Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516

Physically-Based Simulation of Automotive LiDAR

L. Dudzik[†], M. Roschani[*], A. Sielemann[*], K. Trampert[†], J. Ziehn[*], J. Beyerer[*‡] and C. Neumann[†]
[†]Light Technology Institute (LTI), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
Email: {leonhard.dudzik, klaus.trampert}@kit.edu
[‡]Vision and Fusion Laboratory (IES), Karlsruhe Institute of Technology (KIT), 76131 Karlsruhe, Germany
[*]Fraunhofer IOSB, Email: {anne.sielemann, masoud.roschani, jens.ziehn}@iosb.fraunhofer.de

***Abstract*—We present an analytic model for simulating automotive time-of-flight (ToF) LiDAR that includes blooming, echo pulse width, and ambient light, along with steps to determine model parameters systematically through optical laboratory measurements. The model uses physically based rendering (PBR) in the near-infrared domain. It assumes single-bounce reflections and retroreflections over rasterized rendered images from shading or ray tracing, including light emitted from the sensor as well as stray light from other, non-correlated sources such as sunlight. Beams from the sensor and sensitivity of the receiving diodes are modeled with flexible beam steering patterns and with non-vanishing diameter. Different (all non-real time) computational approaches can be chosen based on system properties, computing capabilities, and desired output properties.**

Model parameters include system-specific properties, namely the physical spread of the LiDAR beam, combined with the sensitivity of the receiving diode; the intensity of the emitted light; the conversion between the intensity of reflected light and the echo pulse width; and scenario parameters such as environment lighting, positioning, and surface properties of the target(s) in the relevant infrared domain. System-specific properties of the model are determined from laboratory measurements of the photometric luminance on different target surfaces aligned with a goniometer at 0.01° resolution, which marks the best available resolution for measuring the beam pattern. The approach is calibrated for and tested on two automotive LiDAR systems, the Valeo Scala Gen. 2 and the Blickfeld Cube 1. Both systems differ notably in their properties and available interfaces, but the relevant model parameters could be extracted successfully.

***Index Terms*—LiDAR, laser, perception, simulation, AI**

I. Motivation and State of the Art

THE provision of realistic data for training and testing of safety-critical systems has received additional attention lately. Progress in highly automated or unmanned transportation systems on and off roads and in the air has enabled use cases that demand high safety levels in complex operational design domains [1]. This progress is fueled, to a large extent, by advanced data-driven models based on deep learning. For these, data currently serve both as the primary means of defining functions and as a crucial means of evaluating and validating these functions. Accordingly, new legislations and standards [2], in particular the European AI Act [3], set high standards for the quality of such data. As real-world data acquisition and annotation is a laborious, time-consuming, costly, and error-prone task, synthetic data (mainly from simulations, generative AI, or a combination thereof) are pursued as a plausible alternative. However, to ensure that such data can adequately replace real data with the confidence required for the testing of safety-critical systems, it must be demonstrated that the domain gap between these data and the real world is sufficiently small and sufficiently well understood. Progress has been made in particular for camera-based imaging [4]–[12], which has considerable similarities to LiDAR optics, besides important differences.

The synthetic generation of LiDAR data has also received attention, albeit to a still lesser degree. One of the earliest detailed descriptions using rasterization rendering as a prior for LiDAR simulation is given in [13], where real-time shader artifacts from OpenGL were utilized to generate dense point clouds for airborne sensor data. Applications for using synthetic, shader-based LiDAR data for unmanned aerial vehicle applications are given in [14]. Examples of detailed and application-specific models include [15] for a detailed per-ray model for vegetation monitoring, [16] for ground-based volumetric vegetation perception, or [17] for water appearance. An approach to augment real point clouds of static road environments with synthetic traffic objects is given in [18]. [19] provides a data-driven model based on raycasting and the subsequent application of a U-Net ML model to bridge the domain gap by introducing realistic raydrops (i.e., rays not yielding echoes) based on range, incident angle, intensities, and additional features. This work is expanded in [20] with a modified model for raydrop estimation and surface approximation, and in [21] where an exhaustive description of various effects is provided, including (beyond raydrops) multiple echoes, spurious returns from indirect paths and blooming, noise, and time delays between points, providing a differentiated analysis of the impacts of each aspect on the domain gap. Even more data-driven approaches lie in the generative AI-based creation of point clouds, where [22] gives an overview of approaches and presents a novel diffusion-based model to produce temporally consistent point clouds for traffic scenes.

For this paper, we focus on two aspects that received lesser attention in previous work: The metrological integration of optical phenomena, especially blooming (Sec. I-B), into an efficient rendering pipeline through a transparent, parametric model (Sec. II) and the corresponding derivation of relevant parameters through a dedicated optical characterization of the sensor (Sec. III). This approach is complementary to prior

This publication was written within the framework of the KAMO: Karlsruhe Mobility High Performance Center (www.kamo.one), in the context of the AVEAS research project (www.aveas.org), funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) within the program “New Vehicle and System Technologies”, and supported by the Fraunhofer Internal Programs under Grant No. 40-00736 within the “ALBACOPTER®” project (www.albacopter.fraunhofer.de).

Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516

work in that it integrates optical measurement parameters to more accurately simulate LiDAR data. Depending on the amount of information available about the environment and target objects, and depending on measurement capabilities, approaches may be preferable that can operate exclusively on recorded scenario data. The presented model, in turn, provides analytical solutions for such phenomena for cases where dedicated laboratory measurements are feasible and a minimal domain gap is desired.

A. Physically-based rendering and BRDFs

The model adopts the physically-based rendering (PBR) framework [23], a set of widely used and supported principles for optical image generation, particularly for modeling visual surface properties. The underlying surface model is the bidirectional reflectance distribution function (BRDF) over spherical angles $\Phi = (\phi, \theta)$, describing the relationship between incident and reflected light at a surface interaction, via

$$\beta(\Phi_\mathrm{i}, \Phi_\mathrm{o}) = \mathrm{d}L_\mathrm{o}(\Phi_\mathrm{o})/\mathrm{d}E_i(\Phi_\mathrm{i}), \tag{1}$$

where Φ_i is the incident light direction, Φ_o is the reflected or outgoing light direction, E_i is the incident light *irradience*, and L_o is the outgoing *radiance*. The general framework enables two commonly used sets of units: photometric and radiometric. Photometric units are based on *luminous flux* (as measured in Lumen, with Candela and Lux as derivable units), which in turn is based on the perception of the human eye. Radiometric units use raw physical quantities around *power* (as measured in Watt). Both are interchangeable for the PBR framework, by which light emission intensities and surface interactions are combined to derive received / perceived intensities. For computer graphics, photometric units are generally preferred, as they relate to visual appearances. For LiDAR simulation, however, the response in the laser domain (usually infrared) and power properties of the scanner must be modeled. Hence, radiometric units apply for this use case, with Watt being by far the most common unit.

However, absolute physical units must be considered with caution for inferring simulation parameters, as these are strongly determined by sensor-intrinsic properties—both for echo intensity, which is not recorded directly but through the sensor's AD detection threshold, and the ratio between echo and ambient light intensity at which an echo detection is possible, which depends on the sensor's signal processing. Hence, these properties are determined in a data-driven way as well, by which intensities reduce to relative values.

A particular parametrization of such a BRDF was presented in [24] under the name of "Disney principled BSDF" (where the "S" generalizes "reflectance" to "scattering"), describing the BRDF at each surface point through a set of intuitive parameters, each with plausible values within a common range of $[0, 1]$, such as diffuse, roughness, metalness, and surface normals. This concept sparked a family of related models that have been used not only in renderers (raytracers) such as Mental Ray, Mitsuba, or Cycles, but also in shader-based / rasterization-based game engines, such as Unreal, Unity, or OGRE3D. While generalizations exist for volumetric information of materials, e.g., refraction and subsurface scattering, the most common use cases are pure surface interactions.

B. Blooming phenomena

The point cloud reported by a LiDAR sensor cannot perfectly match the scanned 3D scene. Apart from inevitable noise, point clouds exhibit artifacts, which stem from the measurement principle. One of these artifacts is blooming, which refers to the apparent widening of the contour of objects with a retro-reflective surface. Retro-reflection describes the property of optical surfaces that reflect incoming light back towards its origin, irrespective of the direction of incidence. The ideal BRDF of a retro-reflector approximates a Dirac distribution for $\Phi_\mathrm{o} = \Phi_\mathrm{i}$. The BRDF of real retro-reflectors used in road traffic has a peak reflectance $\beta(\Phi_\mathrm{i}, \Phi_\mathrm{o})$ in the range of 100 to 1 000, and a rapid drop-off for increased angles spanned by Φ_i and Φ_o. Since the working principle of a LiDAR sensor is based on sampling back-reflected radiation, retro-reflecting surfaces always exhibit their peak reflectance from the perspective of the sensor. Compared to diffuse surfaces, whose BRDF cannot exceed π^{-1}, the retro-reflector is approximately three orders of magnitude brighter.

The LiDAR emits a divergent laser beam whose angle-resolved radiant intensity roughly resembles a peak that quickly declines. The outline of any surface can bloom to a limited degree if a subsection of the beam intersecting the border causes enough reflection for the sensor to detect it. The sensor effectively integrates over all incoming radiation, attributing the ToF measurement to the center direction of the beam, which lies outside the target. Because of the high reflectance of retro-reflectors, even minute radiant intensities in the periphery of the emitted pulse cause sufficient reflection to detect the surface, possibly dominating incoming reflections from diffuse surfaces hit by different regions of the pulse. Since blooming can cause the contour to widen by several degrees, and due to the prevalence of retro-reflectors on roads and vehicles, blooming is a relevant aspect of the domain gap for automotive LiDAR use-cases. Most systems developed for assisted or automated driving rely on the absolute distance reference a LiDAR sensor provides, for orientation and classification of the surroundings. Thus, it is essential to test the robustness of these systems towards blooming artifacts.

II. Simulation Model

The following simulation model is implemented in the OCTAS®[1] simulation framework, which provides a modular architecture that enables, in particular, the exchange of different rendering engines—for the given use case, the plugins for the rasterization-based engine OGRE3D[2] and for the raytracing engine Cycles[3] are used, which generate consistent outputs based on the PBR framework.

A. LiDAR model

We establish a *single-bounce model* (cf. Fig. 1), s.t. only a single geometric interaction between a light source and a surface is modeled before the light is received. Interactions

[1] octas.org [2] ogre3d.org [3] cycles-renderer.org

Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516

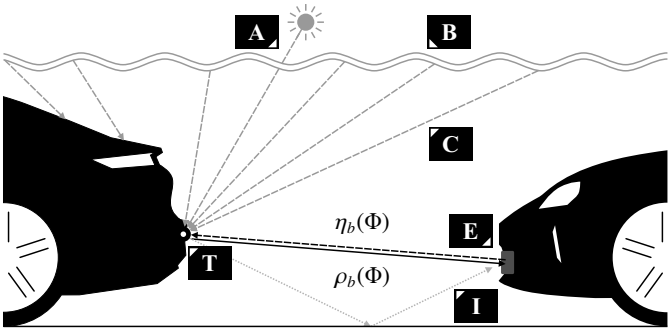


Fig. 1: Included and excluded aspects of the single-bounce model: LiDAR rays originate at the emitter/collector **E** with intensity $\eta_b(\Phi)$, continue to a single target spot **T**, and return at the same angle with intensity $\rho_b(\Phi)$. Stray light travels from a source (e.g., the sun **A**) to **T**, then to **E**. Virtual sources **C**, (e.g., atmospheric scattering **B**), can also be considered. However, indirect paths from/to **E**, such as multiple bounces over the road **I**, are *not* modeled.

that can be included in either the local surface (such as retroreflection or clearcoat) or the ambient light source (such as atmospheric lighting) may also be considered within the model. Generalizations towards multiple bounces, in particular reflections or refractions of LiDAR beams, are omitted here, as they considerably increase computational effort while effects will usually not be dominant (cf. Fig. 7). In the following, we use the spherical angle $\Phi \in \mathcal{A}$ and ranges $r \in \mathcal{R}$.

We model a LiDAR as having a set $\mathcal{B} = \{b_1, b_2, ...\}$ of beams, which we use as indices for particular properties. These properties include the central directional angle per beam, Ψ_b in (sensor-fixed) spherical coordinates; a distribution of emitted intensities $\eta_b : \mathcal{A} \to [0, \infty)$; a resulting reflected intensity of light from the scene at different ranges $\rho_b : \mathcal{A} \times \mathcal{R} \to [0, \infty)$; and an angular sensitivity of the collecting diode $\kappa_b : \mathcal{A} \to [0, 1]$; leading to a signal $\xi_b : \mathcal{R} \to [0, \infty)$ of received intensities over different ranges. These are related via

$$\xi_b(r) = \iint \mathrm{d}\omega_\Phi \; \kappa_b(\Phi)\, \rho_b(\Phi, r) \quad \text{with} \quad \mathrm{d}\omega_\Phi = \sin\theta \, \mathrm{d}\theta \, \mathrm{d}\phi \; . \tag{2}$$

We note that this equation does not explicitly include all of the aforementioned quantities: Ψ_b is implicitly contained in η_b but not required separately, and η_b certainly affects ρ_b through the individual BRDFs β of the surfaces in the scene, along with their geometries. The latter relationship (η, ρ, β) is complex, but—fortunately—solved in a very mature way through PBR frameworks at various levels of detail. Unlike many applications for graphical rendering, which only require $\rho_b(\Phi)$ (a 2D image), we require volumetric information $\rho_b(\Phi, r)$. However, due to the single-bounce approach, we can assume to be in possession of $r(\Phi)$, a unique range at each spherical angle, at least for any target contributing $\rho_b(\Phi, r) > 0$. Often this information is available, though not commonly demanded in the end product of rendering: Raytracing will typically compute depths directly, whereas rasterization-based rendering commonly requires the computation of a z *buffer* to handle occlusions (cf. also Sec. II-D), which stores pixel z coordinates (perpendicular to the image plane), leading to

$$\xi_b(r) = \iint \mathrm{d}\omega_\Phi \; \kappa_b(\Phi)\, \rho_b(\Phi)\, \delta(r - r(\Phi)), \tag{3}$$

where δ is the Dirac delta, meaning that for $\eta_b(r)$ only intensities are accumulated which are at range r. For brevity, we omit that ambient light (e.g., sunlight) is computed identically except for substituting the emitted light by any ambient light sources, acting as the intensity of the noise floor against which echoes must be detected. This statement provides a first complete model to establish received signal intensities ξ_b for a given beam b based on a rendering of scene reflections $\rho_b(\Phi)$ and corresponding depths $r(\Phi)$, and the collector sensitivity κ_b. Yet, computational effort is high when more than a single beam is considered: Each beam requires the rendering of one scene, which is typically prohibitive for modern LiDAR systems with large, dense point clouds. Hence, simplifications are required that will be discussed in the following.

B. Simplifications

Since the scanner's intrinsic geometry is usually very small compared to the scale of the scene, all beams can be considered as originating from a single point coinciding with the collector. Because of this, overlapping beams share the same optical path for a surface point of the scene geometry. In (1), the considered Φ_i and Φ_o, as well as the resulting β will reappear for multiple beams. The same is true for the depth $r(\Phi)$. Since typically the initialization of a new rendering $\rho_b(\Phi)$ is a particularly costly operation, this motivates the combination of multiple beams in a single rendering call.

We define $\hat{\rho}(\Phi)$ as the reflected intensities for a uniform, isotropic emission source (a point light in the scene), s.t.

$$\xi_b(r) = \iint \mathrm{d}\omega_\Phi \; \kappa_b(\Phi) \underbrace{\eta_b(\Phi)\hat{\rho}(\Phi)}_{\rho_b(\Phi)} \delta(r - r(\Phi)) \; , \tag{4}$$

utilizing that in the single-bounce model any rays are directly between sensor and surface along Φ, and no secondary reflections must be considered. Further assuming that κ_b and η_b only depend on b through an angular offset by Ψ, we can define "centered" kernels κ, η to simplify to

$$\xi_b(r) = \iint \mathrm{d}\omega_\Phi \; (\kappa \cdot \eta)(\Phi - \Psi_b)\, \hat{\rho}(\Phi)\, \delta(r - r(\Phi)) \; . \tag{5}$$

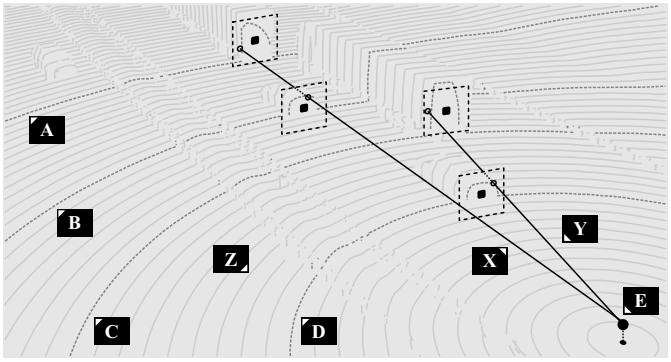
This resembles the convolution or cross-correlation $(\kappa \cdot \eta) * \hat{\rho}$, with a significant difference: The δ effects that individual depths must be considered independently, s.t. a single global convolution (as for camera-based blooming) is not applicable.

C. Algorithmic solutions

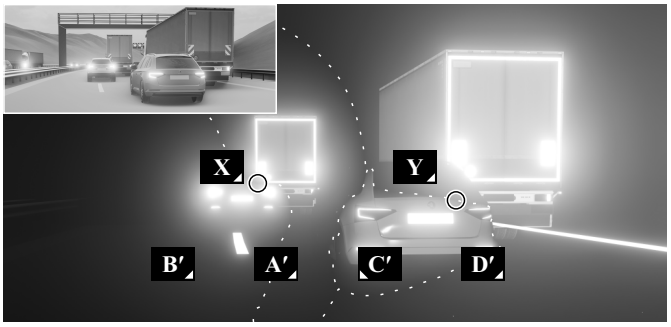
Based on these simplifications, two distinct algorithms are outlined, presented in Alg. 1 (beam iteration) and Alg. 2 (range stacking) respectively. The presented algorithms focus on exposing the key principles in the most concise and clear form, leading to differences in processing aspects and interfaces. It should be noted that both algorithms essentially work on the same types of input and can be used (with additional pre- and post-processing steps) interchangeably. When using identical parameters and outputs, results of either algorithm should correspond to within floating point arithmetic limits.

A common post-processing step (not elaborated in detail here) will entail the computation of echo pulse widths (EPW), which serves as a surrogate for echo intensity for receivers that contain only binary thresholds T at their AD conversion.

Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516



(a) Bird's eye view of the scene with *iso ranges* (not scanning layers)



(b) Exemplary rendering of visual image and LiDAR echo at dominant ranges

Fig. 2: Principle of range-separated blooming for first dominant echo. Lines **Z** in the bird's eye view indicate iso ranges. Signals along lines of sight (**X**, **Y**) do not mix additively across different ranges. Instead, the strongest echo is propagated, forming regions (**A′**–**D′**) in which retroreflections of one of the four front vehicles, and hence their ranges **A**–**D**, dominate. **X** and **Y** are located at spatial angles at which the echoes of the respective car and truck have matching intensity. Optical effects are exaggerated for visibility, and only the range dominance among **A**–**D** is compared.

Emitted pulses have a non-vanishing length in time (and thus range), which we denote $\varpi(r)$. Hence, such scanners will receive $\hat{\xi} = \varpi * \xi$ and record intervals $[r_0, r_1]$ s.t. $\forall r \in [r_0, r_1] : \hat{\xi} > T \wedge \nexists \varepsilon_0, \varepsilon_1 > 0 : \forall r \in [r_0 - \varepsilon_0, r_1 + \varepsilon_1] : \hat{\xi} > T$, i.e., the intervals during which $\hat{\xi}$ exceeds T. For perpendicular targets, $\xi(r) \approx a \cdot \delta(r - r_{\text{target}})$, where a is the reflected amplitude and δ is the Dirac delta, and hence $\hat{\xi}(r) \approx a \cdot \varpi(r - r_{\text{target}})$. Now when measuring the range interval over which $a \cdot \varpi(r - r_{\text{target}}) > T$, its width will usually (e.g., for Gaussian-shaped ϖ) scale with a. Hence, the EPW $r_1 - r_0$ correlates with a, most strongly so for perpendicular targets. For flat incidence angles, EPW will be dominated by the actual geometric range interval over which the object reflects the beam.

Furthermore, for simplicity, the descriptions do not address that raw rendered images will typically follow a pinhole camera projection rather than spherical coordinates, in which kernels $\kappa\eta$ would be inhomogeneous for homogeneous beams in spherical coordinates. Different solutions to this exist, from adapting the kernel (only for Alg. 1), over the reprojection of rendered images, to the subdivision of wider fields of view into smaller, narrower renderings.

1) Beam iteration: This is the more direct and comprehensive solution, described in Alg. 1. Beams are processed individually based on precomputed images, and for each beam,

Input: $\hat{\rho}[U][V]$: rendered intensities
$\varrho[U][V]$: rendered ranges
$\nu[U][V]$: rendered normals
$\kappa\eta[2M+1][2N+1]$: centered kernel
ρ_{min}: threshold intensity
Δr: range resolution
ϱ_{nearest}: maximum range
$\mathcal{B}$: Set of beams
Result: $\xi[|\mathcal{B}|][R]$: received intensities over ranges

$\xi \leftarrow (0)_{|\mathcal{B}| \times R}$;
for $b \in \mathcal{B}$ **do**
  $\Psi \leftarrow \text{beamAngle}(b)$;
  $\hat{\Psi} \leftarrow \text{roundToPixels}(\Psi)$;
  for $(m, n) \in \{-M, ..., M-1, M\} \times \{-N, ..., N-1, N\}$ **do**
    $[u_{\text{offs}}, v_{\text{offs}}] \leftarrow [\hat{\Psi}_{\text{u}} + m, \hat{\Psi}_{\text{v}} + n]$;
    $[r_0, r_1] \leftarrow \text{pixelRangeInterval}(u_{\text{offs}}, v_{\text{offs}}, \varrho, \nu)$;
    for $r_{\text{offs}} \in \{r_0, r_0 + \Delta r, ..., r_1\}$ **do**
      $\xi[b][r_{\text{offs}}] \leftarrow \hat{\rho}[u_{\text{offs}}][v_{\text{offs}}] \cdot \kappa\eta[m][n]$;

Alg. 1: Outline of the beam iteration algorithm as described in Sec. II-C1. See there also for a description of pixelRangeInterval(). Time complexity is $O(M \cdot N \cdot |\mathcal{B}|)$ for single echoes, and $\ldots \cdot (\Delta r)^{-1}$ for multiple echoes.

Input: $\hat{\rho}[U][V]$: rendered intensities
$\varrho[U][V]$: rendered ranges
$\kappa\eta[2M+1][2N+1]$: centered kernel
ρ_{min}: threshold intensity
Δr: range resolution
r_{max}: maximum range
Result: $S_{\text{nearest}}[U][V]$: dominant range indices
$\xi_{\text{nearest}}[U][V]$: dominant intensities

1 $s_{\text{max}} = \lfloor r_{\text{max}} / \Delta r \rfloor$;
2 $\bar{\varrho} \leftarrow \lfloor \varrho / \Delta r \rfloor$;
3 $\xi_{\text{nearest}} \leftarrow (0)_{U \times V}$;
4 $S_{\text{nearest}} \leftarrow (s_{\text{max}})_{U \times V}$;
5 **for** $s \leftarrow s_{\text{max}}$ **to** 1 **do**
6 $\hat{\rho}_{\text{slice}} \leftarrow (0)_{U \times V}$;
7 **for** $(u, v) \in \{1, ..., U\} \times \{1, ..., V\}$ **do**
8 **if** $s = \bar{\varrho}[u][v]$ **then** $\hat{\rho}_{\text{slice}}[u][v] \leftarrow \hat{\rho}[u][v]$;
9 $\xi_{\text{slice}} \leftarrow \hat{\rho}_{\text{slice}} * \kappa\eta$;
10 **for** $(u, v) \in \{1, ..., U\} \times \{1, ..., V\}$ **do**
11 **if** $\xi_{\text{slice}}[u][v] \geq \rho_{\text{min}}$ **then**
12 $\xi_{\text{nearest}}[u][v] \leftarrow \xi_{\text{slice}}[u][v]$;
13 $S_{\text{nearest}}[u][v] \leftarrow s$;

Alg. 2: Outline of the range stacking algorithm for single closest echo above threshold. Note that the convolution in ln. 9 can be optimized for separable kernels, to yield a time complexity of $O((\Delta r)^{-1} \cdot U \cdot V)$, and $\ldots + |\mathcal{B}|$ for sampling actual beam ranges in post-processing.

an array of echo signals over range (or time, respectively) is maintained. This signal array is composed of the contributing intensities over the kernel $\kappa\eta$, where each rendered pixel contributes uniformly to a range interval according to a function pixelRangeInterval(), which yields an estimate for the range interval contained within the pixel by extrapolating from pixel range and surface normal. To obtain a single echo per beam (comparable to Alg. 2), the per-beam intensities can be searched for strongest / nearest / longest echoes. For the former two, only maxima must be stored, and hence only space of $O(|B|)$ must be allocated rather than $O(|B| \cdot R)$, and the post-processing step is avoided.

2) Range stacking: For high numbers of very dense beams, relatively narrow fields of view, and moderate range resolution, it can be beneficial to replace the iteration over the kernel cells

in $\kappa\eta$ by actual convolutions—however, as initially remarked, such convolutions must handle ranges independently.

In this range stacking approach, the rendered images are split into discrete slices according to the depth resolution of the sensor (or an approximation thereof), where typically (as in Alg. 2 only one such slice is kept in memory at a time.

Intensities within the same slice are then convolved, and can iteratively be combined with other slices to propagate, e.g., the strongest echo from far to near ranges. While this will typically involve computing a much denser pattern of beams (namely at pixel level as in Fig. 2) and prohibit the storage of individual echo arrays over range, the benefit lies in replacing the manually implemented pseudo-convolution by a set of true convolutions. These can leverage various optimizations, such as the parallel computation on dedicated hardware (namely GPUs), reducing complexity for (near-) separable kernels (where $\kappa\eta \approx H * V$ with $H^\top \in \mathbb{R}^M, V \in \mathbb{R}^N$, s.t. $\hat{\rho} * \kappa\eta \approx ((\hat{\rho} * H) * V)$, cf. also Sec. III-B) from $O(N \cdot M)$ to $O(N + M)$, or even box kernels via differences in integral images. These optimizations are particularly advantageous when the overlap between beam kernels is significant s.t. avoiding their repeated computation provides a notable gain.

D. Practical rendering considerations

The model can be used with raytracing as with rasterization-based rendering within the PBR framework, where the latter is favorable for many use cases due to its faster rendering speed. The choice of a rendering engine may introduce limitations that require addressing. The following sections give a brief overview of the main relevant effects and solution strategies.

1) z buffer resolution: When used as the source for depth information, the depth resolution of the z buffer must be considered. As it is designed for visual rendering purposes, the resolution is not distributed homogeneously over the z depth (which, in turn, would not correspond to homogeneous range resolution, anyway), but reciprocally between the near and far clipping planes, s.t. close depths are resolved more accurately than distant ones. Accordingly, the resolution of the depth buffer directly affects the range resolution particularly at the far ranges, and particularly through the choice of the *near* clipping plane, since most resolution units are allocated there. Common z buffer depths for realtime rendering (as in OpenGL, for example) are 16, 24, or 32 bits. Fig. 4 shows errors arising based on the variation of the near and far plane.

2) Rasterization: The most commonly found simplification is rasterization: Images are sampled in a regular pixel grid, which typically does not align with the scan pattern of a LiDAR system. This is a pivotal aspect in realtime GPU rendering on graphics cards, hence referred to as rasterization-based rendering. The acceleration structure of parallel computing can be leveraged optimally when perspective projections are the principal geometric transformations. Any nonlinear mappings considerably increase the computational effort and do not generally benefit from optimized existing solutions. Raytracing-based rendering in turn does not generally benefit as significantly from regular, grid-based sampling; however, it is the most commonly used interface for graphics rendering.

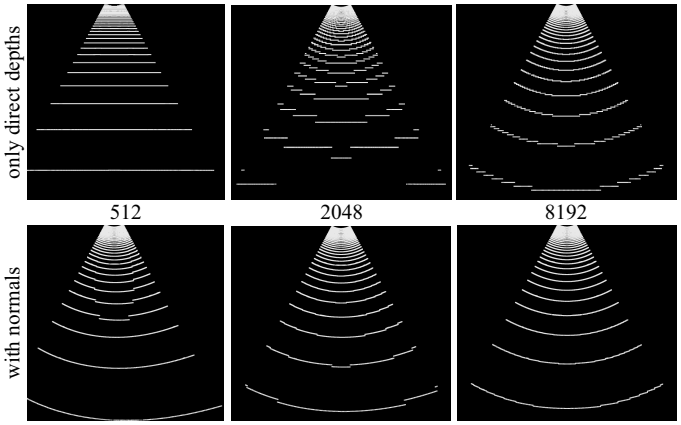


Fig. 3: Comparison of increased resolution vs. the correction (through 10-bit normals) for reducing angular errors. It can be seen that even at high resolutions of 8192 pixels wide, step artifacts remain distinct at far ranges. In contrast, when correcting via normals, residual steps exclusively arise from lack of precision in normals, if applicable. Increased resolution in this case also serves to eliminate extrapolation errors from erroneous normals.

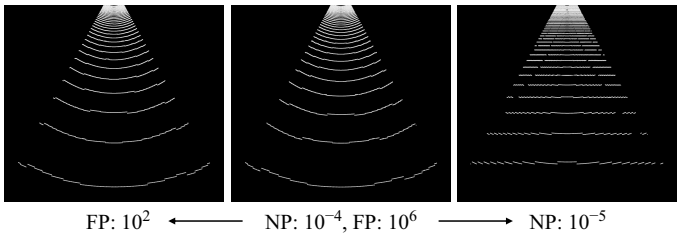


Fig. 4: Effect of clipping planes and thus z buffer resolution with normals correction. A near plane of 10^{-4} and a far plane of 10^6 provide fair results. Pulling the far plane to 10^2 provides little improvement. Pushing the near plane up to 10^{-5}, in turn, yields degenerate, discrete depth levels.

Hence, some renderers such as Mitsuba directly support arbitrary ray directions, whereas others such as Cycles use raster images as their primary interface.

Errors introduced by rasterization are fundamentally angular errors, as information is sampled at discrete Φ which do not generally coincide with the ray directions of the scan pattern. In practice, these angular errors manifest as range errors, since ray directions are assumed to be accurate while ranges are read from the closest pixel coordinates, as seen in Fig. 3.

Interpolation of ranges between pixels is typically out of the question in the given application: Changes in range across the sensor's field of view commonly stem from distinct objects at different depths, where no continuous transition can be assumed (as opposed to, for example, terrain models). Interpolation will thus generate intermediate "ghost points" between targets, whereas LiDAR systems will typically only yield ranges at which actual targets exist in the scene. The scale of the resulting errors is limited only by the maximum distance of the objects in the scene.

To correct angular errors without the need for interpolation between different raster pixels, a linear extrapolation of a given range based on surface normals can be performed according to $r_{\text{corr}} = \boldsymbol{p}^\top \boldsymbol{n} / (\boldsymbol{v}^\top \boldsymbol{n}) \cdot r$ where $\boldsymbol{p}$ is the direction vector towards the pixel center, $\boldsymbol{v}$ is the true direction of the LiDAR ray, $\boldsymbol{n}$ is the surface normal, and r is the range at the pixel center.

It must be noted that normal precision may be limited

in the rendering process, as these are, in realtime graphics applications, commonly used for effects where accuracy is less important (cf. Fig. 4 for the impact of z buffer resolution). Figs. 3 and 4 show 10 bit *normals*, whereas *z buffer* depths of 16 or 32 bits are common. Hence, for well-chosen clipping planes, errors from normals are the more likely error source.

III. Optical Characterization

Both the beam intensity $\eta(\Phi)$ and the sensitivity $\kappa(\Phi)$ are measured with directional dependence relative to the main beam direction. If both η and κ are assumed homogeneous over the sensor's scan pattern, only a single beam direction needs to be characterized and generalized for all measurement directions. Any deviations can be subsequently parameterized as stretched or rotated versions of the reference pulse.

A. Angle-resolved beam intensity

The measurement method used to characterize $\eta(\Phi)$ is borrowed from far field photogoniometry, which deals with the measurement of the angularly resolved luminous intensity of light sources. The measurement setup consists of a detector and a goniometer, which is used to precisely rotate the light source around two rotation axes towards the detector.

The setup in question employs an imaging detector system consisting of a diffuse screen, which is illuminated by the source, and a spectrally and geometrically calibrated camera pointed at the screen. This setup allows measuring the luminous intensity for many directions using a single image with the angular resolution only limited by the camera's resolution. For LiDAR sensors, this setup can be used to measure relative radiant intensities in the near-infrared, where the sensor of the camera is still sensitive. For beam characterization, the camera is used in combination with a long-pass filter, which is transparent in the infrared and blocks visible light.

Measurement of beam intensity poses requirements for the scan pattern. The spacing of the measurement directions needs to be sufficiently large so that they do not overlap. A more general requirement is a constant scan pattern, which allows the camera to integrate over multiple realizations of the same beam direction during the imaging process. The repeatability of a LiDAR sensor is normally accurate enough, so this integration can be reasonably performed. The Blickfeld Cube1 LiDAR exposes many settings to configure the scan pattern and is therefore used for the following exemplary measurements.

Fig. 5a shows a goniometer measurement for a section of the scan pattern of the Blickfeld Cube1. The obtained beam intensity can be used for simulations, but its intensity resolution is not sufficient to simulate blooming artifacts. Since blooming is caused by the very dim outer edges of the pulse shape, a measurement with a high dynamic range is required to resolve the beam intensities across at least six orders of magnitude. To increase the sensitivity, the measurement setup is adapted to use a retro-reflective screen and a camera positioned close to the LiDAR sensor. The screen is constructed from micro prism-based retro-reflective contour marking tape for trucks and trailers. Since the angle spanned by the LiDAR source and the camera is small enough, the received signal is amplified drastically by the reflectance of the retro-reflector. The measurement result for an isolated beam is shown in Fig. 5b. The effective extent of the pulse measures approx. 3°—six times larger than the core pulse shape, providing the optical reason for blooming.

B. Angle-resolved collector sensitivity

The sensitivity of the receiving optics generally decreases for radiation coming from directions that deviate from the beam direction Ψ. While not strictly necessary for measuring the collector's sensitivity, it is beneficial if the LiDAR sensor reports some measure of intensity for the incoming radiation. The output of the Cube 1 includes a measure for ambient (passive) intensity for every measurement direction, which scales linearly with incoming radiation. The input sensitivity was measured by aiming a single beam of the LiDAR sensor at a distant halogen light source using the goniometer. This light source, in turn, creates a constant irradiance on the sensor. The change in the reported ambient intensity is then tracked while the sensor is turned away from the main beam direction in 0.01° steps. This is done for both the horizontal and vertical direction, resulting in two input sensitivity slices, which are treated as two 1D components of a separable 2D filter kernel. The combined input sensitivity is displayed in Fig. 5c.

C. Combined optical domain representation

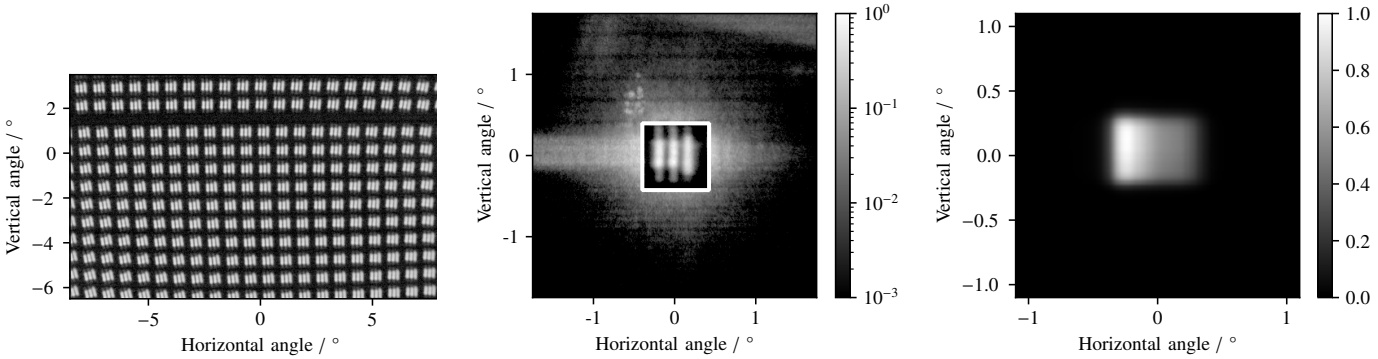
The multiplication of the emitter radiant intensity η and the collector sensitivity κ is employed in the LiDAR model as an effective beam intensity kernel. As the measurements for the emitter radiant intensity η and collector sensitivity κ are in relative quantities, the simulated signal ξ would not allow conclusions about whether it would trigger the sensor. To set these relative values into relation, the detection threshold was determined using the same measurement setup and a dark diffuse target. From this measurement, an equivalent detection threshold signal is deduced.

IV. Real-World Application

As the model is calibrated against the laboratory measurements through defined targets and goniometer, the strong match between the resulting real and synthetic point clouds provides limited insight into the validity of the model. To demonstrate the pipeline in whole and independently, the simulation is applied to real-world data recorded by a Scala Gen. 2 mounted to a research vehicle. In this use case, the limitation is that world properties (geometries and BRDFs) are only approximated in the simulation, enabling mainly qualitative rather than quantitative comparisons. Yet, the presence or absence of effects provides insight into model limitations.

An exemplary frame is shown and discussed in Fig. 6. It can be seen that blooming effects from the retroreflective sign **A** show good correspondence, while the kernel shapes differ in detail. Secondary blooming effects, namely the deformation of ranges near retroreflective road markings **D** also arise similarly in the synthetic and real data. Small road surface irregularities are not modeled in the simulation, yet they

Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516



(a) Relative radiant intensity of the scan pattern. (b) Relative beam intensity η (c) Relative sensitivity κ of the collector diode.

Fig. 5: Results of the optical characterization of the Blickfeld Cube 1. Each group of three spots corresponds to one beam direction, as the sensor employs three laser diodes to construct an approximately square pulse shape. Highly-resolved beam intensity in (b) is measured with retro-reflective screen. The inner section of the pulse shape marked with a white rectangle is scaled down by a factor of 1 000 to visualize the entire pulse within the same range. While offering the highest grade of retro reflectance, the prism optics introduce a higher dependency on the incidence direction and a speckled appearance. Dark stripes result from the horizontal application of the tape. A more uniform retro-reflective surface based on glass beads could be more suitable. Relative sensitivity in (c) over input angles is composed of separate horizontal and vertical measurements via $H * V$.

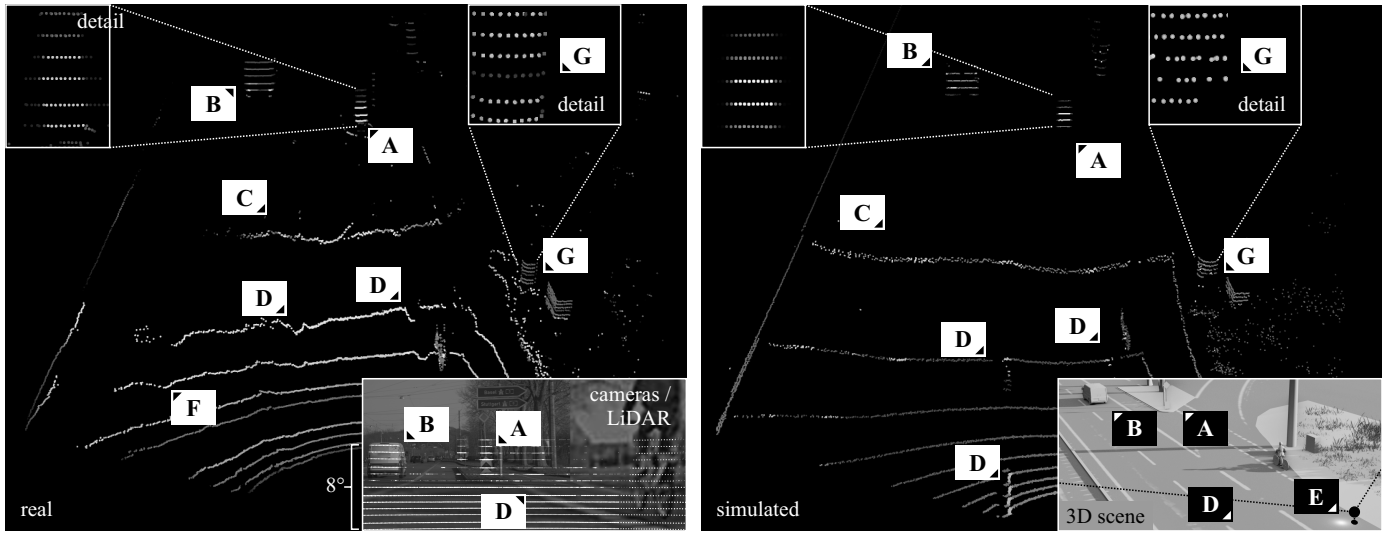


Fig. 6: Comparison of consistently modeled objects between the real point cloud from Valeo Scala Gen. 2 and simulated point cloud (emitter located at **E**), points are colored by echo pulse width. **A** shows the blooming of the retroreflective sign, **B** is a leading van (whose respective intensities are underestimated in the simulation). The increase of the echo pulse width at flat incident angles **C** is qualitatively represented, whereas the unevenness of the road surface is underrepresented in the simulation, leading to weaker distortions at the ground **F**. The shape distortions in the simulated data **D** exclusively stem from blooming at the retroreflective road markings, which are found in the real world data in a comparable way. Furthermore it is seen at the roadside pole **G** that not all layers share the same intensity in the actual Scala sensor due to the diode boundaries. The simulated data, however, assumes constant pulse intensity.

contribute considerable range-based errors at flat incidence angles **F**. Additionally, the idealized assumption that all beams share the same intensity does not hold for the Scala 2 sensor **G** due to emitter diode characteristics. It should be noted that these can be included in the proposed simulation models, but require additional laboratory measurements to calibrate.

The simulated frame contains points from 16 384 beams (more than the true Scala Gen. 2 emits), sampled from a 4096 px wide rendered image by 68 px wide beam kernels. Rendering κ through OGRE3D takes a negligible duration of under 0.01 s on an NVIDIA RTX 4000 Ada; whereas the computation of ξ, including beam diameters, presents the relevant performance bottleneck. Both algorithms are run on an Intel Core i7-13800H @ 2 500 MHz with 14 cores; Alg. 1 was implemented for purely sequential processing, yielding 9.5 s of processing time per frame. Alg. 2 in contrast takes on average 4.3 s for 1 200 range bins (corresponding to $s_{\max}$) at the same resolution, utilizing OpenCV convolutions through parallel processing utilizing the full CPU. Skipping depths with intensities entirely below a threshold sufficient to cause blooming can, for example, decrease computation time to 0.8 s for only 180 remaining depth slices.

Considering non-vanishing beam diameters is hence, despite the discussed optimizations, a very costly task. Options for parallel speedup outlined for the algorithms have thus far only been exploited in a limited way, but even then, the computation will typically be beyond real-time applications such as hardware in the loop (HiL). However, the proposed model and its algorithmic solutions enable synthesizing training and testing datasets including complex blooming artifacts and physically

Dudzik, L., Roschani, M., Sielemann, A., Trampert, K., Ziehn, J., Beyerer, J., & Neumann, C. (2025, September). Physically-Based Simulation of Automotive LiDAR. In 2025 IEEE International Automated Vehicle Validation Conference (IAVVC), doi: 10.1109/IAVVC61942.2025.11219516

Fig. 7: Deviations between the proposed single-bounce intensities for ρ_1 and a full computation of 128 bounces ρ_{128} (each 128 samples/px) using the Cycles raytracer for a single beam with an extreme radius of 8°. Intensity differences are around 10^{-2} on the rough steel wheels, mostly however below 10^{-4}.

based surface interactions. Fig. 7 shows the practical effects of the single-bounce assumption on result intensities, indicating that this model assumption does not severely impact result quality, while providing substantial speedup.

V. Conclusion and Outlook

We have presented a model for the physically-based simulation of automotive LiDAR systems with a focus on the less considered aspects of beam geometries and surface interactions through PBR, such that model parameters can be obtained through laboratory measurements of the LiDAR systems rather than from sensor output data under real-world operations. The presented model enables the simulation of blooming and light interactions at different levels of detail via systematic simplifications and adaptations to established image rendering principles. It was shown that the model parameters can be extracted systematically for ToF LiDAR systems to yield simulations that reproduce known artifacts in LiDAR systems for automotive environments, in particular complex relationships arising without explicit modeling. Relevant computational and geometric considerations are provided to support practical applications in common rendering pipelines.

The presented model and its evaluation have several limitations that are left for future work. The model, as presented, requires a considerably more comprehensive demonstration on different LiDAR systems and the evaluation against heterogeneous use cases, in comparison with existing models. Primarily, this will include the generation of synthetic datasets to evaluate the efficacy of the model for creating testing and training data for AI / ML applications, quantifying the domain gap, as demonstrated in [21]. A second prerequisite for quantitative comparisons in complex environments is a means of deriving world object geometry and surface parameters with sufficient detail. A fundamental concern in simulations, particularly analytic models, is computational effort. While different levels of detail, simplifications, and two algorithmic solutions are discussed, options for leveraging parallel processing are not discussed in detail. Neither variant is realtime-capable on regular hardware, requiring several seconds per frame to compute. Hence, additional accelerations will be required to enable the approach for HiL testing, as opposed to dataset generation. Finally, several aspects of the optical characterization are not addressed herein, especially kernel derivation for scanners that do not provide single beam emissions and ambient (passive) light reception, and the relationship between AD threshold vs. ambient noise, which remain for future work.

References

[1] L. Eisemann, M. Fehling-Kaschek *et al.*, "A Joint Approach Towards Data-Driven Virtual Testing for Automated Driving: The AVEAS Project," *arXiv preprint arXiv:2405.06286*, 2024.
[2] S. Genovesi, M. Haimerl *et al.*, "Evaluating Dimensions of AI Transparency: A Comparative Study of Standards, Guidelines, and the EU AI Act," in *Symposium on Scaling AI Assessments (SAIA)*, 2024.
[3] European Commission, "European Parliament legislative resolution of 13 March 2024 on the proposal for a Regulation of the European Parliament and of the Council Laying Down Harmonised Rules on Artificial Intelligence (Artificial Intelligence Act) and Amending Certain Union Legislative Acts (Texts Adopted, COM(2021)0206 – C9-0146/2021 – 2021/0106(COD))," Mar. 2024.
[4] G. Ros, L. Sellart, J. Materzynska *et al.*, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.
[5] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*. Springer, 2016, pp. 102–118.
[6] M. Johnson-Roberson, C. Barto, R. Mehta *et al.*, "Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?" *arXiv preprint arXiv:1610.01983*, 2016.
[7] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual Worlds as Proxy for Multi-Object Tracking Analysis," in *CVPR*, 2016.
[8] Y. Cabon, N. Murray, and M. Humenberger, "Virtual KITTI 2," *arXiv preprint arXiv:2001.10773*, 2020.
[9] S. Bullinger, C. Bodensteiner, and M. Arens, "A photogrammetry-based framework to facilitate image-based modeling and automatic camera tracking," *arXiv preprint arXiv:2012.01044*, 2020.
[10] J. R. Ziehn, M. Roschani, M. Ruf, D. Bruestle, J. Beyerer, and M. Helmer, "Imaging vehicle-to-vehicle communication using visible light," *Advanced Optical Technologies*, vol. 9, no. 6, pp. 339–348, 2020.
[11] A. Sielemann, S. Wolf *et al.*, "Synset Boulevard: A Synthetic Image Dataset for VMMR," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 9146–9153.
[12] A. Sielemann, L. Loercher *et al.*, "Synset Signset Germany: A Synthetic Dataset for German Traffic Sign Recognition," in *IEEE 27th International Conference on Intelligent Transportation Systems (ITSC)*, 2024.
[13] N. Peinecke, T. Lueken, and B. R. Korn, "Lidar simulation using graphics hardware acceleration," in *2008 IEEE/AIAA 27th Digital Avionics Systems Conference*. IEEE, 2008, pp. 4–D.
[14] J. Riordan, M. Manduhu *et al.*, "LiDAR simulation for performance evaluation of UAS detect and avoid," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2021, pp. 1355–1363.
[15] X. Yang, Y. Wang *et al.*, "Comprehensive LiDAR simulation with efficient physically-based DART-Lux model (I)," *Remote Sensing of Environment*, vol. 272, p. 112952, 2022.
[16] B. Browning, J.-E. Deschaud *et al.*, "3D Mapping for high-fidelity unmanned ground vehicle lidar simulation," *The International Journal of Robotics Research*, vol. 31, no. 12, pp. 1349–1376, 2012.
[17] H. Abdallah, N. Baghdadi *et al.*, "Wa-LiD: A new LiDAR simulator for waters," *IEEE Geoscience & remote sensing letters*, vol. 9, no. 4, 2012.
[18] J. Fang, D. Zhou *et al.*, "Augmented LiDAR simulator for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, 2020.
[19] S. Manivasagam, S. Wang *et al.*, "Lidarsim: Realistic lidar simulation by leveraging the real world," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 167–11 176.
[20] C. Li, Y. Ren, and B. Liu, "PCGen: Point Cloud Generator for LiDAR Simulation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 11 676–11 682.
[21] S. Manivasagam, I. A. Bârsan *et al.*, "Towards zero domain gap: A comprehensive study of realistic lidar simulation for autonomy testing," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 8272–8282.
[22] V. Zyrianov, H. Che *et al.*, "Lidardm: Generative lidar simulation in a generated world," *arXiv preprint arXiv:2404.02903*, 2024.
[23] M. Pharr, W. Jakob, and G. Humphreys, *Physically based rendering: From theory to implementation*. MIT Press, 2023.
[24] B. Burley and Walt Disney Animation Studios, "Physically-Based Shading at Disney," in *ACM Siggraph*, 2012.