Supplementary Material for *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

In the supplementary material, we provide additional details on our evaluation setting, include additional analysis, and finally note limitations. In Sec. 1, we provide additional explanations for how we simulate the same sensor platform and LiDAR configuration, and incorporate pulse and scanning effects. In Sec. 2, we explain how we build the digital twin asset geometry representations for paired-scenarios. We then include additional details about the perception, prediction, and planning modules for the autonomy system under test (Sec. 3). Next, we showcase additional visualizations and metrics for our domain gap analysis in Sec. 4, including different distance thresholds for $\text{ModifyPoints}(\delta_{lo}, \delta_{hi})$. *Finally, we analyze the limitations of our analysis in Sec. 6 and propose future analysis directions. The video associated with this paper provides an overview of our methodology and visual results showing the impact of LiDAR phenomena on the domain gap.*

1. LiDAR Simulation Details

1.1. Ray Generation:

Ray creation: We now describe how we generate the rays for a given LiDAR sensor extrinsics and intrinsics. Given the sensor's azimuth angle $\theta \in [0, 2\pi]$, the azimuth angle offset $\theta^i_{\text{off}} \in [0, 2\pi]$ for laser i, the laser's elevation angle ϕ^i, the ray $\mathbf{r}^i_\theta \in \mathbb{R}^3$ is defined as:

$$\mathbf{r}^i_\theta = [\cos\theta^i \cos\phi^i; \sin\theta^i \cos\phi^i; \sin\phi^i]$$

where $\theta^i = \theta + \theta^i_{\text{off}}$. The sensor origin $\mathbf{o} \in \mathbb{R}^{\nvDash}$ is initially $\mathbf{0}$. The azimuth angles are sampled according to the azimuth resolution specified in each LiDAR's spec sheet (approximately 0.2° and 0.35° for the long-range and mid-range LiDARs respectively). This ensures that in the range view image, each pixel has ≤ 1 points, save for dual returns. To transform the

*Indicates equal contribution.

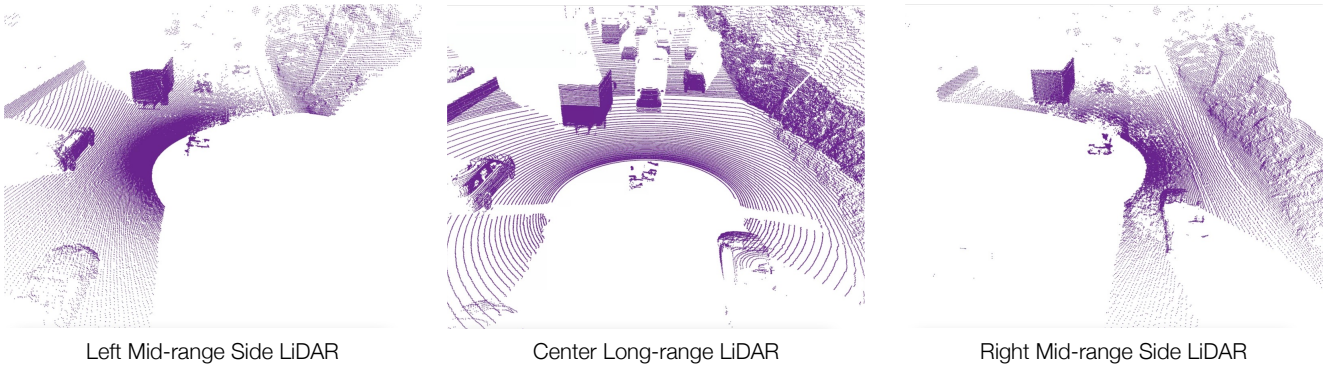

Figure 1. Real LiDAR Setup of the vehicle platform under test for measuring LiDAR simulation domain gap.

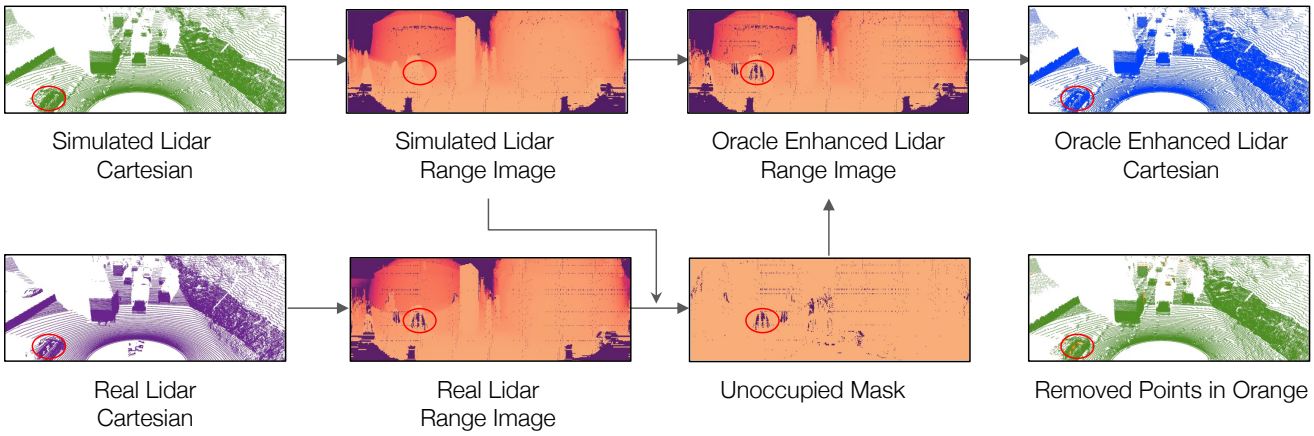

Figure 2. Example of applying DropPoints oracle to the simulated LiDAR. We convert both point clouds into the spherical image representation and identify pixels not occupied in the real LiDAR that are occupied in the simulated LiDAR, generating the *Unoccupied Mask*. This mask is applied to the simulated LiDAR to generate the *Oracle-Enhanced* LiDAR.

LiDAR ray from sensor coordinates to vehicle coordinates, we apply a transform to both $\mathbf{o}, \mathbf{r}$ according to the sensor extrinsics $\mathbf{T}_{\text{sen}}^{\text{veh}} := [\mathbf{R}_{\text{sen}}^{\text{veh}}, \mathbf{t}; \mathbf{0}^T, 1]$, such that $\mathbf{r}_{\text{veh}} = \mathbf{R}_{\text{sen}}^{\text{veh}} \mathbf{r}_{\theta}^{i}$, $\mathbf{o}_{\text{veh}} = \mathbf{T}_{\text{sen}}^{\text{veh}} \mathbf{o}$.

Spherical image: The above ray $\mathbf{r}$ is in cartesian coordinates in $\mathbb{R}^3$, but can also be projected into a 2D image representation according to the laser index i, sorted by elevation angle, and binned azimuth angle $\left\lceil \frac{\theta^i}{\text{azimuth res.}} \right\rceil$. These discretized coordinates can access pixel values in the spherical image representation $\mathbf{I}$, such as depth or occupancy. Fig. 2 left depicts projection of cartesian LiDAR data into spherical image space.

Self-occlusion: Depending on the sensor mount points, some rays sent by the sensor will hit the vehicle platform itself, i.e., self-hits. These self-hits are typically filtered out before giving the LiDAR inputs to autonomy. To match this in simulation, we generate a self-occlusion mask $\mathbf{I}^{\text{mask}}$ to filter these rays out prior to ray-casting. Similar to [8], we aggregate the LiDAR in spherical-image space from recorded log snippets and generate an averaged depth image. We filter rays that fall into pixel values that have an average distance of less than one meter.

Sensor Configuration and Dataset Details: We simulate three time-of-flight (ToF) proprietary 128-beam LiDARs on our vehicle platform. Two mid-range LiDARs are on the side of the vehicle with range up to 120 meters. One long-range LiDAR in the center has range up to 200 meters. Each LiDAR has their own calibrated extrinsics and intrinsics. Please see Fig. 1 for an example visualization of all three LiDARs in vehicle coordinates. For the 20 scenarios evaluated in *Multi-LiDAR-Highway*, we labeled dynamic objects that are only on the same highway as the SDV and do not label traffic on the opposite side, as they do not affect autonomy. We do not simulate opposite side traffic, and we use the map to filter autonomy outputs generated on the opposite highway.

1.2. Applying the Scanning Effects

Rolling Shutter: LiDAR is a temporally scanning sensor, where each ray has an exact firing time t. We thus need to transform the origin and direction of the LiDAR ray according to the position of the SDV at time t. Given the poses of the SDV $\mathbf{T}_{\text{veh}}^{\text{world}}(t_0), \mathbf{T}_{\text{veh}}^{\text{world}}(t_1)$ at sampled time points t_0 and t_1 denoting the start and end of the LiDAR sweep, and a target time t where $t_0 \leq t \leq t_1$, we apply SLERP [18] interpolation on the rotation components in quaternion space, and linear interpolation of the translation, obtaining $\mathbf{T}_{\text{veh}}^{\text{world}}(t)$. This transform is then applied to the ray $\mathbf{r}_{\text{veh}}$ and origin $\mathbf{o}_{\text{veh}}$ from Sec. 1.1 to transform the ray into the scene coordinates for primary raycasting. If rolling shutter is not applied, then the rays are transformed according to the pose of the LiDAR sensor at the end of sweep time t_1, $\mathbf{T}_{\text{veh}}^{\text{world}}(t_1)$.

Motion Blur: Similarly, motion blur is applied through SLERP [18] interpolation on the pose of each dynamic actor by querying the continuous time trajectory for each actor at the start, middle, and end of the LiDAR sweep (three time-steps) and

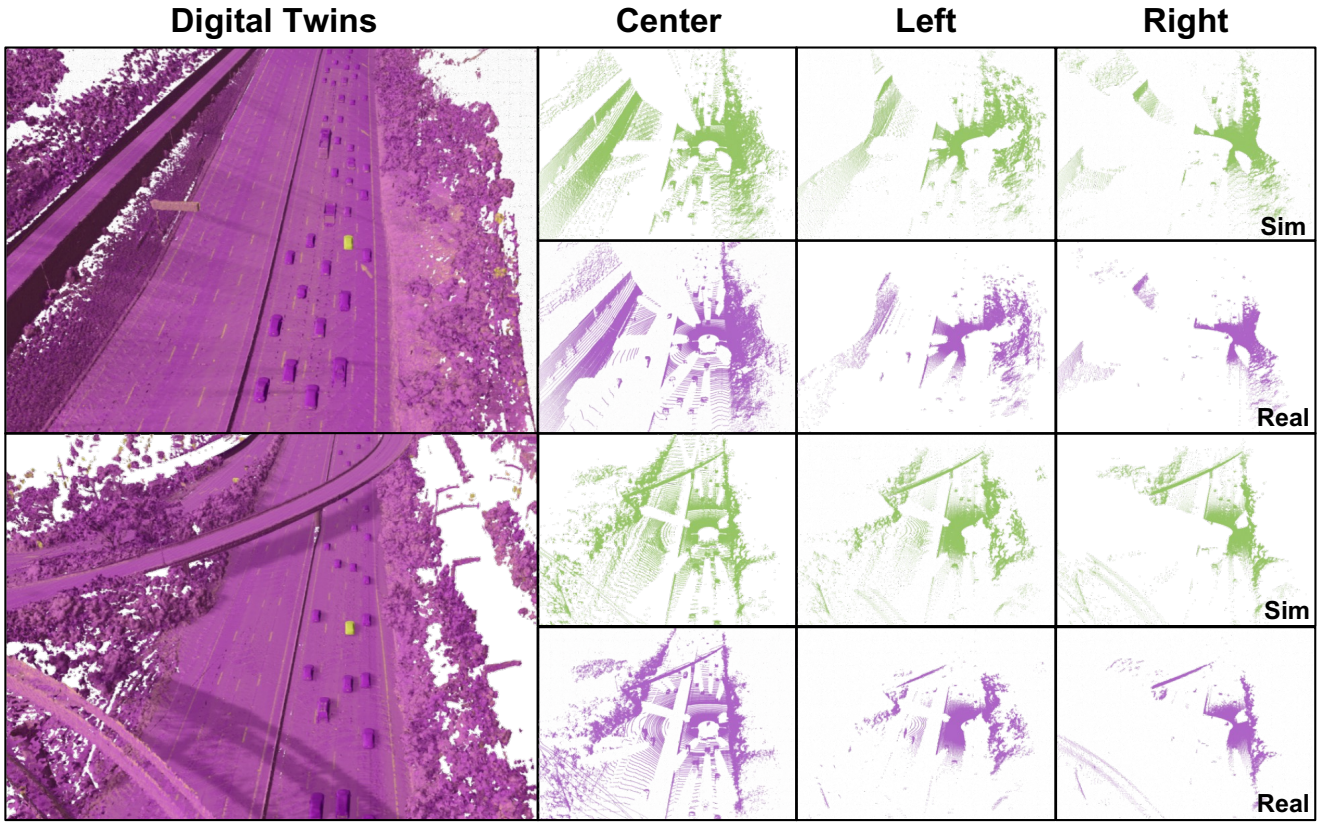

Figure 3. Virtual worlds for multi-LiDAR simulation, with examples of resulting rendered LiDAR on the right.

extracting $\mathbf{T}_{\text{actor}}^{\text{world}}(t_0)$, $\mathbf{T}_{\text{actor}}^{\text{world}}(t_{0.5})$, $\mathbf{T}_{\text{actor}}^{\text{world}}(t_1)$. The ray-tracer will then apply SLERP on the position of the geometry to get the geometry posed at time t when the ray $\mathbf{r}_t$ is casted into the scene. This is supported in off-the-shelf ray-tracers such as Optix [16].

Calibrated Intrinsics: For "Naive" intrinsics, we specify the elevation angle ϕ^i for each laser i by assuming linear spacing between $[\phi_{\text{min}}, \phi_{\text{max}}]$ for all 128 lasers. All three LiDAR have approximately 40° vertical field-of-view, so the elevation angle difference between lasers is approximately 0.3125°. We also set the azimuth offset $\theta_{\text{off}}^i = 0$. This is the typical calibration setting available in current LiDAR simulation systems [6]. In contrast, the calibrated intrinsics for each sensor is typically non-uniform and varies for each manfactured LiDAR sensor, similar to how each camera has a unique intrinsics matrix.

1.3. Applying the Oracle Pulse Effects

DropPoints Example: We show an example of performing DropPoints to enhance the simulated LiDAR point cloud with unreturned pulses. Please see Fig. 2 for a diagrammatic explanation. We project the simulated LiDAR and real LiDAR for the same scene into the spherical image, determine the unreturned pulses in the real LiDAR that do return in simulation, and remove those points in simulation via array-masking to generate the oracle-enhanced LiDAR. Similar operations are applied for $\text{AddPoints}, \text{AddEchoes}, \text{ModifyPoints}(\delta_{\text{lo}}, \delta_{\text{hi}})$.

2. Asset Creation Details

2.1. Surfel Creation for Base-LiDAR

As described in the main paper, surfel geometry assets were built from real LiDAR and used for the domain gap analysis of pulse effects and scanning effects. We describe this construction process in more detail. Following existing works [23, 14], we utilize LiDAR scans to build surfel meshes for the 3D world. To create virtual background with high realism and sufficient coverage, we collected driving data by driving over the same scene multiple times. Then, we associated multiple LiDAR sweeps to a common map coordinate system. Then we aggregate the LiDAR points across all the frames and perform automatic

dynamic point removal using [19]. For dynamic actors, we aggregate the LiDAR points per driving snippet (around 20s each) inside the bounding boxes in the object-centric coordinate. We then mirror the aggregated point cloud along the vehicle's heading axis and concatenate with the original point cloud for a more complete shape. Given the aggregated points, we then estimate per-point normals from 200 nearest neighbors with a radius of $20\mathrm{cm}$ and orient the normals upwards for flat ground reconstruction, outwards for more complete dynamic actors. We downsample the LiDAR points into $4\mathrm{cm}^3$ voxels and create per-point triangle faces (radius $5\mathrm{cm}$) according to the estimated normals.

Compensating Motion Blur for Foreground Actors: As shown in the main paper, motion blur occurs for every object with a non-zero velocity w.r.t. the sensor. We find that the relative motion between ego vehicle and dynamic actors can therefore lead to noisy LiDAR aggregation results. Therefore, for each point p inside the bounding box $\mathcal{B}$ (observed at n-th frame with observation timestamp t_p), we assume the actors are rigid and apply spherical linear interpolation (SLERP) [18] to map the points to the end of sweep time t_n using the actor label trajectory, where $t_{n-1} < t_p \leq t_n$.

Colored-ICP Alignment for Foreground Actors: To further refine the shape and account for errors in point cloud alignment for moving objects, we apply an iterative color-ICP (point-to-point) algorithm [3] before frame-wise aggregation, where we use the intensity value as feature. Following [20], we register the current LiDAR frame to the aggregated LiDAR points (with a window up to 10 frames). The parameter settings are shown in Table 1. We update point clouds only if ICP can increase fitness and reduce the RMSE for all inlier correspondences.

Parameter	Value	Comment
inlier threshold	0.3	discard the correspondence if their point-to-point distance is larger than this threshold
min points	100	apply color-ICP if the number of aggregated points is larger than this value
max iteration	30	maximum number of iterations that the color-ICP algorithm will perform
window size	10	window size aggregated LiDAR used for for registeration

Table 1. Color-ICP Registration Parameters.

Foreground Actor Retrieval: The original surfel assets for each actor can be directly used to simulate the same scenario in simulation as observed in the real world. For the analysis in Sec. 5.4, where CAD models or a combination of curated CAD models and surfel assets are leveraged, we retrieve the asset for each actor that has the closest bounding box size and same actor class.

Discussion on Surfel Representation: Surfel meshes are a popular choice for representing 3D driving scenes due to their efficiency and scalability in capturing surface geometry [23, 14]. Surfel meshes may also "bake" some material modeling and real-world noise in the geometry itself via holes in the mesh and blurry geometry. For example, meshes may capture the transparency of windows from certain observed viewpoints, or noisy observations of a vehicle's antenna. Moreover, the aggregation process may also accumulate the noise and introduce artifacts or inaccuracies during the LiDAR raycasting.

2.2. Asset bank with CAD models

We purchased over 120 artist-created CAD models from TurboSquid [1] for a wide range of rigid actors, such as vehicles, motorcycles, barriers and animals. We re-scale the CAD meshes to the real-world metric scale. In our CAD asset library, The purchased CAD assets are classified into ten categories, with cars being the most common at 51.05%, followed by barriers at 13.29%, truck tractor units at 9.79%, mini truck at 6.99%, animal at 5.59%, bus at 4.90% and construction vehicle at 4.90%.

2.3. Road-only Background Mesh

Similar to surfel creation, we aggregate the LiDAR points using multiple LiDAR scans and perform automatic dynamic point removal using the method proposed by Thomas et al. [19]. We then rasterize the LiDAR points into 2D height maps $H \times W \times 1$. Finally, we initialize a grid plane and query the z for all vertices (x, y) to obtain the road-only mesh. We note that road-only mesh is only accurate in the road region and cannot handle trees and other background items well.

Area	Parameter	Value
Detection	LiDAR RoI	$x = [-50, 200]$; $y = [-50, 50]$; $z = [0, 5]$
	LiDAR resolution	15cm
	Input LiDARs	3× 128-beam proprietary LiDARs
	LiDAR frequency	10Hz
	Input Sweeps	3×5 (500ms of data)
Prediction	History	0.5s @ 10Hz (5 history steps)
	Horizon	7s @ 2Hz (14 waypoints)

Table 2. Autonomy system parameters

2.4. Neural Surface Reconstruction for Background

Inspired by recent success in the implicit surface reconstruction and neural radiance field, we also consider the neural meshes reconstructed by the state-of-the-art approach UniSim [24] for our experiments. Different from surfel creation that are conducted with multiple data scan, UniSim takes one single pass with all the camera and LiDAR sequences, performs neural rendering to learn a 3D geometry representation, and then reconstructs triangle meshes using marching cubes. Specifically, it uses LiDAR points to initialize the 3D sparse voxels for efficient volume rendering. UniSim leverages popular multi-resolution feature grid and hash encoding [15] to predict the SDF values for each voxel. Compared to surfel aggregation, the geometry for UniSim is smoother and less noisy.

Figure 3 displays qualitative examples of reconstructed digital twins together with the simulated and real multi-LiDAR data. The depicted assets are all surfel-based.

3. Autonomy Details

In this section we provide additional details about the autonomy system under test in the main paper. Note that no parts of our analysis depend on specific details of the autonomy stack, besides (a) LiDAR input, and (b) intermediate perception and motion forecasting output.

Perception & Prediction: Our perception and prediction system takes as input multiple LiDAR sweeps from multiple spinning sensors and outputs a series of 2D bounding boxes with trajectory forecasts in bird's-eye view (BEV). We describe the key parameters of our autonomy system in Table 2.

The detection net is based on a two-stage variant of the PIXOR architecture propsed by [22]. LiDAR inputs are voxelized at the specified resolution. HD maps are also rasterized for the same RoI as the LiDAR, using a separate channel for each kind of geometry [4]. The information included in the HD maps consists in lane centerlines, lane boundaries, and shoulders. The rasterized map channels are concatenated with the input voxelized LiDAR.

The first stage of the detector consists in a ResNet backbone, followed by an FPN [13] neck. Features from the top 500 detections from the first stage are extracted using a 3×3 rotated ROI [21] and further refined using both RoI-level self-attention as well as cross-RoI attention. The final outputs of the second layer are computed with two MLPs–one for the classification score and one for box refinement.

The prediction net is goal-based and uses a lanegraph representation for its input [5], which also includes the HD map[1]. Detection features are passed to prediction through the actor features by pooling the respective detection's RoI features. Detection and prediction are trained jointly from scratch using a dataset which combines 80 labeled real data snippets with simulated snippets. All snippets are 20 seconds. We perform this straight-forward real+sim data augmentation to test the domain gap of the autonomy in a more realistic deployment setting where the autonomy is trained on both simulated and real data.

Planner: We employ a sampling-based planner [17] which optimizes the following costs: collision (avoid colliding with actors), burden (avoid forcing other actors to decelerate), headway (maintain a safety buffer ahead), comfort (minimize acceleration, jerk, and trajectory curvature), corridor (avoid crossing lane boundaries), solid lane boundary (prevent crossing

[1]Note that the HD map is provided separately to the detector and to the predictor, albeit using rasterization for the former and a lane graph representation for the latter.

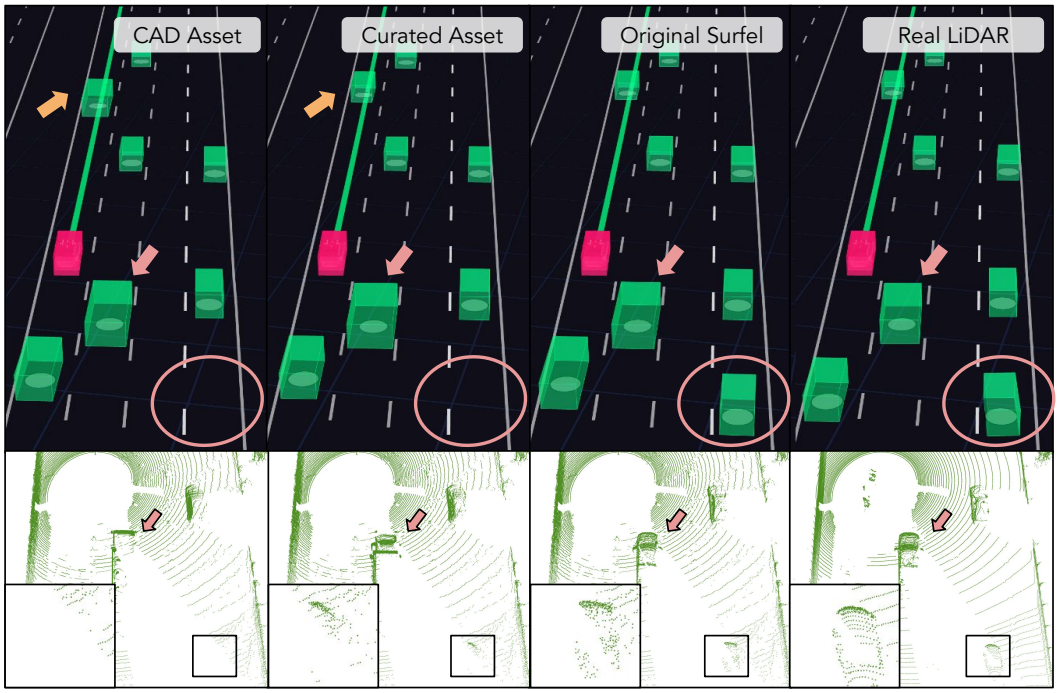

Figure 4. **Differences in domain gap due to the geometry discrepancy in foreground actors.** Different vehicle geometries, despite being similar size, can result in differences in domain gap compared to the Real LiDAR. Orange: curated hybrid assets are more diverse in geometry thus beneficial for more accurate localization; Pink: the autonomy is unable to detect the actor in the rightmost lane (highlighted by pink circle) when using CAD or curated hybrid assets. In contrast, the autonomy can detect that actor when using original surfel meshes, albeit with some localization discrepancies.

solid lane boundaries), speed limit, progress (along the trajectory), cross-track (keep close to centerline), and route (satisfy the desired high-level route, which is fixed in our case to the original trajectory as we are operating in open loop).

4. Additional Analysis

4.1. Detection Agreement at Different Ranges

We report bucketized detection agreement metrics for the same analysis in the main paper in Table 3. As reported in Sec. 5.2, AddPoints and AddEchoes help improve detection agreement especially for actors at long range. We note that row 8 in Table 3, which applies DropPoints and ModifyPoints$(\delta_{\text{lo}}, \delta_{\text{hi}})$ achieves detection agreement AP and Recall of 1.0 for $[0, 40]$ meters, and also achieves the lowest plan discrepancy metric. This indicates that these phenomena are important for perception performance of actors close to the SDV, which are likely actors of interest for motion planning.

4.2. Additional Point Cloud Metrics

In addition to autonomy domain gap metrics, we report LiDAR point cloud metrics using the correspondence established between real and simulated LiDAR with spherical image representation. Occupied pixels in the simulated LiDAR spherical image are compared pair-wise with pixels in the real LiDAR spherical image, and precision, recall, median L_1 error are reported in Table 4. As expected, applying ModifyPoints$(\delta_{\text{lo}}, \delta_{\text{hi}})$ from $[0, 200]$ reduces the L_1 error to 0 meters, while DropPoints and AddPoints improve precision and recall respectively with respect to real LiDAR. AddEchoes does not affect the point cloud metrics as they are only computed on the first return point of each occupied pixel.

4.3. Additional Pulse Phenomena Analysis

In Table 5 we report the domain gap for additional variations and combinations of LiDAR pulse phenomena affects. Specifically, we report the effect of ModifyPoints$(\delta_{\text{lo}}, \delta_{\text{hi}})$ for different error range thresholds. Applying ModifyPoints$(\delta_{\text{lo}}, \delta_{\text{hi}})$ to nearby points, such as $[0, 2]$ meters has a substantial reduction in domain gap, indicating that generating realistic LiDAR noise and accurate geometry reconstruction are key. We also observe domain gap improvements for modifying simulated points with

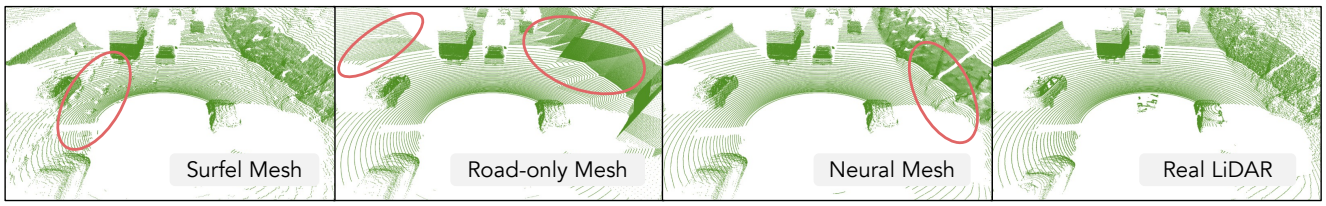

Figure 5. From Left to Right: Simulated LiDAR using Surfel Mesh (default), Road-only Mesh, and Neural Mesh compared to Real LiDAR. Note the smoother road for the Neural Mesh while preserving details, better matching the real LiDAR. The reduction in Planning Discrepancy error with this mesh type indicates the geometry reconstruction of the background plays a role in domain gap.

					Detection Agreement AP				**Detection Agreement Recall**			
#	DropP	AddE	AddP	ModP	[0, 40]	[40, 80]	[80, 150]	[150, 200]	[0, 40]	[40, 80]	[80, 150]	[150, 200]
1					0.95	0.83	0.60	0.35	0.95	0.86	0.69	0.50
2	✓				0.94	0.82	0.58	0.36	0.95	0.85	0.64	0.45
3		✓			0.95	0.84	0.64	0.39	0.96	0.87	0.72	0.55
4			✓		0.95	0.85	0.67	0.58	0.96	0.87	0.75	0.71
5		✓	✓		0.96	0.86	0.69	0.63	0.97	0.88	0.77	0.76
6				[0, 200]	0.98	0.93	0.77	0.55	0.98	0.94	0.83	0.68
7	✓		✓		0.95	0.84	0.67	0.65	0.96	0.86	0.72	0.71
8	✓			[0, 200]	1.00	0.98	0.90	0.62	1.00	0.98	0.91	0.68
9			✓	[0, 200]	0.98	0.94	0.82	0.80	0.99	0.95	0.87	0.89
10		✓	✓	[0, 200]	0.98	0.94	0.82	0.80	0.99	0.95	0.87	0.89
oracle	✓	✓	✓	✓	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 3. **LiDAR Pulse Phenomena Detection Agreement Metrics**: Bucketized by range for analysis with ray propagation effects such as unreturned pulses (**DropP**oints), multi-path (**AddE**choes), spurious points (**AddP**oints), and noisy points (**ModP**oints)

range errors of larger than 10 meters (row 20). This indicates that there are differences in occlusion modelling, or material modelling causing points with larger geometry errors. An example of a difference in material modelling causing a large range error is where the pulse may go through a transparent surface and return a point at an opaque surface further away, but in simulation the point is detected as a hit on the closer by surface according to the object geometry.

4.4. Additional Virtual World Creation Analysis

Fig. 4 shows that differences in domain gap can be attributed to the geometry discrepancies in foreground actors, such as vehicles with different geometries. Even when these geometries are of similar size, they can still lead to differences in the domain gap when compared to real LiDAR data. Specifically, we can observe that queried CAD assets are larger than original actor thus occlude the actor behind in the rightmost lane (see Fig. 4 pink circle). Querying the asset bank with the hybrid curated assets (CAD + surfel) result in better alignment in terms of geometry (orange arrow), however, the autonomy cannot detect the actor located in pink circle due to underwhelming LiDAR observation. Finally, using original surfel assets get the perfect match in the geometry but still cause some discrepancies in the detection outputs due to the geometry error.

Fig. 5, presents a visual comparison of three distinct simulated LiDAR datasets generated using Surfel Mesh, Road-only Mesh, and Neural Mesh, respectively. It is noteworthy that the Neural Mesh produces a smoother road surface while preserving intricate details, making it more closely resemble the real LiDAR data. The road-only mesh is also smoother compared to the surfel mesh and has a smaller planning discrepancy. However, it cannot model non-road regions with fine-grained details. It explains why road-only mesh produces larger detection and prediction domain gap compared to using surfel background.

We believe these findings underline the importance of accurate virtual world creation for reducing the domain gap in LiDAR simulation, and emphasize the need for further research into novel techniques for improved geometry reconstruction in both foreground and background.

5. Additional Autonomy Systems

We also investigate the impact of the various LiDAR oracles on two variants of the autonomy system under test described in the main paper. One version simply re-trains the object detector with enhanced data augmentation, while the other leverages a completely different paradigm consisting of nonparametric motion forecasting (i.e., semantic occupancy prediction).

					Left Mid-range LiDAR			Center Long-range LiDAR			Right Mid-range LiDAR		
#	DropP	AddE	AddP	ModP	Precision	Recall	L_1	Precision	Recall	L_1	Precision	Recall	L_1
1					0.79	0.85	0.27	0.96	0.95	0.26	0.75	0.85	0.22
2	✓				1.00	0.85	0.27	1.00	0.95	0.26	1.00	0.85	0.22
3		✓			0.79	0.85	0.27	0.96	0.95	0.28	0.75	0.85	0.22
4			✓		0.82	1.00	0.09	0.96	1.00	0.21	0.78	1.00	0.10
5		✓	✓		0.82	1.00	0.09	0.96	1.00	0.23	0.78	1.00	0.10
6				[0, 200]	0.79	0.85	0.00	0.96	0.95	0.00	0.75	0.85	0.00
7	✓		✓		1.00	1.00	0.09	1.00	1.00	0.21	1.00	1.00	0.10
8	✓			[0, 200]	1.00	0.85	0.00	1.00	0.95	0.00	1.00	0.85	0.00
9			✓	[0, 200]	0.82	1.00	0.00	0.96	1.00	0.00	0.78	1.00	0.00
10		✓	✓	[0, 200]	0.82	1.00	0.00	0.96	1.00	0.00	0.78	1.00	0.00
oracle	✓	✓	✓	✓	1.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00

Table 4. **LiDAR Pulse Phenomena Ray-Based Metrics**: for analysis with ray propagation effects such as unreturned pulses (**DropP**oints), multi-path (**AddE**choes), spurious points (**AddP**oints), and noisy points (**ModP**oints)

					Detection				Prediction	Planning
#	DropP	AddE	AddP	ModP	ΔAP ↓	ΔRecall ↓	DA AP ↑	DA Recall ↑	minADE ↓	PD@5s ↓
1					0.047	0.032	0.77	0.80	1.74	3.22
6				[0, 200]	0.052	0.041	0.88	0.90	0.98	1.80
16				[0, 2]	0.048	0.036	0.80	0.83	1.45	2.47
17				[0, 10]	0.046	0.036	0.82	0.85	1.24	2.21
18				[2, 10]	0.040	0.031	0.79	0.82	1.53	2.62
19				[2, 200]	0.051	0.043	0.84	0.87	1.21	2.33
20				[10, 200]	0.057	0.046	0.83	0.86	1.28	2.75
21	✓	✓			0.036	0.034	0.77	0.79	1.66	2.68
22	✓	✓		[0, 2]	0.026	0.025	0.79	0.81	1.36	2.30

Table 5. **LiDAR Pulse Phenomena Additional Analysis**: of ray propagation effects such as noisy points at different error ranges (**ModP**oints) as well as more combinations, such as with unreturned pulses (**DropP**oints) and multi-path (**AddE**choes).

Improved Data Augmentation. The original model is trained with global data augmentation (translation and rotation). We retrained the same detector architecture while also employing per-point noise augmentation by adding Gaussian offsets in 3D using $\sigma = 0.5\,\text{cm}$.

Instance-free Autonomy. The main autonomy system under test was an instance-based autonomy system that generated bounding box detections and trajectory forecasts. We further analyze an occupancy-based autonomy system similar to existing work in nonparametric perception such as UniAD [11] and Agro et al. [2] that uses the single long range central lidar to generate an instance-free perception and prediction representation that can be used for planning.

Results. We present the results of the additional autonomy experiments in Table 6. We only report planning discrepancy for the instance-free autonomy system. The trends are very similar to those observed in the main paper, Table 1. Point noise data augmentation results in similar or slightly reduced domain gap. More sophisticated data augmentation may improve results further [12]. Finally, while errors in the occupancy-based autonomy stack are generally higher than the more mature object-centric one, we note that the Oracle trends are similar. Point modification makes the biggest difference, and the top performing setting in terms of Planning Divergence is ModP + DropP in both cases. These results validate the generality of the conclusions from the main paper, emphasizing the importance of accurate geometry and material modeling.

6. Limitations and Future Directions

We believe our domain gap analysis is the first step towards better understanding the importance of different LiDAR pulse and scanning effects for an autonomy system under test. We note that our current analysis was performed with respect to the domain gap for a single autonomy system. Our analysis is general and can work for any autonomy system, and future work will include understanding how the domain gap varies for different types of autonomy systems, such as range-view based

					Improved Data Augmentation (Object-Based)						Occupancy
					Detection				Prediction	Planning	Planning
#	DropP	AddE	AddP	ModP	AP #	Recall#	DA AP "	DA Recall"	minADE #	PD@5s#	PD@5s#
1					0.033	0.025	0.79	0.81	1.89	3.07	15.58
2	3				0.030	0.034	0.76	0.79	1.99	3.14	15.44
3		3			0.045	0.036	0.80	0.83	1.65	2.47	15.58
4			3		0.042	0.035	0.83	0.86	1.45	3.02	15.01
5		3	3		0.050	0.044	0.85	0.87	1.32	2.15	15.01
6				3	0.051	0.043	0.88	0.90	0.90	1.49	5.65
7	3		3		0.024	0.018	0.82	0.84	1.49	2.95	14.53
8	3			3	0.009	0.010	0.93	0.94	0.40	0.69	4.28
9			3	3	0.053	0.047	0.92	0.93	0.85	1.35	4.70
10		3	3	3	0.053	0.047	0.92	0.93	0.85	1.35	4.70

Table 6. Analysis on Additional Autonomy Con gurations. The oracles are the same as in previous Tables, except we analyze two new autonomy system con gurations: the con guration from the main paper, except trained with improved data augmentation, and a completely different stack with occupancy (nonparametric) outputs. There are no object-centric metrics like precision and recall for the occupancy-based perception stack, as its nonparametric outputs are not amenable to them. Instead, we present just the planning divergence. Divergence is higher for the occupancy-based method as it only uses a single LiDAR, a lower quality background, and its hyperparameters have been tuned less.

perception models [7], separately trained perception and prediction modules, or end-to-end neural planners [26]. Additionally, the autonomy system under test used the LiDAR point cloud geometry to perceive the scene. Further analysis would include exploring autonomy systems that also leverage per-point intensity features as input and understand its impact on the domain gap for different LiDAR phenomena. We also note that our initial analysis was performed on dataset of 20 scenarios collected in canonical operating conditions. Future data collection and analysis will involve understanding the importance of pulse and scanning effects in more extreme weather operating conditions such as in fog, heavy rain and snow [10, 9], etc. We hope this rst analysis provides exciting future directions to further improving existing LiDAR simulation methods to account for these effects better and build better virtual world assets, and to make autonomy systems more robust to these LiDAR effects.

References

[1] Turbosquid. https://d8ngmj9xfkzvpqj0jfc39d8.roads.com/. Accessed: 2023-03-06. 4

[2] Ben Agro, Quinlan Sykora, Sergio Casas, and Raquel Urtasun. Implicit occupancy ow elds for perception and prediction in self-driving. In CVPR, 2023. 8

[3] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In Sensor fusion IV: control paradigms and data structures, volume 1611, pages 586–606. Spie, 1992. 4

[4] Sergio Casas, Wenjie Luo, and Raquel Urtasun. IntentNet: Learning to predict intention from raw sensor data. In CoRL, 2018. 5

[5] Alexander Cui, Sergio Casas, Kelvin Wong, Simon Suo, and Raquel Urtasun. Gorela: Go relative for viewpoint-invariant motion forecasting. In ICRA, 2023. 5

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In CoRL, 2017. 3

[7] Lue Fan, Xuan Xiong, Feng Wang, Naiyan Wang, and Zhaoxiang Zhang. Rangedet: In defense of range view for lidar-based 3d object detection. In ICCV, 2021. 9

[8] Jin Fang, Dingfu Zhou, Jingjing Zhao, Chulin Tang, Cheng-Zhong Xu, and Liangjun Zhang. Lidar-cs dataset: Lidar point cloud dataset with cross-sensors for 3d object detection. arXiv preprint arXiv:2301.12515, 2023. 2

[9] Martin Hahner, Christos Sakaridis, Mario Bijelic, Felix Heide, Fisher Yu, Dengxin Dai, and Luc Van Gool. LiDAR snowfall simulation for robust 3D object detection. In CVPR, pages 16364–16374, 2022. 9

[10] Martin Hahner, Christos Sakaridis, Dengxin Dai, and Luc Van Gool. Fog simulation on real LiDAR point clouds for 3D object detection in adverse weather. arXiv preprint arXiv:2108.05249, 2021. 9

[11] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, et al. Planning-oriented autonomous driving. In CVPR, 2023. 8

[12] Zhaoqi Leng, Guowang Li, Chenxi Liu, Ekin Dogus Cubuk, Pei Sun, Tong He, Dragomir Anguelov, and Mingxing Tan. LiDAR augment: Searching for scalable 3D LiDAR data augmentations. In ICRA, pages 7039–7045. IEEE, 2023. 8

[13] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In CVPR, 2017. 5