LiFTeR: Generating LiDAR Gaussian Fields from Surround-View Cameras

Anonymous authors
Paper under double-blind review

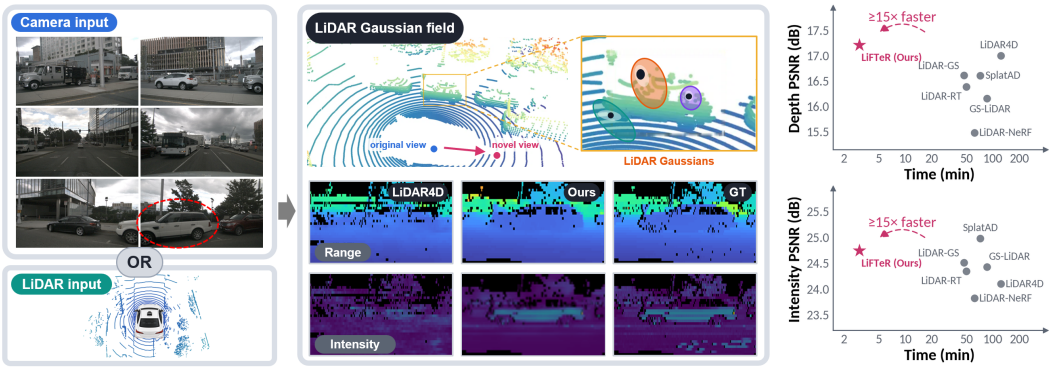


Figure 1: We propose **LiFTeR**, a framework that generates a LiDAR Gaussian field from surround-view cameras or a LiDAR scan. It runs at least $15\times$ faster than per-scene LiDAR Gaussian reconstruction while maintaining high fidelity in rendered range and intensity maps.

Abstract

High-fidelity LiDAR simulation requires modeling both scene geometry and sensor-specific return characteristics. Existing LiDAR generation methods often synthesize range, intensity, and ray-drop maps in image space, without an explicit 3D representation underlying these observations. Gaussian-based LiDAR reconstruction offers such a representation, but typically relies on per-scene optimization with real LiDAR measurements. We present LiFTeR, a cross-modal generative framework that bridges these approaches by producing renderable LiDAR Gaussian fields from surround-view images, without requiring target-scene LiDAR measurements or per-scene optimization at inference time. A diffusion transformer generates a latent representation conditioned on multi-view images, capturing scene geometry and LiDAR return characteristics. A ray-anchored Gaussian decoder then maps this latent representation to Gaussian primitives with explicit geometric parameters, intensity attributes, and ray-drop probabilities, which a panoramic Gaussian splatting renderer converts into range, intensity, and ray-drop maps. The framework additionally supports feed-forward conversion of observed LiDAR into Gaussian fields, enabling generation and reconstruction within a shared representation. On nuScenes, LiFTeR achieves the state-of-the-art MMD and JSD among LiDAR generation methods. Without target-scene LiDAR, it outperforms per-scene reconstruction methods on most depth and intensity metrics, while reducing the time to construct a scene representation from about an hour to 3 minutes. With LiDAR input, LiFTeR reduces the depth and intensity RMSE of novel view synthesis relative to baselines within 0.5 minutes per scene.

1 Introduction

LiDAR sensors are widely used in autonomous driving for 3D perception and planning. However, labeling 3D point clouds for training perception models is costly and time-consuming. LiDAR simulation, which generates realistic point clouds, provides a practical solution and has attracted

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

increasing attention. It not only reduces annotation costs but also enables systematic testing of perception models under diverse and safety-critical scenarios.

Existing LiDAR simulation methods mainly follow two paradigms: generative modeling and scene reconstruction. Generative approaches synthesize LiDAR scans without target-scene LiDAR observations at inference time, using adversarial, variational, or diffusion models (Caccia et al., 2019; Zyrianov et al., 2022; Ran et al., 2024; Hu et al., 2024). Recent works also explore joint camera-LiDAR generation and image-conditioned LiDAR synthesis (Xie et al., 2025; Wu et al., 2024b; Liu et al., 2025). Most of these methods produce individual scans rather than explicit scene representations for novel view synthesis. LidarDM (Zyrianov et al., 2025) instead generates scene meshes and synthesizes scans through raycasting. Reconstruction-based approaches recover neural fields (Huang et al., 2023; Tao et al., 2024) or Gaussian representations (Chen et al., 2024; Zhou et al., 2025) from captured LiDAR scans. These representations support novel view synthesis, with Gaussian-based methods attaching intensity and ray-drop to explicit geometric primitives. However, these reconstruction methods require target-scene LiDAR observations and per-scene optimization. This motivates directly generating LiDAR scene representations that encode geometry, intensity, and ray-drop for novel view synthesis, without target-scene LiDAR or per-scene optimization.

To address the limitations above, we introduce LiFTeR, a framework that generates LiDAR Gaussian fields from surround-view images or observed LiDAR scans for novel view scan synthesis without per-scene optimization. LiFTeR consists of a diffusion transformer (DiT) and a ray-anchored LiDAR Gaussian decoder. The DiT learns to denoise LiDAR latents conditioned on surround-view images. At inference time, this learned denoising process generates LiDAR latents from noise under the guidance of the input images. The Gaussian decoder then turns these latents into an explicit scene representation, rather than an individual range image. It directly predicts 2D Gaussian primitives with intensity and ray-drop attributes. To connect Gaussian geometry with the LiDAR sensing process, the decoder anchors each primitive to a calibrated ray and predicts its distance along that ray. Panoramic Gaussian splatting renders the field into a range map, which is compared with the ground-truth LiDAR range map to supervise the predicted geometry. The same decoder also accepts latents encoded from observed LiDAR scans. This shared latent space allows both image-based generation and scan-based reconstruction to produce the same Gaussian field representation.

Experiments on nuScenes (Caesar et al., 2020) show that LiFTeR achieves state-of-the-art performance in both camera-to-LiDAR generation and novel view synthesis, outperforming both generation and per-scene reconstruction methods. The generated scans can be directly used for downstream applications such as 3D detection.

In summary, our three main contributions are as follows.

- We introduce LiFTeR, a unified framework that bridges camera-conditioned LiDAR generation and feed-forward LiDAR scene reconstruction through a shared latent space and Gaussian representation. LiFTeR generates explicit LiDAR Gaussian fields from surround-view images without requiring target-scene LiDAR or per-scene optimization.
- We propose a ray-anchored Gaussian decoder that places each primitive along a calibrated LiDAR ray, supervises its geometry through rendered range maps, and jointly predicts intensity and ray-drop.
- We achieves the state-of-the-art camera-to-LiDAR generation performance and consistent novel-pose rendering from a single generated field.

2 Related Work

LiDAR novel view synthesis. LiDAR novel view synthesis reconstructs a scene from captured scans and renders realistic scans at unseen sensor poses. Early methods raycast surfel assets built from real scans (Manivasagam et al., 2020), and later methods represent the scene with neural fields (Mildenhall et al., 2021; Huang et al., 2023; Tao et al., 2024; Zheng et al., 2024). Recent methods adopt Gaussian primitives (Kerbl et al., 2023) for faster training and rendering and model intensity and ray-drop with them (Chen et al., 2024; Jiang et al., 2025; Hess et al., 2025; Zhou et al., 2025). However, these methods only reconstruct scenes from observed LiDAR scans, requiring per-scene optimization. LiFTeR adopts the Gaussian representation and generates it from surround-view images through latent diffusion followed by feed-forward Gaussian decoding, without per-scene optimization.

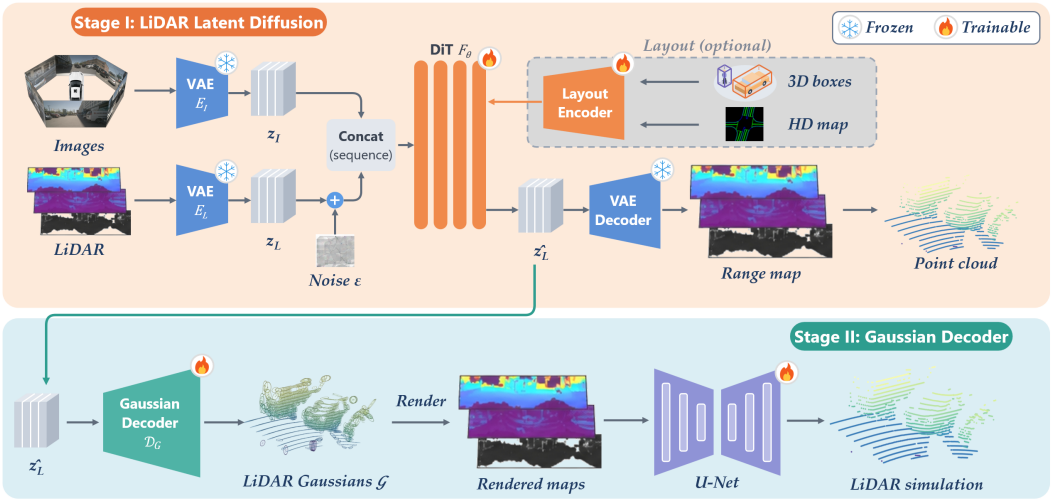


Figure 2: **Overview of LiFTeR.** In Stage I, the diffusion transformer is trained to generate LiDAR latents conditioned on surround-view images and optional layout information. Stage II freezes the diffusion model and consists of two steps. The Gaussian decoder is first trained through the panoramic renderer, and a U-Net is then trained with the decoder frozen to refine the rendered ray-drop map.

LiDAR generation. Generative approaches learn scan distributions to produce realistic geometry and sensor returns. Early methods apply GANs and VAEs to range images (Caccia et al., 2019; Nakashima & Kurazume), and subsequent methods adopt diffusion models (Zyrianov et al., 2022; Nakashima & Kurazume, 2024; Ran et al., 2024; Hu et al., 2024) and incorporate text (Wu et al., 2024a) or camera images (Ran et al., 2024; Xie et al., 2025; Wu et al., 2024b; Liu et al., 2025). However, their outputs remain range images or point clouds tied to the sensor pose assumed at generation. These outputs can be reprojected to new poses but cannot recover newly visible surfaces. LidarDM (Zyrianov et al., 2025) and UniScene (Li et al., 2025) render LiDAR from a generated 3D scene, but the scene is generated from a layout without camera images. LiFTeR generates the Gaussian field from camera images, with intensity and ray-drop attached to the primitives.

Feed-forward Gaussian reconstruction. Feed-forward methods predict Gaussian primitives from images in a single pass. pixelSplat (Charatan et al., 2024) estimates per-pixel depth distributions from two views, while GS-LRM (Zhang et al., 2024) directly regresses Gaussians with a transformer. Long-LRM (Ziwen et al., 2025) supports long input sequences with Mamba-2 and transformer blocks, and Lyra (Bahmani et al., 2026) decodes Gaussians from video diffusion latents. These methods primarily target image reconstruction under photometric supervision, without explicitly modeling LiDAR scanning geometry or return attributes. LiDAR sensing emits rays along calibrated directions in the sensor frame, and each ray either returns a measured range and intensity or yields no return. LiFTeR extends latent-space Gaussian decoding to LiDAR by anchoring primitives to sensor rays.

3 Method

3.1 Overview

Given surround-view images or LiDAR scans, LiFTeR produces LiDAR Gaussian fields $\mathcal{G}$ that can be rendered into range, intensity, and ray-drop maps. As shown in Fig. 2, a diffusion transformer F_θ first generates the LiDAR latent $\hat{\mathbf{z}}_L$ conditioned on the image latents (Sec. 3.2). A Gaussian decoder $\mathcal{D}_G$ then converts $\hat{\mathbf{z}}_L$ into $\mathcal{G}$, and a U-Net refines the rendered ray-drop map (Sec. 3.3). Training and inference are described in Sec. 3.4 and Sec. 3.5.

3.2 Camera-Conditioned LiDAR Latent Diffusion

Paired camera-LiDAR data are scarce compared with camera data. LiFTeR therefore builds on a multi-view diffusion transformer pre-trained on large-scale data (Agarwal et al., 2025) and formulates

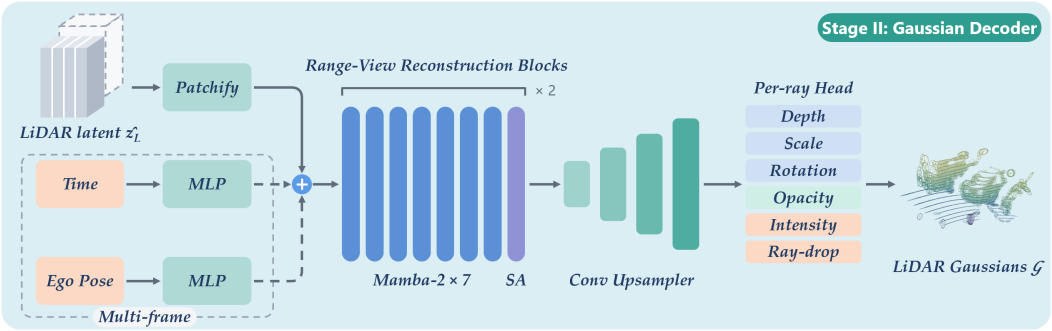


Figure 3: **Architecture of the LiDAR Gaussian decoder.** The decoder patchifies the LiDAR latent $\hat{\mathbf{z}}_L$ into tokens, processes them with reconstruction blocks that interleave Mamba-2 and self-attention layers, and upsamples the features to the ray grid, where a per-ray head predicts the Gaussians $\mathcal{G}$.

camera-to-LiDAR generation as a multi-view generation task, in which the surround-view images serve as conditions for synthesizing the corresponding LiDAR range map.

Latent encoding. We use two separate VAE encoders to encode the images and LiDAR data. The Cosmos VAE encoder (Agarwal et al., 2025) encodes surround-view images into $\mathbf{z}_I^v = E_I(I^v)$ with $8\times$ spatial compression and 16 latent channels, while the LiDAR VAE encoder (Hu et al., 2024) jointly encodes the range map D and the intensity map P into $\bar{\mathbf{z}}_L = E_L(D, P) \in \mathbb{R}^{4\times H/4\times W/4}$ with $4\times$ spatial compression. Both encoders remain frozen during diffusion training. To match the backbone's 16-channel input interface, we replicate each LiDAR latent channel four times to obtain $\mathbf{z}_L \in \mathbb{R}^{16\times H/4\times W/4}$. After denoising, we map the output back to four channels by averaging within each group of replicated channels.

Multi-view denoising. The denoiser F_θ adopts the multi-view diffusion transformer of Cosmos-Predict1 (Agarwal et al., 2025) and is initialized from its pre-trained weights. The image latents $\{\mathbf{z}_I^v\}_{v=1}^V$ and the noisy LiDAR latent $\mathbf{z}_L^\sigma$ are zero-padded to a common spatial size and concatenated as views of the backbone. The image views are kept clean as conditions, and only the LiDAR view is denoised, which yields $\hat{\mathbf{z}}_L$ after the channel averaging described above. The training objective is given in Sec. 3.4.

Layout conditioning. When layout information (such as HD maps and 3D bounding boxes) is available, it can be incorporated as additional conditioning to guide the spatial structure of the generated scene. Implementation details are provided in Appendix A.

3.3 LiDAR Gaussian Decoder

A range image decoded from $\hat{\mathbf{z}}_L$ is tied to the sensor pose assumed at generation and cannot be rendered at other poses. The LiDAR Gaussian decoder $\mathcal{D}_G$ is therefore designed to transform $\hat{\mathbf{z}}_L$ into an explicit 3D representation that can be rendered from arbitrary sensor poses. Gaussian splatting is chosen for its explicit representation and fast rendering, and its LiDAR variants attach intensity and ray-drop to the primitives (Chen et al., 2024; Jiang et al., 2025; Zhou et al., 2025). The decoder operates on the latent directly, so a LiDAR latent generated by the diffusion transformer and a latent encoded from an observed scan are both decoded in the same way. Fig. 3 details the decoder.

Architecture. The LiDAR Gaussian decoder $\mathcal{D}_G$ first maps the LiDAR latent $\hat{\mathbf{z}}_L$ into the hidden dimension of the reconstruction blocks and outputs Gaussian parameters (Fig. 3). Specifically, the input latent is patchified into tokens and processed by the reconstruction blocks of Long-LRM (Ziwen et al., 2025), which interleave Mamba-2 (Dao & Gu, 2024) layers with self-attention layers. The self-attention layers model global interactions among all tokens, and the Mamba-2 layers scale linearly with the number of tokens and keep decoding efficient. A convolutional upsampler restores the tokens to the $H \times W$ ray grid, and a per-ray head maps the features to 12 Gaussian channels on each ray, namely the ray-distance logit u^d, the rotation quaternion $\mathbf{q}$, the tangential scales (s^u, s^v), the opacity α, and four color channels of the rasterizer, two of which carry the intensity ρ and the ray-drop probability r. Formally, the decoder is expressed as $\mathcal{G} = \mathcal{D}_G(\hat{\mathbf{z}}_L)$.

Ray-anchored Gaussian field. Each primitive is a 2D Gaussian (Huang et al., 2024), and its center is parameterized by the distance along its ray,

$$\boldsymbol{\mu}_{ij} = \mathbf{o}_{ij} + \ell_{ij}\,\mathbf{d}_{ij}, \qquad \ell_{ij} = d_{\min} + (d_{\max} - d_{\min})\,\varsigma\big(u^d_{ij}\big), \tag{1}$$

where $\mathbf{o}_{ij}$ and $\mathbf{d}_{ij}$ are the ray origin and direction, ℓ_{ij} is the distance along the ray bounded by $[d_{\min}, d_{\max}]$, and ς is the sigmoid function. The field is rendered into range, intensity, and ray-drop maps by the panoramic rasterizer, and a U-Net (Zheng et al., 2024) takes the three maps as input to refine the ray-drop map over the full range image. For multi-frame input, the embeddings of the timestamp t_k and ego pose $\mathbf{T}_k$ of each frame are added to its tokens, and the field is the union of the primitives of all frames in the world frame as $\mathcal{G} = \bigcup_{k=1}^{K} \mathcal{D}_G(\hat{\mathbf{z}}^k_L, t_k, \mathbf{T}_k)$.

3.4 Training

We train LiFTeR in two stages, first training camera-conditioned LiDAR latent diffusion and then training the Gaussian decoder with rendering supervision.

Stage I: LiDAR latent diffusion. With both VAE encoders frozen, we optimize the DiT using the EDM objective (Karras et al., 2022). Given a LiDAR latent $\mathbf{z}_L$ and the corresponding camera latents $\mathbf{c} = \{\mathbf{z}^v_I\}_{v=1}^{V}$, the denoiser learns to recover $\mathbf{z}_L$ from its noise-corrupted counterpart:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{\mathbf{z}_L, \mathbf{c}, \sigma, \boldsymbol{\epsilon}} \left[w(\sigma)\, \| F_\theta(\mathbf{z}_L + \sigma\boldsymbol{\epsilon}; \sigma, \mathbf{c}) - \mathbf{z}_L \|_2^2 \right], \tag{2}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, F_θ denotes the EDM-preconditioned DiT, and $w(\sigma) = (\sigma^2 + \sigma^2_{\text{data}})/(\sigma \cdot \sigma_{\text{data}})^2$ is the loss weight.

Stage II: Gaussian decoding. The VAE encoders and the trained diffusion model are frozen, and the Gaussian decoder is trained through differentiable rendering. Running multi-step diffusion sampling for each training example is costly. The decoder inputs are therefore constructed by perturbing latents encoded from ground-truth scans with noise at $\sigma \le \sigma_{\max}$, followed by a single denoising pass conditioned on the corresponding camera latents. The decoder predicts a Gaussian field from these latents, which is rendered at the sensor poses of the input frames and supervised by the corresponding ground-truth scans.

The training loss combines range, intensity, and ray-drop supervision with geometric regularization:

$$\mathcal{L}_{\text{dec}} = \mathcal{L}_{\text{range}} + \lambda_{\text{int}}\mathcal{L}_{\text{int}} + \lambda_{\text{drop}}\mathcal{L}_{\text{drop}} + \mathcal{L}_{\text{geo}}. \tag{3}$$

Here, $\mathcal{L}_{\text{range}}$ and $\mathcal{L}_{\text{int}}$ are L_1 losses averaged over rays with valid returns, while $\mathcal{L}_{\text{drop}}$ is binary cross-entropy between the rendered ray-drop probabilities and the ground-truth no-return mask over all rays. The geometric term $\mathcal{L}_{\text{geo}}$ comprises range-gradient, normal-consistency, truncated Chamfer, and distortion losses. Detailed definitions and training configurations are provided in the appendix.

3.5 Inference

After training, LiFTeR supports two inference modes.

Camera-to-Gaussian generation: Given surround-view images, we sample LiDAR latents through multi-step denoising conditioned on the image latents, without requiring input LiDAR scans. The sampled latents are processed by the same Gaussian decoding and rendering pipeline.

LiDAR-to-Gaussian reconstruction: Given observed LiDAR scans, we encode their range and intensity maps into LiDAR latents using the frozen encoder E_L. The Gaussian decoder maps these latents to a Gaussian field, which is rendered at the query sensor pose to synthesize a LiDAR scan.

4 Experiments

4.1 Experimental Setup

Datasets. Experiments are conducted on nuScenes (Caesar et al., 2020), captured by six surround-view cameras and a 32-beam LiDAR. The 2 Hz keyframes are used, and each scan is projected into a 32×1024 range image. Training uses the 700 training scenes, and camera-to-LiDAR generation is

Table 1: **Comparison of LiDAR generation on nuScenes.** For each column, the best value is highlighted by **bold**, and the second best one is denoted by underline.

Modality	Method	Point clouds quality	
		MMD↓	JSD↓
LiDAR	LiDARGen (Zyrianov et al., 2022)	1.9×10^{-3}	0.160
	RangeLDM (Hu et al., 2024)	1.9×10^{-4}	0.054
Text→LiDAR	Text2LiDAR (Wu et al., 2024a)	2.3×10^{-3}	0.089
Layout→LiDAR	UniScene (Li et al., 2025)	3.4×10^{-3}	0.103
	OpenDWM (ope, 2025)	5.4×10^{-4}	0.091
Camera→LiDAR	X-Drive (Xie et al., 2025)	1.0×10^{-4}	0.052
	LiFTeR (Ours)	$\mathbf{6.5 \times 10^{-5}}$	**0.033**

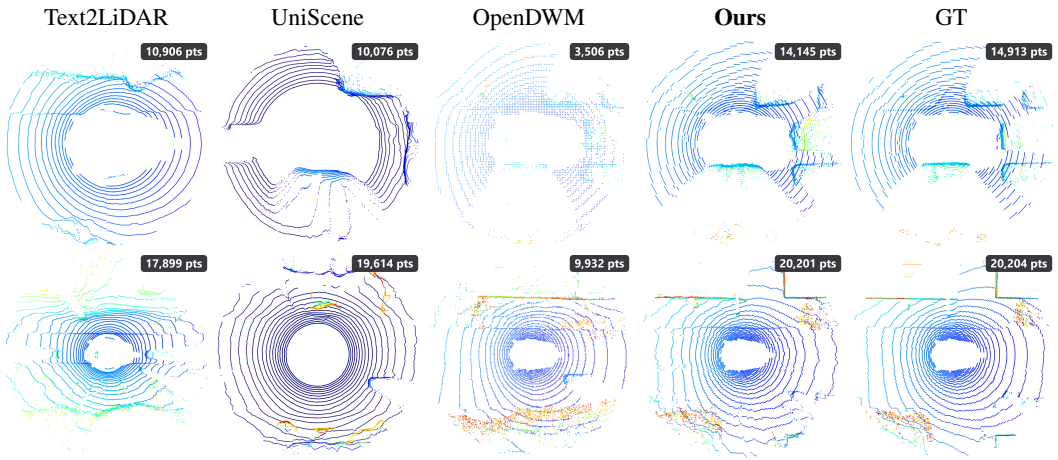


Figure 4: **Qualitative comparison of LiDAR generation.**

evaluated on the 150 validation scenes. The comparison with per-scene methods uses 10 validation scenes, where 5 frames per scene are held out for testing and per-scene methods are optimized on the remaining frames. These test frames are also unseen by LiFTeR, which uses only camera images. For novel view synthesis, every other frame is held out for testing. The reported time is the per-scene optimization time for per-scene methods and the end-to-end inference time for LiFTeR.

Baselines. Generative baselines include the unconditional LiDARGen (Zyrianov et al., 2022) and RangeLDM (Hu et al., 2024), the text-conditioned Text2LiDAR (Wu et al., 2024a), the layout-conditioned UniScene (Li et al., 2025) and OpenDWM (ope, 2025), and the camera-conditioned X-Drive (Xie et al., 2025). Per-scene reconstruction baselines include LiDAR-NeRF (Tao et al., 2024), LiDAR4D (Zheng et al., 2024), LiDAR-GS (Chen et al., 2024), GS-LiDAR (Jiang et al., 2025), SplatAD (Hess et al., 2025), and LiDAR-RT (Zhou et al., 2025).

Metrics. Following GS-LiDAR (Jiang et al., 2025), point clouds are evaluated by CD and F-score at a 5 cm threshold, and rendered depth and intensity maps by RMSE, MedAE, LPIPS, SSIM, and PSNR. Following X-Drive (Xie et al., 2025), generated and real point clouds are projected onto 100×100 BEV histograms, and the MMD and JSD between the histograms are reported.

Implementation details. The DiT is trained for 20k iterations on 64 NVIDIA H20 GPUs, and the Gaussian decoder for 500k iterations on 16 NVIDIA H20 GPUs. Inference uses 16 denoising steps. The loss weights are $\lambda_{\text{int}} = 1$, $\lambda_{\text{drop}} = 1$, $\lambda_{\text{grad}} = 0.2$, $\lambda_{\text{nrm}} = 0.1$, $\lambda_{\text{cd}} = 0.05$, and $\lambda_{\text{dist}} = 0.01$. The ray-drop U-Net is trained on 4 GPUs with a learning rate of 10^{-4} and a weight decay of 0.1.

Table 2: **Comparison with per-scene LiDAR reconstruction methods on nuScenes.** All methods are evaluated on held-out test frames. Per-scene methods are optimized on the remaining LiDAR scans of each scene, and LiFTeR uses cameras only. The best and second best results are colored.

Method	Point Cloud		Depth					Intensity					Time↓
	CD↓	F-score↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	
LiDAR-NeRF (Tao et al., 2024)	1.8041	0.5116	8.5872	0.8053	0.1705	0.4349	15.4756	0.0671	0.0191	0.2571	0.3202	23.8179	62min
LiDAR4D (Zheng et al., 2024)	1.1075	0.6101	7.4088	0.1821	0.1152	0.5587	16.9947	0.0513	0.0109	0.0742	0.5157	24.0961	124min
LiDAR-GS (Chen et al., 2024)	1.2645	0.5129	7.4896	0.4108	0.1417	0.5109	16.6076	0.0615	0.0173	0.1094	0.4775	24.5061	47min
LiDAR-RT (Zhou et al., 2025)	1.3692	0.5213	7.7374	0.3575	0.1532	0.4922	16.3817	0.0623	0.0160	0.0978	0.4858	24.3398	50min
SplatAD (Hess et al., 2025)	1.2230	0.5911	7.5005	0.3223	0.1407	0.4966	16.6021	0.0581	0.0195	0.0971	0.4851	24.9742	72min
GS-LiDAR (Jiang et al., 2025)	1.0503	0.5533	7.9395	0.2586	0.1770	0.5771	16.1514	0.0616	0.0153	0.1103	0.5106	24.4226	86min
LiFTeR (Ours)	1.5955	0.6118	7.0761	0.1263	0.1032	0.7550	17.2053	0.0449	0.0073	0.1191	0.6154	24.7396	3min

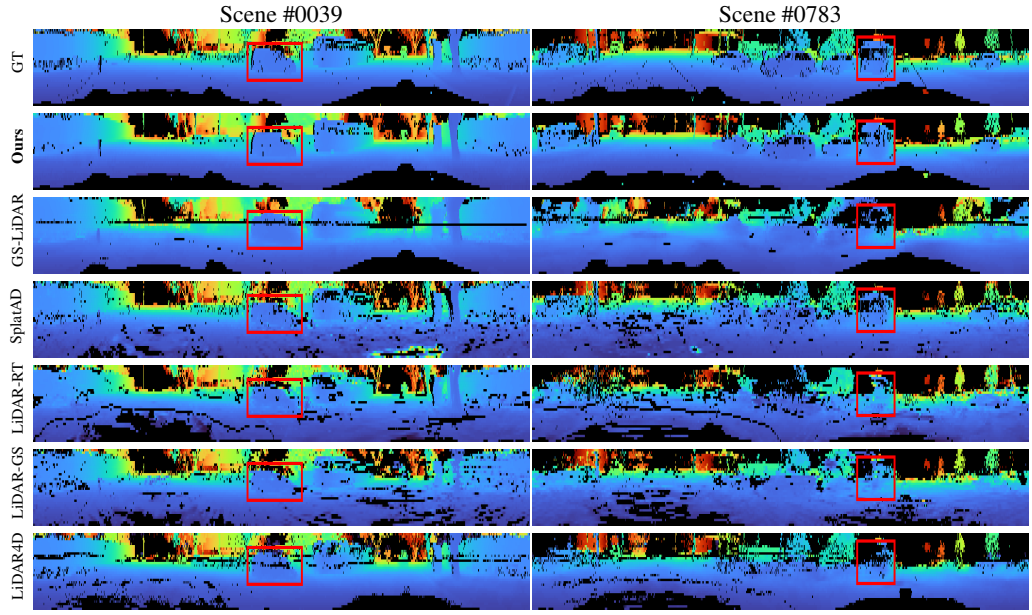


Figure 5: **Qualitative comparison with per-scene LiDAR reconstruction methods.**

4.2 Evaluation on Camera-to-LiDAR Generation

Comparison with generative models. As shown in Tab. 1, LiFTeR achieves the lowest MMD and JSD among all generative models, reducing them by 35% and 36.5% relative to the camera-conditioned X-Drive. In Fig. 4, LiFTeR preserves the native scanline structure, while the voxel-based OpenDWM and the occupancy-based UniScene disrupt the scanning pattern.

Comparison with per-scene reconstruction. Tab. 2 compares each generated scan with the ground truth of the same frame, measuring the correspondence to the input cameras. Without any LiDAR of the evaluated scenes, LiFTeR attains the best result on nine of the twelve metrics, reducing the median depth error by 30.6% and the median intensity error by 33.0% relative to LiDAR4D. Its Chamfer distance remains higher than that of most per-scene methods, which fit geometry directly to the LiDAR of each scene. Qualitative comparisons are shown in Fig. 5.

4.3 Novel View Synthesis

Each test frame is rendered from the field decoded from its neighboring training frames through multi-frame inference. The latents of these frames are generated from camera images in LiFTeR (C→L) and encoded from LiDAR scans in LiFTeR (L→L). Generating half of the frames in each scene takes LiFTeR (C→L) 18 min. As shown in Tab. 3 and Fig. 6, without LiDAR input, LiFTeR (C→L) achieves rendering quality comparable to the per-scene methods, with a depth PSNR on par with LiDAR4D and a higher intensity SSIM than all per-scene methods except GS-LiDAR, while running at least 2.4 times faster. With LiDAR input, LiFTeR (L→L) reduces the depth and intensity RMSE by 2.4% and 5.9% relative to SplatAD within 0.5 min.

Table 3: **Comparison of LiDAR novel view synthesis on nuScenes.** The notations are consistent with Tab. 2.

Method	Point Cloud		Depth					Intensity					Time↓
	CD↓	F-score↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	RMSE↓	MedAE↓	LPIPS↓	SSIM↑	PSNR↑	
LiDAR-NeRF (Tao et al., 2024)	1.4231	0.5321	8.4504	0.5044	0.1693	0.4412	15.5955	0.0681	0.0201	0.2743	0.2844	23.6024	75min
LiDAR4D (Zheng et al., 2024)	1.1945	0.5561	7.9318	0.3611	0.1472	0.5532	16.0416	0.0599	0.0135	0.1755	0.4951	25.6137	157min
LiDAR-GS (Chen et al., 2024)	1.4647	0.5101	7.9832	0.4465	0.1523	0.4973	16.3027	0.0645	0.0189	0.1375	0.4825	24.3141	49min
LiDAR-RT (Zhou et al., 2025)	1.2762	0.5127	8.0514	0.4211	0.1488	0.4303	16.0004	0.0641	0.0176	0.0993	0.4661	24.1014	44min
SplatAD (Hess et al., 2025)	1.1441	0.5981	7.6435	0.3241	0.1419	0.4878	16.4269	0.0573	0.0187	0.0977	0.4853	25.0631	75min
GS-LiDAR (Jiang et al., 2025)	1.3218	0.5841	7.9665	0.2186	0.1689	0.5726	16.1026	0.0614	0.0144	0.1043	0.5071	24.4801	78min
LiFTeR (C→L)	1.7357	0.4733	8.3949	0.4120	0.1574	0.4862	16.0364	0.0661	0.0189	0.1493	0.5033	24.3280	18min
LiFTeR (L→L)	1.4352	0.5293	7.4609	0.3066	0.1411	0.5567	16.6716	0.0539	0.0136	0.1227	0.5125	25.5878	0.5min

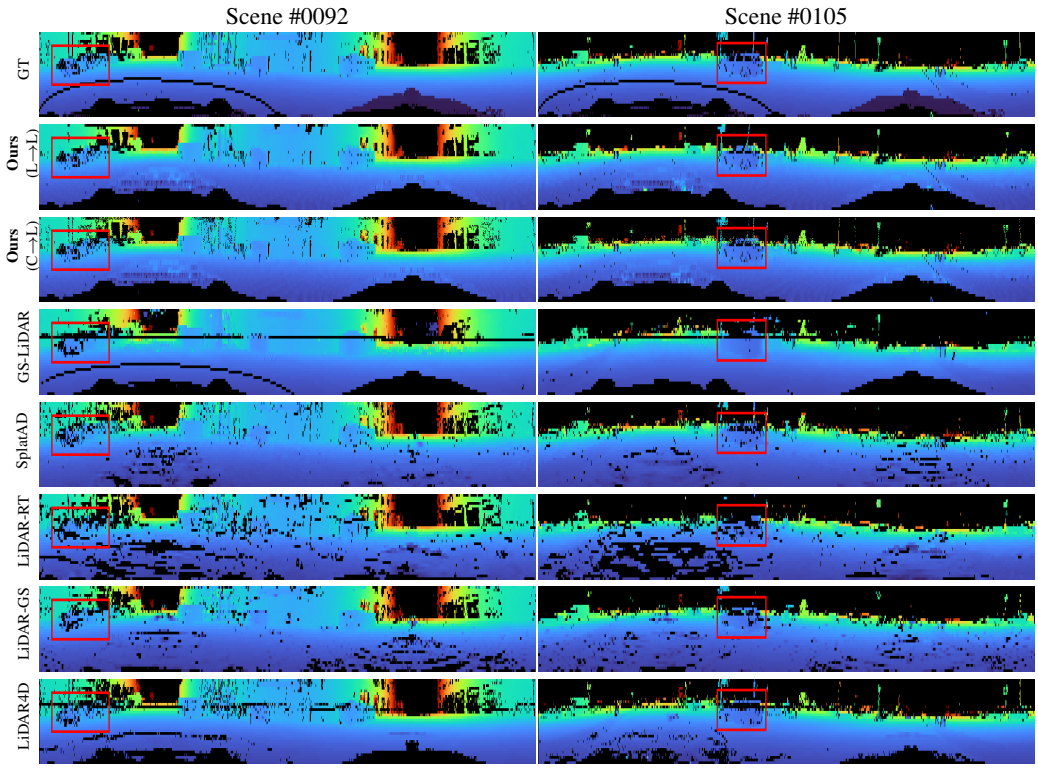


Figure 6: **Qualitative comparison of LiDAR novel view synthesis.**

4.4 Ego-Trajectory Extrapolation

Ego-trajectory extrapolation evaluates a fixed generated field at novel sensor poses. We generate a single field from a single reference keyframe and render it at subsequent keyframes ($T = 5, 10, 20, 50$) without regeneration, using a scene with limited object motion. Ground surfaces, road boundaries, and facades remain aligned with corresponding measured scans (Fig. 8). Depth PSNR decreases from 21.35 dB at $T = 0$ to 16.66 dB at $T = 20$ and 13.37 dB at $T = 50$, as the sensor moves beyond the initially observed region. Compared with reprojecting a range image decoded from the same latent, Gaussian rendering starts at similar reference-pose accuracy but preserves better surface coverage under viewpoint changes, achieving a 1.20 dB advantage at $T = 30$ (Fig. 7).

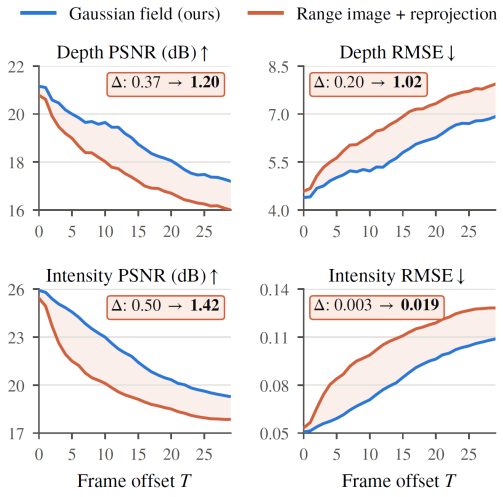


Figure 7: **Comparison of the Gaussian field and range-image reprojection.**

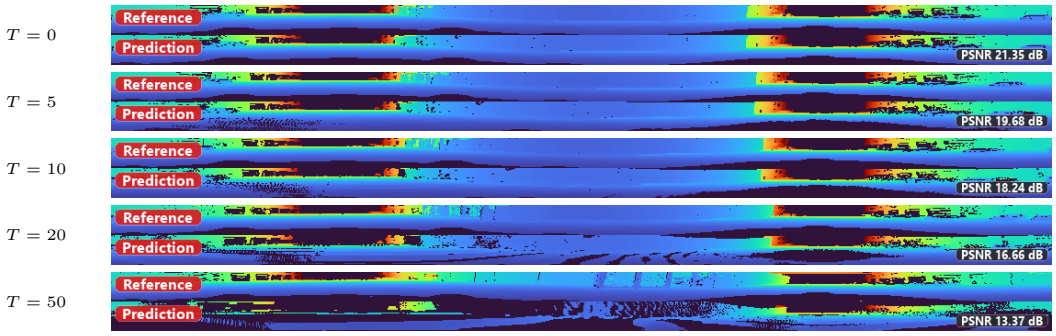


Figure 8: **Ego-trajectory extrapolation from a single generated Gaussian field.**

4.5 Ablation Study

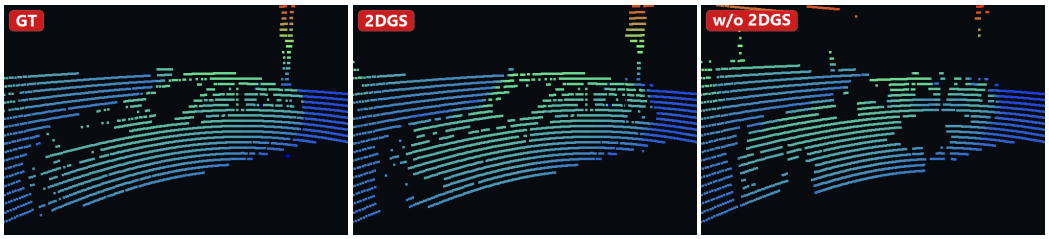
Tab. 4 shows that layout conditioning improves the scene structure. Adding the HD map reduces the MMD from 6.5 to 6.3 and the JSD from 3.3 to 3.1, and adding the 3D boxes further reduces the JSD to 3.0. Tab. 5 evaluates the components of the Gaussian decoder. Replacing 2D Gaussians with 3D Gaussians increases the Chamfer distance by 4.9%, since volumetric primitives do not fit the thin nature of surfaces (Huang et al., 2024) and yield less accurate geometry (Fig. 9). Removing $\mathcal{L}_{\text{geo}}$, which regularizes range gradients, surface normals, point-level alignment, and the depth distribution along each ray, reduces the F-score by 2.7%. Removing the ray-drop refinement leads to the largest increase in Chamfer distance (6.1%) and increases the intensity MedAE by 21.9%.

Table 4: **Ablation on the diffusion model.** MMD in 10^{-5} and JSD in 10^{-2}.

	Map	Box	MMD↓	JSD↓
Camera only			6.5	3.3
+ HD map	✓		6.3	3.1
+ HD map & boxes	✓	✓	6.3	3.0

Table 5: **Ablation on the Gaussian decoder.**

	Point Cloud		Depth		Intensity	
	CD↓	F-score↑	MedAE↓	PSNR↑	MedAE↓	PSNR↑
Full model	1.5955	0.6118	0.1263	17.2053	0.0073	24.7396
w/o 2DGS	1.6733	0.5921	0.1316	16.8016	0.0082	24.5775
w/o $\mathcal{L}_{\text{geo}}$	1.6032	0.5954	0.1297	17.0574	0.0083	24.5012
w/o ray-drop refinement	1.6931	0.5863	0.1322	16.9761	0.0089	24.5160



GT Full model w/o 2DGS

Figure 9: **Comparison of 2DGS and 3DGS primitives.**

5 Conclusion

This paper presents LiFTeR, a cross-modal generative framework that produces explicit LiDAR Gaussian fields from surround-view camera images without requiring target-scene LiDAR or per-scene optimization. LiFTeR builds on a pre-trained multi-view diffusion transformer that generates LiDAR latents conditioned on the camera views. A ray-anchored Gaussian decoder maps LiDAR latents to 2D Gaussian primitives, each carrying geometry, intensity, and ray-drop probability.

The resulting field can be rendered from novel sensor poses without rerunning the generative model. Through a shared LiDAR latent space and Gaussian decoder, LiFTeR also supports feed-forward reconstruction from observed LiDAR scans. Experiments on nuScenes show that LiFTeR achieves the tate-of-the-art MMD and JSD among generation methods. Without target-scene LiDAR, LiFTeR outperforms per-scene reconstruction methods on most depth and intensity metrics at held-out frames and remains competitive in novel view synthesis. These results show that cross-modal generation and an explicit, queryable LiDAR representation can be unified within a single framework.

AI USE STATEMENT

Large language models were used only to assist with literature searches and to polish the writing of this paper, including grammar correction and wording refinement. They were not used to design the method, implement the code, conduct the experiments, or analyze the results. All references identified and text edited with AI assistance were reviewed and verified by the authors, who take full responsibility for the content of this work.

REFERENCES

Opendwm: Open driving world models, 2025. https://github.com/SenseTime-FVG/OpenDWM.

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical ai. *arXiv preprint arXiv:2501.03575*, 2025.

Sherwin Bahmani, Tianchang Shen, Jiawei Ren, Jiahui Huang, Yifeng Jiang, Haithem Turki, Andrea Tagliasacchi, David Lindell, Zan Gojcic, Sanja Fidler, et al. Lyra: Generative 3d scene reconstruction via video diffusion model self-distillation. In *International Conference on Learning Representations*, volume 2026, pp. 82850–82880, 2026.

Lucas Caccia, Herke Van Hoof, Aaron Courville, and Joelle Pineau. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5034–5040. IEEE, 2019.

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. In *2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11618–11628. IEEE, 2020.

David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19457–19467. IEEE, 2024.

Qifeng Chen, Sheng Yang, Sicong Du, Tao Tang, Rengan Xie, Peng Chen, and Yuchi Huo. Lidar-gs: Real-time lidar re-simulation using gaussian splatting. *arXiv preprint arXiv:2410.05111*, 2024.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 605–613, 2017.

Georg Hess, Carl Lindström, Maryam Fatemi, Christoffer Petersson, and Lennart Svensson. Splatad: Real-time lidar and camera rendering with 3d gaussian splatting for autonomous driving. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11982–11992. IEEE, 2025.

Qianjiang Hu, Zhimin Zhang, and Wei Hu. Rangeldm: Fast realistic lidar point cloud generation. In *European Conference on Computer Vision*, pp. 115–135. Springer, 2024.

Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pp. 1–11, 2024.

Shengyu Huang, Zan Gojcic, Zian Wang, Francis Williams, Yoni Kasten, Sanja Fidler, Konrad Schindler, and Or Litany. Neural lidar fields for novel view synthesis. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18190–18200. IEEE, 2023.

Junzhe Jiang, Chun Gu, Yurui Chen, and Li Zhang. Gs-lidar: Generating realistic lidar point clouds with panoramic gaussian splatting. In *International Conference on Learning Representations*, volume 2025, pp. 56631–56645, 2025.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Bohan Li, Jiazhe Guo, Hongsi Liu, Yingshuang Zou, Yikang Ding, Xiwu Chen, Hu Zhu, Feiyang Tan, Chi Zhang, Tiancai Wang, et al. Uniscene: Unified occupancy-centric driving scene generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11971–11981. IEEE, 2025.

Alan Liang, Youquan Liu, Yu Yang, Dongyue Lu, Linfeng Li, Lingdong Kong, Huaici Zhao, and Wei Tsang Ooi. Lidarcrafter: Dynamic 4d world modeling from lidar sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18406–18414, 2026.

Youquan Liu, Lingdong Kong, Weidong Yang, Ao Liang, Jianxiong Gao, Yang Wu, Xiang Xu, Xin Li, Linfeng Li, Runnan Chen, et al. Veila: Panoramic lidar generation from a monocular rgb image. *arXiv preprint arXiv:2508.03690*, 2025.

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 *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11164–11173. IEEE, 2020.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Kazuto Nakashima and Ryo Kurazume. Learning to drop points for lidar scan synthesis. in 2021 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 222–229.

Kazuto Nakashima and Ryo Kurazume. Lidar data synthesis with denoising diffusion probabilistic models. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 14724–14731. IEEE, 2024.

Haoxi Ran, Vitor Guizilini, and Yue Wang. Towards realistic scene generation with lidar diffusion models. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14738–14748. IEEE, 2024.

Tang Tao, Longfei Gao, Guangrun Wang, Yixing Lao, Peng Chen, Hengshuang Zhao, Dayang Hao, Xiaodan Liang, Mathieu Salzmann, and Kaicheng Yu. Lidar-nerf: Novel lidar view synthesis via neural radiance fields. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pp. 390–398, 2024.

OD Team et al. Openpcdet: An open-source toolbox for 3d object detection from point clouds. 2020.

Yang Wu, Kaihua Zhang, Jianjun Qian, Jin Xie, and Jian Yang. Text2lidar: Text-guided lidar point cloud generation via equirectangular transformer. In *European Conference on Computer Vision*, pp. 291–310. Springer, 2024a.

Zehuan Wu, Jingcheng Ni, Xiaodong Wang, Yuxin Guo, Rui Chen, Lewei Lu, Jifeng Dai, and Yuwen Xiong. Holodrive: Holistic 2d-3d multi-modal street scene generation for autonomous driving. *arXiv preprint arXiv:2412.01407*, 2024b.

Yichen Xie, Chenfeng Xu, Chensheng Peng, Shuqi Zhao, Nhat Ho, Alexander Pham, Mingyu Ding, Masayoshi Tomizuka, and Wei Zhan. X-drive: Cross-modality consistent multi-sensor data synthesis for driving scenarios. In *International Conference on Learning Representations*, volume 2025, pp. 49268–49284, 2025.

Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11784–11793, 2021.

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 1–19. Springer, 2024.

Zehan Zheng, Fan Lu, Weiyi Xue, Guang Chen, and Changjun Jiang. Lidar4d: Dynamic neural fields for novel space-time view lidar synthesis. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5145–5154. IEEE, 2024.

Chenxu Zhou, Lvchang Fu, Sida Peng, Yunzhi Yan, Zhanhua Zhang, Yong Chen, Jiazhi Xia, and Xiaowei Zhou. Lidar-rt: Gaussian-based ray tracing for dynamic lidar re-simulation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1538–1548. IEEE, 2025.

Chen Ziwen, Hao Tan, Kai Zhang, Sai Bi, Fujun Luan, Yicong Hong, Li Fuxin, and Zexiang Xu. Long-lrm: Long-sequence large reconstruction model for wide-coverage gaussian splats. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4349–4359. IEEE, 2025.

Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision*, pp. 17–35. Springer, 2022.

Vlas Zyrianov, Henry Che, Zhijian Liu, and Shenlong Wang. Lidardm: Generative lidar simulation in a generated world. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6055–6062. IEEE, 2025.

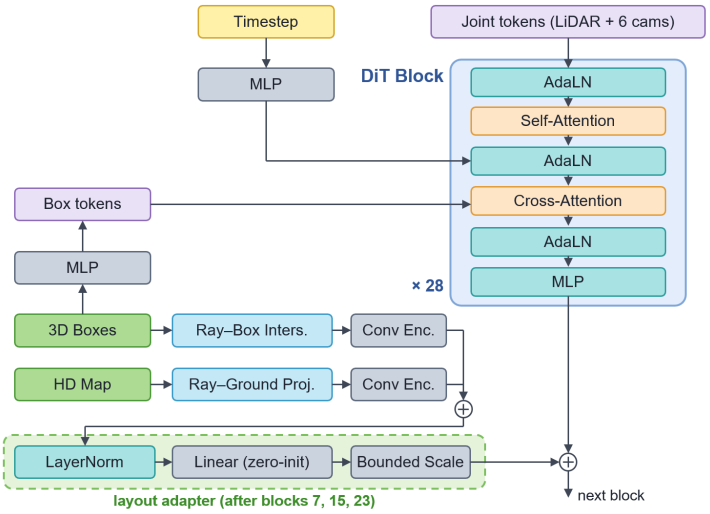


Figure 10: **Layout conditioning.** HD maps and 3D boxes are projected onto the LiDAR rays and injected into the LiDAR tokens through bounded residual adapters. The boxes are additionally encoded as cross-attention tokens.

A Layout Conditioning Details

Range-view layout maps. The layout conditions are converted into maps on the 32×1024 ray grid of the range image, so that each condition is spatially aligned with the LiDAR latent. Specifically, the road segments and lanes within a $100\,\mathrm{m} \times 100\,\mathrm{m}$ region around the sensor are rasterized into a BEV mask at a resolution of $0.1\,\mathrm{m}$, and each ray samples the mask at its intersection with the ground plane. The resulting HD map encodes the drivable mask, the validity of the intersection, the normalized intersection distance, and the signed distance to the drivable boundary. For the 3D boxes, each ray is intersected with the oriented boxes and retains the nearest one, and the resulting map encodes the hit mask, the normalized entry and exit distances, and the object class.

Layout encoding. Each map is processed by a convolutional encoder with two 3×3 layers of width 128, pooled to the resolution of the LiDAR tokens, and projected to the hidden dimension of the diffusion transformer. The encoded HD map and box maps are summed into layout tokens $\mathbf{l}$. The range-view box map keeps only the nearest box along each ray, so occluded boxes and boxes covering only a few rays are partially lost. The 16 nearest boxes within $50\,\mathrm{m}$ are therefore also encoded as cross-attention tokens. Following LiDARCrafter (Liang et al., 2026), the box center is normalized by the spatial range, the box size s is encoded as $\log(1+s)/\log 11$ to compress its dynamic range, and the yaw angle is represented by its sine and cosine for continuity. These features are embedded by an MLP, summed with a class embedding, and mapped to 1024-dimensional tokens by a zero-initialized output layer. The box tokens are appended to the cross-attention context of each view.

Bounded residual adapters. The layout tokens are injected into the LiDAR tokens $\mathbf{h}_L$ through residual adapters after blocks 7, 15, and 23 of the diffusion transformer. To preserve the camera-conditioned behavior, the magnitude of each residual is bounded relative to that of the LiDAR tokens. Formally, each adapter is expressed as

$$\mathbf{h}_L \leftarrow \mathbf{h}_L + \gamma(i)\, s\, \mathbf{x}, \qquad s = \min\Big(1,\, \frac{\kappa\, \mathrm{RMS}(\mathbf{h}_L)}{\max(\mathrm{RMS}(\mathbf{x}), \epsilon)}\Big), \qquad \mathbf{x} = \mathbf{W}\, \mathrm{LN}(\mathbf{l}), \tag{4}$$

where $\mathbf{W}$ is a zero-initialized linear projection, $\kappa = 0.05$ bounds the relative magnitude, $\epsilon = 10^{-8}$ ensures numerical stability, and $\gamma(i) = \min(1, i/300)$ linearly warms up the residual over the first 300 iterations. The layout model is fine-tuned from the camera-conditioned checkpoint, and the zero-initialized adapters leave the LiDAR tokens unchanged at initialization.

B Real-World Data Experiments

B.1 Camera-to-LiDAR Generation

The nuScenes dataset records each scan with a single 32-beam LiDAR, which is considerably sparser than the sensor suites of current production vehicles. To examine LiFTeR under a denser sensor configuration, the camera-to-LiDAR generation stage is further trained on real-world driving data. Each vehicle carries six surround-view cameras, including three wide-angle cameras with a 120° field of view, two rear-side cameras with a 70° field of view, and one rear telephoto camera with a 30° field of view, and all images are resized to 1280×720. The returns of four LiDARs are merged in the vehicle frame and projected into a 128×1792 range image following the elevation angles of a 128-beam sensor, with a maximum range of 100 m. The data are organized into clips of 300 frames at 10 Hz. Fig. 11 shows the generated scans together with the generated points projected onto the six camera images.

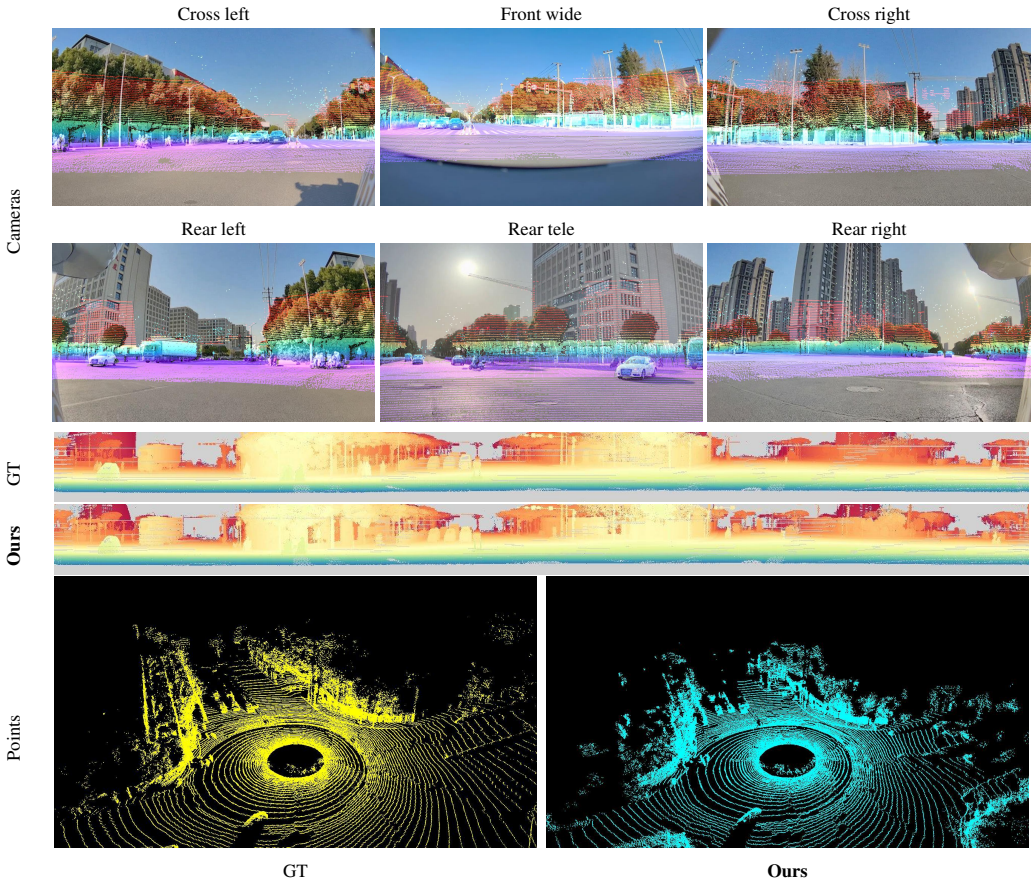


Figure 11: **Camera-to-LiDAR generation on multi-LiDAR driving data.**

B.2 Downstream Tasks

To examine whether the generated scans are usable by downstream LiDAR perception, we apply a CenterPoint (Yin et al., 2021) detector from OpenPCDet (Team et al., 2020), pretrained on nuScenes, directly to the generated LiDAR without any fine-tuning. For reference, the same detector is run on the corresponding real LiDAR. Both inputs are single keyframes and share the same preprocessing and confidence threshold.

To inspect spatial consistency with the conditioning images, we transform the points and predicted boxes into the front-camera frame using the sensor calibration and ego pose, and project them

onto the image (Fig. 12). On the generated scans, nearby objects form coherent returns, and the detector produces boxes that align with the visible vehicles in position and heading, consistent with its detections on real LiDAR. These qualitative results show that an off-the-shelf detector operates on the generated scans without adaptation.

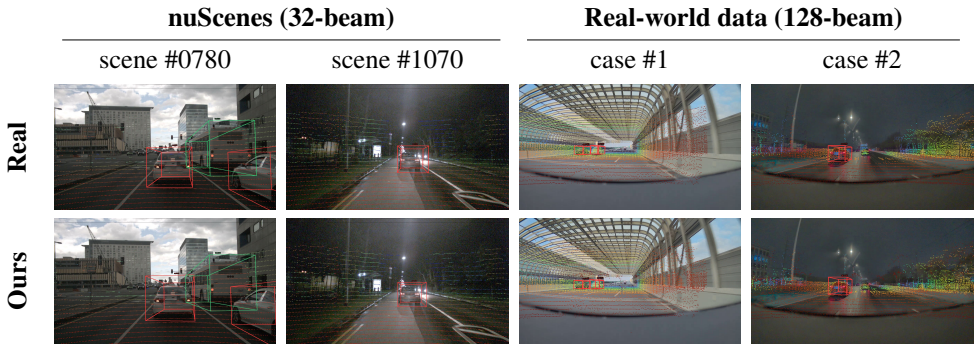


Figure 12: **Off-the-shelf 3D detection on generated LiDAR.** A pretrained CenterPoint is applied, without fine-tuning, to real LiDAR (top) and to LiDAR generated by LiFTeR (bottom), on nuScenes (32-beam, left) and real-world data (128-beam, right). Points and predicted boxes are projected onto the front camera.

C Limitations and Failure Cases

Temporal horizon. The LiDAR VAE and the Cosmos VAE encode each frame independently, and the latent sequence grows linearly with the number of frames. This confines generation to a short temporal horizon. A spatiotemporal LiDAR autoencoder and a video tokenizer would bound the sequence length and enable long-horizon 4D LiDAR Gaussian fields.

Dependence on camera observations. The generated scan depends on the scene information in the camera views. Fig. 13 shows two failure cases. In night scenes, only regions lit by headlights and street lamps are visible, and distant buildings and vegetation are missed or distorted. In high-speed scenes, motion blur removes thin structures such as poles, and the boundaries of surrounding objects are misplaced. Such scenes are also rare in nuScenes. Conditioning on sensors robust to illumination and weather, such as radar, is a promising direction.

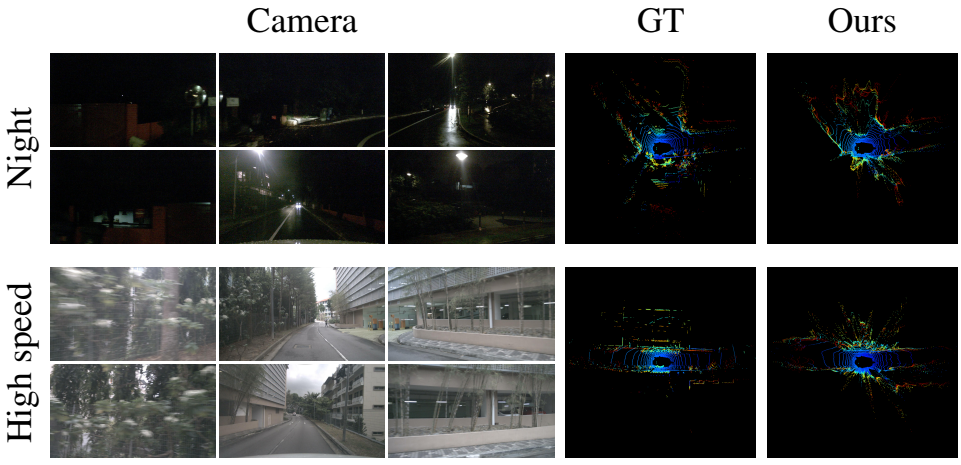


Figure 13: **Failure cases of LiFTeR.** Night scene (scene #1062, top) and high-speed scene (scene #0924, bottom). In both cases, the camera images provide insufficient information.

D GEOMETRIC LOSSES

Let $\hat{D}$ and D denote the rendered and measured range maps on the $H \times W$ ray grid, and let M be the mask of rays with valid returns. Following GS-LiDAR (Jiang et al., 2025), the geometric term is

$$\mathcal{L}_{\text{geo}} = \lambda_{\text{grad}}\mathcal{L}_{\text{grad}} + \lambda_{\text{nrm}}\mathcal{L}_{\text{nrm}} + \lambda_{\text{cd}}\mathcal{L}_{\text{cd}} + \lambda_{\text{dist}}\mathcal{L}_{\text{dist}}. \tag{5}$$

Range-gradient loss. To preserve depth discontinuities and local surface slopes, $\mathcal{L}_{\text{grad}}$ matches the gradients of the rendered and measured range maps,

$$\mathcal{L}_{\text{grad}} = \frac{1}{|M|} \sum_{(h,w)\in M} \left\| \nabla\hat{R}(h,w) - \nabla R(h,w) \right\|_1. \tag{6}$$

Normal-consistency loss. Each 2D Gaussian has a well-defined normal, and the rendered normal map $\hat{N}$ is obtained by alpha-blending the normals of the Gaussians along each ray. Following 2DGS (Huang et al., 2024), $\mathcal{L}_{\text{nrm}}$ aligns $\hat{N}$ with the pseudo-normal map $\tilde{N}$ computed from the gradients of the range map,

$$\mathcal{L}_{\text{nrm}} = \frac{1}{|M|} \sum_{(h,w)\in M} \Big(1 - \hat{N}(h,w)^\top \tilde{N}(h,w)\Big). \tag{7}$$

Truncated Chamfer loss. The rendered and measured range maps are back-projected into point clouds $\hat{\mathcal{P}}$ and $\mathcal{P}$, and the Chamfer distance (Fan et al., 2017) is computed with a truncation threshold τ to suppress the influence of outliers,

$$\mathcal{L}_{\text{cd}} = \frac{1}{|\hat{\mathcal{P}}|} \sum_{\hat{p}\in\hat{\mathcal{P}}} \min\Big(\min_{p\in\mathcal{P}} \|\hat{p} - p\|_2,\, \tau \Big) + \frac{1}{|\mathcal{P}|} \sum_{p\in\mathcal{P}} \min\Big(\min_{\hat{p}\in\hat{\mathcal{P}}} \|p - \hat{p}\|_2,\, \tau \Big). \tag{8}$$

Depth-distortion loss. Following 2DGS (Huang et al., 2024), $\mathcal{L}_{\text{dist}}$ concentrates the Gaussians along each ray on the surface by reducing the distance between ray-splat intersections,

$$\mathcal{L}_{\text{dist}} = \sum_{i,j} \omega_i \omega_j \, |\ell_i - \ell_j|, \tag{9}$$

where ω_i is the blending weight of the i-th intersection and ℓ_i is its distance along the ray.