LiDAR-EVS: Enhance Extrapolated View Synthesis for 3D Gaussian Splatting with Pseudo-LiDAR Supervision

Yiming Huang[1,2⋆], Xin Kang[1⋆], Sipeng Zhang[1], Hongliang Ren[2], Weihua Zhang[1], Junjie Lai[1†]

[1] NVIDIA
[2] The Chinese University of Hong Kong
{yimingh,joeyk,sipengz,weihuaz,julienl}@nvidia.com

Abstract. 3D Gaussian Splatting (3DGS) has emerged as a powerful technique for real-time LiDAR and camera synthesis in autonomous driving simulation. However, simulating LiDAR with 3DGS remains challenging for extrapolated views beyond the training trajectory, as existing methods are typically trained on single-traversal sensor scans, suffer from severe overfitting and poor generalization to novel ego-vehicle paths. To enable reliable simulation of LiDAR along unseen driving trajectories without external multi-pass data, we present LiDAR-EVS, a lightweight framework for robust extrapolated-view LiDAR simulation in autonomous driving. Designed to be plug-and-play, LiDAR-EVS readily extends to diverse LiDAR sensors and neural rendering baselines with minimal modification. Our framework comprises two key components: (1) pseudo extrapolated-view point cloud supervision with multi-frame LiDAR fusion, view transformation, occlusion curling, and intensity adjustment; (2) spatially-constrained dropout regularization that promotes robustness to diverse trajectory variations in real-world driving. Extensive experiments demonstrate that LiDAR-EVS achieves SOTA performance on extrapolated-view LiDAR synthesis across three datasets, making it a promising tool for data-driven simulation, closed-loop evaluation, and synthetic data generation in autonomous driving. Code and demos are available at: https://lastbasket.github.io/LiDAR-EVS_page/.

Keywords: 3D Gaussian Splatting · LiDAR Simulation · Novel View Synthesis

1 Introduction

A central requirement for realistic closed-loop autonomous driving simulation [10, 21] is the ability to synthesize accurate LiDAR point clouds and photorealistic RGB images from novel viewpoints. While neural radiance fields (NeRF)-based methods [28, 34] first introduced neural rendering for joint camera and LiDAR

⋆ equal contribution, †corresponding author

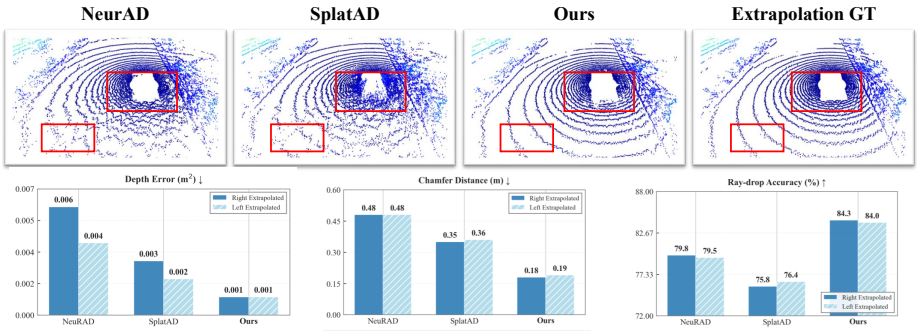

Fig. 1: LiDAR-EVS enables robust LiDAR synthesis for extrapolated views beyond the training trajectory. Given single-traversal data, our framework generates pseudo supervision for novel viewpoints, achieving accurate LiDAR simulation on extrapolation view, outperforming existing methods that overfit to the training path.

simulation, 3D Gaussian Splatting (3DGS) [12] offers high-fidelity LiDAR [3,11] and camera [33,37] rendering with significantly more promising real-time performance. In addition to superior rendering capabilities, the explicit 3D representation of 3DGS provides flexible controllability, enabling the reconstructed pedestrian and other dynamic actors in the scene to be edited [5,9]. These rendering and control abilities of 3DGS enable more scalable closed-loop evaluation and synthetic data generation, allowing autonomous driving systems to perform extensive validation across diverse scenarios before real-world deployment.

Although recent advances [9,11,36] have demonstrated substantial progress in applying 3D Gaussian Splatting (3DGS) to LiDAR simulation, extrapolated view LiDAR synthesis still remains a significant challenge [2]. Unlike dense, texture-rich cameras, LiDAR sensors produce sparse, geometry-focused point clouds lacking color and high-frequency details. Consequently, LiDAR simulation receives weaker supervision than RGB modalities, degrading generalization capability. Additionally, most existing 3DGS-based methods [3,9,11] are trained exclusively on single-traversal sensor data collected along the ego-vehicle's original path, resulting in severe overfitting to specific trajectories. Consequently, when evaluated on extrapolated views, these methods frequently suffer from pronounced artifacts, missing geometry, unrealistic intensity patterns, and degraded fidelity in sparse or distant regions, as shown in Fig. 1. While prior work [2] attempts to improve generalization ability through pre-trained diffusion priors, this approach introduces substantial computational overhead and strong external dependencies. With the pre-trained model, its capability heavily depends on the distribution of the training data, making it vulnerable to out-of-distribution extrapolated views from unseen scenarios. In addition, such prior-based models struggle to generalize to novel LiDAR settings due to differences in sensor intrinsics, surface reflectivity, and occlusion patterns.

To address these limitations, we introduce LiDAR-EVS (Extrapolated View Synthesis), a framework designed for robust extrapolated-view autonomous driving simulation using 3D Gaussian Splatting. We observe that by the geometric invariance in the world coordinate system, static LiDAR points can be fused across frames and re-projected from the original to the shifted poses. Inspired by the previous observation, we present our key idea: effective generalization to novel trajectories can be achieved through pseudo-LiDAR supervision combined with lightweight spatial regularization. Our framework consists of two major components. First, we introduce a pseudo extrapolated-view supervision pipeline that enriches the training distribution through the dedicated designed pseudo LiDAR point clouds. The curation of pseudo supervision includes multi-frame LiDAR fusion, view transformation, adaptive intensity adjustment, and occlusion-aware point curling. This generates plausible LiDAR observations for viewpoints beyond the training trajectory, providing direct supervision for the extrapolated view synthesis. Second, we propose spatially-constrained dropout regularization, which selectively perturbs Gaussian parameters during training to prevent overfitting to specific spatial configurations while preserving local geometric coherence. Together, these components form a plug-and-play solution that generalize on various LiDAR sensor setups and can be integrated seamlessly with all neural rendering pipelines.

Extensive experiments on large-scale autonomous driving datasets demonstrate that LiDAR-EVS achieves state-of-the-art performance on both interpolated and extrapolated novel-view synthesis. Notably, our method significantly outperforms existing approaches on LiDAR point cloud rendering for out-of-distribution viewpoints, with minimal computational overhead compared to standard 3DGS training. Our contributions are summarized as follows:

- We propose LiDAR-EVS, a plug-and-play framework that augments existing 3DGS-based renderer with robust extrapolated-view LiDAR synthesis.
- We design a sensor-agnostic pseudo-LiDAR curation pipeline that produces high-fidelity extrapolated-view LiDAR data via multi-frame fusion, view transformation, occlusion-aware curling, and intensity adjustment.
- We introduce spatially constrained dropout regularization that perturbs Gaussians according to LiDAR-specific viewing geometry, substantially improving the robustness of extrapolated-view synthesis.
- Extensive experiments on three public datasets demonstrate state-of-the-art results on extrapolated-view LiDAR rendering.

2 Related Works

Neural Rendering for Autonomous Driving Simulation. Simulation has become indispensable for autonomous driving development, enabling safe testing of perception and planning systems [7, 21]. Traditional graphics-based simulators [7, 22] rely on artist-created assets and physics engines, which struggle to achieve the photorealism required for sensor simulation. Recent approaches [21] leverage neural rendering to reconstruct real-world scenes directly from sensor

data. Neural Radiance Fields (NeRF) [17] have been explored for autonomous driving scene reconstruction [28,29,34], but their volumetric rendering is computationally expensive for large-scale dynamic environments. 3D Gaussian Splatting (3DGS) [12] has emerged as a compelling alternative, offering real-time rendering through explicit 3D Gaussian primitives. Several works have adapted 3DGS for driving scenarios [5, 37], which combine 3DGS with deformable models for dynamic objects [4], introduce compact representations for large-scale scenes [15], and [5, 33] combine multi-camera for surround-view synthesis. However, these methods primarily focus on RGB rendering and often assume interpolated viewpoints, leaving extrapolated-view synthesis underexplored.

LiDAR Simulation and Synthesis. Accurate LiDAR simulation is critical for validating perception systems that rely on geometric cues. Physics-based LiDAR simulators [8, 25] model ray casting and material reflectance but require detailed scene geometry and are computationally intensive. Learning-based approaches [34] have gained traction for their ability to capture real-world sensor characteristics from data. Early neural LiDAR synthesis methods employed point cloud completion [32] or range image generation [38], but these operate in 2D projection space, losing 3D geometric consistency. More recent works integrate LiDAR into neural scene representations, [27, 28] extend NeRF for range image rendering, while [5, 9] jointly optimize camera and LiDAR observations within 3DGS frameworks. Despite these advances, existing methods typically train on single-traversal data, leading to overfitting and poor generalization when simulating novel ego-vehicle paths, a requirement for closed-loop simulation where the vehicle may deviate from the recorded trajectory.

Generalization in Novel-View Synthesis. Generalization to unseen viewpoints remains a fundamental challenge in neural rendering. For interpolated views within the training camera trajectory, various regularization techniques have been proposed, including depth smoothness constraints [24], confidence guidance [35], and geometric priors [23]. However, extrapolated-view synthesis rendering from viewpoints outside the training distribution presents significantly greater difficulty. Several strategies have been explored to enhance extrapolation capabilities. Pre-trained depth estimation networks can provide geometric guidance [16, 23], but introduce domain shift issues and computational overhead. Data augmentation through camera pose perturbation [30] enriches training distributions but lacks explicit supervision for the extrapolated domain. Generative priors [2, 14, 26] have also shown promise but require complex procedures. For LiDAR specifically, the discrete, sparse nature of point clouds and the viewpoint-dependent occlusion patterns make direct application of camera-oriented techniques insufficient. Our work addresses this gap through pseudo-LiDAR supervision that generates explicit training signals for extrapolated views, combined with regularization tailored to the geometric characteristics of LiDAR sensing.

3 Preliminary

3D Gaussian Splatting has been utilized in the autonomous driving simulation [9] with the scene representation as a set of properties: mean $\boldsymbol{\mu} \in \mathbb{R}^3$, opacity $o \in (0, 1)$, covariance matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{3\times3}$, feature vector $\mathbf{f}^{\text{rgb}} \in \mathbb{R}^3$ and $\mathbf{f} \in \mathbb{R}^{D_\mathbf{f}}$. The covariance $\boldsymbol{\Sigma}$ depends on scale $\mathbf{S} \in \mathbb{R}^3$ and quaternion $\mathbf{q} \in \mathbb{R}^4$, $\mathbf{f}^{\text{rgb}}$ is the color feature, and $\mathbf{f}$ is used for both view-dependent effects and LiDAR properties. For camera rendering, the features are rasterized with α-blending:

$$[\mathbf{F}_\mathbf{p}^{\text{rgb}}, \mathbf{F}_\mathbf{p}] = \sum_i [\mathbf{f}_i^{\text{rgb}}, \mathbf{f}_i] \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

where $\mathbf{p}$ is the pixel position, α value is computed as in [9]. The final camera colors are then adjusted by the view-dependent feature with a small MLP network. For LiDAR rendering, the mean $\boldsymbol{\mu} = [x, y, z]$ is transformed into the spherical coordinate $\boldsymbol{\mu}^S = \left[\phi, \omega, r\right]$ where $\phi = \arctan2(y, x)$, $\omega = \arcsin(z/r)$, and $r = \sqrt{x^2 + y^2 + z^2}$. The covariance matrix is transformed using the Jacobian, $\boldsymbol{\Sigma}^S = \mathbf{J}^S \boldsymbol{\Sigma}^L (\mathbf{J}^S)^\top$, where $\boldsymbol{\Sigma}^L$ is the covariance in LiDAR coordinates and $\mathbf{J}^S$ and $\boldsymbol{\Sigma}^S$ are the spherical Jacobian and covariance matrix as in [9]. During rasterization, $\mathbf{f}_i$ are α-blended similarly to Eq. (1) to produce the LiDAR range map and blended feature, where the feature are concatenated with ray directions, and decoded via a small MLP to predict intensity and ray-drop probability.

4 LiDAR-EVS

We present LiDAR-EVS (Fig. 2), a plug-and-play framework for robust extrapolated-view LiDAR synthesis using 3DGS. It seamlessly integrates with arbitrary LiDAR sensors and existing neural rendering pipelines.

4.1 Sensor Agnostic Pseudo LiDAR Curation

To enable robust extrapolated-view synthesis, we generate pseudo LiDAR observations for viewpoints beyond the training trajectory. Our curation pipeline enriches the training distribution through multi-frame fusion, geometric transformation, and physically-informed augmentation, providing explicit supervision for novel ego-vehicle paths without requiring additional data collection.

Multi-Frame LiDAR Fusion. We observe that when the viewpoint changes to the extrapolated pose, some of the scene points are occluded, resulting in missing areas. To provide comprehensive coverage of the underlying geometry at the extrapolated viewpoint, we aggregate LiDAR measurements from N temporally adjacent frames to densify the point cloud, explicitly focusing on static scene elements. Specifically, we remove dynamic objects such as vehicles, pedestrians, and cyclists from each frame with SAM2 [20], ensuring that only geometrically invariant static structures such as road surfaces and buildings are fused for the

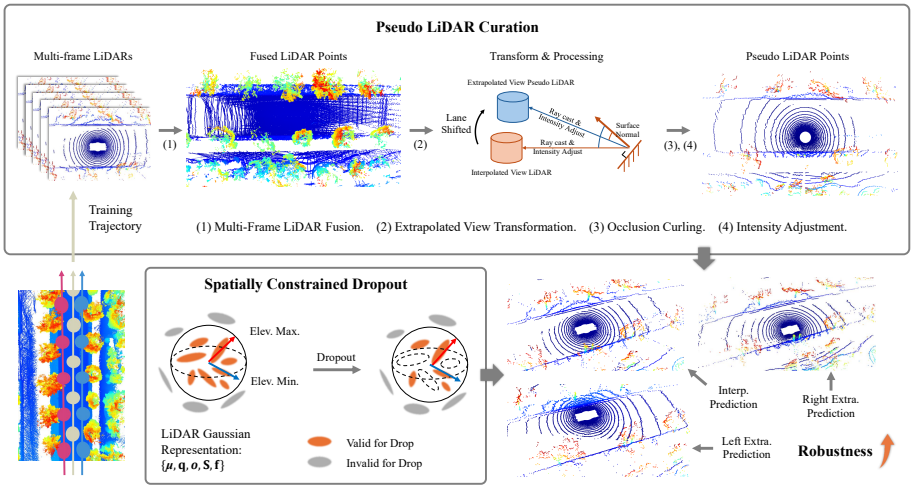

Fig. 2: LiDAR-EVS Pipeline. Our framework consists of two key modules: Pseudo LiDAR Curation and Spatially Constrained Dropout. Pseudo LiDAR curation include the following steps: (1) Multi-frame fusion, (2) Extrapolated view transformation, (3) Occlusion curling, (4) Intensity adjustment. With the proposed framework, we can optimize the Gaussian scene representation to achieve robust LiDAR synthesis for both interpolated and extrapolated view rendering.

additional frames. For each frame i, we transform LiDAR points into the world frame via the ego-pose $\mathbf{T}_i \in \mathrm{SE}(3)$. Let $\mathcal{P}_t$ denote the full point cloud at the current frame (including dynamic objects), and $\mathcal{P}_i^{\text{static}}$ denote the static subset at other frames. We construct a fused point cloud as

$$\mathcal{P}^{\text{fused}} = \mathbf{T}_t^{-1}\mathcal{P}_t \cup \left(\bigcup_{i \in \{1,\ldots,N\}\setminus\{t\}} \mathbf{T}_i^{-1}\mathcal{P}_i^{\text{static}} \right), \tag{2}$$

which preserves dynamic content from the current frame while densify static structures using temporally adjacent observations.

Extrapolated View Transformation. Exploiting geometric invariance, static LiDAR points maintain consistent world-coordinate positions across frames. We warp the fused static point cloud to target extrapolated poses outside the training trajectory, generating explicit pseudo-LiDAR supervision for novel viewpoints. Given a target extrapolated pose $\mathbf{T}^{\text{extra}}$, we transform the fused point cloud to the novel viewpoint: $\mathcal{P}^{\text{extra}} = \mathbf{T}^{\text{extra}}\mathcal{P}^{\text{fused}}$, yielding geometrically consistent point positions for the target view. Extrapolated poses are sampled by perturbing the training trajectory with lane-width shifts of autonomous driving scenarios, providing direct supervision for the unseen extrapolated viewpoints.

Algorithm 1 LiDAR Occlusion Curling

Require: Point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$, range map height h, width w
Initialize occlusion mask $\mathbf{m}_{\text{occ}} \leftarrow \mathbf{0}^N$
Spherical projection: $\mathcal{P} \mapsto (\theta, \phi, r)$
Range map coords: $(\theta, \phi, r) \mapsto (u, v)$
Valid FoV mask: $\mathbf{m}_{\text{FoV}} \leftarrow (v \in [0, h)) \wedge (u \in [0, w))$
$\mathbf{uv} \leftarrow \text{round}([v, u])$
$\mathbf{m}_{\text{cast}} = \mathit{Raycast}(\mathcal{P}, \mathbf{uv})$ ▷ *Raycast* computes the nearest hit along ray direction
$\mathbf{m}_{\text{occ}}[\mathbf{m}_{\text{cast}}] \leftarrow 1$
return $\mathcal{P} \odot (\mathbf{m}_{\text{occ}} \wedge \mathbf{m}_{\text{FoV}})$

Occlusion Curling. Although multi-frame fusion can significantly densify the point cloud on the new extrapolated view, some of the redundant points become occluded or dis-occluded under viewpoint change and require special handling. To eliminate these gaps, we apply the occlusion curling based on the LiDAR-specific parameters as shown in Algorithm 1. Intuitively, we retain only the nearest hit along each LiDAR ray from the extrapolated pose, and discard farther points that would be self-occluded, thus avoiding unrealistic 'see-through' geometry at extrapolated views. We first transform all points into the sensor space and convert to the LiDAR spherical coordinate, then we perform ray casting from the extrapolated view pose $\mathbf{T}^{\text{extra}}$ to identify occluded points in the range-map. Based on the geometry relationship for points on the same ray, we remove those further outliers and preserve those casted points within the sensor visibility constraints. This prevents overfitting to additional fused point clouds while maintaining realistic point cloud density. The curated pseudo LiDAR $\tilde{\mathcal{P}}^{\text{extra}} = \{(\mathbf{p}_j, I_j)\}_{j=1}^{M}$ serves as supervision signal for training the 3DGS representation, with each point providing position $\mathbf{p}_j \in \mathbb{R}^3$ and intensity $I_j \in \mathbb{R}$ targets for the extrapolated viewpoint.

Intensity Adjustment. LiDAR intensity measurements depend on surface reflectance, incidence angle, and sensor-to-point distance. To model intensity variation under viewpoint change, we first estimate surface normals for the fused point cloud using nearest neighbor search. Given the original pose $\mathbf{T}^{\text{ori}}$ and extrapolated pose $\mathbf{T}^{\text{extra}}$, we compute viewing ray directions $\mathbf{r}_{\text{ori}}$ and $\mathbf{r}_{\text{extra}}$ from sensor positions to each point. As inspired by [6], we apply the incident normalization to the original intensity I^{ori} to get the adjusted intensity I^{extra} based on the incidence angle between surface normal $\mathbf{n}$ and viewing direction:

$$I^{\text{extra}} = I^{\text{ori}} \frac{\mathbf{n} \cdot \hat{\mathbf{r}}_{\text{extra}}}{\mathbf{n} \cdot \hat{\mathbf{r}}_{\text{ori}}}, \tag{3}$$

where $\hat{\mathbf{r}}_{\text{extra}}, \hat{\mathbf{r}}_{\text{ori}}$ denotes normalized ray direction for the original and extrapolated view. Note that we omit the distance correction factor, as the LiDAR's distance coefficient is extremely small and negligible. For numerical stability and realism, we clamp the predicted intensity to $[0, 1]$.

4.2 Spatially-Constrained Dropout Regularization

To prevent overfitting to specific trajectory configurations and enhance generalization to novel viewpoints, we introduce spatially-constrained dropout regularization inspired by [19]. Instead of uniformly perturbing all Gaussian parameters, our approach selectively drops Gaussians based on their spatial relationship to the sensor, preserving geometric coherence in critical regions while encouraging robustness to viewpoint variations. Given Gaussian means $\boldsymbol{\mu} \in \mathbb{R}^{N\times 3}$ and sensor pose $\mathbf{T} \in \mathbb{R}^{4\times 4}$, we first compute the observation view vector $\mathbf{v} = \boldsymbol{\mu} - \mathbf{T}_{:3,3}$ and distance $d = \|\mathbf{v}\|_2$. The normalized view direction $\hat{\mathbf{v}}$ then yields the elevation angle $\phi = \arcsin(\hat{v}_z) \in [-\pi/2, \pi/2]$. We define the region-of-interest (ROI) mask based on sensor-specific elevation bounds $[\phi_{\min}, \phi_{\max}]$ and distance threshold $d_{\max}$: $\mathbf{m}_{\text{ROI}} = (d \le d_{\max}) \wedge (\phi_{\min} \le \phi < \phi_{\max})$. Dropout is applied to the 3DGS within the ROI by the mask $\mathbf{m}_{\text{drop}}$ with dropout rate r_{drop}:

$$\mathbf{m}_{\text{drop}}[i] = \begin{cases} \mathbb{I}[u_i < r_{\text{drop}}] & \text{if } \mathbf{m}_{\text{ROI}}[i] = 1 \\ 0 & \text{otherwise} \end{cases}, \tag{4}$$

where $u_i \sim \mathcal{U}(0,1)$ is the uniform random sampling. This spatially-constrained mechanism ensures that: (1) distant and out-of-elevation Gaussians remain stable for background consistency during the optimization, (2) near-field regions are regularized to prevent trajectory overfitting. During inference, we follow [19] to compensate for the expected dropout effect to ensure consistent rendering quality. Given the same ROI in training, we scale their opacities by the expected retention probability:

$$\tilde{o}_i = \begin{cases} o_i \cdot (1 - r_{\text{drop}}) & \text{if } \mathbf{m}_{\text{ROI}}[i] = 1 \\ o_i & \text{otherwise} \end{cases}, \tag{5}$$

where o_i is the original opacity and r_{drop} is the training dropout rate. This scaling accounts for the expected attenuation from dropout regularization, ensuring that the Gaussians' density matches the training-time marginal distribution.

4.3 Training Strategy and Optimization

To prevent trajectory overfitting, we apply random lateral shifts with lane-width δ during training, randomly selecting either the left or right direction at each iteration. This ensures the augmented viewpoints correspond to realistic neighboring lane positions while providing balanced exposure to both sides of the training path. The shift transform updates LiDAR sensor pose:

$$\mathbf{T}^{\text{extra}} = \mathbf{T}\mathbf{T}^{\text{shift}}, \quad \text{where } \mathbf{T}^{\text{shift}} = \begin{bmatrix} \mathbf{I} & [0, \delta, 0]^\top \\ \mathbf{0} & 1 \end{bmatrix}. \tag{6}$$

With the transformed lane shifted pose $\mathbf{T}^{\text{extra}}$, the pseudo-LiDAR curation (Sec. 4.1) generates the corresponding shifted point cloud and projects to the range map for the shifted viewpoints supervision.

Following [9], we optimized the Gaussians with the following loss:

$$\begin{aligned}\mathcal{L} =&\lambda_r\mathcal{L}_1 + (1-\lambda_r)\mathcal{L}_{\text{SSIM}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{los}}\mathcal{L}_{\text{los}} + \\ &\lambda_{\text{inten}}\mathcal{L}_{\text{inten}} + \lambda_{\text{raydrop}}\mathcal{L}_{\text{BCE}} + \lambda_{\text{MCMC}}\mathcal{L}_{\text{MCMC}},\end{aligned} \tag{7}$$

where $\mathcal{L}_1$ and $\mathcal{L}_{\text{SSIM}}$ are L1 and SSIM losses for images. $\mathcal{L}_{\text{depth}}$ and $\mathcal{L}_{\text{inten}}$ are L2 losses for the LiDAR range map and intensity. $\mathcal{L}_{\text{los}}$ is a line-of-sight loss, $\mathcal{L}_{\text{BCE}}$ is a binary cross-entropy loss for ray-drop probability, and $\mathcal{L}_{\text{MCMC}}$ is the opacity and scale regularization used in [13].

5 Experiments

5.1 Settings

Datasets. We conduct all experiments on three public autonomous driving datasets with diverse scenarios and sensor configurations. nuScenes [1] provides driving scenes with 32-beam LiDAR and 6 cameras in urban environments. PandaSet [31] contains scenes with a 64-beam LiDAR and 6 cameras. For both nuScenes and PandaSet, we generate pseudo extrapolated views by shifting the trajectory laterally by $\pm 4m$ to simulate lane change maneuvers. Para-Lane [18] is a specialized dataset featuring parallel traversals of the identical road, providing real extrapolated views with both left-shifted and right-shifted trajectories relative to a reference path. We evaluate 5 scenes for each dataset. For each dataset, we hold out 50% of frames for testing and report metrics on both interpolated (original trajectory) and extrapolated (shifted trajectory) views as in [9].

Baselines and Metrics. We compare LiDAR-EVS against state-of-the-art neural rendering methods for autonomous driving simulation, which include RGB+LiDAR methods such as NeuRAD [28] with neural radiance fields, OmniRE [5], and SplatAD [9] with 3D Gaussian Splatting. In terms of LiDAR-only methods, we choose the state-of-the-art 3DGS-based LiDAR synthesis method LiDAR-GS [3]. We do not compare against LiDAR-GS++ [2] as its official implementation is not publicly available. Furthermore, our core contributions on extrapolated view synthesis are orthogonal to LiDAR-GS++ [2], meaning our method does not conflict with theirs. All baselines are retrained on our datasets using official implementations with default hyper-parameters and evaluated with identical metrics for fair comparison. For LiDAR evaluation, we report Depth Error (m^2) as median square error, Chamfer Distance (m) for geometric fidelity, Intensity RMSE, and Ray-drop Accuracy (%). For the Paralane dataset, we use multi-traversal LiDAR scans as ground truth to compute these metrics; for nuScenes and Pandaset, we use pseudo-LiDAR as ground truth for extrapolated views. For camera rendering evaluation, we report PSNR, SSIM, and LPIPS.

Implementation Details. We build LiDAR-EVS in PyTorch with SplatAD [9] as baseline and conduct all experiments on NVIDIA H20 GPUs. Unless otherwise specified, we adopt SplatAD [9] as our baseline and the same hyper-parameters

Table 1: Quantitative Results on the Para-Lane Dataset. Lane Right/Left Extra. indicates the extrapolated view is right/left-shifted with lane width. Inter. represents the interpolated view at original recorded trajectory.

Methods	Lane Right Extra. LiDAR			Lane Left Extra. LiDAR			Lane Inter. LiDAR			Lane Inter. Image		
	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑	PSNR ↑	SSIM ↑	LPIPS ↓
LiDAR-GS [3]	0.018	0.45	87.8	0.017	0.45	87.6	0.0026	0.29	92.8	-	-	-
NeuRAD [28]	0.006	0.48	79.8	0.004	0.48	79.5	0.0012	0.22	84.6	22.86	0.647	0.323
NeuRAD+Ours	0.002	0.25	85.2	0.002	0.25	85.2	0.0006	0.18	82.6	**22.88**	0.648	0.301
LiDAR-RT [36]	0.054	0.49	76.2	0.055	0.50	76.1	0.003	0.28	98.9	-	-	-
LiDAR-RT+Ours	0.004	0.28	78.0	0.005	0.32	77.8	0.001	0.28	**98.9**	-	-	-
GS-LiDAR [11]	0.008	0.56	93.5	0.010	0.56	94.4	0.003	0.36	98.6	-	-	-
GS-LiDAR+Ours	0.003	0.38	**94.2**	0.002	0.41	**95.8**	0.002	0.34	98.6	-	-	-
SplatAD [9]	0.003	0.35	75.8	0.002	0.36	76.4	**0.0005**	0.19	80.6	22.37	**0.692**	**0.192**
Ours	**0.001**	**0.18**	84.3	**0.001**	**0.19**	84.0	0.0007	**0.15**	87.7	22.22	0.691	0.295

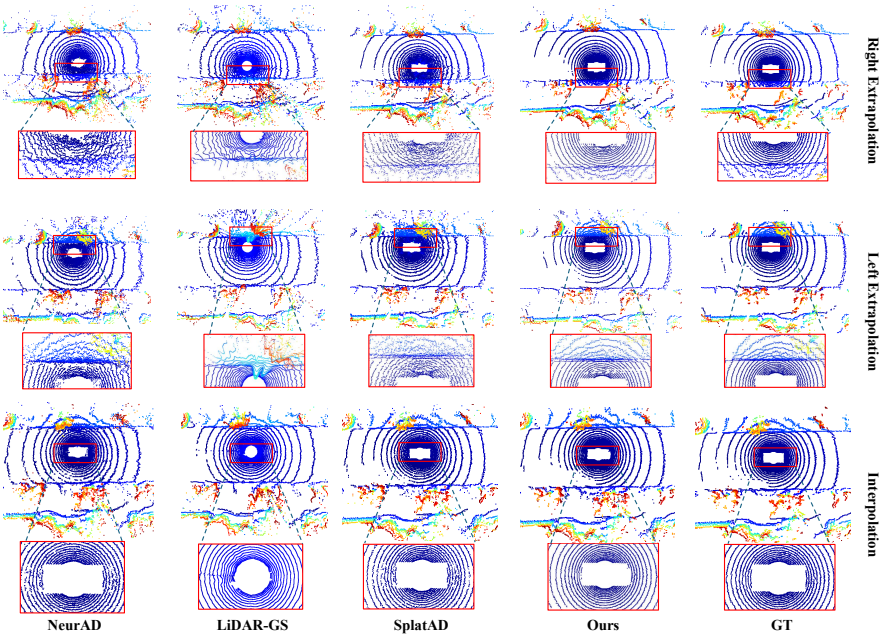

Fig. 3: Qualitative LiDAR Rendering Results on Para-Lane Dataset.

for optimization. For spatially-constrained dropout, we set the dropout rate to 0.5 for Para-Lane and 0.2 for others, which achieves the optimal balance between extrapolation robustness and geometric fidelity. Dataset-specific lane widths are configured according to regional driving standards: we use $\delta = 4m$ for nuScenes and PandaSet, and $\delta = 3m$ for Para-Lane. The number of multi-frame fusions is 10. The maximum LiDAR sensing distance $d_{\max}$ is set to 200 meters for all of nuScenes, PandaSet, and Para-Lane based on sensor specifications. Training takes approximately 1.5 hours per scene on a single H20 GPU with batch size 1

Table 2: Quantitative LiDAR Rendering Results on nuScenes Dataset.

Methods	Lane Right Extra. LiDAR				Lane Left Extra. LiDAR				Lane Inter. LiDAR			
	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑
NeuRAD [28]	1.611	1.23	0.087	51.9	3.567	1.44	0.094	52.1	0.018	0.42	0.042	92.7
OmniRE [5]	3.242	2.42	-	-	3.122	2.26	-	-	1.440	1.74	-	-
SplatAD [9]	1.399	2.07	0.084	57.8	1.314	1.34	0.091	57.9	**0.009**	0.41	**0.038**	**93.7**
Ours	**0.072**	**0.55**	**0.064**	**74.4**	**0.306**	**0.54**	**0.068**	**73.3**	0.011	**0.41**	0.041	92.7

Table 3: Quantitative LiDAR Rendering Results on Pandaset Dataset.

Methods	Lane Right Extra. LiDAR				Lane Left Extra. LiDAR				Lane Inter. LiDAR			
	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑
NeuRAD [28]	0.011	0.90	0.097	40.9	0.009	0.88	0.098	39.8	0.005	0.34	0.061	95.5
OmniRE [5]	3.071	2.43	-	-	2.465	3.26	-	-	2.467	3.15	-	-
SplatAD [9]	0.014	0.94	0.097	41.3	0.011	0.99	0.098	40.4	0.008	0.31	**0.058**	**97.2**
Ours	**0.011**	**0.36**	**0.067**	**79.1**	**0.011**	**0.38**	**0.069**	**79.0**	**0.008**	**0.31**	0.061	96.1

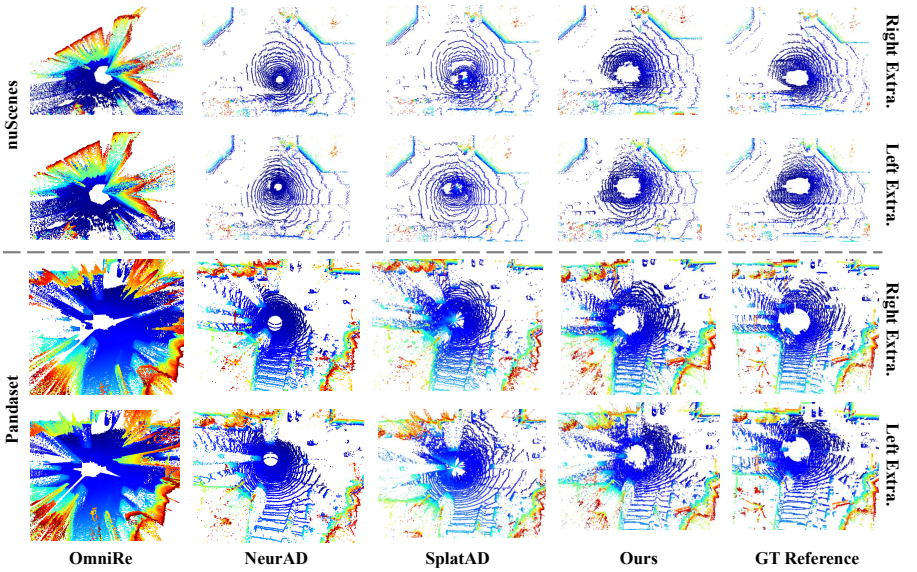

Fig. 4: Qualitative LiDAR Rendering Results on nuScenes and Pandaset.

for 30K iterations. More details for the computational overhead and additional experiments are in the supplement material.

5.2 Result Analysis

Extrapolated View LiDAR Rendering. We evaluate LiDAR-EVS against state-of-the-art methods on both real and pseudo extrapolated views across three datasets. Table 1 presents results on real extrapolations of the Para-Lane dataset with physically captured left/right-shifted trajectories. LiDAR-EVS achieves sig-

Table 4: Quantitative Image Rendering Results on nuScenes and Pandaset.

Methods	nuScenes			Pandaset		
	PSNR ↑	SSIM ↑	LPIPS ↓	PSNR ↑	SSIM ↑	LPIPS ↓
NeuRAD [28]	25.96	0.786	0.318	26.28	0.774	0.230
OmniRE [5]	22.77	0.706	**0.210**	23.63	0.706	0.223
SplatAD [9]	25.97	0.833	0.317	27.12	0.838	**0.178**
Ours	**26.11**	**0.837**	0.339	**27.27**	**0.841**	0.192

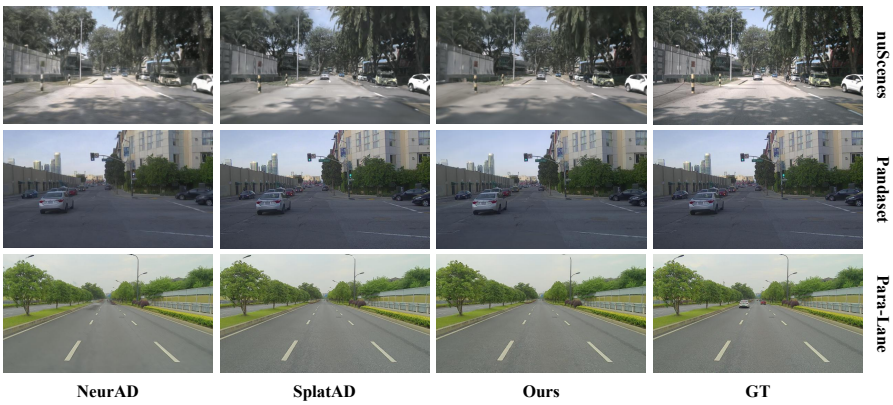

Fig. 5: Qualitative Image Rendering Results on nuScenes, Pandaset and Para-Lane Dataset.

nificant improvements over all methods on extrapolated-view LiDAR synthesis. Notably, our method reduces Chamfer Distance (CD) by 48% with $0.18m$ against $0.35m$ and Depth Error by 67% with $0.001m^2$ against $0.003m^2$ compared to SplatAD on right-shifted views, demonstrating strong generalization to real unseen trajectories. Built upon SplatAD, our approach maintains comparable RGB quality of 22.22 PSNR while dramatically improving LiDAR fidelity. Figure 3 provides qualitative comparisons on the Para-Lane dataset, where LiDAR-EVS generates significantly more complete and accurate point clouds, faithfully reconstructing distant structures, road boundaries, and fine-grained scene geometry. Crucially, these excellent results on real extrapolated trajectories validate the rationality and effectiveness of our pseudo LiDAR data. This strong empirical evidence justifies our use of pseudo LiDAR as the ground truth for evaluating extrapolation metrics on the remaining two datasets.

In Table 2, we present results on the pseudo-extrapolated LiDAR from nuScenes with $4m$ lateral trajectory shifts. LiDAR-EVS exhibits dramatic improvements on extrapolated views: Depth Error reduces from $1.399m^2$ of SplatAD to $0.072m^2$ on the right-extrapolated view and $1.314m^2$ to $0.306m^2$ on the left extrapolated view. Chamfer Distance decreases on right-shifted views from $2.07m$ to $0.55m$. These gains confirm that our method effectively bridges the domain gap to novel

Table 5: Pseudo LiDAR Ablation Results on Para-Lane Dataset.

Methods	Lane Right Extra. LiDAR			Lane Left Extra. LiDAR			Lane Inter. LiDAR		
	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑
w/o Pseudo	0.003	0.35	75.8	0.002	0.36	76.4	**0.0005**	0.19	80.6
Left Only	0.002	0.23	71.1	0.001	0.20	77.8	0.0007	0.16	83.5
Right Only	0.001	0.19	78.4	0.002	0.26	71.3	0.0007	0.16	83.5
Our Full	**0.001**	**0.18**	**84.3**	**0.001**	**0.19**	**84.0**	0.0007	**0.15**	**87.7**

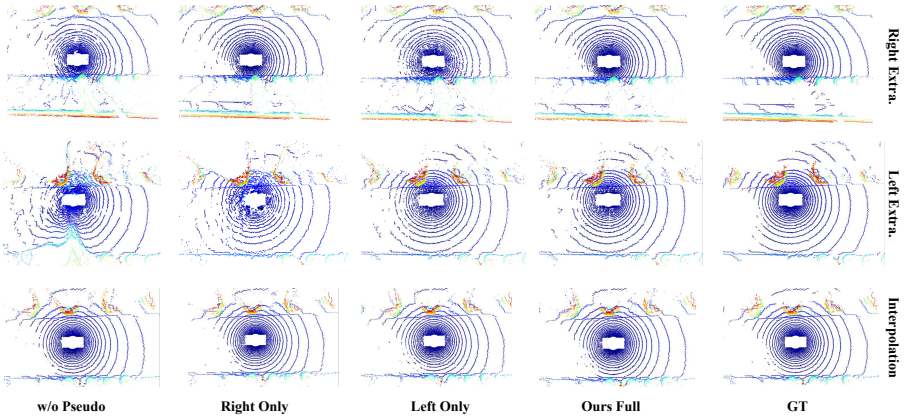

Fig. 6: Qualitative Results for Ablation Study on Para-Lane Dataset.

viewpoints. On interpolated views, our method matches SplatAD, validating that extrapolation enhancement does not compromise interpolation quality.

The result in Table 3 demonstrates consistent improvements on PandaSet with 64-beam LiDAR. LiDAR-EVS achieves 62% lower Chamfer Distance on right-shifted views from $0.94m$ to $0.36m$ and maintains strong intensity accuracy by 0.067 RMSE. The ray-drop accuracy improves from 41.3% to 79.1%, indicating better geometric understanding of sensor visibility patterns. These results generalize across different LiDAR configurations (32-beam, 64-beam), validating the sensor-agnostic design of our approach. Figure 4 confirms our improvements, showing that LiDAR-EVS produces more complete geometry with fewer artifacts compared to others across both 32-beam and 64-beam sensors.

Camera Rendering Quality. Table 4 summarizes camera rendering qualities across the nuScenes and PandaSet datasets. LiDAR-EVS achieves competitive or superior results compared to SplatAD PSNR 25.97 and ours 26.11 on nuScenes, SplatAD PSNR 27.12 and ours 27.12 on PandaSet, demonstrating that our LiDAR-focused enhancements benefit photorealistic rendering. Figure 5 provides qualitative comparisons, showing that our method renders RGB images with visual quality on par with or exceeding other methods.

Table 6: Dropout Regularization Ablation Results on Paralane Dataset.

Dropout Rate	Lane Right Extra. LiDAR			Lane Left Extra. LiDAR			Lane Inter. LiDAR		
	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑	Depth ↓	CD ↓	Raydrop ↑
0.0	0.0009	0.16	76.77	0.0010	0.18	75.77	0.0004	0.15	81.94
0.1	0.0009	0.16	78.26	0.0010	0.18	77.60	0.0004	0.15	83.33
0.2	0.0009	0.16	79.76	0.0011	0.18	79.13	0.0005	0.15	84.74
0.3	0.0010	0.16	81.70	0.0011	0.18	80.90	0.0005	0.15	86.07
0.4	0.0011	0.16	83.72	0.0011	0.18	82.76	0.0006	0.15	87.01
0.5	0.0012	0.16	85.48	0.0011	0.18	85.01	0.0006	0.15	87.80
0.6	0.0014	0.17	86.56	0.0014	0.18	86.19	0.0008	0.16	88.15
0.7	0.0018	0.19	87.31	0.0021	0.21	87.12	0.0011	0.18	88.09
0.8	0.0039	0.25	87.44	0.0049	0.28	87.24	0.0029	0.24	87.83

5.3 Ablation Study

We conduct comprehensive ablations to validate the contribution of each component in LiDAR-EVS. All experiments are on Para-Lane with real extrapolated views to ensure rigorous evaluation. In the supplementary material, we provide extra experiments by applying our method to other baselines and ablation experiments for the multi-frame fusion and intensity adjustment.

Pseudo LiDAR Supervision. Table 5 and Figure 6 investigate the impact of pseudo-LiDAR supervision strategies. The baseline without pseudo-supervision (w/o Pseudo) corresponds to standard SplatAD training, showing poor extrapolation performance with CD $0.35m$, Depth $0.003m^2$ on right-shifted views. Adding unidirectional supervision, either only the left-shifted pseudo LiDAR or the right-shifted pseudo LiDAR improves performance on the corresponding shift direction. For example, training with left-shifted pseudo-LiDAR achieves strong left-shift results of CD $0.20m$ but is limited on right-shifted views with CD of $0.23m$, revealing directional bias in unilateral training. Our full strategy with random right or left direction shifted pseudo-LiDAR supervision achieves the best overall performance on both extrapolated directions and the interpolated view with balanced ray-drop accuracy. This confirms that diverse directional exposure is critical for symmetric generalization, preventing the model from overfitting to specific trajectory patterns.

Spatially Constrained Dropout. Table 6 shows the dropout rate impact on Para-Lane. Without dropout, extrapolation suffers a degraded ray-drop of 76.77% / 75.77%. While increasing the dropout rates improves extrapolation: 79.76% / 79.13% at 0.2, 83.72% / 82.76% at 0.4, and 85.48% / 85.01% at 0.5, dropout rates beyond 0.5 plateau result in degraded geometry, where the Depth will be decreased to $0.0014m^2 - 0.0049m^2$. We use the dropout rate 0.5 as final, maximizing extrapolation robustness while maintaining interpolation fidelity.

Table 7: Different Iteration Setup on nuScenes Dataset.

	Depth ↓	CD ↓	Intensity ↓	Raydrop ↑	Training (H) ↓
SplatAD (30K)	0.009	0.41	0.038	93.7	1.2
SplatAD (40K)	0.009	0.40	0.038	93.7	1.45
Ours (30K)	0.011	0.41	0.041	92.7	0.85
Ours (40K)	0.009	0.40	0.038	93.5	1.1

Table 8: Runtime Analysis on Para-Lane Dataset.

1 × H20	Preprocess (H)↓	Training (H)↓	Total (H)↓
SplatAD	-	3.84	3.84
SplatAD+Ours	0.3	**1.69**	**1.99**

Training Iterations Trade-off on Interpolation-Extrapolation. As shown in Table 7, pseudo-extrapolation supervision shifts capacity toward unseen views under the default 30K iterations setting. With 40K iterations, our approach achieves near Depth, CD, and Intensity compared to SplatAD, confirming that the trade-off is not a fundamental limitation but a budget allocation.

Runtime breakdown for pseudo-curation and training. As in the Table 8, our method reduce the total overhead. The pseudo-curation uses only 0.3 hours on 190 frames, where the processing time for each frame are: Segmentation(0.5s), Multi-frame fusion(0.3s), Projection(1.5s), Occlusion curling(0.06s), Intensity Adjustment(1.0s), Saving(2.3s).

6 Conclusion

We present LiDAR-EVS, a lightweight framework for robust extrapolated-view LiDAR synthesis for autonomous driving simulation. Our pipeline produces high-quality LiDAR renderings for viewpoints beyond the original sensor trajectory by combining pseudo-LiDAR supervision with spatially constrained dropout. Extensive experiments on three public datasets show state-of-the-art extrapolated-view performance: LiDAR-EVS significantly reduces Chamfer Distance and depth error compared to prior methods while preserving competitive camera-rendering quality. Its modular, plug-and-play design enables seamless integration with a variety of LiDAR sensors and neural rendering baselines. Future work will focus on enforcing temporal consistency for dynamic objects. LiDAR-EVS thus provides a practical, scalable foundation for data-driven driving simulation, helping to bridge recorded data and closed-loop evaluation needs.

Acknowledgments This work was supported by the Ministry of Science and Technology (MOST) of China Key Project 2025YFE0122500, NSFC/RGC Joint Research Scheme (N_CUHK420/22).

References

1. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: CVPR. pp. 11621–11631 (2020)
2. Chen, Q., Liu, J., Xie, R., Tang, T., Du, S., Zhao, Y., Huo, Y., Yang, S.: Lidar-gs++: Improving lidar gaussian reconstruction via diffusion priors. arXiv preprint arXiv:2511.12304 (2025)
3. Chen, Q., Yang, S., Du, S., Tang, T., Xie, R., Chen, P., Huo, Y.: Lidar-gs: Real-time lidar re-simulation using gaussian splatting. arXiv preprint arXiv:2410.05111 (2024)
4. Chen, Y., Gu, C., Jiang, J., Zhu, X., Zhang, L.: Periodic vibration gaussian: Dynamic urban scene reconstruction and real-time rendering. International Journal of Computer Vision **134**(3), 83 (2026)
5. Chen, Z., Yang, J., Huang, J., de Lutio, R., Esturo, J.M., Ivanovic, B., Litany, O., Gojcic, Z., Fidler, S., Pavone, M., Song, L., Wang, Y.: Omnire: Omni urban scene reconstruction. In: The Thirteenth International Conference on Learning Representations (2025)
6. Dai, W., Chen, S., Huang, Z., Xu, Y., Kong, D.: Lidar intensity completion: Fully exploiting the message from lidar sensors. Sensors **22**(19), 7533 (2022)
7. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
8. Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., Yang, R.: Augmented lidar simulator for autonomous driving. IEEE Robotics and Automation Letters **5**(2), 1931–1938 (2020)
9. Hess, G., Lindström, C., Fatemi, M., Petersson, C., Svensson, L.: Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 11982–11992 (2025)
10. Hu, X., Li, S., Huang, T., Tang, B., Huai, R., Chen, L.: How simulation helps autonomous driving: A survey of sim2real, digital twins, and parallel intelligence. IEEE Transactions on Intelligent Vehicles **9**(1), 593–612 (2023)
11. Jiang, J., Gu, C., Chen, Y., Zhang, L.: Gs-lidar: Generating realistic lidar point clouds with panoramic gaussian splatting. In: International Conference on Learning Representations. pp. 1–15 (2025)
12. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3d gaussian splatting for real-time radiance field rendering. TOG **42**(4), 139–1 (2023)
13. Kheradmand, S., Rebain, D., Sharma, G., Sun, W., Tseng, Y.C., Isack, H., Kar, A., Tagliasacchi, A., Yi, K.M.: 3d gaussian splatting as markov chain monte carlo. In: NeurIPS (2024), `https://openreview.net/forum?id=UCSt4gk6iX`
14. Kong, H., Yang, X., Wang, X.: Generative sparse-view gaussian splatting. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 26745–26755 (2025)
15. Kulhanek, J., Rakotosaona, M.J., Manhardt, F., Tsalicoglou, C., Niemeyer, M., Sattler, T., Peng, S., Tombari, F.: LODGE: Level-of-detail large-scale gaussian splatting with efficient rendering. In: The Thirty-ninth Annual Conference on Neural Information Processing Systems (2025), `https://openreview.net/forum?id=Iqu63cYI3z`
16. Li, W., Zhang, Z., Ye, Z., Liu, S., Qiao, P.: Regularizing sparse input 3d gaussian splatting with geometry priors. In: International Conference on Neural Information Processing. pp. 123–137. Springer (2024)

17. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
18. Ni, Z., Du, S., Hou, Z., Wu, C., Yang, S.: Para-lane: Multi-lane dataset registering parallel scans for benchmarking novel view synthesis (2025), https://arxiv.org/abs/2502.15635
19. Park, H., Ryu, G., Kim, W.: Dropgaussian: Structural regularization for sparse-view gaussian splatting. In: Proceedings of the computer vision and pattern recognition conference. pp. 21600–21609 (2025)
20. Ravi, N., Gabeur, V., Hu, Y.T., Hu, R., Ryali, C., Ma, T., Khedr, H., Rädle, R., Rolland, C., Gustafson, L., et al.: Sam 2: Segment anything in images and videos. arXiv preprint arXiv:2408.00714 (2024)
21. Ren, X., Lu, Y., Cao, T., Gao, R., Huang, S., Sabour, A., Shen, T., Pfaff, T., Wu, J.Z., Chen, R., Kim, S.W., Gao, J., Leal-Taixe, L., Chen, M., Fidler, S., Ling, H.: Cosmos-drive-dreams: Scalable synthetic driving data generation with world foundation models. arXiv preprint arXiv:2506.09042 (2025)
22. Shah, S., Dey, D., Lovett, C., Kapoor, A.: Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In: Field and service robotics: Results of the 11th international conference. pp. 621–635. Springer (2017)
23. Shih, M.L., Chen, Y.H., Liu, Y.L., Curless, B.: Prior-enhanced gaussian splatting for dynamic scene reconstruction from casual video. In: Proceedings of the SIGGRAPH Asia 2025 Conference Papers. pp. 1–13 (2025)
24. Song, M., Lin, X., Zhang, D., Li, H., Li, X., Du, B., Qi, L.: D^2gs: Depth-and-density guided gaussian splatting for stable and accurate sparse-view reconstruction (2025), https://arxiv.org/abs/2510.08566
25. Tallavajhula, A., Mericli, C., Kelly, A.: Off-road lidar simulation with data-driven terrain primitives. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 7470–7477. IEEE (2018)
26. Tan, K., Shen, Y., Zhu, H., Zhan, Z., Zhao, S., Tu, M., Luo, H., Sun, H., Wang, B., Chen, G., et al.: Extrags: Geometric-aware trajectory extrapolation with uncertainty-guided generative priors. arXiv preprint arXiv:2508.15529 (2025)
27. Tao, T., Gao, L., Wang, G., Lao, Y., Chen, P., Zhao, H., Hao, D., Liang, X., Salzmann, M., Yu, K.: Lidar-nerf: Novel lidar view synthesis via neural radiance fields. In: Proceedings of the 32nd ACM International Conference on Multimedia. pp. 390–398 (2024)
28. Tonderski, A., Lindström, C., Hess, G., Ljungbergh, W., Svensson, L., Petersson, C.: Neurad: Neural rendering for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14895–14904 (2024)
29. Turki, H., Zhang, J.Y., Ferroni, F., Ramanan, D.: Suds: Scalable urban dynamic scenes. In: CVPR. pp. 12375–12385 (2023)
30. Wu, Z., Jiang, M., Lin, Z., Song, Y., Ma, H., Wu, Q., Zhang, D., Pu, G.: Curigs: Curriculum-guided gaussian splatting for sparse view synthesis. arXiv preprint arXiv:2511.16030 (2025)
31. Xiao, P., Shao, Z., Hao, S., Zhang, Z., Chai, X., Jiao, J., Li, Z., Wu, J., Sun, K., Jiang, K., Wang, Y., Yang, D.: Pandaset: Advanced sensor suite dataset for autonomous driving. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 3095–3101 (2021). https://doi.org/10.1109/ITSC48978.2021.9565009

32. Xiong, Y., Ma, W.C., Wang, J., Urtasun, R.: Learning compact representations for lidar completion and generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1074–1083 (2023)
33. Yan, Y., Lin, H., Zhou, C., Wang, W., Sun, H., Zhan, K., Lang, X., Zhou, X., Peng, S.: Street gaussians for modeling dynamic urban scenes. In: ECCV (2024)
34. Yang, Z., Chen, Y., Wang, J., Manivasagam, S., Ma, W.C., Yang, A.J., Urtasun, R.: Unisim: A neural closed-loop sensor simulator. In: CVPR. pp. 1389–1399 (June 2023)
35. Zhao, B., Yu, S., Ding, G., Hu, Y., Wang, H.: Pseudo-view enhancement via confidence fusion for unposed sparse-view reconstruction. arXiv preprint arXiv:2602.21535 (2026)
36. Zhou, C., Fu, L., Peng, S., Yan, Y., Zhang, Z., Chen, Y., Xia, J., Zhou, X.: Lidar-rt: Gaussian-based ray tracing for dynamic lidar re-simulation. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 1538–1548 (2025)
37. Zhou, X., Lin, Z., Shan, X., Wang, Y., Sun, D., Yang, M.H.: Drivinggaussian: Composite gaussian splatting for surrounding dynamic autonomous driving scenes. In: CVPR. pp. 21634–21643 (2024)
38. Zyrianov, V., Zhu, X., Wang, S.: Learning to generate realistic lidar point clouds. In: European Conference on Computer Vision. pp. 17–35. Springer (2022)