LiDAR Scene Synthesis with Triplane Diffusion Models via LiDAR Simulation-based Coarse-to-Fine Point Sampling

Anonymous authors
Paper under double-blind review

Abstract

Realistic LiDAR scene synthesis is critical for autonomous driving perception systems, yet existing methods struggle to simultaneously capture the distinctive scanning patterns, geometric structures, and semantic properties of real-world data. To tackle these challenges, we present the first two-stage framework, called **LiTDM**, that leverages triplane representation as a latent space proxy for diffusion models tailored to LiDAR scene synthesis. First, a triplane-based autoencoder is trained to predict point-based occupancy and intensity with a novel ray sampling strategy. Then, we sample the diffusion model trained on triplane features to generate realistic LiDAR scenes. With triplane features as a proxy, our method achieves pattern realism, geometric accuracy, and semantic coherence. Moreover, our proposed LiDAR Simulation-based Point Sampling (LSPS) ensures the characteristic scanning patterns of real sensors. Experimental results show that our framework generates highly realistic LiDAR scenes against the baselines. Furthermore, the flexibility of LiTDM extends beyond unconditional synthesis, enabling diverse conditional generation tasks such as camera-to-LiDAR and text-to-LiDAR generation. Code will be released to support reproducibility.

1 Introduction

The increasing deployment of autonomous vehicles demands a sophisticated understanding of the 3D environment, with LiDAR sensors playing a pivotal role in perception systems. Beyond conventional physics-based simulation of LiDAR patterns (Manivasagam et al., 2020; Li et al., 2022), generating realistic synthetic LiDAR data has emerged as a critical component for robust training and validation of perception of autonomous systems. Such synthetic data must not only replicate the distinctive scanning patterns of LiDAR sensors but also preserve semantic and geometric fidelity to real-world environments. Despite notable advances in generative models (Goodfellow et al., 2014; Kingma & Welling, 2013; Ho et al., 2020), achieving this level of realism remains a formidable challenge in autonomous driving. To ensure robust performance across varied environments, these systems must be exposed to a wide range of scenarios during development. However, acquiring and annotating real-world LiDAR data is both resource-intensive and inherently limited in diversity. Synthetic data generation offers a scalable means to augment training datasets, yet existing LiDAR scene synthesis methods often fail to concurrently capture the intricate sensor-specific patterns, geometric structures, and semantic attributes required for high-fidelity simulation.

Traditional methods for LiDAR scene generation have evolved along several distinct trajectories. Physics-based simulation approaches (Manivasagam et al., 2020; Li et al., 2022) effectively model sensor characteristics and environmental interactions, yet they often struggle with computational efficiency and encounter notable domain gaps. Recently, diffusion-based methods, notably LiDARGen (Zyrianov et al., 2022), demonstrated promising LiDAR generation ability, but have been limited by challenges in maintaining structural coherence and spatial consistency. The advent of LiDAR Diffusion Models (LiDM) (Ran et al., 2024), which employ curve-based representations, has improved the preservation of characteristic scanning patterns and object geometries, thereby enhancing pattern, geometric, and object realism through specialized compression and encoding techniques.

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.

The triplane representation, a compact 3D feature grid factorized onto three orthogonal 2D planes, has recently gained traction as an effective framework for a variety of 3D tasks. Pioneering work such as EG3D (Chan et al., 2022) has demonstrated the efficacy of triplanes for 3D-aware image synthesis, and subsequent studies like SemCity (Lee et al., 2024) have integrated triplane representations into diffusion models for semantic scene generation in outdoor environments. Gleaning insights on the efficient triplane representation, we introduce a novel triplane-based diffusion model approach to LiDAR scene synthesis, which remains largely unexplored to date. Specifically, we leverage the following advantages of triplane representation for effective and efficient LiDAR scene synthesis: (1) a compact, memory-efficient encoding of 3D information that preserves spatial relationships while mitigating the sparsity issues inherent to point-based methods, (2) their structured nature facilitates the efficient learning of complex 3D distributions using conventional 2D convolutional architectures, and (3) they adeptly capture both local geometric details and global semantic context, which are critical for generating coherent and realistic LiDAR scenes.

Building on the triplane representation, we formulate **LiTDM**, LiDAR Triplane Diffusion Models, as a two-stage framework for LiDAR scene synthesis. In the first stage, a triplane autoencoder encodes sparse LiDAR point clouds into triplane features through a sparse convolutional encoder and axis-wise pooling. Given sampled query points, the decoder retrieves the corresponding triplane features and predicts point-wise occupancy, point-wise intensity, and ray-wise emptiness. In the second stage, we arrange the learned triplane features into a tiled 2D feature map and train a diffusion model to learn their distribution. During generation, triplane features sampled by the diffusion model are decomposed back into plane-wise features and passed to the pre-trained decoder to synthesize LiDAR point clouds.

To synthesize points from triplane features efficiently, we introduce LiDAR Simulation-based Point Sampling (LSPS). LSPS first constructs LiDAR rays from a range-view grid by modeling the azimuth and elevation angles of each pixel. Along each ray, it samples a set of candidate points and queries their triplane features for occupancy prediction. The sampling is performed in a coarse-to-fine manner, where coarse samples cover the ray segment and fine samples are placed in a smaller region around the selected coarse location. The final LiDAR point on each ray is obtained by selecting the candidate with the highest predicted occupancy, while the ray-wise emptiness prediction is used to suppress invalid returns.

We contribute to the field of LiDAR scene synthesis in that:

- We introduce LiTDM, a triplane latent diffusion framework tailored to LiDAR scene synthesis. By learning the data distribution in the triplane feature space, LiTDM combines compact 3D scene representation with efficient 2D diffusion-based generation.
- We propose LiDAR Simulation-based Point Sampling, a ray-based coarse-to-fine sampling strategy for decoding triplane features into LiDAR points.
- Through extensive comparisons and ablation studies, we show that LiTDM improves LiDAR scene generation quality across perceptual and statistical metrics, while also demonstrating conditional generation settings such as camera-to-LiDAR and text-to-LiDAR synthesis.

2 Related Work

2.1 Data Representation for LiDAR Generation

The choice of representation for LiDAR point clouds significantly impacts the quality, efficiency, and applicability of generative models with varying trade-offs between fidelity, computational efficiency, and semantic preservation.

Compact Representations. Curve-based representations in LiDMs (Ran et al., 2024), preserve the characteristic scanning patterns of LiDAR sensors through curve-wise compression, thus significantly improving pattern realism while achieving a huge speedup compared to point-based diffusion models. UltraLiDAR (Xiong et al., 2023) introduces a data-driven framework using discrete representations from large-scale LiDAR point clouds. The approach utilizes a compact, discrete representation (codebook) that encodes geometric structure, is robust to noise, and facilitates easy

manipulation. CURL (Zhang et al., 2022) transforms LiDAR scans into compact spherical harmonics representations, consequently enabling continuous reconstruction of much denser point clouds while achieving high storage space savings. Hence, this approach addresses both the sparsity challenges of raw LiDAR data and the need for efficient storage and transmission.

Triplane representations (Chan et al., 2022; Gupta et al., 2023; Shue et al., 2023; Wang et al., 2023; Lee et al., 2024) have been demonstrated to efficiently re-parameterize volumetric data by decomposing a 3D scene into more manageable forms of 2D orthogonal feature planes, facilitating efficient 3D generation tasks while maintaining rich geometric details. This approach enables efficient 2D sampling, substantially reducing both computational and memory overhead compared to processing dense voxel grids. Notably, triplane representation-based 3D generative modeling works (Chan et al., 2022; Lee et al., 2024) achieved high-fidelity reconstruction of complex geometric and semantic details while maintaining scalability, and further integrated triplane geometry into diffusion models to synthesize urban-scale 3D scenes, respectively.

2.2 Diffusion Models for Realistic Data Generation

3D diffusion models have recently paved a crucial pathway towards LiDAR scene generation for autonomous driving applications (Wu et al., 2024; Bian et al., 2024; Zyrianov et al., 2022; Xiong et al., 2023; Nakashima & Kurazume, 2024; Ran et al., 2024; Yan et al., 2024; Nakashima et al., 2024; Liu et al., 2025; Xiang et al., 2025; Wu et al., 2025; Zhu et al., 2025; Nakashima et al., 2025; Zhang & Liu, 2024). In this field, LiDARVAE and LiDARGAN (Caccia et al., 2019) first applied generative models to generate larger-scale and complex LiDAR scenes, showcasing the potential of generative models in the 3D domain. To generate more realistic LiDAR scenes, UltraLiDAR (Xiong et al., 2023) learns discrete representations from vector-quantized variational autoencoder (Van Den Oord et al., 2017), and LiDARGen (Zyrianov et al., 2022) employs a score-matching energy-based diffusion model (Song & Ermon, 2019) to produce diverse and high-quality point cloud samples with controllable attributes. Diffusion-based R2DM (Nakashima & Kurazume, 2024) further generates diverse and high-fidelity 3D scene point clouds using image representations of range and reflectance intensity. LiDMs (Ran et al., 2024) operate in a latent space tailored to capture LiDAR scene realism by incorporating geometric priors, focusing on pattern, geometry, and object realism. Text2LiDAR (Wu et al., 2024) introduces an equirectangular transformer for text-controllable LiDAR point cloud generation, enabling semantic control through textual prompts. These approaches collectively aim to improve the quality, efficiency, and controllability of LiDAR scene generation, addressing challenges such as maintaining geometric consistency, handling sparse inputs, and generating diverse yet realistic outputs for autonomous driving research and development.

3 Proposed Method

In our LiTDM framework, we propose an effective approach for synthesizing realistic LiDAR scenes by learning compact triplane representations of LiDAR data. We first design a triplane autoencoder that predicts point-wise occupancy and intensity values. To avoid inefficient querying across the entire 3D space, we introduce a LiDAR Simulation-based coarse-to-fine Point Sampling (LSPS) strategy that samples points along LiDAR rays and refines them to synthesize accurate LiDAR points. After training the autoencoder, we train a diffusion model on the learned triplane representations. At inference time, the generated triplane features are decoded by the pre-trained triplane decoder, and LSPS synthesizes realistic LiDAR points guided by the predicted occupancy values. An overview of our LiTDM framework is illustrated in Fig. 1.

3.1 LiDAR Triplane Feature Learning

To learn accurate LiDAR triplane features, we introduce a triplane autoencoder network in the triplane feature learning stage, as shown in Fig. 1. The proposed triplane autoencoder encodes sparse LiDAR data into a triplane representation and uses LiDAR simulation-based sampling to retrieve corresponding triplane features for occupancy, intensity, and mask prediction.

Triplane Feature Extraction. The set of LiDAR points, denoted as $\{P_i\}_{i=1}^{N} \in \mathbb{R}^{N\times 3}$, serves as the input to the triplane autoencoder. These points are encoded by the triplane sparse convolutional encoder f_θ into point features $\{F_i\}_{i=1}^{N}$. To compress $\{F_i\}_{i=1}^{N}$ into triplane features, three orthogonal

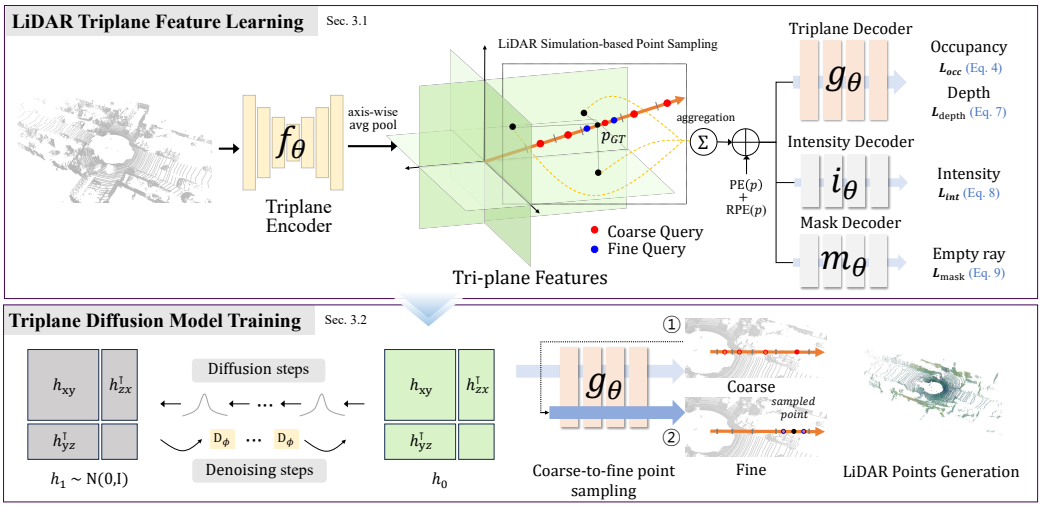


Figure 1: **Overview of LiTDM.** The proposed LiTDM framework consists of two stages. In the feature learning stage, a triplane autoencoder encodes LiDAR point clouds into triplane representations, which are queried at sampled point locations to predict point-wise occupancy and intensity. During triplane diffusion model training, the model learns the distribution of triplane features, enabling the generation of novel LiDAR points. In both stages, we use LiDAR simulation-based coarse-to-fine point sampling (LSPS): coarse points are first sampled along each ray, and fine points are then sampled around the coarse point with the highest occupancy.

2D grids are generated along the z-axis, y-axis, and x-axis. The center point of the LiDAR data, o, is positioned at the center of each grid, and the features $\{F_i\}_{i=1}^{N}$ are projected onto these grids. Finally, by averaging the features projected onto each grid, the triplane features $h_{xy} \in \mathbb{R}^{X\times Y\times C}$, $h_{yz} \in \mathbb{R}^{Y\times Z\times C}$, and $h_{zx} \in \mathbb{R}^{Z\times X\times C}$ are obtained.

LiDAR Simulation-based Point Sampling. The extracted triplane feature can be used to predict the occupancy of an arbitrary point $\mathbf{p} = (x, y, z)$ in space by aggregating its feature from h_{xy}, h_{yz}, and h_{zx}. To obtain LiDAR points from triplane features, we sample multiple points along each LiDAR ray, evaluate their occupancy, and select the point with the highest occupancy. This avoids inefficient querying across the entire 3D space while preserving the geometric structure inherent in LiDAR data.

LiDAR sensors acquire data through ray tracing, ensuring that each LiDAR ray intersects only one point. We simulate this characteristic of LiDAR sensors to sample points and perform ray-wise occupancy prediction on the sampled points. In the LiDAR ray equation, $R = \mathbf{o} + t\mathbf{d}$, the direction vector $\mathbf{d} = (d_x, d_y, d_z) \in \mathbb{R}^3$ can be modeled using the azimuth angle θ and elevation angle ϕ. Next, we create a range view image grid of $W \times H$, as in Kong et al. (2023). By modeling rays corresponding to the θ and ϕ values of each pixel in the grid, we can generate $W \times H$ rays. θ_u and ϕ_v corresponding to the (u, v) coordinates in the grid are calculated as follows:

$$\begin{pmatrix} \theta_u \\ \phi_v \end{pmatrix} = \begin{pmatrix} \pi - \frac{u}{W} 2\pi \\ (1 - \frac{v}{H}) f + f_{down} \end{pmatrix}, \tag{1}$$

where u and v range from 0 to $W-1$ and 0 to $H-1$, respectively. f_{down} and f denote downward field-of-view (FOV) and vertical FOV. The direction vector $\mathbf{d}_{u,v}$ of a ray corresponding to angles θ_u and ϕ_v is calculated as Eq. 2:

$$\begin{aligned} d_x(u, v) &= \cos(\theta_u) \times \cos(\phi_v) \\ d_y(u, v) &= -\sin(\theta_u) \times \cos(\phi_v) \\ d_z(u, v) &= \sin(\phi_v) \end{aligned} \tag{2}$$

The proposed LiDAR simulation-based coarse-to-fine sampling method consists of two stages: (1) coarse sampling, where points along the ray are selected using stratified sampling, and (2) fine sampling, where a denser set of points is sampled within a smaller region. This coarse-to-fine strategy

facilitates the generation of realistic and accurate LiDAR points while maintaining computational efficiency. The details of point sampling and region selection differ between the training phase, where ground-truth LiDAR points are available, and the inference phase, where they are not.

3.1.1 TRAINING TRIPLANE AUTOENCODER.

When training the triplane autoencoder, we sample points from the modeled $W \times H$ rays. In the coarse sampling stage, we sample M_c points $\mathcal{S}_c(R_{u,v}) = \{\mathbf{p}_\ell^c\}_{\ell=1}^{M_c}$ along each ray $R_{u,v} = \mathbf{o} + t\mathbf{d}_{u,v}$, using the stratified sampling strategy. We divide the ray segment defined by $t \in [t_{min}, t_{max}]$ into M_c bins of uniform length. Within each of these M_c bins, we uniformly sample one point.

Subsequently, in the fine sampling stage, we sample M_f additional points $\mathcal{S}_f(R_{u,v}) = \{\mathbf{p}_\ell^f\}_{\ell=1}^{M_f}$ for each ray $R_{u,v}$ in the vicinity of P_i^{proj}, where P_i^{proj} is obtained by projecting the given occupied point P_i onto the nearest ray. Specifically, the fine sampling range is defined by the midpoints between the coarse sample in the bin containing P_i^{proj}, and its two adjacent coarse samples. This methodology ensures a more concentrated search space around the expected surface. The pseudo-code for the proposed LSPS is presented in *Appendix*. This sampling methodology results in a total of $M_c + M_f$ points sampled within each ray in the 3D space. Considering $W \times H$ rays, the total number of sampled points is $W \times H \times (M_c + M_f)$.

To generate LiDAR points, we decode triplane features using the MLP decoder g_θ for occupancy prediction. For each sampled point $\mathbf{p} = (x, y, z) \in \mathbb{R}^3$ from M_c and M_f, we query the triplane features and concatenate them with positional embeddings as input to g_θ. The feature of $\mathbf{p}$ is obtained by summing bilinearly interpolated features from three planes: $h(\mathbf{p}) = h_{xy}(x, y) + h_{yz}(y, z) + h_{zx}(z, x)$. For positional embedding, we use both PE and RPE (Lee et al., 2024), while further details are in *Appendix*. The concatenation of $(h(\mathbf{p}), \mathrm{PE}(\mathbf{p}), \mathrm{RPE}(\mathbf{p}))$ is fed into g_θ, which predicts occupancies for all $W \times H \times (M_c + M_f)$ sampled points across $W \times H$ rays. Finally, for each ray, we select the point with the highest occupancy among the $(M_c + M_f)$ candidates, ensuring that each ray contributes at most one occupied point.

Considering the limited depth range measurable by LiDAR sensors, rays without any detected points are possible. To address this, we additionally train a mask decoder that predicts an emptiness logit $m_{u,v} = m_\theta(R_{u,v})$ from the max-pooled features for each ray $R_{u,v}$. Then, we threshold the predicted empty-ray probability $\mathrm{sigmoid}(m_{u,v})$ with a predefined mask threshold τ and refrain from generating points for rays classified as invalid, ensuring that the generated LiDAR scenes accurately reflect sensor limitations. Furthermore, to generate intensity values for LiDAR points, we train an intensity decoder i_θ that regresses the intensity of each selected occupied point from its triplane feature and positional embeddings. Given a generated point p, concatenated feature (h(p), PE(p), and RPE(p)) is used as the input to i_θ, which outputs the corresponding intensity value. This allows the generated LiDAR scenes to model both geometric occupancy and point-wise reflectance.

Gaussian-kernel Occupancy Labels. For each valid ray R, we assign soft occupancy labels to sampled points according to their distance from the ground-truth LiDAR point $\mathbf{p}_{GT}$ on the ray. Specifically, the occupancy label of a sampled point $\mathbf{p}$ is defined as

$$y_{\mathrm{occ}}(\mathbf{p}) = \exp\left(-\frac{\|\mathbf{p} - \mathbf{p}_{GT}\|^2}{2\sigma^2}\right), \tag{3}$$

where σ controls the spread of the Gaussian kernel. For invalid rays without a ground-truth return, all sampled points are assigned an occupancy label of 0.

Loss Function. We then train the autoencoder using a combination of the following losses. First, we utilize Binary Cross-Entropy (BCE) loss which is suitable for binary classification tasks with labels in the range $[0, 1]$. BCE loss measures the dissimilarity between predicted occupancies and actual labels, defined as Eq. 4:

$$\mathcal{L}_{\mathrm{occ}} = \mathrm{BCE}(\hat{o}(\mathbf{p}), y_{\mathrm{occ}}(\mathbf{p})), \tag{4}$$

where $\hat{o}(\mathbf{p})$ denotes the predicted occupancy of sampled point $\mathbf{p}$.

For valid rays, we further impose expected depth and intensity losses. Given the fine sampled points $\mathcal{S}_f(R)$ on ray R, we compute a soft occupancy weight for each fine sample:

$$w_R(\mathbf{p}) = \frac{\exp(\hat{o}(\mathbf{p}))}{\sum_{\mathbf{q} \in \mathcal{S}_f(R)} \exp(\hat{o}(\mathbf{q}))}, \quad \mathbf{p} \in \mathcal{S}_f(R). \tag{5}$$

The expected depth of the ray R is computed as

$$\hat{d}_R = \sum_{\mathbf{p} \in \mathcal{S}_f(R)} w_R(\mathbf{p})\, d(\mathbf{p}), \tag{6}$$

where $d(\mathbf{p})$ denotes the depth of sampled point $\mathbf{p}$ along ray R. The depth loss is then defined as

$$\mathcal{L}_{\text{depth}} = \text{SmoothL1}(\hat{d}_R, d_R), \tag{7}$$

where d_R is the ground-truth depth of ray R.

For a sampled point $\mathbf{p}$ whose nearest ground-truth point $\mathbf{p}_{\text{GT}}$ satisfies $\|\mathbf{p} - \mathbf{p}_{\text{GT}}\|_2 < \epsilon$ with a predefined window size ϵ, we supervise its predicted intensity $\hat{I}(\mathbf{p})$ using the ground-truth intensity I. The intensity loss is defined as:

$$\mathcal{L}_{\text{int}} = \text{L2}(\hat{I}(\mathbf{p}), I). \tag{8}$$

Furthermore, we supervise ray-wise emptiness prediction with BCE loss. Given the predicted empty-ray logit $m_\theta(R)$ for each ray R and its ground-truth empty-ray label y_R, the mask loss is defined as

$$\mathcal{L}_{\text{mask}} = \text{BCE}(\text{sigmoid}(m_\theta(R)), y_R). \tag{9}$$

The final training objective is

$$\mathcal{L}_{\text{AE}} = \mathcal{L}_{\text{occ}} + \lambda_{\text{depth}}\mathcal{L}_{\text{depth}} + \lambda_{\text{int}}\mathcal{L}_{\text{int}} + \lambda_{\text{mask}}\mathcal{L}_{\text{mask}}. \tag{10}$$

By jointly optimizing occupancy, depth, intensity, and mask prediction losses, our model learns to synthesize LiDAR points effectively, capturing the spatial characteristics inherent in LiDAR data.

3.2 Triplane Diffusion Models

To generate novel LiDAR scenes, we employ the denoising diffusion U-Net and DDPM formulation of R2DM (Nakashima & Kurazume, 2024), leveraging well-established 2D diffusion architectures commonly used in LiDAR scene synthesis. To adapt triplane representations to this framework, we arrange the plane-wise features into a 2D triplane feature map used by the diffusion model. After sampling, the generated feature map is decomposed back into triplane features and passed through the decoder to predict the occupancy and intensity of arbitrary 3D points. Finally, our LiDAR simulation-based coarse-to-fine point sampling strategy synthesizes accurate LiDAR points guided by the predicted occupancies and intensities. Details on the triplane diffusion model and the coarse-to-fine sampling strategy are provided below.

Triplane Feature Map. Given the triplane features h_{xy}, h_{yz}, and h_{zx}, we construct a 2D tiled triplane feature map as

$$h_{\text{tile}} = \begin{bmatrix} h_{xy} & h_{zx}^\top \\ h_{yz}^\top & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{(X+Z)\times(Y+Z)\times C},$$

where $\top$ denotes transposition over the spatial dimensions and $\mathbf{0} \in \mathbb{R}^{Z\times Z\times C}$ is a zero-padding tensor. The diffusion model takes h_{tile} as input and learns its distribution.

Diffusion Model Training. We train the diffusion model to learn the distribution of tiled triplane feature maps using the v-parameterization (Salimans & Ho, 2022) under a continuous-time Gaussian diffusion process with a log-SNR cosine schedule (Kingma et al., 2021). Let h_0 be a clean tiled triplane feature map. For $t \sim \mathcal{U}(0, 1)$, we corrupt h_0 as

$$h_t = \alpha_t h_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{11}$$

where $\alpha_t = \sqrt{\text{sigmoid}(\lambda(t))}$, $\sigma_t = \sqrt{\text{sigmoid}(-\lambda(t))}$, and $\lambda(t)$ is the log-SNR cosine schedule. The velocity target is defined as

$$v_t = \alpha_t \epsilon - \sigma_t h_0, \tag{12}$$

and the denoising network D_ϕ is trained to predict v_t from the noisy feature map h_t:

$$\mathcal{L}_{\text{D}} = \mathbb{E}_{h_0,\epsilon,t}\left[\|v_t - D_\phi(h_t, \mathbf{c}, t)\|_2^2\right], \tag{13}$$

where $\mathbf{c}$ denotes an optional condition for conditional generation. After training D_ϕ, novel tiled triplane feature maps are generated from Gaussian noise $h_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ through iterative denoising from $t = 1$ to $t = 0$, and are then decomposed back into triplane features.

LiDAR Scene Synthesis. For each triplane feature sampled using the triplane diffusion model, LiDAR points are generated through our coarse-to-fine point sampling strategy. We first sample M_c points $\{\mathbf{p}_i^c\}_{i=1}^{M_c}$ along each ray R using stratified sampling. These sampled points are then queried to obtain their corresponding triplane features $h(\mathbf{p}_i^c)$, which are fed into the decoder to compute the occupancies $\hat{o}(\mathbf{p}_i^c)$. Among the sampled points, the one with the highest occupancy is identified:

$$i^\star = \underset{i \in \{1,\dots,M_c\}}{\arg\max}\ \hat{o}(\mathbf{p}_i^c). \tag{14}$$

Then, fine sampling is performed in its vicinity to generate an additional M_f points $\{\mathbf{p}_j^f\}_{j=1}^{M_f}$. The region for fine sampling is defined identically to that in training. Finally, occupancies $\hat{o}(\mathbf{p}_j^f)$ are computed for M_f points, and the point with the highest occupancy is selected as the occupied point:

$$\hat{\mathbf{p}}_R = \mathbf{p}_{j^\star}^f, \qquad j^\star = \underset{j \in \{1,\dots,M_f\}}{\arg\max}\ \hat{o}(\mathbf{p}_j^f). \tag{15}$$

The intensity values are computed using soft occupancy weights over fine samples, as in training:

$$\hat{I}_R = \sum_{j=1}^{M_f} w_R(\mathbf{p}_j^f)\, \hat{I}(\mathbf{p}_j^f). \tag{16}$$

Finally, we apply ray-wise mask filtering to suppress invalid returns. Specifically, the selected point $\hat{\mathbf{p}}_R$ is retained only when $\mathrm{sigmoid}(m_\theta(R)) < \tau$ and is discarded otherwise. The mask prediction is computed using all $M_c + M_f$ samples, consistent with the training phase.

4 Experiments

4.1 Experimental Settings

Datasets. Following the baselines (Zyrianov et al., 2022; Xiong et al., 2023; Nakashima & Kurazume, 2024; Ran et al., 2024), we validate our method on the KITTI-360 (Liao et al., 2022) dataset, which comprises nine driving sequences of 81,106 LiDAR point clouds acquired from a Velodyne HDL-64E sensor. The data are split by the standard protocol (Xiong et al., 2023; Zyrianov et al., 2022; Nakashima & Kurazume, 2024). Additional experiments on the nuScenes (Caesar et al., 2020) dataset are provided in the *Appendix*.

Evaluation Metrics. We evaluate the distributional similarity between real and generated LiDAR scenes using BEV-based metrics (JSD, MMD) as well as perceptual quality measures (FSVD, FPVD), following Wu et al. (2024); Ran et al. (2024).

Implementation Details. We train our LiDAR scene generation using four NVIDIA A6000 GPUs. We use the Adam (Kingma & Ba, 2014) optimizer with a learning rate of $1{\times}10^{-4}$ for training the autoencoder, and $1{\times}10^{-5}$ for the diffusion model with a cosine decay scheduler. In our method, the triplane encoder f_θ utilizes the Minkowski UNet18 (Choy et al., 2019a) architecture. For the denoising model $\mathrm{D}_\phi(\cdot)$, we employ the Efficient U-Net architecture (Saharia et al., 2022) following the baseline (Nakashima & Kurazume, 2024). We use range view grid size of $(H, W) = (64, 1024)$ and triplane shape of $(X, Y, Z) = (160, 160, 32)$ with $C = 16$. While training, we sample $M_c = 64$ and $M_f = 8$ points, whereas for synthesis, we use $M_c = 160$ and $M_f = 20$. To distinguish rays without occupied points, we set the threshold $\tau = 0.5$ and utilize $M_c = 64$ and $M_f = 8$ points. For generating occupancy ground truth labels, we use a threshold $\sigma = 0.1$. We use a window size $\epsilon = 0.5$m for the intensity supervision. We use $\lambda_{depth} = 1.0$, $\lambda_{int} = 1.0$, and $\lambda_{mask} = 0.1$.

4.2 LiDAR Generation Task

Quantitative Results. Table 1 presents the performance of our LiTDM, demonstrating significant improvements over methods utilizing range view representations. Specifically, compared to the existing SOTA baseline model (Nakashima & Kurazume, 2024), our approach achieves a 14.46%

Table 1: Quantitative comparison of LiDAR generation models.

Methods	Perceptual Metric		Statistical Metric	
	FSVD ↓	FPVD ↓	JSD ($\times 10^{-2}$) ↓	MMD ($\times 10^{-4}$) ↓
LiDARVAE (Caccia et al., 2019)	289.68	290.87	15.18	19.20
LiDARGAN (Caccia et al., 2019)	347.98	336.38	21.86	78.20
ProjectedGAN (Sauer et al., 2021)	196.32	206.67	7.45	6.85
LiDM (Ran et al., 2024)	217.28	231.91	17.89	35.40
R2DM (Nakashima & Kurazume, 2024)	34.31	37.22	4.73	4.58
Text2LiDAR (Wu et al., 2024)	113.82	102.41	4.67	4.16
LiTDM (ours)	**29.35**	**35.23**	**4.59**	**1.93**

Table 2: Ablation study on the feature representation.

Representation	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-4}$)
Range View (RV)	34.31	37.22	4.73	4.58
Bird's-Eye View (BEV)	32.48	40.53	6.85	6.19
Voxel	52.76	52.53	5.55	5.21
Curve-based Latent	41.82	47.47	4.63	3.42
LiTDM (Triplane)	**29.35**	**35.23**	**4.59**	**1.93**

Table 3: Ablation study on the ray composition.

$H \times W$	FSVD ↓	FPVD ↓
32×1024	56.89	65.68
64×1024	29.35	35.23
64×2048	**28.84**	**28.51**

Table 4: Ablation study on the diffusion sampling steps.

Steps	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-4}$)
128	32.79	39.27	5.94	1.96
256	30.98	37.16	5.21	1.92
512	29.75	35.74	4.75	**1.83**
1024	**29.35**	**35.23**	**4.59**	1.93

Table 5: Ablation study on the coarse-to-fine sampling points number.

(M_c, M_f)	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-4}$)
(64, 8)	55.28	59.85	4.54	**1.76**
(128, 16)	**28.99**	36.10	4.53	1.89
(160, 20)	29.35	**35.23**	4.59	1.93
(160, 1)	68.77	64.90	**4.50**	1.79

enhancement in FSVD and a 57.86% improvement in MMD. These advancements are attributed to our LiDAR simulation-based point sampling technique, which effectively captures 3D geometric information. We further evaluate the semantic coherence of the generated samples, with detailed results provided in the *Appendix*.

Qualitative Results. Fig. 2 presents the qualitative results of baselines (Nakashima & Kurazume, 2024; Ran et al., 2024; Wu et al., 2024) and our method. Thanks to our LiDAR simulation-based coarse-to-fine sampling, LiTDM effectively learns the distribution of LiDAR data, generating more realistic points when viewed in BEV.

4.3 Ablation Study

LiDAR Feature Representation. To validate our choice of the triplane representation, we compare it with range view (RV), bird's-eye view (BEV), voxel, and curve-based latent representations, where the curve-based variant follows LiDM (Ran et al., 2024), under the same training and evaluation setup. As shown in Table 2, the triplane representation consistently outperforms all variants across all similarity metrics. For the voxel-based variant, we replace the 2D convolutions in the denoiser with 3D convolutions; however, under the same latent budget, the dense voxel grid operates at a coarser resolution and allocates substantial capacity to empty space. This result underscores the triplane's superior capability in efficiently encoding fine-grained 3D geometric structures, which is critical for generating accurate and realistic LiDAR scenes.

Ray Composition. Our model learns 3D space with triplane features, enabling LiDAR point simulation for arbitrary ray compositions. As shown in Table 3, increasing the ray resolution consistently improves generation quality, with 64×2048 achieving the best FSVD and FPVD. This demonstrates the flexibility of our simulation-based sampling strategy across different LiDAR configurations.

Diffusion Sampling Steps. As shown in Table 4, increasing the number of diffusion sampling steps generally improves generation quality across most metrics. We use 1024 steps as the default setting, as it achieves the best FSVD, FPVD, and JSD while maintaining competitive MMD.

Coarse-to-Fine Sampling Points. Table 5 shows that using few samples, such as $(64, 8)$, degrades both metrics due to insufficient ray coverage. Increasing the number of coarse and fine samples improves generation quality, with $(160, 20)$ achieving the best overall performance. The poor results of $(160, 1)$ indicate that fine sampling is essential for accurate surface localization along each ray.

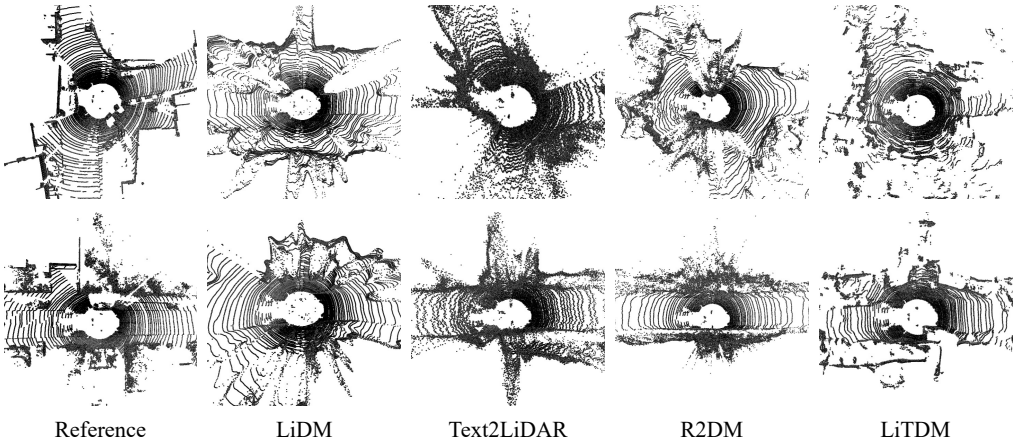
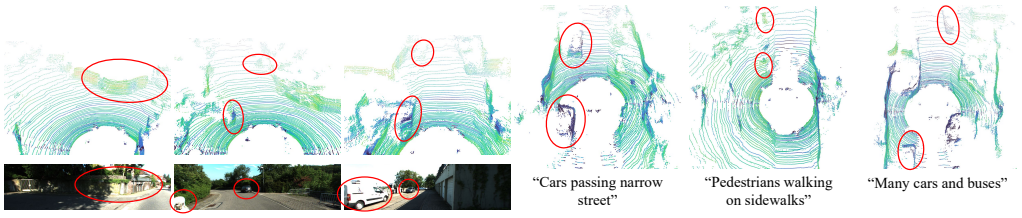


Figure 2: Qualitative comparison of LiDAR scene samples from the KITTI-360 reference data, LiDM (Ran et al., 2024), Text2LiDAR (Wu et al., 2024), R2DM (Nakashima & Kurazume, 2024), and our LiTDM, all trained on the KITTI-360 dataset.



(a) Camera Image (b) Text Prompt

Figure 3: Conditional generation results conditioned on (a) Camera Image and (b) Text Prompt.

4.4 Conditional Generation

Camera-to-LiDAR Generation. Our model is capable of camera-to-LiDAR generation by training the denoiser conditioned on camera images I. We achieve this by computing the normalized CLIP (Radford et al., 2021) embedding $\mathbf{c}_I = f_{\text{CLIP}}(I)/\|f_{\text{CLIP}}(I)\|_2$, and injecting it into the bottleneck via additional cross-attention layers (Rombach et al., 2022). Training uses conditional dropout:

$$\mathcal{L}_{\text{D}}^{\text{cam}} = \mathbb{E}_{h_0,\epsilon,t,I,b}\left\|v_t - D_\phi(h_t,\, b\,\mathbf{c}_I,\, t)\right\|_2^2, \quad b \sim \text{Bernoulli}(1 - p_{\text{drop}}), \tag{17}$$

which enables classifier-free guidance at sampling (Ho & Salimans, 2022).

As shown in Fig. 3 (a), LiTDM generates LiDAR point clouds that are both geometrically and semantically consistent with the input images. Key objects in the image, such as the vehicles and surrounding foliage, are accurately reconstructed, suggesting effective cross-modal alignment.

Text-to-LiDAR Generation. The conditional model trained above can also be used for text-to-LiDAR generation, as shown in Fig. 3 (b). Given text prompts such as "Cars passing narrow street," "Pedestrians walking on sidewalks," and "Many cars and buses," LiTDM generates coherent LiDAR scenes that reflect the described semantic elements, including pedestrians and buses, while forming scene layouts consistent with prompts such as a narrow street.

5 Conclusion

In conclusion, we have presented a novel LiTDM framework that leverages triplane representations as latent spaces for diffusion models, specifically tailored to LiDAR scene synthesis. Additionally, our LSPS strategy ensures that the generated scenes faithfully replicate the characteristic patterns of LiDAR scans, thereby enhancing both fidelity and realism. While effective, our framework involves non-negligible computational costs due to the autoencoder-based representation learning and iterative diffusion-based generation. This work opens new avenues for future research at the intersection of 3D representation learning and generative modeling.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and assisting with translation. We have not used generative AI tools for generating synthetic datasets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results. Providing critical ingredients for proving mathematical claims and assisting in the writing of proofs are not applicable to this work, as it contains no formal proofs. Additionally, we used generative AI tools for drafting parts of a research paper, brainstorming, editing a research paper to improve readability, and identifying relevant literature. We have reviewed all AI-assisted work. All AI-assisted outputs were manually reviewed and verified by the authors for correctness, relevance, and consistency with the final manuscript. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Hengwei Bian, Lingdong Kong, Haozhe Xie, Liang Pan, Yu Qiao, and Ziwei Liu. Dynamiccity: Large-scale lidar generation from dynamic scenes. *arXiv preprint arXiv:2410.18084*, 2024.

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.

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. Efficient geometry-aware 3d generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16123–16133, 2022.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3075–3084, 2019a.

Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3075–3084, 2019b.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for lidar segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 228–240, 2023.

Jumin Lee, Sebin Lee, Changho Jo, Woobin Im, Juhyeong Seon, and Sung-Eui Yoon. Semcity: Semantic scene generation with triplane diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 28337–28347, 2024.

Chenqi Li, Yuan Ren, and Bingbing Liu. Pcgen: Point cloud generator for lidar simulation. *arXiv preprint arXiv:2210.08738*, 2022.

Yiyi Liao, Jun Xie, and Andreas Geiger. Kitti-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (3):3292–3310, 2022.

Jiuming Liu, Zheng Huang, Mengmeng Liu, Tianchen Deng, Francesco Nex, Hao Cheng, and Hesheng Wang. Topolidm: Topology-aware lidar diffusion models for interpretable and realistic lidar point cloud generation. *arXiv preprint arXiv:2507.22454*, 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11167–11176, 2020.

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.

Kazuto Nakashima, Xiaowen Liu, Tomoya Miyawaki, Yumi Iwashita, and Ryo Kurazume. Fast lidar data generation with rectified flows. *arXiv preprint arXiv:2412.02241*, 2024.

Kazuto Nakashima, Xiaowen Liu, Tomoya Miyawaki, Yumi Iwashita, and Ryo Kurazume. Fast lidar data generation with rectified flows. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10057–10063. IEEE, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

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

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10684–10695, 2022.

Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=TIdIXIpzhoI`.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021.

J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20875–20886, 2023.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4563–4573, 2023.

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, 2024.

Yang Wu, Yun Zhu, Kaihua Zhang, Jianjun Qian, Jin Xie, and Jian Yang. Weathergen: A unified diverse weather generator for lidar point clouds via spider mamba diffusion. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 17019–17028, 2025.

Zhengkang Xiang, Zizhao Li, Amir Khodabandeh, and Kourosh Khoshelham. Sg-ldm: Semantic-guided lidar generation via latent-aligned diffusion. *arXiv preprint arXiv:2506.23606*, 2025.

Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Ultralidar: Learning compact representations for lidar completion and generation. *arXiv preprint arXiv:2311.01448*, 2023.

Tianyi Yan, Junbo Yin, Xianpeng Lang, Ruigang Yang, Cheng-Zhong Xu, and Jianbing Shen. Olidm: Object-aware lidar diffusion models for autonomous driving. *arXiv preprint arXiv:2412.17226*, 2024.

Huijie Zhang and Xiaobai Liu. Focal diffusion process for object-aware 3d lidar generation. In *Proceedings of the 6th ACM International Conference on Multimedia in Asia*, pp. 1–7, 2024.

Kaicheng Zhang, Ziyang Hong, Shida Xu, and Sen Wang. Curl: Continuous, ultra-compact representation for lidar. *arXiv preprint arXiv:2205.06059*, 2022.

Dekai Zhu, Yixuan Hu, Youquan Liu, Dongyue Lu, Lingdong Kong, and Slobodan Ilic. Spiral: Semantic-aware progressive lidar scene generation. *arXiv preprint arXiv:2505.22643*, 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.

A Further Implementation Details

Positional embedding. For positional embedding, we employ both Positional Embedding (PE) and Relative Positional Embedding (RPE) methods. PE involves applying sinusoidal positional embedding to the coordinates x, y, and z, enabling g_θ to learn global spatial relationships within the 3D space. We further employ RPE to allow each plane feature to learn more detailed spatial relationships. By computing the relative positions of x, y, z with respect to the centers of the xy, yz, and zx grids, and applying sinusoidal positional embedding to these relative positions, the model captures finer spatial nuances within each plane.

Architectural details of the triplane feature extraction module. Implementation details of the triplane feature extraction module, such as backbone architecture choice, triplane resolution, and number of channels are specified in Table 6.

B Pseudocode of LSPS

We provide the pseudocode of our LiDAR Simulation-based coarse-to-fine Point Sampling (LSPS) algorithm in Algorithm. 1.

Table 6: Complete specification of the triplane feature extraction module.

Stage	Specification
Input	Raw scan points with features $(x, y, z, \text{intensity})$, sparsely quantized at a voxel size of 0.05 m
Encoder	MinkowskiUNet18 (Choy et al., 2019b) in its standard configuration, with 16 output channels per point
Projection	Each point feature is assigned to one cell of each of the three axis-aligned grids
Pooling	Features falling in the same cell are averaged; cells that receive no point remain zero
Triplanes	$\mathbf{h}_{xy}$ of 160×160, $\mathbf{h}_{xz}$ of 160×32 and $\mathbf{h}_{yz}$ of 160×32, all at 1.25 m resolution with $C = 16$, covering $200 \times 200 \times 40$ m
Normalization	Instance normalization followed by a scaled $\tanh$, which bounds the latent that the diffusion model is trained on

Algorithm 1 LiDAR Simulation-based Coarse-to-Fine Point Sampling

Input Ray $R = \mathbf{o} + t\mathbf{d}$, where $t \in [t_{\min}, t_{\max}]$, ground-truth depth d_R if ray R has a return, coarse sample number M_c, fine sample number M_f

Output Sampled on-ray points $\mathcal{S}(R) = \{\mathbf{p}_\ell\}_{\ell=1}^{M_c+M_f}$

$\Delta \leftarrow (t_{\max} - t_{\min})/M_c$
$\mathcal{S}_c(R) \leftarrow \emptyset$
for $i = 1$ to M_c **do**
  $u_i \sim \mathcal{U}(0, 1)$
  $t_i^c \leftarrow t_{\min} + (i-1)\Delta + u_i\Delta, \quad \mathbf{p}_i^c \leftarrow \mathbf{o} + t_i^c\mathbf{d}, \quad \mathcal{S}_c(R) \leftarrow \mathcal{S}_c(R) \cup \{\mathbf{p}_i^c\}$
end for
if R has a return **then**
  $i^\star \leftarrow \lfloor (d_R - t_{\min})/\Delta \rfloor + 1$
else
  $i^\star \sim \mathcal{U}\{1, 2, \ldots, M_c\}$
end if
$\mathcal{S}_f(R) \leftarrow \emptyset$
$a \leftarrow (t_{i^\star-1}^c + t_{i^\star}^c)/2, \quad b \leftarrow (t_{i^\star}^c + t_{i^\star+1}^c)/2$
for $j = 1$ to M_f **do**
  $u_j \sim \mathcal{U}(0, 1)$
  $t_j^f \leftarrow a + u_j(b - a), \quad \mathbf{p}_j^f \leftarrow \mathbf{o} + t_j^f\mathbf{d}, \quad \mathcal{S}_f(R) \leftarrow \mathcal{S}_f(R) \cup \{\mathbf{p}_j^f\}$
end for
return $\mathcal{S}_c(R) \cup \mathcal{S}_f(R)$

C Further Ablation Studies

Mask Threshold. Table 7 denotes the ablation study on the mask threshold τ and its effect on generation quality. Increasing τ generally improves FSVD and FPVD, indicating better feature-level consistency, while overly large thresholds degrade JSD and MMD. Thus, we set $\tau = 0.5$ as the default value, as it achieves the best overall trade-off across all evaluation metrics.

Candidate Selection in LSPS. Table 8 compares the default argmax selection in LSPS with a first-hit rule that selects the nearest candidate whose occupancy exceeds τ. The first-hit rule slightly improves BEV JSD and MMD, from 4.59 to 4.54 and from 1.93 to 1.80, respectively, in the reported scales. Conversely, argmax achieves marginally lower FSVD and FPVD, with differences of only 0.06 and 0.09. These results indicate a modest trade-off between BEV distributional similarity and perceptual metrics. We retain argmax as the default, while first-hit selection offers a competitive alternative.

Table 7: Ablation study on the mask threshold τ.

τ	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-4}$)
0.1	42.62	49.86	7.97	6.97
0.3	32.33	38.63	4.82	2.21
0.5	**29.35**	**35.23**	**4.59**	**1.93**
0.7	27.69	33.24	4.89	2.71
0.8	26.62	31.87	5.28	3.86

Table 8: Effect of the candidate selection rule. The first-hit rule takes the closest candidate whose occupancy exceeds the threshold τ instead of the candidate with the highest occupancy.

Selection rule	BEV JSD ↓ ($\times 10^{-2}$)	BEV MMD ↓ ($\times 10^{-4}$)	FSVD ↓	FPVD ↓
argmax (Eq. (14) and (15))	4.59	1.93	**29.35**	**35.23**
first hit	**4.54**	**1.80**	29.41	35.32

D Further Experimental Results

Semantic Coherence. Table 9 evaluates semantic coherence on 10,000 real and 2,000 generated scans using the same pretrained segmentation model used for FSVD. The evaluation covers ten classes appearing in at least 90% of real frames and accounting for 99.5% of all points. LiTDM achieves the lowest BEV JSD and MMD among other baselines, at 19.50×10^{-2} and 6.58×10^{-3}, respectively. The prediction entropy for LiTDM is also closest to the real reference, indicating that the generated scenes contain semantically well-defined structures comparable to those in real LiDAR scans. The per-class results in Tables 10 and 11 further show that LiTDM achieves the best JSD in seven classes and MMD in six, leading on both metrics for road, sidewalk, fence, and pole.

Cross-Sensor LiDAR Generation. Table 12 evaluates cross-sensor generation on nuScenes dataset using models trained exclusively on KITTI-360, with evaluation restricted to the elevation band shared by both sensors. While the baselines require reprojection onto the nuScenes grid, LiTDM directly generates points in the target sensor configuration without retraining or retuning. Our method achieves the lowest FSVD and FPVD of 34.41 and 26.97, respectively, outperforming both reprojected baselines. It also obtains the lowest MMD of 9.34×10^{-3}, while its JSD remains close to the best score achieved by Text2LiDAR. These results support the flexibility of LSPS in accommodating different sensor configurations while preserving competitive generation quality.

Evaluation on nuScenes. Table 13 evaluates unconditional generation on nuScenes dataset, with all methods trained on nuScenes. LiTDM achieves the lowest FSVD and FPVD, reducing the scores from 19.86 to 13.39 and from 13.64 to 11.69, respectively, compared with R2DM. It also obtains the lowest MMD of 1.06×10^{-4}. R2DM achieves the lowest JSD, whereas our method ranks second with 5.23×10^{-2}. These results demonstrate the effectiveness of our framework beyond KITTI-360, with improvements in three of the four evaluated metrics.

Computational Cost and Memory Usage. Table 14 compares computational cost and peak memory on a single RTX 3090 with a batch size of 1. With the same denoiser parameter count as R2DM, LiTDM reduces FLOPs per step from 229.0 G to 90.3 G, a reduction of 60.6%. It achieves the shortest denoising time of 15.4 ms per step. Including decoding, generation takes 18.0 s per scene at 1,000 steps, making it the fastest among the compared methods. Peak memory reaches 2.24 GiB during sampling and 5.55 GiB during decoding, while maintaining efficient inference.

Table 9: Semantic coherence evaluation. Mean per-class BEV occupancy metrics and mean per-frame semantic prediction entropy.

Method	BEV JSD ↓ ($\times 10^{-2}$)	BEV MMD ↓ ($\times 10^{-3}$)	Prediction Entropy
Real	10.06	0.33	0.5366
LiDARVAE	37.76	58.5	0.8195
LiDARGAN	41.92	64.4	0.9625
ProjectedGAN	35.55	47.7	0.7789
LiDM	32.37	24.4	0.8012
R2DM	21.83	18.8	0.7055
Text2LiDAR	39.02	50.0	0.6708
LiTDM (ours)	**19.50**	**6.58**	**0.6375**

Table 10: Per-class BEV occupancy JSD.

Method	car	truck	other-vehicle	road	sidewalk	building	fence	vegetation	terrain	pole	average
Real	0.053	0.294	0.125	0.016	0.087	0.059	0.062	0.053	0.036	0.220	0.101
LiDARVAE	0.263	0.645	0.430	0.262	0.485	0.255	0.306	0.333	0.205	0.592	0.378
LiDARGAN	0.289	0.784	0.541	0.370	0.520	0.339	0.254	0.210	0.352	0.533	0.419
ProjectedGAN	0.242	0.621	0.414	0.305	0.466	0.321	0.257	0.215	0.210	0.503	0.356
LiDM	0.254	0.430	0.315	0.249	0.416	0.210	0.358	0.275	0.239	0.491	0.324
R2DM	0.133	**0.370**	0.232	0.075	0.305	0.123	0.262	0.135	**0.129**	0.419	0.218
Text2LiDAR	0.301	0.655	0.632	0.347	0.395	0.380	0.351	**0.133**	0.166	0.543	0.390
LiTDM (ours)	**0.119**	0.376	**0.168**	**0.051**	**0.243**	**0.121**	**0.181**	0.164	0.154	**0.373**	**0.195**

Table 11: Per-class BEV occupancy MMD ($\times 10^{-3}$).

Method	car	truck	other-vehicle	road	sidewalk	building	fence	vegetation	terrain	pole	average
Real	0.13	0.66	0.71	0.02	0.52	0.11	0.33	0.10	0.16	0.60	0.33
LiDARVAE	5.28	120	79.9	102	148	3.00	10.7	14.7	12.8	87.9	58.5
LiDARGAN	5.78	210	57.8	22.5	126	7.31	11.2	3.44	21.1	180	64.4
ProjectedGAN	6.93	121	83.3	25.9	116	6.07	9.42	8.67	17.4	82.6	47.7
LiDM	3.69	18.2	**12.6**	8.52	68.0	1.46	13.7	6.14	9.98	102	24.4
R2DM	**1.58**	15.1	15.8	1.41	41.2	**0.94**	7.05	**1.73**	3.12	100	18.8
Text2LiDAR	14.6	120	110	55.4	38.1	6.68	33.3	3.39	12.5	106	50.0
LiTDM (ours)	1.71	**0.81**	17.7	**0.33**	**24.1**	1.26	**3.63**	2.40	**2.53**	**11.4**	**6.58**

Table 12: Cross-sensor generation on nuScenes dataset. All models are trained on KITTI-360. LiTDM generates directly in the nuScenes sensor configuration without retraining or retuning, while the baselines are reprojected onto the nuScenes grid. Scores are computed over the elevation band covered by both sensors.

Method	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-3}$)
R2DM (reprojected)	51.61	39.05	25.9	12.0
Text2LiDAR (reprojected)	70.27	59.49	**25.6**	10.4
LiTDM (ours)	**34.41**	**26.97**	25.7	**9.34**

Table 13: Unconditional generation on nuScenes dataset in the matched setting, where every method is trained on nuScenes.

Method	FSVD ↓	FPVD ↓	JSD ↓ ($\times 10^{-2}$)	MMD ↓ ($\times 10^{-4}$)
R2DM	19.86	13.64	**4.91**	1.22
Text2LiDAR	32.66	24.91	5.89	2.28
LiTDM (ours)	**13.39**	**11.69**	5.23	**1.06**

Table 14: Computational cost and peak memory, measured on a single RTX 3090 with a batch size of 1, with decoding included for the two-stage models. Memory is reported as sampling / decoding for the two-stage models.

Method	Denoiser params	FLOPs per step	Time per step	Time per scene (1000 steps)	Peak memory (sampling / decoding)
LiDM	257.8 M	171.7 G	20.8 ms	20.8 s	1.24 / 1.38 GiB
R2DM	31.1 M	229.0 G	27.6 ms	27.6 s	0.93 GiB
Text2LiDAR	45.8 M	1132.0 G	98.1 ms	98.1 s	2.36 GiB
LiTDM (ours)	31.1 M	**90.3 G**	**15.4 ms**	**18.0 s**	2.24 / 5.55 GiB