulation module

Tracking algorithms combine the current and previous measurements under the assumptions of certain motion models for estimating the number of objects along with their respective pose. In lidar, object detection can be performed after clustering (detection of coherent subsets in the PCL) and segmentation of the PCL and the tracking algorithm assigns PCL segments to objects.

For computation efficiency and as a solution for lacking the actually used algorithm within the sensor system, the tracking simulation module implemented in the modular lidar system model uses GT information to assign the points from the cloud to the objects. For each object in the sensor's field of view, the algorithm maps corresponding points from the (simulated) lidar PCL to the objects that are within the field of view of the sensor. When sharing the same coordinate frame as the PCL, all points inside the object's bounding box under consideration of its orientation, which is available from GT, are now assigned to the respective object. The result is a set of ideally separated segments without any segmentation errors.

Actual tracking algorithms are designed in such a way that an object is initialized, e.g., after five consecutive detections, or five out of ten, and deleted again if it was not detected in, e.g., ten consecutive measurements. Because of the GT usage, all points in the PCL are assigned correctly to segments and always detected. This would cause all segments to be treated as objects after, e.g., five time steps, also in cases when they consist of only a single point. For higher fidelity, a faithful track management model is designed. It is assumed that the existence probability P is directly related to the (varying) number of points n that is assigned to a segment as shown in Fig. 9: If the number of points n of the PCL segment exceeds a predefined threshold

Fig. 9 Simulated track management

N , the existence probability P is increased by a fixed increment with every time step, otherwise it is decreased. If the object has already existed in the last OL, the threshold is set to one point, as real lidar sensors like the Ibeo LUX 2010 do not lose leaving objects without occlusion, once detected. Finally, all objects exceeding a predefined existence probability are listed in the output OL. For reliable results, the calculation of the existence probability is expanded to cover all conditions where the ideal and real OL differ.

In addition, the segmented PCL is utilized to determine realistic bounding box size, position, and heading for the objects, as the ideal values from GT often differ significantly: As an example, lidar sensors that comprise only a small number of layers are not expected to report a height that is even marginally correct. Figure 10 shows the difference in height of GT and real bounding boxes, where the real bounding box, calculated from the measured points, will most likely be smaller than the ideal one. As for segmentation and track management, the fidelity of the simulated PCL directly influences the fidelity of bounding box dimension, position, and orientation.

Finally, the focus lies on the reference point calculation. It is known from the Ibeo LUX 2010 lidar sensor system that the reference point of the object is, for example, calculated as center of gravity of the segmented object candidate, or as its geometric center [13]. The tracking simulation module

Fig. 10 Bounding box height of a nearby object from GT and calculated from PCL (denoted by black stars)

provides both implementations. Additionally, a third implementation is presented, where the corners of the L-shaped PCL segments are captured as the reference point during tracking, which promises more stable behavior than the first ones. The implementation shows the model's parameterization capability to serve different possible requirements and test purposes.

Detection ability for objects is integrated, as presented. However, false positives as well as the separation ability for objects that are close to each other are not yet considered. As already mentioned, the fidelity of the results highly depends on the fidelity of the simulated PCLs. This reflects the actual tool chain of the real sensor system and, therefore, is without any alternative, when realistic error propagation simulation is required. In summary, it combines efficiency in computation with fidelity. Overall, the complexity of object detection and tracking is reduced to a small number of parameters, which can be set by the user to mimic a particular implementation of a tracking algorithm.

3.4 Classification simulation module

A detailed description of the classification simulation module is not in scope here. Nevertheless, a high-density PCL is expected to facilitate correct classification, for which a

wide body of methods exists [17, 18]. It has been found that classification shows distance sensitivity for key features (i.e., dimensions, speed, relative position) to classify the objects [19]. In this case, for modeling the performance of classification algorithms, combining simulated PCL and GT like in the proposed tracking module is promising.

4 Evaluations with real measurements

To evaluate the presented approach, an experiment is carried out on a test site and replicated in simulation. The comparison of simulation and measurement is done by means of objective metrics as required in an overall validation methodology [20].

4.1 Experiment description

The chosen intersection scenario as illustrated in Fig. 11 reflects a typical urban traffic situation: The ego vehicle (car_{ego}) drives up to a stationary vehicle (car_{stat}) and waits for car_{move} to turn. During the cornering process, car_{move} is occluded by (car_{stat}). The ego car carries two Ibeo LUX 2010 [13] sensors mounted at front-left and front-right in combination with Ibeo's ECU for sensor fusion, reporting both OL and PCL. The position of moving objects is recorded using a global navigation satellite system (GNSS) receiver with real-time kinematic (RTK) accuracy, allowing for recreation of their trajectories in simulation.

4.2 First results

For evaluation, the simulated OLs are compared with the OLs from the real lidar sensor system and the reference trajectory from RTK together with the GT bounding box dimensions. A basic and applicable metric in this case is the root mean squared error (RMSE) as proposed by Morton [21]. It quantifies the deviations of the center point trajectory's coordinates and bounding box dimension's trends of the cars of simulated and real measurement from GT. In this

Fig. 11 Schematic and photograph of the scenario [8]

work, the RMSEs are normalized with the corresponding GT values $v_{GT,1}$ for the first time step of the test run ($t=1$) to achieve a dimensionless metric value between 0 and 1 that has 0 as its optimum. The mean within the RMSEs is the mean of the squared errors at every time step t of the vectors $\vec{v}_i$ and $\vec{v}_j$ of the length T . So, the RMSEs for each dimension/coordinate $\vec{v}$ are calculated as.

$$\overline{\text{RMSE}}(\vec{v}_i, \vec{v}_j, \vec{v}_{GT}) = \frac{\text{RMSE}(\vec{v}_i, \vec{v}_j)}{v_{GT,1}} = \frac{\sqrt{\frac{\text{mean}_{t=1, \dots, T}((v_{i,t} - v_{j,t})^2)}{v_{GT,1}}}}$$

where $i, j \in \{\text{sim.}, \text{real}, \text{GT}\}$.

The normalizing causes the deviations of, e.g., the x -direction of the positions that are usually much higher in absolute values than in z -direction to be better comparable as they are in fractions of the first entry of the corresponding GT values $\vec{v}_{GT}$.

Table 1 shows the calculated $\overline{\text{RMSE}}$ s for the scenario as described in Sect. 4.1 lasting 7.62 s. In the evaluated simulation, the faithful sensor model with super-sampling of the beam was used for PCL generation that was processed by the OL tracking model to show the ability of the sequential approach. The sensor simulation in this case reflects two sensor models, parameterized to reflect the real sensors from the measurements (here two Ibeo LUX 2010).

The behavior can be seen in Fig. 12, where the time steps are visualized by the change of the color saturation

from light to dark. The GT bounding boxes are visualized in dotted black lines, with car_{ego} starting at (0, 0, 0), car_{stat} in the center and car_{move} performing the turn around it. In green, the PCL is plotted, the bounding boxes from the real measurement are plotted in blue, and the simulated OL is plotted in red. In this matter, it visualizes the resulting values for the different RMSEs from Table 1.

As targeted, comparing the bounding box's length $\vec{l}$, width $\vec{w}$, and height $\vec{h}$ for simulated and real measurement, the RMSEs are very low, i.e., the fidelity is very high. By considering the point cloud, as described in Sect. 3.3, especially the simulated values for the height show significantly higher fidelity than an ideal model giving only GT values. For the same reason, the simulated trajectory of the moving car shows higher fidelity in $\vec{z}$ than the GT trajectory does. Nevertheless, in $\vec{x}$ and $\vec{y}$ coordinates the fidelity of the used model is lower, as the OL from real measurements shows higher deviations from GT than the simulated OL, which is caused by a tracking model that is outperforming real algorithms by design. To achieve higher fidelity of the simulation in this case, the actual tracking algorithms could be used instead of the tracking model.

Fig. 12 Simulation of the scenario

Table 1 $\overline{\text{RMSE}}(\vec{v}_i, \vec{v}_j, \vec{v}_{GT})$ in fractions of the first entry of the corresponding GT values $v_{GT,1}$ for real/simulated measurements of the GT dimensions and trajectories (0 means identity)

i vs. j	$\overline{\text{RMSE}}(\dots)$ for the moving car						$\overline{\text{RMSE}}(\dots)$ for the static car					
	$\vec{x}_i, \vec{x}_j$	$\vec{y}_i, \vec{y}_j$	$\vec{z}_i, \vec{z}_j$	$\vec{l}_i, \vec{l}_j$	$\vec{w}_i, \vec{w}_j$	$\vec{h}_i, \vec{h}_j$	$\vec{x}_i, \vec{x}_j$	$\vec{y}_i, \vec{y}_j$	$\vec{z}_i, \vec{z}_j$	$\vec{l}_i, \vec{l}_j$	$\vec{w}_i, \vec{w}_j$	$\vec{h}_i, \vec{h}_j$
Real vs. GT	0.023	0.143	0.438	0.088	0.296	0.622	0.025	0.027	0.685	0.138	0.025	0.806
Sim. vs. GT	0.007	0.071	0.501	0.092	0.286	0.583	0.015	0.016	0.753	0.137	0.029	0.789
Sim. vs. real	0.029	0.139	0.095	0.048	0.242	0.105	0.014	0.015	0.085	0.026	0.016	0.048

5 Conclusion and outlook

In conclusion, first results are promising that the sequential and modular approach for lidar sensor system simulation is able to generate simple or faithful PCLs that lead, combined with the available GT, to faithfully reproduced OLs, as required for simulation-based safety validation of HAD. The modular structure following standardized interfaces allows replacing a proxy algorithm with the actual implementation, when available and needed. With the modular approach and parameterizable fidelity in the plugged-in modules, however needed, it allows to be used for all kinds of simulations where a sensor system is involved. As a consequence, the presented sequential approach with exchangeable modules for different functions within the simulated sensor system will be used as a role model within the community for simulation-based safety validation of automated driving, at first in the research project *SET Level 4to5*.

The approach for the data processing model to use GT and a simulated PCL as basis for OL simulation can be used as a digital twin of a sensor system, when parameterized as such. Possible use cases include algorithm development on planning or behavior level, or stress testing of data processing pipelines. The presented approach also addresses the topic of requirements engineering, for example to obtain the required quality of tracking or classification that needs to be provided by the sensor system. Avoiding the algorithmic complexity of tracking and classification by a parameterizable GT manipulation using simulated PCLs causes deviations that should be considered.

As an outlook, for the existence probability during tracking, this simulation module could consider the dimension of the segment in addition to its number of points. Furthermore, the module for tracking could be extended with, e.g., false positives, as well as the separation ability for objects that are close to each other. The module for classification could be implemented, e.g., by calculating classification probabilities based on object features like size, velocity and number of points. Finally, to reflect processing other than object detection, there could be implemented modules for, e.g., occupancy grids for free-space estimation.

Acknowledgements This paper was partially supported by the projects *SET Level 4to5* and *PEGASUS*, funded by the German Federal Ministry for Economic Affairs and Energy (A 15012 Q, 19 A 19004E) based on a decision of the Deutsche Bundestag. It was also partially supported by the project *ENABLE-S3* that has received funding from the ECSEL Joint Undertaking under Grant Agreement no. 692455. This joint undertaking receives support from the European Union's HORIZON 2020 research and innovation program and Austria, Denmark, Germany, Finland, Czech Republic, Italy, Spain, Portugal, Poland, Ireland, Belgium, France, Netherlands, United Kingdom, Slovakia, Norway.

Funding Open Access funding provided by Projekt DEAL.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Rosenberger, P., Holder, M. F., Huch, S., Winner, H., Fleck, T., Zofka, M. R., Zöllner, J. M., D'Hondt, T., Wassermann, B.: Benchmarking and Functional Decomposition of Automotive Lidar Sensor Models. In: 2019 IEEE intelligent vehicles symposium (IV), Paris, France (2019)
2. International Organization for Standardization: SO/DIS 23150: Road vehicles - data communication between sensors and data fusion unit for automated driving functions - logical interface. Standard ISO/DIS 23150:2020 (2020)
3. Hanke, T., Hirsenkorn, N., van Driesten, C., Garcia Ramos, P., Schiementz, M., Schneider, S., Biebl, E.: Open simulation interface: a generic interface for the environment perception of automated driving functions in virtual scenarios. (2017). <https://www.hot.ei.tum.de/forschung/automotive-veroeffentlichungen/>. Accessed 12 Feb 2020
4. MODELICA Association, Project FMI: Functional Mock-up Interface for Co-Simulation. (2017). https://svn.modelica.org/fmi/branches/public/specifications/v1.0/FMI_for_CoSimulation_v1.0.1.pdf. Accessed 12 Feb 2020
5. Van Driesten, C., Schaller, T.: Overall approach to standardize AD sensor interfaces: simulation and real vehicle. In: Bertram, T. (ed.) Fahrerassistenzsysteme 2018, pp. 47–55. Springer Fachmedien Wiesbaden, Wiesbaden (2019)
6. Herrmann, M., Schön, H.: Efficient sensor development using raw signal interfaces. In: Bertram, T. (ed.) Fahrerassistenzsysteme 2018, pp. 30–39. Springer Fachmedien Wiesbaden, Wiesbaden (2019)
7. Dietmayer, K.: Predicting of machine perception for automated driving. In: Maurer, M., Gerdes, J.C., Lenz, B., Winner, H. (eds.) Autonomous driving: technical, legal and social aspects, pp. 407–424. Springer, Berlin Heidelberg (2016)
8. Aust, P.: Entwicklung eines lidartypischen Objektlisten-Sensormodells. M. Sc. Thesis, Technische Universität Darmstadt, Darmstadt, Germany (2019)
9. Open Simulation Interface Documentation. (2019). <https://opensimulationinterface.github.io/open-simulation-interface/>. Accessed 12 Feb 2020
10. Open Simulation Interface 3.1: *OSI sensor model packaging*. (2019). <https://opensimulationinterface.github.io/osi-documentat>

- [ion/osi-sensor-model-packaging/doc/specification.html](#). Accessed 12 Feb 2020
11. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software. Addison-Wesley Longman Publishing Co. Inc, Boston (1995)
 12. Rosenberger, P., Holder, M. F., Zirulnik, M., Winner, H.: Analysis of real world sensor behavior for rising fidelity of physically based lidar sensor models. In: 2018 IEEE intelligent vehicles symposium (IV), Changshu, Suzhou, China (2018)
 13. Ibeo Automotive Systems GmbH: Operating Manual ibeo LUX 2010 Laserscanner, vol. 1 p. 6. (2014)
 14. Rothkirch, A.: Systematische Bestimmung der bidirektionalen spektralen Reflexionsfunktion (BRDF) von städtischen Flächen aus multispektralen Luftbildern und Labormessungen. Ph. D. Thesis, Universität Hamburg, Hamburg, Germany (2001)
 15. Shell, J. R.: Bidirectional reflectance: an overview with remote sensing applications & measurement recommendations. Center for imaging science, Rochester Institute of Technology, Rochester, New York, USA. (2014)
 16. Tamm-Morschel, J. F.: Erweiterung eines phänomenologischen Lidar-Sensormodells durch identifizierte physikalische Effekte. M. Sc. Thesis, Technische Universität Darmstadt, Darmstadt, Germany (2020)
 17. Lang, A. H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: PointPillars: fast encoders for object detection from point clouds. In: The IEEE conference on computer vision and pattern recognition (CVPR) (2019)
 18. Zhou, Y., Tuzel, O.: VoxelNet: end-to-end learning for point cloud based 3d object detection. In: The IEEE conference on computer vision and pattern recognition (CVPR) (2018)
 19. Holder, M. F., Rosenberger, P., Bert, F., Winner, H.: Data-driven derivation of requirements for a lidar sensor model. In: 2018 Graz symposium virtual vehicle (GSVF), Graz, Austria (2018)
 20. Rosenberger, P., Wendler, J. T., Holder, M. F., Linnhoff, C., Berghöfer, M., Winner, H., Maurer, M.: Towards a generally accepted validation methodology for sensor models - challenges, metrics, and first results. In: 2019 Graz symposium virtual vehicle (GSVF), Graz, Austria (2019)
 21. Morton, P., Douillard, B., Underwood, J.: An evaluation of dynamic object tracking with 3D LIDAR. In: Proceedings of the australasian conference on robotics & automation (ACRA) (2011)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.