Accepted article of 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, June 09-12, 2019. DOI: 10.1109/IVS.2019.8814081.

Benchmarking and Functional Decomposition of Automotive Lidar Sensor Models

Philipp Rosenberger[1], Martin Holder[1], Sebastian Huch[1], Hermann Winner[1],
Tobias Fleck[2], Marc René Zofka[2], J. Marius Zöllner[2],
Thomas D'hondt[3] and Benjamin Wassermann[4]

***Abstract*— Simulation-based testing is seen as a major requirement for the safety validation of highly automated driving. One crucial part of such test architectures are models of environment perception sensors such as camera, lidar and radar sensors. Currently, an objective evaluation and the comparison of different modeling approaches for automotive lidar sensors are still a challenge. In this work, a real lidar sensor system used for object recognition is first functionally decomposed. The resulting sequence of processing blocks and interfaces is then mapped onto simulation methods. Subsequently, metrics applied to the aforementioned interfaces are derived, enabling a quantitative comparison between simulated and real sensor data at different steps of the processing pipeline. Benchmarks for several existing sensor models at a concrete selected interface are performed using those metrics by comparing them to measurements gained from the real sensor. Finally, we outline how metrics on low-level interfaces can correlate with results on more abstract ones. A major achievement of this work lies within the commonly accepted interfaces and a common understanding of real and virtual lidar sensor systems and, even more important, an initial guideline for the quantitative comparison of sensor models with the ambition to support future validation of virtual sensor models.**

I. INTRODUCTION

Simulation-based testing is seen as a requirement for the validation of the safety of automated driving [1]. In consequence, the research project ENABLE-S3 [2] has been established to find cross-domain test methods, including software-, hardware-, and vehicle-in-the-loop testing, for safety validation of Automated Cyber-Physical systems (ACPS). A central part of most ACPS is formed by its perception sensor systems. For conducting X-in-the-loop (software, hardware, vehicle) tests, corresponding simulation models for all components are in demand: In SiL tests, sensor models are required to feed the implemented functions of the ACPS. In HiL and ViL tests, real-time sensor models can be used for sensor stimulation, where real sensor hardware is exposed to synthetic data.

[1]Philipp Rosenberger, Martin Holder, Sebastian Huch and Hermann Winner are with Institute of Automotive Engineering (FZD), Technische Universität Darmstadt, 64287 Darmstadt, Germany {holder, rosenberger, winner}@fzd.tu-darmstadt.de, contact@sebastianhuch.de

[2]Tobias Fleck, Marc René Zofka and J. Marius Zöllner are with the FZI Research Center for Information Technology, 76131 Karlsruhe, Germany {tfleck, zofka, zoellner}@fzi.de

[3]Thomas D'hondt is with Siemens Industry Software NV, 3000 Leuven, Belgium thomas.dhondt@siemens.com

[4]Benjamin Wassermann is with TWT GmbH, Ernsthaldenstraße 17, 70565 Stuttgart, Germany, benjamin.wassermann@twt-gmbh.de

Alternatively, sensor hardware can be replaced by directly injecting synthetic data into the functional units of the ACPS. Even combinations of simulated and real sensor signals can be used for stimulation [3]. Before virtual sensor data can be used in such tests, they must be validated in order to provide a high level of quality and fidelity. Sensor model validation, however, is a complex task and globally accepted quality criteria for sensor models have not been established yet. In this work, the focus is put on automotive lidar sensors, which are frequently used in addition to camera and radar sensors for environment perception in automated driving applications, such as valet parking. By deriving metrics for the comparison of synthetic against real data on different interfaces, this work takes a first step towards enabling sensor model validation.

The remainder of this work is organized as follows: At first, a description of lidar sensors is given where we carefully derive a differentiation between sensors and sensor systems for which we deploy the method of functional decomposition. As a result, the different output interfaces of existing lidar sensor systems are listed and specified. Quantitative metrics are then proposed to benchmark and evaluate synthetic sensor data at the previously derived interfaces. Finally, an exemplary application of the derived metrics is given by evaluating different virtual lidar sensor models against data from performed real-world measurements.

II. RELATED WORK

The method of functional decomposition was introduced in [4] in the context of safety validation for automated driving. There, six functional layers of the driving function were derived to reduce the number of concrete scenarios that have to be covered for safety validation. The functional layers of *information reception* and *information processing* are now further functionally decomposed here for an example of a lidar sensor systems. It enables to derive generic interfaces of such systems later used for sensor model validation. Related to functional decomposition, a generic architecture for sensor models has been proposed [5].

For sensor model validation, AIAA definitions [6] are often applied [7]–[9]. Validation in this context means the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended use of the model. It is mostly based on the comparison of real and synthetic data, as presented e.g. in [10]. The data is compared using metrics and, therefore, it

Accepted article of 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, June 09-12, 2019. DOI: 10.1109/IVS.2019.8814081.

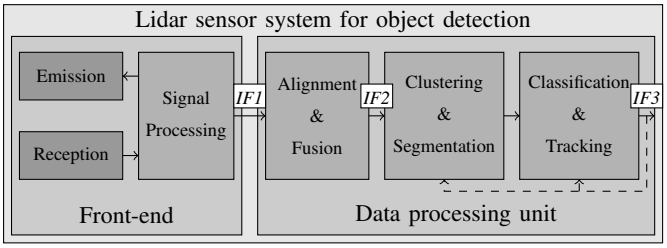


Fig. 1: Lidar Sensor System for Object Detection

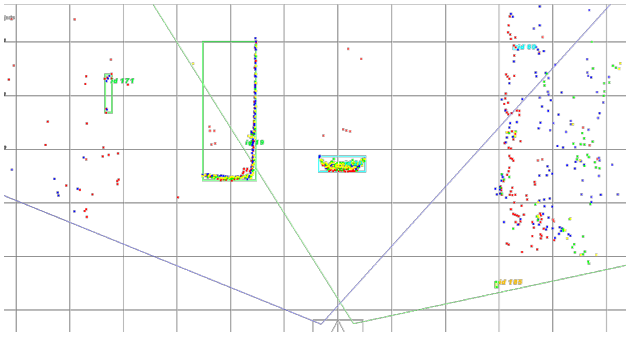


Fig. 2: Object list and point cloud of two fused Ibeo LUX 2010 lidar systems, taken during scenario 5

must be ensured that the collected data and the performed scenarios contain the aspects and effects that shall be tackled by the model. The results of the metrics still have to be interpreted by one or more "experts", though it is not clear what qualifies to be one. Additionally, [8] shows a comparison of real and synthetic lidar data in a static scenario is shown. The sensor data is compared on occupancy grid level using quantitative metrics. However, this study does not cover other lidar data types, such as point clouds or object lists.

Prerequisite for validation is a list of valid requirements, which is predefined by the stakeholders of the model and not by its developer. In fact, requirements indirectly define the metrics and scenarios to be selected for validation. But, no method for deriving decent requirements for sensor models has been reported, yet. Thereby, a paradox appears in this context, as typically no stakeholder has the complete overview and knowledge about the required and the possible fidelity of sensor models, yet, which are generally not identical. While [8] is directly related to the work in the following, progress here is provided by benchmarking different models in different simulation tools. In fact, this enables writing requirements and interpreting metrics from the insight into lidar sensor systems and from first benchmarks of different modeling approaches and interfaces.

III. FUNCTIONAL LIDAR SENSOR SYSTEM DECOMPOSITION

In the scope of this work, the focus is put on the decomposition of lidar sensor systems for object detection, landmark detection, target tracking and classification tasks, but can be easily extended to additional applications.

As a first definition, there is a distinction between the lidar sensor *front-end* and the *data processing unit*. While the former focuses on emitting laser pulses in the environment and measuring their echoes, the latter includes all steps of data reduction towards an *object list*. Fig. 1 shows the decomposition of a lidar sensor system. The output of the sensor front-end is defined as the *raw scan* (*IF1*), whereas the output of the processing unit is an object list that contains geometric and physical properties of inferred objects, such as position, velocity or classification (*IF3*). In between, the *point cloud* (*IF2*) is an intermediate interface that consists of a Cartesian representation of the radially measured distances of a lidar scan. It may contain data from multiple lidar sensors. In this case, temporal and spatial calibration into a common reference frame has been performed previously.

As simulating signals at the possible interface right after digitization and sampling is not very common and mostly not available in current simulation tools, it is not considered as a generic interface for functional decomposition and evaluation. Consequently, the raw scan is chosen as a first generic interface *IF1* for the functional decomposition of lidar systems for simplicity. Therefore, thresholding for echo detection while excluding false negative detections and producing and handling of false positive echoes is already included in the first generic interface *IF1*. A raw scan consists of a list of tuples of measured values, as for example the measured distance and additional values like intensity or echo-pulse-width, depending on the sensor implementation. Also depending on that implementation, one or more echoes are saved in the raw scan for each transmitted beam. This involves a decision, which and how many of them are going into the raw scan. Therefore, the number of saved tuples equals the number of transmitted beams multiplied by the number of saved echoes per transmitted beam. The header of one raw scan list has to provide that number, and e.g. temporal information and the angular and radial resolution, as well as the maximal and minimal angular and radial range. Thus, *IF1* is a spherical list representation of measured data.

Once the raw scan is provided, the next generic step for multi-sensor lidar sensor systems is to fuse several sensors into a single coordinate system, mostly Cartesian and originated at the center of the ego vehicle's rear axle. The vehicle coordinate system is orientated with x to the front, y to the left and z up in accordance with ISO 8855:2011 [11]. Again, in single-sensor systems, fusion and transformation are not mandatory.

The result of the possible alignment and fusion steps, which includes extrinsic calibration, is named point cloud *IF2* and can be used as a starting point for several use cases, as already mentioned. If only one sensor is used, alignment to the vehicle coordinate system can still be necessary for further usage.

In the use case of lidar-based object perception, the point cloud is typically processed continuously by a perception pipeline that consists of a clustering and segmentation

Accepted article of 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, June 09-12, 2019. DOI: 10.1109/IVS.2019.8814081.

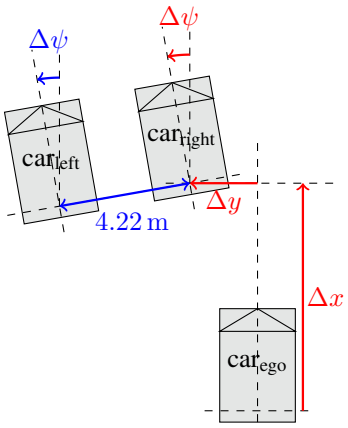


Fig. 3: Traffic jam scenario overview

TABLE I: Scenario parameters

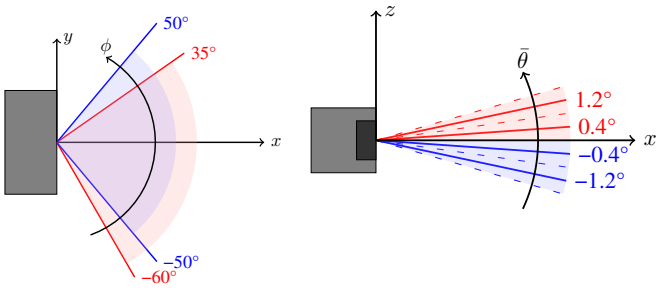
Scenario #	Δx / m	Δy / m	$\Delta\psi$ / °
1	50.06	−0.35	0.1
2	39.95	0.45	1.3
3	29.97	0.10	0.8
4	19.99	0.16	1.1
5	9.97	0.17	1.3

algorithm, as in [12], and of temporal fusion models such as Bayesian filters (e.g. Extended Kalman Filter, UKF or Multiple-Model-Filters [13], [14]) to keep track of objects over time. Hence, the combination of time data with motion models of target objects allows to reduce noise on the measurements and to estimate characteristics of the target that were not initially measured (e.g. velocity). Meanwhile, the object candidates get classified using information from tracking. This is achieved through mostly rule- or machine learning-based algorithms [15]. The point clusters and their corresponding object candidates are illustrated in Fig. 2.

IV. GENERATION OF REAL LIDAR DATA

A. Scenarios for model benchmarking

A measurement campaign was executed to obtain sensor data for benchmarking the performance of several simulation models. At first, a static traffic jam scenario was studied, where the ego car (car_{ego}) was placed behind two preceding vehicles (car_{left} and $\text{car}_{\text{right}}$), as shown in Fig. 3. During the campaign, the ego car was placed at five different longitudinal distances Δx to the cars for each independent measurement, which resulted in slightly different lateral distances Δy and orientation angles $\Delta\psi$ of the ego car with respect to the others. Both cars stayed in place for all measurements. The left car had at a lateral distance of 4.22 m to the right car with respect to their reference points and the rear axles of the cars in front were aligned. The different evaluated values for Δx, Δy and $\Delta\psi$ are summarized in Tab. I. The distance of the two cars was limited to 50 m to limit the influence of beam divergence on measurements. Ground truth positions for the ego vehicle and the right car were acquired using an RTK



(a) Top view (b) Side view

Fig. 4: Top view and side view of the Ibeo LUX 2010 lidar and its layers. In side view, center lines of layers are drawn in full lines, beam divergence in dashed lines.

TABLE II: Ibeo sensor position and orientation, relative to the vehicle coordinate frame.

Ibeo #	x / m	y / m	z / m	ψ / °
1	3.47	0.63	0.36	17.9
2	3.49	−0.58	0.36	−17.9

GPS system with an accuracy of 0.1 m for the position and 1° for the heading. Besides, all positions were double-checked with measurement tape.

For the evaluation of the object list interface (*IF3*) a dynamic scenario was carried out. In this scenario, the ego car was placed on a street and remained stationary. A second car in front of the ego car drove at walking pace from a distance greater than the sensors' maximal radial range straight towards the ego car. This scenario was stopped when the second car reached the ego car up to a vehicle length. As in the static scenario, both cars were equipped with RTK GPS systems to measure ground truth positions.

B. Details of the real lidar sensor system

The ego vehicle was equipped with two Ibeo LUX 2010 lidars, as described in [16]. Their position and orientation in the car is summarized in Tab. II, where all the coordinates refer to the vehicle frame located on the ground under the center of the rear axle and oriented as described in ISO 8855:2011 [11]. The maximal radial range of the sensor is 200 m and the minimal radial range is 0.3 m. The sensors perceive the environment using four layers spread along the elevation angle θ of the sensor. Those layers are each scanned independently by a laser moving along the azimuth angle ϕ, which is different for the two bottom and two top layers (Fig. 4) and leads to a total horizontal angular range of 110°.

The horizontal angular resolution of both sensors was set to 0.25°. In fact, this horizontal resolution here is obtained by a regular resolution of 0.5° and a zipper-like shift of the two upper layers with respect to the lower ones, as visualized in Fig. 5. The real sensor system shows beam divergence, as well, with 0.8° in vertical and 0.08° in horizontal direction. The resulting beam size is schematically shown in Fig. 5 for a radial distance of 20 m and indicated by the colored beams in

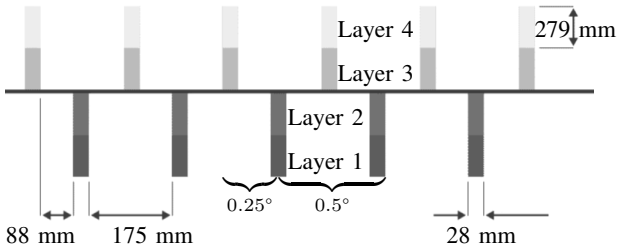


Fig. 5: Zipper-like layer orientation and beam size of Ibeo LUX 2010 at $20\,\mathrm{m}$ with an angular resolution of $0.25°$ [16]

Fig. 4. There, the middle of the beams is marked, which are located at −1.2°, −0.4°, 0.4°, and 1.2°. Including the beam divergence, this leads to an overall vertical angular range of 3.2°. Other effects on the real data like noise behavior, influences on the received signal's intensity and temporal effects are explained in detail in [17] and specifically for the presently used sensor in [18]. In the scenarios evaluated in this work, beam divergence at $\Delta x = 50\,\mathrm{m}$ is equal to 70 cm in the elevation angle θ and 7 cm in the azimuth angle ϕ.

Finally, the output interface of the sensor system for the static scenarios was set to obtain a raw scan, which corresponds to the previously defined *IF1*. For dynamic scenarios, the tracking part of the object list is evaluated, so this data is provided by the real system, as well.

V. GENERATION OF SYNTHETIC LIDAR DATA

There are different approaches for sensor simulation. Cao distinguishes them by white-box, black-box and gray-box models [19]. While white-box models focus on detailed simulation of sensor properties including physical phenomena or physical structures while neglecting run time, black box models focus on the general properties of a perception sensor and favor the run time. Gray-box models are a trade-off between white- and black-box models that try to offer necessary reality and efficiency for the validation of ACPS.

The interface that should be addressed by the sensor model plays a crucial role when reasoning about modeling approaches. For object lists on *IF3*, the task is to reproduce existence, state and class uncertainties of the actual sensor system. This is possible without knowledge about the actual physical coherences with data-driven black-box models as e.g. in [20] In case of low-level data simulation on *IF1* or *IF2* with high degree of fidelity, more physical aspects have to be modeled. This includes signal propagation models as used in ray tracing-like approaches, as well as noise models, beam divergence or temporal effects that occur. Additionally, the physical properties of the environment have to be mapped to the virtual world accordingly.

Nevertheless, physics can be considered in models on object list level, as well, which results in gray-box or phenomenological models as in [21]. Still, there is no clear separation when to call a model gray-box or black-box, stochastic, phenomenological, or data-based. In this work, we evaluate data on different interfaces, where in case of the object list data, it is derived from formerly generated point cloud data that is then processed by object perception algorithms developed to serve as state-of-the-art reference. At first, the point cloud data needs to be created, which will be described in the following.

A. Ray tracing methods

Ray casting and ray tracing both have their origins in computer graphics and describe techniques for the rendering of a three-dimensional scene. Ray casting [22, pp. 305–312] is a fast method for rendering, but is now mainly used in the context of volume visualization. Ray tracing [22, pp. 219–266], on the other hand, is used to generate synthetic photo-realistic images. The basic principle is to shoot rays into the scene from the perspective of a sensor. If a ray hits an object in the scene, the intersection point of the object relative to the sensor can be determined.

Further steps are considered in ray tracing. These include direct and indirect beam calculations between light sources in the scene and the intersection point. In combination with the material description of the hit object, the reflective energy perceived by the sensor is calculated. In order to determine further effects such as reflections, refraction, etc., further beams can be calculated from the intersection point depending on the material properties. This process can be repeated as often as desired until, for example, a maximal reflection depth is reached or the contribution of energy becomes too low.

Since lidar works according to the same principle, the ray casting method can be used for the purpose of lidar simulation. The rays are calculated directly by the sensor and the intersection points with the objects of the scene are collected as a result. A more advanced approach would be to use ray tracing where the "light source" of the active sensor is also modeled, as are material properties, atmospheric effects, etc. which influence the reflected energy, as they are mentioned in Sec. IV-B. In this work, however, only the simple ray casting is applied and the maximal range is considered.

The simple approach can easily be extended by including material, atmospheric, and distance in signal attenuation properties. In addition, multiple beams or oversampling can be used to simulate beam divergence.

In order to run a lidar simulation with ray casting effectively with today's hardware, the calculations are strongly parallelized by calculating several beams simultaneously. The scene remains fixed until all rays are calculated. With mechanical lidar sensors, however, each scan (horizontal or vertical) measures only one series of points per time step. This leads to a distortion of the resulting point cloud and is also known as rolling shutter for camera images, which is lost by using strong parallelization. However, this effect is negligible as long as there are no objects with higher velocities in the scene. In addition, a new generation of lidar sensors is currently being developed, as already mentioned, where the temporal order of collecting the echoes will be different and in case of flash lidars more similar to the parallel ray casting models.

Accepted article of 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, June 09-12, 2019. DOI: 10.1109/IVS.2019.8814081.

B. Depth buffering

Based on the simplified approach, a lidar sensor can be simulated very effectively via the pipeline of today's graphics cards. The Z-buffer (also referred to as depth buffer), which is needed for the concealment calculation during image rendering, can also be stored as depth map. It contains the depth information of the image relative to the camera sensor. With this information it is possible to reconstruct the beams and their intersections in the scene of a lidar sensor that has the same location as the camera sensor. The drawback, compared to the real ray casting, is the required resampling step to transfer the beams of the pinhole camera model to those of a lidar model. However, if oversampling is used, the error is small and the effect of beam divergence can also be taken into account.

C. Environment modeling

At last, it should be mentioned that the generation of synthetic data relies strongly on the environment models and simulation. Roads, buildings and all other objects need to represent the real world that the sensor is facing with respect to their materials, shapes and surfaces. The immense effort e.g. for modeling static objects in adequate precision like 5 cm deviation in geometric dimensions of the models themselves and their geo-referenced position is described in [8]. In the scope of this work, static scenarios are used in which no environment is modeled and only cars are considered to avoid complications caused by differences of the environment simulation compared to reality, as stated in Sec. V. This way, the influence of non-observable effects is minimized and the feasibility of the scenarios in the real world is ensured.

VI. METRICS FOR OBJECTIVE DATA COMPARISON

In this section, rather than visually comparing real and synthetic data in a subjective fashion, a list of possible metrics dealing with point cloud (*IF2*) and tracked object list (*IF3*) allowing for an objective comparison is given. Nevertheless, most metrics on *IF2* do work on *IF1*, as well.

A. Metrics on point cloud interface (IF2)

There are multiple metrics for comparing point clouds as described in [23]. They can be categorized as follows. The metrics of the first category can be applied directly onto the two point clouds without prior preparations of the clouds. An example for these metrics is the point cloud distance metric mentioned in [24]. This metric is based on the calculation of the minimal Euclidean distance from every point in the real point cloud $\mathbb{P} = \{p_1, ..., p_M\}$ to the simulated point cloud $\tilde{\mathbb{P}} = \{\tilde{p_1}, ..., \tilde{p_N}\}$, with the 3D-points $p_m, \tilde{p_n} \in \mathbb{R}^3$.

$$D_{\mathrm{PP}}^{'}(\mathbb{P}, \tilde{\mathbb{P}}) = \sum_{m} \min_{n} ||p_m - \tilde{p_n}||$$

And because this is a non-symmetrical metric, the worst case

$$D_{\mathrm{PP}}(\mathbb{P}, \tilde{\mathbb{P}}) = \max(D_{\mathrm{PP}}^{'}(\mathbb{P}, \tilde{\mathbb{P}}), D_{\mathrm{PP}}^{'}(\tilde{\mathbb{P}}, \mathbb{P}))$$

is assumed.

For comparison purposes, we divide the point cloud distance metric by the maximum number of points of either the real or simulated point cloud, to get

$$\bar{D}_{\mathrm{PP}}(\mathbb{P}, \tilde{\mathbb{P}}) = \frac{D_{\mathrm{PP}}(\mathbb{P}, \tilde{\mathbb{P}})}{\max(|\mathbb{P}|, |\tilde{\mathbb{P}}|)}.$$

To take the impact of noise into account, we calculate this metric for K consecutive scans and compute the average $\bar{D}_{\mathrm{PP}}^{K}(\mathbb{P}, \tilde{\mathbb{P}})$.

For the second category of metrics, two-dimensional occupancy grids are generated from the point clouds in a first step. When comparing different occupancy grids, various metrics can be found in the literature, like the overall error or the Barons cross correlation coefficient described in [8]. The occupied cells ratio (OCR) metric is chosen here, which is based on a cell-wise comparison of the real and simulated occupancy grid. The OCR describes the relation of true classified, occupied cells in the simulated occupancy grid $\tilde{\mathbb{G}} = \{\tilde{c_1}, ..., \tilde{c_J}\}$ to the total number of occupied cells in the real occupancy grid $\mathbb{G} = \{c_1, ..., c_I\}$ [25],

$$\mathrm{OCR}^K = \frac{\sum \zeta(\tilde{c}_j)}{\sum \zeta(c_i)}, \text{where}$$

$$\zeta(\tilde{c}_j) = \begin{cases} 1, & \text{if } P(\tilde{p_n} \text{ in } \tilde{c}_j) > 0.5 \\ 0, & \text{else} \end{cases}, \quad \zeta(c_i) = \begin{cases} 1, & \text{if } P(p_m \text{ in } c_i) > 0.5 \\ 0, & \text{else} \end{cases}.$$

For the calculation of OCR^K, K consecutive scans are considered for the generation of the real and simulated occupancy grids. With grids that were accumulated over time, the impact of noise is taken into account. Thus, $P(p_m \text{ in } c_i)$ indicates how often a cell was occupied during K scans. In case of $\bar{D}_{\mathrm{PP}}^{K}$, 0 m would be the optimum and in case of OCR^K, it would be 1.

B. Metrics on object list interface (IF3)

The evaluation of the object list interface recorded in a dynamic scenario can be done on different levels. On the first level, the quality of the tracking algorithm is assessed by the well-known Optimal Subpattern Assignment (OSPA) metrics, which are based on the Wasserstein distance. Several versions of the OSPA metric are compared in [23], but because of its applicability in dynamic scenarios with multiple objects, the Optimal Subpattern Assignment for Multiple Tracks (OSPA-MT) metric, which is described in [26], is selected. This metric measures the distance between two finite sets of tracks while also considering false and missed tracks. Additionally, the Jaccard index [27] is calculated, which is typically used for the assessment of detection and tracking algorithms, in which geometric information of objects is available. It is based on the comparison of the ground truth and estimated object bounding box. To verify the results of the segmentation and classification algorithms, a confusion matrix can be built as shown in [28]. This matrix is the basis for calculating the three metrics precision, recall and F1 Score, which in turn are a quality measure for the algorithms. The results of the Jaccard index and OSPA-MT metric are not shown in a single scenario, but used to show a correlation, as described in the following.

Accepted article of 2019 IEEE Intelligent Vehicles Symposium (IV), Paris, France, June 09-12, 2019. DOI: 10.1109/IVS.2019.8814081.

TABLE III: Overview of implemented and benchmarked models

Model #	M1	M2
Rendering	Ray casting	Ray casting
ϕ resolution	0.25°	0.25°
ϕ range	−60° to 50°	−60° to 50°
θ resolution	0.8°	0.8°
θ range	−1.2° to 1.2°	−1.2° to 1.2°
r resolution	Float32	0.04 m
r range	0.30 m to 200 m	0.30 m to 200 m
Layer shift (Fig. 4)	Yes	Yes
Zipper shift (Fig. 5)	Yes	Yes
Additional effects	No	No

Model #	M3	M4
Rendering	Z-Buffer	Ray casting
ϕ resolution	0.25°	0.25°
ϕ range	−55° to 55°	−50° to 50°
θ resolution	0.8°	0.8°
θ range	−1.2° to 1.2°	−1.2° to 1.2°
r resolution	24Bit non linear*	0.04 m
r range	0.30 m to 200 m	0.30 m to 200 m
Layer shift (Fig. 4)	No	No
Zipper shift (Fig. 5)	No	No
Additional effects	No	No

C. Correlation between metrics on different interfaces

As a last step of the evaluation, the correlation between metrics on point cloud interface (*IF2*) and on object list interface (*IF3*) is evaluated by computing the Pearsons correlation coefficient. This shows the influence of *IF2* on *IF3* and allows us to precisely detect errors in the tool chain. Therefore, metrics on *IF2* and *IF3* during the dynamic scenario need to be aligned over time. Afterwards, a scan-by-scan comparison for every time stamp, in which the metrics of *IF2* and *IF3* are computed, is made by calculating the Pearsons correlation coefficient over all time stamps, as described in [23].

VII. EVALUATED MODELS AND BENCHMARKS

A. Metric evaluation by car misplacement

As metric evaluation with respect to their intended usage is a required step during validation, both metrics have been evaluated regarding their sensitivity to misplacement of objects in simulation. Fig. 6 shows the results of a variation of the distance between car_{left} and $\text{car}_{\text{right}}$ at a distance Δx to car_{ego} of 10 m. OCR^{250} was calculated for 250 scans at 12.5 Hz of real point clouds, due to the noise, and shows very high sensitivity due to its binary character, whereas $\bar{D}_{\text{PP}}^{250}$, with the same 250 scans, is ascending with higher error in distance, as expected. For the simulated data, as no model is considering noise in this study, it does not matter if 1 or 250 scans are included in the evaluation. As the misplacement of an object can be caused by errors in reference data or small sensor calibration errors, this has to be taken into account, when results are presented in the following section.

*In this case a logarithmic 24 bit depth buffer was used. In contrast to a linear depth buffer, the 24 bits are not distributed evenly over the visible area, but logarithmically. This has an advantage if the visible range is very large and the depth resolution becomes very coarse in the linear case.

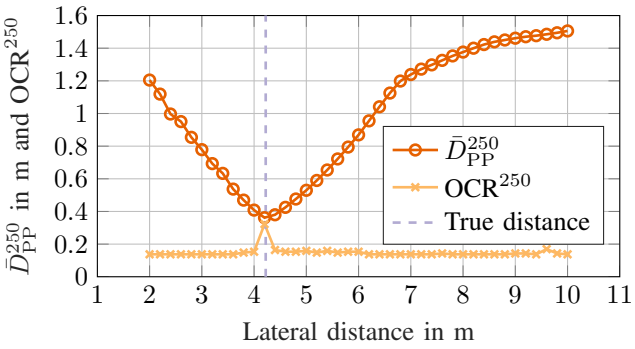


Fig. 6: Results of $\bar{D}_{\text{PP}}^{250}$ and OCR^{250} for different lateral distances between car_{left} and $\text{car}_{\text{right}}$

The error of the reference data in the described scenario is approximately 10 cm, as RTK GPS on both cars is approximately 1 cm off and the measuring error while obtaining the position of the devices in the cars is about 4 cm. For the measurements, [16] lists a distance resolution of 4 cm with a repeat accuracy (1 σ) of 10 cm for the Ibeo LUX 2010.

B. Static scenario for benchmarking of point cloud models

For the model benchmarking on point cloud interface (*IF2*), only the two objects described in Sec. IV-A with identical 3D models for the cars are used in each simulation. The environment is not considered, as stated in Sec. V. The cell size for OCR^{250} in all benchmarks here is set to 0.1 m. Different simulation tools are used to generate the data and the rendering techniques, as well as the considered effects and specifications of the real sensor in the lidar sensor model implementations can be seen in Tab. III. The so-called additional effects could be e.g. noise, beam divergence, multi-path reflections of the signal, multiple echoes per beam, rolling shutter, and motion blur. All of them offer potential for improvements of the models' fidelity. Additionally, no signal intensities or echo pulse widths are simulated, as they would need different metrics for their evaluation. Besides, only the first echo of the real lidar sensors have been considered for comparison with simulated data in this work.

The obtained results can be observed in Fig. 7 for $\bar{D}_{\text{PP}}^{250}$ and Fig. 8 for OCR^{250} for their exact values. At first, they show the tendency that $\bar{D}_{\text{PP}}^{250}$ increases with increasing Δx as real data is influenced by beam divergence and radial measurement accuracy of the real sensor. Then, OCR^{250} is falling with increasing Δx, which is most likely based on less occupied cells in the real data compared to the simulated one because of the not considered signal attenuation in all models. If the simulated and real point clouds are identical, but shifted just one cell in the occupancy grid for the chosen cell size, OCR^{250} gets very low or even 0, as potentially happening for model M1.

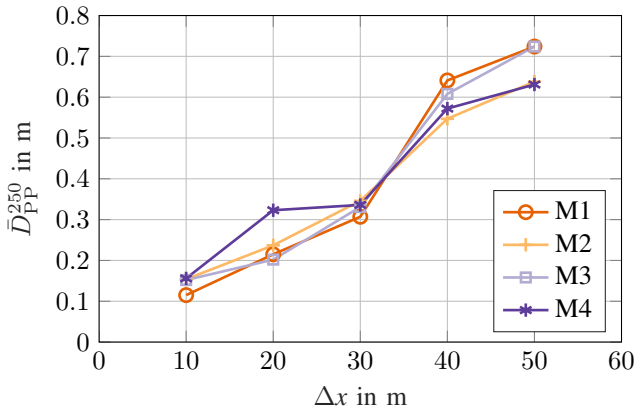


Fig. 7: Results of $\bar{D}_{\mathrm{PP}}^{250}$ for different models (optimum: 0 m)

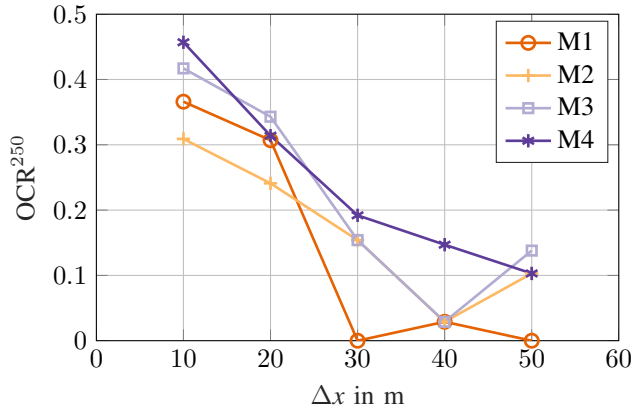


Fig. 8: Results of OCR^{250} for different models (optimum: 1)

C. Dynamic scenario to show correlation of metrics

The analysis of the correlation between the metrics on *IF2* and *IF3* is done by calculating the two point cloud interface metrics for every time stamp of the dynamic scenario and comparing them with the two object-list interface metrics, which are calculated for every time stamp as well. Only the results of M1 were considered in this calculation. Furthermore, we limit the evaluation of *IF3* to the two tracking metrics described before. Tab. IV shows the Pearsons correlation coefficient for all combinations of *IF2* and *IF3* metrics. As the distance between the second car and the ego car decreases, both the *IF2* and *IF3* metrics improve. In case of the OCR^{250} and Jaccard index, the values rise with lower distance, while the values of OSPA-MT and $\bar{D}_{\mathrm{PP}}^{250}$ metrics fall as their optimum is zero. For this reason, two correlation coefficients show a negative sign, nevertheless this means a high correlation. Overall, the correlation coefficient demonstrates the existence of a correlation between the interfaces. This knowledge can be used to specifically show the origin of an error in the processing chain.

TABLE IV: Correlation between *IF2* and *IF3*

		Point cloud metrics (*IF2*)	
		$\bar{D}_{\mathrm{PP}}^{250}$ in m	OCR^{250}
Tracking metrics (*IF3*)	OSPA-MT	0.77	-0.67
	Jaccard index	-0.78	0.73

VIII. CONCLUSION

In the present work, automotive lidar sensor systems for object detection are functionally decomposed and internal interfaces are defined in order to provide a methodology that supports the validation of virtual sensor models. Quantitative metrics were proposed for different interfaces that enable the comparison of synthetic and real sensor data. The proposed metrics were evaluated by using real sensor measurements that were compared to sensor data from four different virtual sensor models in a static scenario from the automotive domain. First investigations of influences and correlations between metrics from different interfaces were made for a target tracking scenario.

Since all investigated models use simple but common abstractions from the physical principles of real lidar sensors, each of the models performed poorly regarding the introduced metrics. In particular, it was shown that none of the models can generate sensor data for far objects precisely and ray launching methods have not shown to outperform depth buffer rendering for the task of point cloud generation, despite their theoretical advantages. However, the objective nature of the metrics derived in this work will support further developments of sensor models towards a level that can be considered to be sufficient for virtual validation of ACPS. This leads to the conclusion that functional decomposition of the sensor system and investigation of sensor data on different processing levels is key for the analysis of modeling errors.

In order to finally approach the validation of virtual sensor models, additional investigations have to be made in the future. At first, this includes a sensitivity analysis on interface metrics in large field tests which will make the case for the sensor system validation. Secondly, additional physical properties as well as their related metrics and influences on different use cases have to be encountered. Finally, an extension towards interfaces related to different use cases (e.g. lidar maps for SLAM) has to be made in order to combine them into a general validation argument for sensor models in examination.

IX. ACKNOWLEDGEMENT

This work was conducted within the ENABLE-S3 project that has received funding from the ECSEL Joint Undertaking and the German Federal Ministry of Education and Research (BMBF) under Grant Agreement no. 692455. Additionally, this work was partially supported under the German Federal Ministry of Education and Research (BMBF) grant 16ESE0132.

We gratefully acknowledge the helpful comments received from anonymous reviewers.

REFERENCES

[1] W. Wachenfeld and H. Winner, "The Release of Autonomous Vehicles", in *Autonomous Driving*, Springer Berlin Heidelberg, 2016, pp. 425–449.

[2] AVL LIST GMBH. (2018). Enable S3 Enable S3 official page. Accessed: 2019-01-25, [Online]. Available: https://www.enable-s3.eu/about-project/.

[3] M. R. Zofka, M. Essinger, T. Fleck, R. Kohlhaas, and J. M. Zollner, "The sleepwalker framework: Verification and validation of autonomous vehicles by mixed reality LiDAR stimulation", in *2018 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, 2018.

[4] C. Amersbach and H. Winner, "Functional Decomposition: An Approach to Reduce the Approval Effort for Highly Automated Driving", in *8. Tagung Fahrerassistenz*, 2017.

[5] T. Hanke, N. Hirsenkorn, B. Dehlink, A. Rauch, R. Rasshofer, and E. Biebl, "Generic architecture for simulation of ADAS sensors", in *2015 16th International Radar Symposium (IRS)*, 2015.

[6] *Guide for the Verification and Validation of Computational Fluid Dynamics Simulations (AIAA G-077-1998(2002))*. American Institute of Aeronautics and Astronautics, Inc., Jan. 1998.

[7] W. L. Oberkampf and T. G. Trucano, "Verification and validation benchmarks", *Nuclear Engineering and Design*, vol. 238, no. 3, pp. 716–743, Mar. 2008.

[8] A. Schaermann, A. Rauch, N. Hirsenkorn, T. Hanke, R. Rasshofer, and E. Biebl, "Validation of vehicle environment sensor models", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[9] J. T. Wendler, T. Menzel, M. Steimle, and M. Maurer, "Sensormodelle für die simulative Absicherung von automatisierten Fahrfunktionen", in *8. Tagung Fahrerassistenz*, 2017.

[10] E. Roth, T. Dirndorfer, K. v. Neumann-Cosel, *et al.*, "Analysis and validation of perception sensor models in an integrated vehicle and environment simulation", in *Proceedings of the 22nd Enhanced Safety of Vehicles Conference (ESV)*, 2011.

[11] International Organization for Standardization, "ISO 8855:2011: Road vehicles – Vehicle dynamics and road-holding ability – Vocabulary", Geneva, CH, Tech. Rep., 2011.

[12] A. Nguyen and B. Le, "3D point cloud segmentation: A survey", in *2013 6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013.

[13] M. Schreier, "Bayesian Environment Representation, Prediction, and Criticality Assessment for Driver Assistance Systems", Ph.D. thesis, Technische Universität Darmstadt, 2016.

[14] K. Granstrom, M. Baum, and S. Reuter, "Extended Object Tracking: Introduction, Overview and Applications", 2016. eprint: arXiv:1604.00970.

[15] S. Deshpande, W. Muron, and Y. Cai, "Vehicle Classification", in *Computer Vision and Imaging in Intelligent Transportation Systems*. John Wiley & Sons, Ltd, 2017, ch. 3, pp. 47–79.

[16] Ibeo Automotive Systems GmbH, *Operating Manual ibeo LUX 2010 Laserscanner*.

[17] Z. Wang, Y. Liu, Q. Liao, H. Ye, M. Liu, and L. Wang, "Characterization of a RS-LiDAR for 3D Perception", 2017. eprint: arXiv:1709.07641.

[18] P. Rosenberger, M. Holder, M. Zirulnik, and H. Winner, "Analysis of real world sensor behavior for rising fidelity of physically based lidar sensor models", in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018.

[19] P. Cao, "Modeling active perception sensors for real-time virtual validation of automated driving systems", Ph.D. thesis, Technische Universität Darmstadt, 2018.

[20] T. A. Wheeler, M. Holder, H. Winner, and M. J. Kochenderfer, "Deep stochastic radar models", in *2017 IEEE Intelligent Vehicles Symposium (IV)*, 2017.

[21] S. Bernsteiner, Z. Magosi, D. Lindvai-Soos, and A. Eichberger, "Phänomenologisches Radarsensormodell zur Simulation längsdynamisch regelnder Fahrerassistenzsysteme", *Elektronik im Fahrzeug*, VDI-Berichte, vol. 2188, pp. 639–650, 2013.

[22] A. Watt and M. Watt, *Advanced Animation and Rendering Techniques*. New York, NY, USA: ACM, 1991.

[23] S. Huch, "Entwicklung einer umfassenden Metrik für die Bewertung einer Lidar-Sensor-Simulation durch Betrachtung mehrerer aufeinander folgender Verarbeitungsebenen", M.Sc. thesis, Technische Universität Darmstadt, 2018.

[24] B. Browning, J.-E. Deschaud, D. Prasser, and P. Rander, "3D Mapping for high-fidelity unmanned ground vehicle lidar simulation", *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1349–1376, Oct. 2012.

[25] R. Grewe, M. Komar, A. Hohm, S. Lueke, and H. Winner, "Evaluation method and results for the accuracy of an automotive occupancy grid", in *2012 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2012.

[26] T. Vu and R. Evans, "Optimal Subpattern Assignment Metric for Multiple Tracks (OSPAMT Metric)", 2018. eprint: arXiv:1808.02242.

[27] P. Jaccard, "The Distribution of the Flora in the Alpine Zone", *New Phytologist*, vol. 11, no. 2, pp. 37–50, Feb. 1912.

[28] P. Babahajiani, L. Fan, J.-K. Kämäräinen, and M. Gabbouj, "Urban 3D segmentation and modelling from street view images and LiDAR point clouds", *Machine Vision and Applications*, vol. 28, no. 7, pp. 679–694, May 2017.