Towards Zero Domain Gap: A Comprehensive Study of Realistic LiDAR Simulation for Autonomy Testing

Sivabalan Manivasagam[1,2*] Ioan Andrei Bârsan[1,2*] Jingkang Wang[1,2] Ze Yang[1,2] Raquel Urtasun[1,2]
[1]Waabi [2]University of Toronto
{siva, andrei, jwang, zyang, urtasun}@waabi.ai

Abstract

Testing the full autonomy system in simulation is the safest and most scalable way to evaluate autonomous vehicle performance before deployment. This requires simulating sensor inputs such as LiDAR. To be effective, it is essential that the simulation has low domain gap with the real world. That is, the autonomy system in simulation should perform exactly the same way it would in the real world for the same scenario. To date, there has been limited analysis into what aspects of LiDAR phenomena affect autonomy performance. It is also difficult to evaluate the domain gap of existing LiDAR simulators, as they operate on fully synthetic scenes. In this paper, we propose a novel "paired-scenario" approach to evaluating the domain gap of a LiDAR simulator by reconstructing digital twins of real world scenarios. We can then simulate LiDAR in the scene and compare it to the real LiDAR. We leverage this setting to analyze what aspects of LiDAR simulation, such as pulse phenomena, scanning effects, and asset quality, affect the domain gap with respect to the autonomy system, including perception, prediction, and motion planning, and analyze how modifications to the simulated LiDAR influence each part. We identify key aspects that are important to model, such as motion blur, material reflectance, and the accurate geometric reconstruction of traffic participants. This helps provide research directions for improving LiDAR simulation and autonomy robustness to these effects. For more information, please visit the project website: https://waabi.ai/lidar-dg

1. Introduction

Accurately testing the behavior of robots such as self-driving vehicles (SDVs) is of paramount importance to ensure their safe deployment in the real world. The safest, most scalable and sustainable way to test the autonomy system is through simulation. To assess the safety of the full system, it is critical to evaluate the complete autonomy stack in such a simulator. This is a must, as small changes in one sub-component (e.g., a missed detection) can cause a chain reaction of downstream effects that significantly alter the outcome, and might result in a safety hazard. To evaluate the full autonomy, we must simulate all the inputs to the system. This requires high fidelity sensor simulation with low domain gap with respect to the real world. That is, the performance of the autonomy system in simulation on all scenarios should match the real world performance.

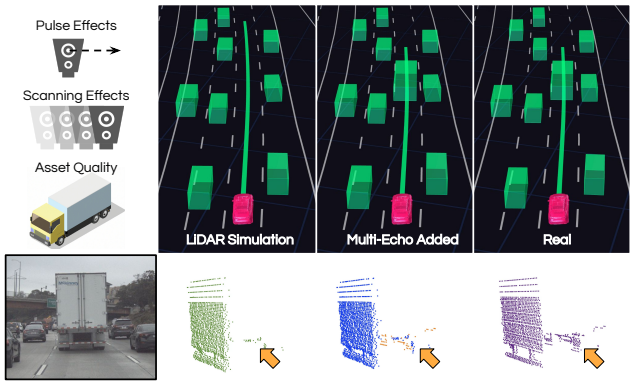

Figure 1. **Analysis Overview.** We study the impact of pulse effects, scanning effects, and asset quality on LiDAR simulation realism. We depict one example of a *pulse effect* domain gap: failure to model multiple echoes causes the object detector to fail, resulting in an unsafe autonomy plan. The bottom row depicts the front-camera view, for reference only, followed by the relevant LiDAR: original simulation, added multi-echoes, and real. We denote multi-echoes re-added by the middle method in orange. Subtle differences in the area highlighted with the arrow stem from weak returns on truck's rear wheels, impacting the domain gap.

In the past few years, a wide range of LiDAR simulation systems have been developed, as LiDAR is the primary sensor in many modern autonomy stacks [11, 53, 37, 62]. Most LiDAR simulation systems first perform virtual world creation to build a 3D scene, either using CAD models [16, 23], data-driven assets [55, 35], or a combination [18]. They then perform sensor rendering, which simulates the LiDAR scanning process to generate point clouds, either through

*Indicates equal contribution.

physics and graphics [22, 16, 18, 23], data-driven modelling [72, 6, 55] or hybrid approaches [35, 24, 7]. Despite its importance, there has been little investigation into what aspects of LiDAR simulation matter for autonomy testing, i.e., what is important for drawing insights about real-world system performance from simulation.

In this paper, we propose a novel approach for evaluating the domain gap of a sensor simulator. Unlike standard sensor simulators [16, 50] that create fully synthetic worlds that cannot be directly compared with the real world, we devise a "paired-scenario" setting, where we recreate in simulation a digital twin of the exact same scenario observed in the real world during data collection. The digital twin occurs on the same map, with the same traffic participant (actor) placement and behaviors as observed in the real world. With this digital twin, we can then simulate the LiDAR data for this scenario according to the same platform and sensor configuration it was observed with, and compare the simulated LiDAR against the real LiDAR to evaluate their differences. Through this "paired-scenario" setting, we can run autonomy on both the simulated LiDAR and real LiDAR and compare autonomy outputs, such as differences in motion plans (see Fig.1). This unique setting allows us to directly measure the domain gap of a sensor simulator with respect to autonomy.

We then conduct the first analysis of what aspects of LiDAR simulation are critical to simulate with high fidelity to ensure close matching performance of the autonomy system between the simulator and the real world. We investigate the effect of multiple sensor phenomena, including LiDAR pulse effects such as multi-path reflections and material modelling, and scanning LiDAR effects such as realistic motion blur and rolling shutter. We also investigate different ways to build the virtual world, such as with CAD models or 3D reconstruction. Since each part of the autonomy may be affected differently by the simulator's domain gap, we assess both the full stack as well as its subsystems, including perception, motion forecasting, and motion planning. Our results show that LiDAR realism is strongly affected by phenomena like motion blur and multi-echo returns, which are often not simulated in standard LiDAR simulators. We also find that standard perception metrics typically used in the community to evaluate algorithm performance during sim-to-real and real-to-sim, such as detection precision and recall, are not strong indicators of the domain gap of the autonomy system on the end task of motion planning.

While proposing a specific novel simulator to address these challenges remains an open topic, we believe that our analysis sheds light into what directions are most important for the community to tackle in order to improve simulation realism and to develop robotic systems that are less susceptible to domain gap.

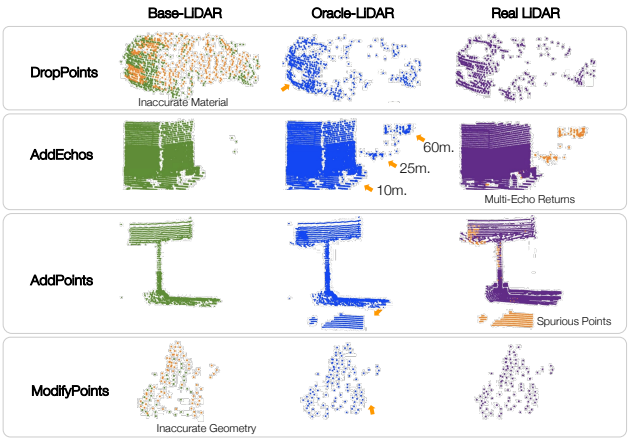

Figure 2. Given **Base-LiDAR**, we leverage an "oracle" real LiDAR point cloud to make it more realistic, such as dropping points, adding multi-path returns, spurious returns, or correcting for inaccurate geometry in simulation. The orange points indicate points that are either modified in **Base-LiDAR** or points taken from the real LiDAR and added to **Base-LiDAR**.

2. Related Work

Sensor simulation in computer vision: The computer vision community has primarily used sensor simulation for offline training data generation for boosting the performance of perception models [44]. As simulators provide inexpensive ground truth for segmentation and detection, many synthetic camera [47, 46, 21, 9] and LiDAR [63, 67, 28] datasets have been generated to train better perception models. To overcome domain gap in both content diversity and modelling realism, one line of work performs domain randomization to encourage model generalization [3, 26, 70, 45], or generates synthetic content optimized for real world task performance [43, 30, 15]. Another line of work performs domain adaptation with source synthetic data for real-world target data (e.g., segmentation [64, 71, 67]). These works want synthetic data to provide value during training, but do not investigate the realism of simulated data for perception evaluation. Prior work analyzed camera perception training [3], but there is less research on LiDAR. Our work analyzes the necessary factors for autonomy systems to perform consistently in both simulated and real-world environments.

Sensor simulation in robotics: The robotics community has leveraged LiDAR simulation primarily for sim-to-real training. Sensor simulation allows for safe and cost efficient development of the full autonomy system, as the robot can learn to interact in the virtual world environment safely and at scale [5, 8, 59]. Sensor simulators have been developed for indoor scene navigation [49, 66, 65, 54], manipulation [32, 57], and for self-driving [16, 50, 6]. Like in the vision community, several domain randomization [4, 56] and

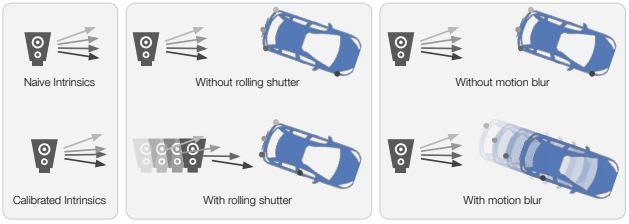

Figure 3. We analyze the importance of different ToF scanning LiDAR effects such as accurate intrinsics for generating rays, motion blur to account for actor motion during the sweep, and rolling shutter to model movement of the LiDAR sensor during the sweep.

adaptation techniques [58] have been applied to improve autonomy real world performance. But there has been limited analysis of what parts of the sensor simulation are most important for autonomy. Some works perform "simulator identification" to tune the simulation parameters to ensure better sim-to-real robot performance [17, 29, 25]. Our work focuses on understanding how LiDAR sensor phenomena affect the simulation realism when evaluating autonomy.

LiDAR Simulation for autonomy: Standard LiDAR simulators for robotics and vision often leverage classical graphics and game development techniques. Artists build 3D assets to create synthetic worlds, which are then used to simulate LiDAR with rasterization or raytracing [23, 16, 22, 50, 27]. However, these approaches often lack realism, require large amounts of computation and expert manual tuning to be realistic [22], and may lack object and scenario diversity [27]. One alternative are data-driven generative models that synthesizes LiDAR conditioned on a scene representation [10, 61, 72]. While realistic and computationally efficient, these methods often sacrifice controllability and may fail to generalize to new environments. Hybrid methods [24, 7, 35, 25, 33] combine classic rendering with learning, gaining the flexibility and controllability of physics-based rendering as well as the realism granted by data-driven priors. Our work aims to better understand the domain gap of hybrid simulators that have achieved the most success, and to better understand what directions the community should pursue to further improve realism for accurate autonomy evaluation.

3. Analyzing LiDAR Simulation

Our goal is to identify actionable insights that can help improve the quality of existing LIDAR simulation methods. While parts of our analysis apply to all LiDAR types, we focus on the mechanical spinning time-of-flight (ToF) LiDAR, the most common LiDAR sensor type utilized by SDVs. In this section, we review ToF LiDAR principles and describe the base state-of-the-art ToF LiDAR simulation system we perform analysis with (Sec. 3.1). Then, we explain how we modify this base LiDAR simulator to analyze sensor phenomena (Sec. 3.2).

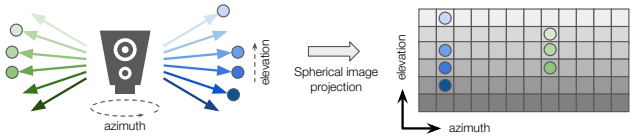

Figure 4. We compute correspondence between the simulated and real LiDAR by projecting to a spherical-image.

3.1. Base LiDAR Simulator

ToF LiDAR: ToF LiDAR measures the distance to an object by sending out pulses of collimated light in the infrared spectrum and measuring how long it takes for the received pulse to return [38]. A collocated receiver continually measures the returning photons at that wavelength, and when the returned signal strength is high enough and within a time window, it detects a return by recording the time-of-flight time Δt. The measured distance d of the detection can be computed as: $d = \frac{c\Delta t}{2}$, where c is the speed of light propagating through air, which is a constant. Based on the direction $r \in \mathbb{R}^3$ of the light pulse and the sensor origin $\mathbf{o} \in \mathbb{R}^3$, a 3D point y is extracted: $\mathbf{o} + \mathbf{r}d = \mathbf{y}$. A mechanical LiDAR sends several pulses of light by rotating a column of laser receivers/detectors around its z-axis to generate a point cloud $Y : \{\mathbf{y}_1, \ldots, \mathbf{y}_n\}$. The lasers are oriented at specific elevation angles and scan at fixed azimuth intervals over the full 360°, which derives the direction $\mathbf{r}$ of each pulse.

LiDAR Range Equation: After pulse transmission, the returned signal strength after hitting a target can be modelled by the following equation [31, 60, 38]:

$$P_r = \frac{P_t D_r^2 \eta_{\text{atm}} \eta_{\text{sys}} \sigma}{4\pi R^4 \beta_t^2}, \tag{1}$$

where P_r is the received power, P_t is the transmitted power, D_r is the receiver aperture diameter, β_t is the beam width, σ is the target cross section, R is distance travelled to target, and $\eta_{\text{atm}}\eta_{\text{sys}}$ are atmospheric and system losses. Due to sunlight and other background noise, ToF LiDARs require a minimum signal-to-noise ratio between the received pulse and the background light to detect a return. Through signal processing on the waveform of the returned P_r over a time window, the flight time t is computed to generate a LiDAR point if above the minimum detectable value.

Simulating ToF LiDAR: Most ToF LiDAR simulation systems for robotics [23, 32] and self-driving [16, 35, 18] model the world as 3D geometric assets and free space and model the LiDAR sensor pulses as sets of light rays interacting with these geometries. To better understand the domain gap of such systems, we build a similar LiDAR simulator, **Base-LiDAR**, to analyze. At each timestep t of the scenario being simulated, we first place a set of assets representing

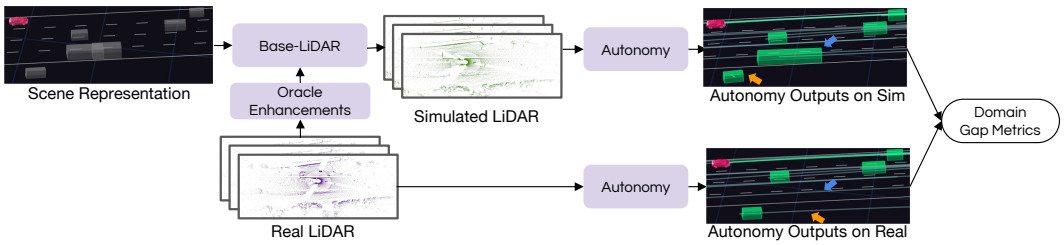

Figure 5. Given paired simulated and real LiDAR for the same scenario, we run autonomy on both in open-loop and compare the domain gap for the autonomy under test.

the actors and background into a 3D scene. We then model each light pulse as a ray with origin at the laser location, and direction based on the specified azimuth and elevation angle of the laser. This can be derived from the LiDAR extrinsics and intrinsics. We thus generate the full set of rays for each LiDAR in the scene. We model the interaction of the light pulse with the scene by computing ray-geometry intersections [39] through primary ray-casting [41], where the first intersected geometry element is a hit point and terminates. We assume all hit rays terminate and return at the first intersection, generating a simulated point cloud $\mathbf{Y}_t^s$. Please see the supplementary material for details.

3.2. Modelling LiDAR Phenomena

We now specify a LiDAR simulation taxonomy that breaks down the sensor effects in ToF LiDAR and how we incorporate them into **Base-LiDAR** to understand its affect on the domain gap with respect to the autonomy system. To properly analyze certain effects, we assume access to an "oracle" that has knowledge of the real LiDAR $\mathbf{Y}_t^r$ for the exact same scene. We describe how we access this oracle in Sec. 4. Please see Fig. 2 for visuals of the different oracle-enhanced LiDAR operations, and Fig. 3 for different scanning LiDAR effects we model and analyze.

3.2.1 Unreturned pulses

Cause: One reason for undetected returns that hit a target is due to the pulse's power decay ($\frac{1}{R^4}$) as it travels, along with other losses $\eta_{\text{atm}}, \eta_{\text{sys}}$ that affect the return signal strength, such as particle scattering and receiver efficiency. They may also not return if the target cross section σ is low, which can be due to a low incidence angle between the target surface and the pulse (especially on specular surfaces such as cars, see Fig. 2, row 1), or low reflectivity due to material composition.

Simulation: To model unreturned pulses, we project the real LiDAR $\mathbf{Y}_t^r$ and simulated **Base-LiDAR** $\mathbf{Y}_t^s$ into 2D "spherical"-coordinate images [64], where the x-axis is the binned azimuth angle from $[-2\pi, 2\pi]$, and y-axis is the discrete laser id for each sensor, sorted by elevation angle (see Fig. 4). This representation enable us to build a correspondence between simulated points and real ones falling within the same pixel. We then apply DropPoints, where points in the simulated sweep that do not have correspondence to any real point are removed.[1]

3.2.2 Multiple echoes

Cause: Sometimes, a transmitted LiDAR pulse will partially bounce on different surfaces one or more times as it travels to and from the sensor before coming back to the receiver. This can result in the LiDAR response waveform having multiple peaks within a time window. Depending on the LiDAR firmware, this can produce multiple returned points. "Multi-echo" returns can be caused by partially porous or refractive medium, such as vegetation or glass, in which multi-echoes may correspond to physical surfaces, or by reflective surfaces such as metal or water, where one or more returns may correspond to non-existing surfaces, also known as "ghosting" (e.g., floating points behind a truck due to multiple bounces in Fig. 2, row 2).

Simulation: We assume the oracle has knowledge whether each point $\mathbf{y}_i^r$ is the result of a single bounce or multiple; this metadata is commonly available from the LiDAR firmware. Points from the real LiDAR that are due to multiple bounces are added to the simulated point cloud: $\text{AddEchoes}(\mathbf{Y}_t^s, \{\mathbf{y}_{\text{multi}}^r\}) \rightarrow \mathbf{Y}_t^{s\prime}$.

3.2.3 Spurious returns

Cause: Spurious returns in LiDAR can also occur due to multi-path, blooming, beam divergence, and volume scattering effects. Multi-path is similar to multi-echo, but the returned pulse arrives from a different angle than what it was transmitted at, so it is not detected by the LiDAR sensor as an additional echo (Fig. 2, row 3). In blooming, highly reflective surfaces can produce unexpectedly strong returns, causing the return to bleed into adjacent photodiodes [34]. In beam divergence, the initially narrow pulse diameter expands as it travels. If part of the beam gets reflected by a surface, it can come back as a returned point. This can especially occur for distant objects. Spurious returns may also

[1] DropPoints can be seen as the upper bound of the learned ray dropping network from Manivasagam et al. [35].

be due to interference from particles such as exhaust or fog.

Simulation: To model these effects, we propose the AddPoints operator. Similar to DropPoints, we project the real and simulated point clouds to the spherical-image representation to compute correspondences. If real points within a binned pixel have no corresponding simulated points, we add the real points to the simulated point cloud.

3.2.4 Noisy points

Cause: Noisy points can occur where the peak in the waveform is ambiguous, resulting in inaccurate calculation of return time t. This can occur for thin structures, retro-reflectors, and inherent aleatoric noise in the real world [31].

Simulation: To model noisy points, we propose $\text{ModifyPoints}(\delta_{\text{lo}}, \delta_{\text{hi}})$. Once again, we compute correspondence between simulated and real LiDAR in the spherical image space. For simulated points that have correspondence with real, but have a range difference between the returns that lies within the specified δ range, we replace those simulated points with the corresponding real points with new distance values. The δ range pertains to the distance between the sensor and a particular point, expressed in meters. In addition to inherently noisy points in real LiDAR, $\text{ModifyPoints}(\delta_{\text{lo}}, \delta_{\text{hi}})$ can also correct inaccuracies in geometry and material modelling in simulation, such as differences in shape for a motorcycle actor (Fig. 2, row 4), or pulses going through transparent surfaces (e.g, windows) and returning the interior.

3.2.5 Spinning sensor ray generation

Cause: In addition to proper modelling of the light pulse interacting with the environment, realistic LiDAR simulation also involves simulation of the beam steering mechanism that allows the LiDAR to scan the scene. The LiDAR intrinsics specify the calibrated azimuth and elevation angles for each laser, which affect the pulse pattern and alter the point cloud density and sensor's field-of-view.

Simulation: Simulators typically use a generic calibration just by specifying the number of lasers and their field-of-view bounds without specifying exact intrinsics [16]. We test this simulation mode and also enhance **Base-LiDAR** with the calibrated intrinsics for each of the LiDARs on the SDV platform to generate the set of rays (Fig. 3 col. 1).

3.2.6 Rolling shutter and motion blur

Cause: Spinning mechanical LiDARs gather measurements over time: It typically takes 100 ms for a 360° LiDAR

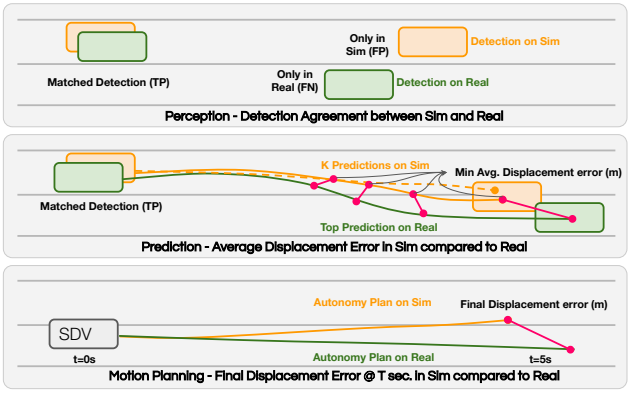

Figure 6. Given autonomy outputs on simulated and real LiDAR we compute the agreement between the two with outputs on real LiDAR being the "ground-truth".

to scan the full scene. If the SDV (ego-vehicle) is moving during this time, the scan will be distorted as the pose of the sensor changes during the scan (i.e., rolling shutter in Fig. 3 col. 2). Similarly, the motion of dynamic actors can blur the generated LiDAR point cloud (Fig. 3 col. 3), and can produce changes in the location of where the actor is observed in the sweep. Depending on the relative direction of movement, actors can appear compressed or elongated in the scene. These effects increase at highway speeds, where the actors and SDV are moving around 30 m/s. We separate these effects in our analysis because rolling shutter can be corrected by ego-motion estimation, but motion blur cannot, as it depends on the motion of each individual actor.

Simulation: We enhance **Base-LiDAR** to simulate these effects by assigning each ray a firing timestamp, and modifying the sensor origin of the ray by applying SLERP, [52] which interpolates the sensor pose over time using the recorded trajectory of the SDV. Similarly, for motion blur, we specify the trajectories of the actor poses and apply SLERP to the actors' geometries during the ray casting to ensure dynamic actors are at the correct position for each ray's firing time. To summarize, the simulator is physics-based with data-driven assets, hence hybrid, but does not make any direct use of machine learning. Please see the supplementary material for details.

4. Measuring Domain Gap

Paired Scenario Setting: To directly measure the domain gap between a LiDAR simulator and the real world for the autonomy system under test, we propose to analyze the domain gap in a novel "paired"-scenario setting (see Fig. 5 for illustration). Given the real scenario we want to simulate, we construct a "digital-twin" of this exact same scenario (e.g., map location, actor placement) in simulation. This allows us to compare the simulated LiDAR directly with the real LiDAR in a pair-wise fashion.

#	DropP	AddE	AddP	ModP	Detection ΔAP ↓	Detection ΔRecall ↓	Detection DA AP ↑	Detection DA Recall ↑	Prediction minADE ↓	Planning PD@5s ↓
1					0.047	0.032	0.77	0.80	1.74	3.22
2	✓				0.035	0.035	0.75	0.78	1.87	3.30
3		✓			0.044	0.034	0.79	0.82	1.58	2.71
4			✓		0.046	0.036	0.81	0.85	1.43	3.21
5		✓	✓		0.056	0.046	0.83	0.86	1.35	2.50
6				[0, 200]	0.052	0.041	0.88	0.90	0.98	1.80
7	✓		✓		0.029	0.023	0.81	0.83	1.46	3.22
8	✓			[0, 200]	**0.009**	**0.011**	**0.93**	**0.93**	**0.42**	**0.92**
9			✓	[0, 200]	0.053	0.045	0.91	**0.93**	0.93	1.69
10		✓	✓	[0, 200]	0.053	0.045	0.91	**0.93**	0.93	1.69
oracle	✓	✓	✓	✓	0.000	0.000	1.00	1.00	0.00	0.00

Table 1. **LiDAR Pulse Phenomena**: Enhancing **Base-LiDAR** with ray propagation effects such as unreturned pulses (**DropP**oints), multi-path (**AddE**chos), spurious points (**AddP**oints), and noisy points (**ModP**oints)

We define a scenario $\mathbf{X}$ as a temporal sequence of scene representations $\mathbf{X} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$. Each $\mathbf{x}_t$ defines the state of the world, the environment, and where the relevant traffic participants are with respect to the simulated sensor's location. The scenario can be extracted through human annotation or automatic offline-labelling of real-world data collects containing a sequence of real LiDAR point clouds $\mathbf{Y}_1^r \ldots \mathbf{Y}_T^r$ for the LiDAR sensors exploited by the autonomy system under test. We then can compose virtual geometry assets to match the scene $\mathbf{x}_t$ and simulate the LiDAR for the SDV platform as described in Sec. 3.1 in order to evaluate its domain gap. For domain gap, "similarity" is defined as whether the autonomy performs the same on either simulated or real LiDAR.

Autonomy System Under Test: The autonomy system takes as input a sequence of LiDAR data and additional information k_{aux} such as the map and generates autonomy outputs $\mathcal{Z}$: $f(\{\mathbf{Y}_t, \ldots \mathbf{Y}_{t+4}\}, k_{\text{aux}}) \rightarrow \mathcal{Z}$. We evaluate the autonomy system in an open-loop setting, where at each timestep we provide as input a sequence of either simulated or real LiDAR, which generates autonomy outputs $\mathcal{Z}_t^s, \mathcal{Z}_t^r$, respectively. We then compute domain gap metrics between $\mathcal{Z}_t^s$ and $\mathcal{Z}_t^r$, where $\mathcal{Z}_t^r$ is the desired target. Specifically, we toggle different LiDAR pulse and scanning effects on the simulated LiDAR and evaluate their effect on the domain gap. By leveraging this paired domain-gap setting, we can access the real LiDAR $\mathbf{Y}_t^r$ at each time-step as our oracle.

We compute domain gap metrics using a modular autonomy system that performs perception (object detection), motion forecasting (actor trajectories), and motion planning (output planned trajectory of the SDV). We focus on the evaluation setting, where the autonomy system is already trained and we want to evaluate the realism of the LiDAR simulator with respect to the autonomy system. The autonomy system takes as input the past 0.5 seconds of LiDAR history from *all three LiDARs*, generates actor detections, 6 second trajectory forecasts for each detected actor, and a 5 second planned trajectory. Detection is performed using a two-stage LiDAR detector [51, 36] followed by a lane-graph network which encodes trajectories and map information to forecast actor motion [13]. Finally, a sampling-based path-lateral time (PLT) planner [48] plans an ego trajectory that balances safety, comfort, and route progress costs. Our analysis is general and can be performed for any autonomy system; see the supplementary material for results on additional autonomy configurations and further implementation details. We now describe the metrics in more detail (see Fig. 6 for illustration).

Detection Distributional Agreement (ΔAP, ΔRecall): We measure the absolute differences in autonomy performance when run on the simulated and real LiDAR dataset using average precision (AP) and recall. We therefore define $\Delta\text{AP} = |\text{AP}_{\text{real}} - \text{AP}_{\text{sim}}|$ for AP, and an equivalent metric for recall, under a pre-specified IoU threshold. A perfect simulator would mean that autonomy has the same performance between the simulated and real datasets, resulting in ΔAP and ΔRecall of 0. We include this metric for completeness, as it is used commonly in the field.

Detection Agreement (DA): Distributional metrics only measure perception performance in aggregate over the full dataset. To assess whether an actor in simulation at time t in the same frame of the paired-scenario is mis-detected or correctly detected in both the simulated and real LiDAR, we report detection agreement (DA). Intuitively, detection agreement is a non-symmetric measure of similarity for two sets of model outputs, computed by treating one of the sets as pseudo-labels. In practice we consider the autonomy model outputs computed on the real LiDAR as "labels", and the simulated LiDAR model outputs as the proposals. We then compute the average precision and recall under differ-

ent IoU thresholds.[2] DA AP and DA Recall of 1 mean that an autonomy systems generates the same output detection set on both simulated and real LiDAR.

Prediction Error (minADE): We compute the differences between motion forecasting outputs run when the input to the autonomy system is simulated LiDAR and real LiDAR respectively. We use the displacement error of each actor as the disagreement metric. Since prediction models are often multi-modal, to handle diverse futures [14], we use the most-likely mode in the real LiDAR as the ground-truth, and we compute the minimum average displacement error (minADE) between all the trajectory modes predicted by the autonomy system when run with the simulated LiDAR as input for that actor. Since different simulation methods result in different actor prediction sets, we evaluate the prediction results at a fixed detection recall.

Plan Discrepancy (PD): As we perform open-loop planning on both real and simulated LiDAR sequences, we can measure the discrepancies between planner outputs. Planning metrics are helpful in understanding the impact of sensor realism on the decision making of the SDV [42]. Specifically, we measure the ℓ_2 error between planner waypoints at a fixed time in the future, for every log frame:

$$\text{PlanDiscrepancy} = \sum_{i=1}^{N} \sum_{m=i}^{i+P} \left\| \tau_m^{(s)} - \tau_m^{(r)} \right\|_2^2,$$

where N is the length of simulation, P the planning horizon, and τ_m denotes the plan trajectory at time m, under either real or simulated data.

5. What Matters for LiDAR Realism?

We now describe the dataset of paired scenarios we evaluate on (Sec. 5.1) and perform our analysis in three parts. We first analyze different LiDAR *pulse modelling* effects by applying different oracles (Fig. 2) in isolation and also in combination to understand their impact on the domain gap (Sec. 5.2). All scanning LiDAR effects are enabled in this analysis. We then analyze different *scanning LiDAR* effects such as calibrated intrinsics, rolling shutter, and motion blur, with no oracle enhancements applied (Sec. 5.3). Finally, we investigate different ways of building the virtual world geometry meshes for both the foreground and background, with scanning LiDAR effects enabled. (Sec. 5.4).

[2]Unlike common AP and recall in the detection literature, we do not use any human annotation labels for this metric. This is in contrast to prior variants of this metric, such as the agreement used by Manivasagam et al. [35], which relied on first matching all detections to labels, thereby not measuring agreement in terms of false positives.

5.1. Evaluation Setting Details

Dataset: To our knowledge, no public dataset [53, 11, 12] provides the detailed LiDAR information necessary for our analysis, such as multi-return metadata, LIDAR intrinsics, and per-point timestamps. We therefore captured a *Multi-LiDAR-Highway* dataset to perform our analysis, which consists of twenty 20 second annotated snippets captured on US-101 in California with different traffic densities and vehicle types. To analyze different ToF LiDAR, we equip the data collection vehicle with three mechanical spinning LiDARs - a central long-range LiDAR (up to 200m) as well as two medium-range (up to 80m) 128-beam side LiDARs. The long-range LiDAR provides up to two returns per pulse.

5.2. Analyzing LiDAR Pulse Phenomena

Table 1 reports the effect of different LiDAR phenomena on the domain gap. We evaluate all perception metrics at IoU=0.7 [35], and report prediction metrics at a fixed recall of 0.3. Higher recall could not be set, as certain evaluation variations could not achieve it. We report several findings.

We find that multi-echo points from the long-range LiDAR, which only accounts for ∼5% of the total input points, substantially improves domain gap metrics. Qualitatively, AddEchoes (row 4) enables the simulated LiDAR to model multiple echoes and alters object detection, enabling better matching with the real LiDAR, including false positives (Fig. 1), and ensuring agreement with downstream planning. AddPoints alone improves detection agreement and prediction while not reducing planning discrepancy. It helps especially for better detection agreement at long range, suggesting modelling spurious points may matter in these regions. We also find that, while on average, performing DropPoints alone harms domain gap (row 2), certain situations show it better matching motion planning outputs w.r.t real LiDAR (Fig. 7, left). Furthermore, pairing it with $\text{ModifyPoints}(\delta_{\text{lo}}, \delta_{\text{hi}})$ (row 8) results in the best realism gain over all oracle policies. This indicates that better geometry reconstruction of the actors and scene in conjunction with better material modelling are key to better realism.

We also find detection distributional agreement may not be sufficient for measuring realism. Counter-intuitively, a setting may have smaller ΔAP while producing higher output disagreement (i.e., row 1 vs. row 5). This is because metrics such as AP can look similar even if autonomy makes different mistakes between simulated and real LiDAR. Paired metrics better reflect the true task setting.

5.3. Analyzing Scanning LiDAR Effects

Table 2 reports the domain gap metrics for different scanning LiDAR effects ablated. "Naive" intrinsics correspond to linearly spaced laser elevations, as real LiDARs employ uneven patterns to maximize long range coverage [1]. No rolling shutter or no motion blur means we place the SDV

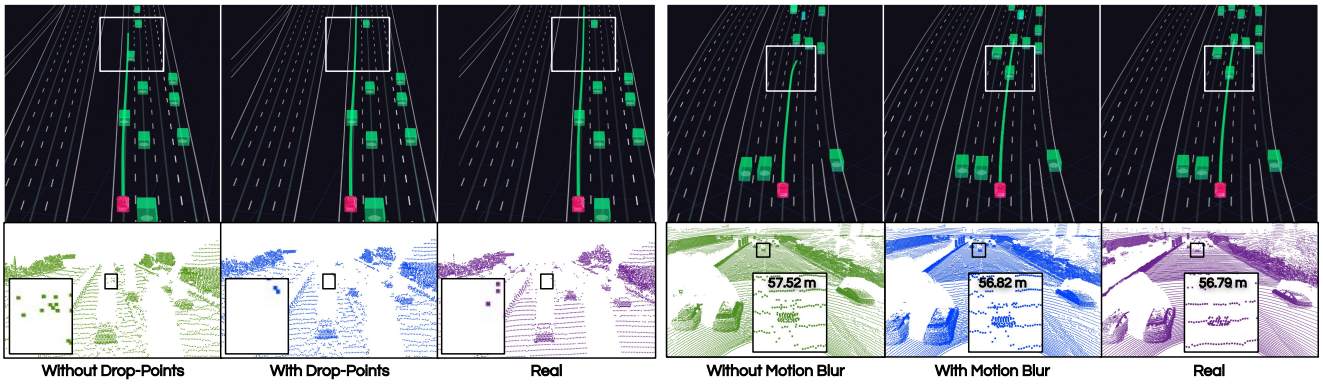

Figure 7. Qualitative examples on enhancing Base-LiDAR with DropPoints (left) and motion blur (right) to reduce the domain gap.

#	Intrinsics	RS	MB	Detection ΔAP ↓	Detection ΔRecall ↓	Detection DA AP ↑	Detection DA Recall ↑	Prediction minADE ↓	Planning PD@5s ↓
11	Naive	✓	✓	0.04	0.03	0.73	0.76	1.95	3.35
12	Calibrated	✓	✓	0.05	0.03	0.77	0.80	1.74	3.22
13	Calibrated			0.20	0.15	0.49	0.55	2.70	4.57
14	Calibrated		✓	0.05	0.03	0.77	0.80	1.76	3.15
15	Calibrated	✓		0.20	0.15	0.49	0.55	2.73	4.62

Table 2. **Spinning LiDAR Effects**: Analyzing the the domain gap for calibrated intrinsics, rolling shutter (RS) and motion blur (MB).

and the actors respectively at their end-of-sweep position and keep them static for the 100ms sweep duration. We observe increased domain gap with naive intrinsics, indicating that autonomy is not invariant to the LiDAR scanning pattern, which causes certain spatial regions to have different point density between the simulated and real LiDAR. More significantly, we find that modelling actor motion during the LiDAR sweep is critical to ensure matching autonomy outputs. Where in space and time the actor is observed affects the autonomy's outputs significantly (see Fig 7, right). Most LiDAR simulation systems used by the self-driving research community do not account for this effect [16]. Surprisingly, we find toggling ego rolling-shutter has fluctuations in the domain gap, reducing domain gap on its own, but slightly harming with motion blur. We conjecture this is because the autonomy under test consumes motion-compensated LiDAR, a standard practice in most benchmarks [11, 53]. Analyzing autonomy operating on raw LiDAR is a promising research direction which would also test state estimation [20].

5.4. Analyzing Virtual World Creation

One key design choice for LiDAR simulation is how the mesh representations are constructed during virtual world creation. As observed in Sec. 5.2, dropping and modifying LiDAR points together can substantially boost realism. We therefore decided to investigate further how different ways of building the virtual world geometry can affect domain gap. We divide the analysis into two main areas: modelling background regions such as the road, and foreground actors such as vehicles. For **Base-LiDAR**, we follow [68, 35, 55] and perform LiDAR-aggregation of collected logs to build surfel asset meshes for the actors and static background. Please see the supplementary material for details. While faithfully matching the observations, surfel meshes may suffer from topological problems, and their construction is unable to account for sensor noise.

Asset bank	Detection ΔAP ↓	Detection ΔRecall ↓	Detection DA AP ↑	Detection DA Recall ↑	Prediction minADE ↓	Planning PD@5s ↓
Base (Surfels) [35]	0.047	0.032	0.77	0.80	1.74	3.22
Background Reconstruction						
Road-only Mesh	0.057	0.043	0.75	0.79	1.95	3.13
Neural Mesh	0.055	0.042	0.77	0.80	1.57	2.98
Foreground Asset Bank						
CAD Assets	0.084	0.069	0.68	0.74	1.96	3.41
CAD + Surfel Assets	0.077	0.050	0.72	0.77	1.93	3.06

Table 3. Effect of different virtual world creation approaches.

Background Creation: Besides surfel aggregation, we explored two other approaches for background creation: heuristic road-only meshing, and neural reconstruction [69]. For the road-only mesh, we adopt RANSAC plane fitting [19] to identify ground points and then create a road-only mesh grid based on the ground height. For neural reconstruction, we adopt state-of-the-art neural reconstruction approaches [69, 40] for large scenes and extract geometry using marching cubes. As shown in Table 3, using either road-only mesh or neural mesh can reduce the motion planning domain gap.

Foreground Modelling: To model the foreground actors, we also consider artist-created CAD models [16], as well

as a combination of the two, where we combine CAD models with a manually curated set of 942 actor meshes that have cleaner geometry. For CAD models, we purchase over 120 CAD models from TurboSquid [2] for a wide range of rigid actors, such as vehicles and motorcycles. As shown in Table 3, using only CAD models lead to a larger domain gap compared to real-world reconstruction. We find using a combination of CAD models and surfel assets, despite having higher perception and prediction domain gap, improves the planning discrepancy. We hypothesize this might be due to better modelling of the actors of interest that affect the motion planning, such as actors directly in front or behind the SDV. Please see the supplementary material for details.

6. Conclusion

In this paper, we analyze what aspects of LiDAR pulse and scanning effects, as well as virtual asset creation, impact domain gap for autonomy testing. We proposed a paired-scenario setting for domain gap evaluation, and designed autonomy metrics specifically for measuring the domain gap between simulated and real LiDAR. We find there are several effects that are important to model which are missing in existing LiDAR simulation, including the ability to simulate multiple echoes and unreturned rays.

There are also several limitations with the presented analysis. We focused on open-loop evaluation, as it is necessary for ensuring identical behaviors for all other traffic participants. Subsequent work and metrics are necessary for analyzing simulator domain gap in closed loop, i.e., when the SDV itself acts differently than it did in the original log.

We hope our insights shed light onto new research directions for more realistic LiDAR simulation, as well as onto autonomy robustness to these differences, enabling simulation to be a better testing tool.

Acknowledgements: We thank the Waabi team for their valuable assistance and support. In particular, we would like to thank Carter Fang for his insights into perception models, Melinda Lu for her help on domain gap analysis, and Nathan Chau for providing guidance on visualizing interesting cases for the paper videos.

References

[1] Pandar-128 user manual. https://perceptionengine.jp/pdf/hesai/manual/Pandar128_User_Manual.pdf. Accessed: 2023-03-08. Fig 1.5 covers the laser vertical distribution. 7

[2] Turbosquid. https://www.turbosquid.com/. Accessed: 2023-03-06. 9

[3] David Acuna, Jonah Philion, and Sanja Fidler. Towards optimal strategies for training self-driving perception models in simulation. *NeurIPS*, 34:1686–1699, 2021. 2

[4] Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019. 2

[5] Alexander Amini, Igor Gilitschenski, Jacob Phillips, Julia Moseyko, Rohan Banerjee, Sertac Karaman, and Daniela Rus. Learning robust control policies for end-to-end autonomous driving from data-driven simulation. *IEEE Robotics and Automation Letters*, 2020. 2

[6] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. *arXiv preprint arXiv:2111.12083*, 2021. 2

[7] Benjamin Attal, Eliot Laidlaw, Aaron Gokaslan, Changil Kim, Christian Richardt, James Tompkin, and Matthew O'Toole. Törf: Time-of-flight radiance fields for dynamic scene view synthesis. *NeurIPS*, 34, 2021. 2, 3

[8] Arunkumar Byravan, Jan Humplik, Leonard Hasenclever, Arthur Brussee, Francesco Nori, Tuomas Haarnoja, Ben Moran, Steven Bohez, Fereshteh Sadeghi, Bojan Vujatovic, et al. Nerf2real: Sim2real transfer of vision-guided bipedal motion skills using neural radiance fields. *arXiv preprint arXiv:2210.04932*, 2022. 2

[9] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. *arXiv preprint arXiv:2001.10773*, 2020. 2

[10] Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of LiDAR data. In *IROS*, 2019. 3

[11] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. *CVPR*, 2020. 1, 7, 8

[12] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D tracking and forecasting with rich maps. *CVPR*, 2019. 7

[13] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. In *ICRA*, 2023. 6

[14] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In *ICRA*, 2019. 7

[15] Jeevan Devaranjan, Amlan Kar, and Sanja Fidler. Metasim2: Unsupervised learning of scene structure for synthetic data generation. *ECCV*, 2020. 2

[16] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *CoRL*, 2017. 1, 2, 3, 5, 8

[17] Yuqing Du, Olivia Watkins, Trevor Darrell, Pieter Abbeel, and Deepak Pathak. Auto-tuned sim-to-real transfer. In *ICRA*, 2021. 3

[18] Jin Fang, Dingfu Zhou, Feilong Yan, Tongtong Zhao, Feihu Zhang, Yu Ma, Liang Wang, and Ruigang Yang. Augmented

lidar simulator for autonomous driving. *IEEE Robotics and Automation Letters*, 5(2):1931–1938, 2020. 1, 2, 3

[19] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. 8

[20] Davi Frossard, Shun Da Suo, Sergio Casas, James Tu, and Raquel Urtasun. StrObe: Streaming object detection from LiDAR packets. In *CoRL*, 2021. 8

[21] Adrien Gaidon, Qiao Wang, Yohann Cabon, and Eleonora Vig. Virtual worlds as proxy for multi-object tracking analysis. *CVPR*, 2016. 2

[22] Adam A Goodenough and Scott D Brown. Dirsig5: Next-generation remote sensing data and image simulation framework. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 10(11):4818–4833, 2017. 2, 3

[23] Michael Gschwandtner, Roland Kwitt, Andreas Uhl, and Wolfgang Pree. Blensor: Blender sensor simulation toolbox. In *International Symposium on Visual Computing*, pages 199–208. Springer, 2011. 1, 2, 3

[24] Benoît Guillard, Sai Vemprala, Jayesh K Gupta, Ondrej Miksik, Vibhav Vineet, Pascal Fua, and Ashish Kapoor. Learning to simulate realistic LiDARs. In *IROS*, 2022. 2, 3

[25] Eric Heiden, Ziang Liu, Ragesh K Ramachandran, and Gaurav S Sukhatme. Physics-based simulation of continuous-wave lidar for localization, calibration and tracking. In *ICRA*, 2020. 3

[26] Jiaxing Huang, Dayan Guan, Aoran Xiao, and Shijian Lu. FSDR: Frequency space domain randomization for domain generalization. In *CVPR*, 2021. 2

[27] Sebastian Huch, Luca Scalerandi, Esteban Rivera, and Markus Lienkamp. Quantifying the LiDAR sim-to-real domain shift: A detailed investigation using object detectors and analyzing point clouds at target-level. *IEEE Transactions on Intelligent Vehicles*, 2023. 3

[28] Braden Hurl, Krzysztof Czarnecki, and Steven Waslander. Precise synthetic image and lidar (presil) dataset for autonomous vehicle perception. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2522–2529. IEEE, 2019. 2

[29] Yifeng Jiang, Tingnan Zhang, Daniel Ho, Yunfei Bai, C Karen Liu, Sergey Levine, and Jie Tan. SimGAN: Hybrid simulator identification for domain adaptation via adversarial reinforcement learning. In *ICRA*, 2021. 3

[30] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. *ICCV*, 2019. 2

[31] Alireza G Kashani, Michael J Olsen, Christopher E Parrish, and Nicholas Wilson. A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration. *Sensors*, 15(11):28099–28128, 2015. 3, 5

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *IROS*, 2004. 2, 3

[33] Chenqi Li, Yuan Ren, and Bingbing Liu. PCGen: Point cloud generator for LiDAR simulation. In *ICRA*, 2022. 3

[34] Derek D Lichti, Stuart J Gordon, and Taravudh Tipdecho. Error models and propagation in directly georeferenced terrestrial laser scanner networks. *Journal of surveying engineering*, 131(4):135–142, 2005. 4

[35] Sivabalan Manivasagam, Shenlong Wang, Kelvin Wong, Wenyuan Zeng, Mikita Sazanovich, Shuhan Tan, Bin Yang, Wei-Chiu Ma, and Raquel Urtasun. LiDARsim: Realistic LiDAR simulation by leveraging the real world. In *CVPR*, 2020. 1, 2, 3, 4, 7, 8

[36] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid R-CNN: Towards better performance and adaptability for 3D object detection. In *ICCV*, 2021. 6

[37] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, et al. One million scenes for autonomous driving: ONCE dataset. *NeurIPS*, 2021. 1

[38] Paul F McManamon. *Field Guide to Lidar*. SPIE Press, 2015. 3

[39] Tomas Möller and Ben Trumbore. Fast, minimum storage ray/triangle intersection. In *ACM SIGGRAPH 2005 Courses*, pages 7–es. 2005. 4

[40] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *SIGGRAPH*, 2022. 8

[41] Steven G Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, et al. Optix: a general purpose ray tracing engine. *ACM transactions on graphics (TOG)*, 29(4):1–13, 2010. 4

[42] Jonah Philion, Amlan Kar, and Sanja Fidler. Learning to evaluate perception models using planner-centric metrics. In *CVPR*, 2020. 7

[43] Aayush Prakash, Shaad Boochoon, Mark Brophy, David Acuna, Eric Cameracci, Gavriel State, Omer Shapira, and Stan Birchfield. Structured domain randomization: Bridging the reality gap by context-aware synthetic data. In *ICRA*, 2019. 2

[44] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. *ECCV*, 2016. 2

[45] Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, 2(8):428–436, 2020. 2

[46] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 2

[47] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. *CVPR*, 2016. 2

[48] Abbas Sadat, Mengye Ren, Andrei Pokrovsky, Yen-Chen Lin, Ersin Yumer, and Raquel Urtasun. Jointly learnable behavior and trajectory planning for self-driving vehicles. In *IROS*, 2019. 6

[49] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied AI research. In *ICCV*, 2019. 2

[50] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics: Results of the 11th International Conference*, pages 621–635. Springer, 2018. 2, 3

[51] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019. 6

[52] Ken Shoemake. Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254, 1985. 5

[53] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. *CVPR*, 2020. 1, 7, 8

[54] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. In *NeurIPS*, 2021. 2

[55] Abhijeet Tallavajhula, Cetin Mericli, and Alonzo Kelly. Off-road lidar simulation with data-driven terrain primitives. In *ICRA*, pages 7470–7477. IEEE, 2018. 1, 2, 8

[56] Gabriele Tiboni, Karol Arndt, and Ville Kyrki. DROPO: Sim-to-real transfer with offline domain randomization. *arXiv preprint arXiv:2201.08434*, 2022. 2

[57] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IROS*, 2012. 2

[58] Joanne Truong, Sonia Chernova, and Dhruv Batra. Bi-directional domain adaptation for sim2real transfer of embodied navigation agents. *IEEE Robotics and Automation Letters*, 6(2):2634–2641, 2021. 3

[59] Tsun-Hsuan Wang, Alexander Amini, Wilko Schwarting, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. Learning interactive driving policies via data-driven simulation. In *ICRA*, 2022. 2

[60] Claus Weitkamp. *Lidar: range-resolved optical remote sensing of the atmosphere*, volume 102. Springer Science & Business, 2006. 3

[61] Tim A Wheeler, Martin Holder, Hermann Winner, and Mykel J Kochenderfer. Deep stochastic radar models. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 47–53. IEEE, 2017. 3

[62] Benjamin Wilson, William Qi, Tanmay Agarwal, John Lambert, Jagjeet Singh, Siddhesh Khandelwal, Bowen Pan, Ratnesh Kumar, Andrew Hartnett, Jhony Kaesemodel Pontes, et al. Argoverse 2: Next generation datasets for self-driving perception and forecasting. *arXiv preprint arXiv:2301.00493*, 2023. 1

[63] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D lidar point cloud. In *ICRA*, 2018. 2

[64] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019. 2, 4

[65] Fei Xia, Chengshu Li, Kevin Chen, William B Shen, Roberto Martın-Martın, Noriaki Hirose, Amir R Zamir, Li Fei-Fei, and Silvio Savarese. Gibson env v2: Embodied simulation environments for interactive navigation. *Stanford University, Tech. Rep.*, 2019. 2

[66] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 2

[67] Aoran Xiao, Jiaxing Huang, Dayan Guan, Fangneng Zhan, and Shijian Lu. Transfer learning from synthetic to real lidar point cloud for semantic segmentation. *arXiv: 2107.05399*, 2021. 2

[68] Zhenpei Yang, Yuning Chai, Dragomir Anguelov, Yin Zhou, Pei Sun, Dumitru Erhan, Sean Rafferty, and Henrik Kretzschmar. SurfelGAN: Synthesizing realistic sensor data for autonomous driving. *CVPR*, 2020. 8

[69] Ze Yang, Yun Chen, Jingkang Wang, Sivabalan Manivasagam, Wei-Chiu Ma, Anqi Joyce Yang, and Raquel Urtasun. UniSim: A neural closed-loop sensor simulator. In *CVPR*, 2023. 8

[70] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *ICCV*, 2019. 2

[71] Sicheng Zhao, Yezhen Wang, Bo Li, Bichen Wu, Yang Gao, Pengfei Xu, Trevor Darrell, and Kurt Keutzer. ePointDA: An end-to-end simulation-to-real domain adaptation framework for LiDAR point cloud segmentation. In *AAAI*, 2021. 2

[72] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic LiDAR point clouds. In *ECCV*, 2022. 2, 3