Towards Realistic Scene Generation with LiDAR Diffusion Models

Haoxi Ran
Carnegie Mellon University
ranhaoxi@cmu.edu

Vitor Guizilini
Toyota Research Institute
vitor.guizilini@tri.global

Yue Wang
University of Southern California
yue.w@usc.edu

https://lidar-diffusion.github.io

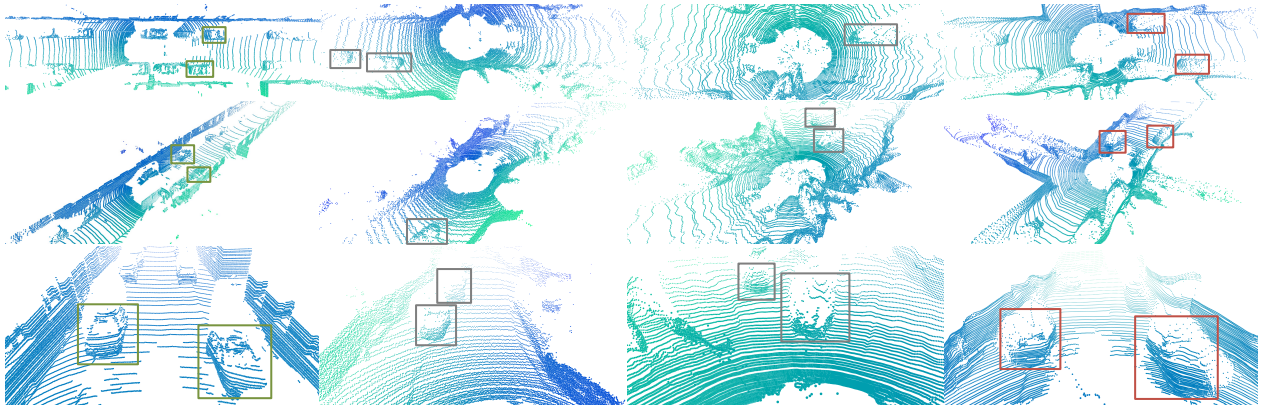


Figure 1. Our method (LiDM) establishes a new state-of-the-art in unconditional LiDAR-realistic scene generation, and marks a milestone towards conditional LiDAR scene generation from different input modalities.

Abstract

*Diffusion models (DMs) excel in photo-realistic image synthesis, but their adaptation to LiDAR scene generation poses a substantial hurdle. This is primarily because DMs operating in the point space struggle to preserve the curve-like patterns and 3D geometry of LiDAR scenes, which consumes much of their representation power. In this paper, we propose **LiDAR Diffusion Models** (LiDMs) to generate LiDAR-realistic scenes from a latent space tailored to capture the realism of LiDAR scenes by incorporating geometric priors into the learning pipeline. Our method targets three major desiderata: pattern realism, geometry realism, and object realism. Specifically, we introduce curve-wise compression to simulate real-world LiDAR patterns, point-wise coordinate supervision to learn scene geometry, and patch-wise encoding for a full 3D object context. With these three core designs, we establish a new state of the art on unconditional LiDAR generation in 64-beam scenario, while maintaining high efficiency compared to point-based DMs (up to 107× faster). Furthermore, by compressing LiDAR scenes into a latent space, we enable the controllability of DMs with various conditions such as semantic maps, camera views, and text prompts.*

1. Introduction

Recent years have observed a surge of conditional generative models that are capable of generating visually appealing and highly realistic images. Among them, diffusion models (DMs) have emerged as one of the most popular methods, thanks to its unexceptionable performance. To enable generation with arbitrary conditions, Latent Diffusion Models (LDMs) [51] combine the cross-attention mechanism with a convolutional autoencoder to generate high-resolution images. Its subsequent extensions (*e.g.*, Stable Diffusion [2], Midjourney [1], ControlNet [72]) further boosted its potential for conditional image synthesis.

This success leads us to inquire: can we apply controllable DMs to LiDAR scene generation for autonomous driving and robotics? For instance, given a collection of bounding boxes, can these models synthesize corresponding LiDAR scenes, thus turning these bounding boxes into high-quality and expensive labeled data? Alternatively, is it possible to generate a 3D scene solely from a set of images? Or even more ambitiously, can we design a language-driven LiDAR generator for controllable simulation? To answer these interleaved questions, our goal is to design DMs that incorporate a diverse set of conditions (*e.g.*, layouts, camera

views, text) to generate LiDAR-realistic scenes.

To that end, we glean insights from recent works of DMs for autonomous driving. In [75], *point-based* DM (*i.e*., LiDARGen) is introduced to unconditional LiDAR scene generation. However, this model tends to produce noisy backgrounds (*e.g*., roads, walls) and ambiguous objects (*e.g*., cars), leading to a failure in generating LiDAR-realistic scenes (*cf*. Fig. 1). In addition, applying diffusion on points without any compression can computationally slow down the inference process. Moreover, the direct application of *patch-based* DMs (*i.e*., Latent Diffusion [51]) to LiDAR scene generation yields unsatisfactory performance both qualitatively and quantitatively (*cf*. Fig. 1).

To enable conditional LiDAR-realistic scene generation, we thereby propose a *curve-based* generator, termed *LiDAR Diffusion Models* (LiDMs), to answer the aforementioned questions and tackle the shortcomings of recent works. LiDMs are capable of processing arbitrary conditions, such as bounding boxes, camera images, and semantic maps. LiDMs leverage range images as the representations of LiDAR scenes, which are prevalent in various downstream tasks such as detection [34, 43], semantic segmentation [44, 66], and generation [75]. This choice is grounded in the reversible and lossless conversion between range images and point clouds, along with the substantial benefits from the highly optimized 2D convolutional operation. To grasp the semantic and conceptual essence of LiDAR scenes during the diffusion process, our approach leverages encoded points of LiDAR scenes into a perceptually equivalent latent space before the diffusion process.

To further improve the realistic simulation of real-world LiDAR data, we focus on three key components: pattern realism, geometry realism, and object realism. First, We leverage curve-wise compression in the auto-encoding process to maintain the curve patterns of points, motivated by [59]. Second, to achieve geometry realism, we introduce point-wise coordinate supervision to imbue our autoencoder with the understanding of scene-level geometry. Lastly, we enlarge the receptive field by incorporating an additional patch-wise down-sampling strategy to capture the full context of visually large objects. Augmented by these proposed modules, the resulting perceptual space enables DMs to efficiently synthesize high-quality LiDAR scenes (*cf*. Fig. 1), while also exhibiting superior performance with a $\times 107$ speedup compared to point-based DMs (assessed on one NVIDIA RTX 3090), and supporting arbitrary types of image-based and token-based conditions. We summarize our key contributions as follows:

- We propose a novel LiDAR Diffusion Model (LiDM), a generative model that consumes arbitrary input conditions for LiDAR-realistic scene generation. To the best of our knowledge, this is the first method capable of LiDAR scene generation from multi-modal conditions.
- We introduce *curve-wise compression* to maintain realistic LiDAR patterns, *point-wise coordinate supervision* to regularize models for scene-level geometry, and *patch-wise encoding* to fully capture the context of 3D objects.
- We introduce three metrics for thorough and quantitative evaluation of the quality of generated LiDAR scenes in the perceptual space, comparing representations including range images, sparse volumes, and point clouds.
- Our method achieves *state-of-the-art* on unconditional scene synthesis under 64-beam scenario, while realizing a $\times 107$ speedup compared to point-based DMs.

2. Related Work

Diffusion Models. Diffusion models (DMs) [57] shows great success in image synthesis [11, 22, 31] in pixel space. Instead of applying diffusion to pixel space, Latent Diffusion Models (LDMs) [51] adopt a perceptually equivalent latent space for DMs. Their larger-scale applications of LDMs (*e.g*., Stable Diffusion [2], Midjourney [1] further boost the community of DMs. Recent applications of DMs, including language-guided DMs (*e.g*., Glide [45], DALL-E2 [49]) and other controllable DMs (*e.g*., ControlNet [72]), also reveal the great potential of DMs.

3D Diffusion Models. 3D diffusion models represent a crucial branch of DMs, offering the capability to generate high-quality samples across various 3D modalities. This includes point clouds [39, 42, 63, 74], meshes [20, 38, 40], and implicit fields [9, 14, 23, 35, 36, 55, 71]. Recently, Point-E [46], a language-guided DM, has demonstrated efficient capabilities in generating high-quality hand-crafted 3D models based on a large-scale 3D dataset.

LiDAR Scene Generation. Given the larger-scale and complex scenes, treating LiDAR point clouds as a point cloud generation task akin to hand-crafted 3D models encounters difficulties. In [7], the authors explore the possibilities of generative models in LiDAR scenes by providing two solutions, LiDARVAE and LiDARGAN. [67] and [75] further introduce vector-quantized variational autoencoder and point-based diffusion model, respectively, to generate satisfactory samples of LiDAR scenes. However, these aforementioned methods still fail to generate LiDAR-realistic scenes as they may overlook the curve-like structures and geometric details inherent in LiDAR data.

LiDAR Scene Simulation. LiDAR simulation produces LiDAR point clouds through physics-based simulators [13, 32] or data-driven simulators [4, 24, 41, 65]. Physics-based LiDAR simulators (*e.g*., CARLA [13]) use raycasting to project rays from the sensor's origin onto the environment's geometry to simulate LiDAR by calculating intersections.

Benefiting from producing realistic LiDAR scenes, data-driven LiDAR simulation gains great attention in the community. A pioneering work LiDARSim [41] adopts deep learning models to produce deviations from physics-based simulations to generate realistic LiDAR point clouds.

3. LiDAR Diffusion Models

In this section, we present **Li**DAR **D**iffusion **M**odels (LiDMs) with details. An overview is shown in Fig. 2. In Sec. 3.1, we discuss our design choice of range images for data representation. In Sec. 3.2, we formulate the task of LiDAR scene generation with LiDMs. In Sec. 3.3, we explore the design of LiDAR compression in terms of both pattern and scene geometry realism. In Sec. 3.4, we present potential applications of LiDM based on multi-modal conditions. In Sec. 3.5, we define the training objectives in the stages of LiDAR compression and LiDAR Diffusion. Finally, in Sec. 3.6, we describe three novel perceptual metrics designed to quantitatively analyze the sample quality for LiDAR scene generation.

3.1. Data Representation

LiDAR data can be represented by different modalities. Since our goal is to simulate raw LiDAR output, we have to choose a representation that is losslessly converted to and from raw data, which eliminates optional voxelization. To this end, we consider two choices, namely 3D point clouds and 2D range images. Both modalities have been successfully explored in previous works, albeit in different settings: point cloud generation [3, 69, 70] often focuses on human-crafted objects due to the efficiency limitation, while range image generation [7, 75] is tailored for larger-scale scenes such as LiDAR scenes. This separation indicates a clear preference towards range images for our settings.

3.2. Problem Formulation

Both the input and the output of our generators are represented as range images, denoted as x and $\hat{x}$, respectively, where $x, \hat{x} \in \mathbb{R}^{H \times W}$, and H and W are the height and width of the range image, respectively. For one pixel, defined by its normalized 2D location and range value, we can directly compute its depth, yaw, and pitch given predefined parameters. Specifically, we define the 3D coordinate $p = [\alpha, \beta, \gamma]$ of the pixel with:

$$\alpha = cos(\text{yaw}) \times cos(\text{pitch}) \times \text{depth}, \quad (1)$$

$$\beta = -sin(\text{yaw}) \times cos(\text{pitch}) \times \text{depth}, \quad (2)$$

$$\gamma = sin(\text{pitch}) \times \text{depth}. \quad (3)$$

Our LiDMs pipeline can be split into two parts: LiDAR compression and diffusion process. For LiDAR compression, we use an encoder $\mathcal{E}$ to compress range images into latent code $z = \mathcal{E}(x)$, where $z \in \mathbb{R}^{h \times w \times d}$, and a decoder $\mathcal{D}$ to decode it into $\hat{x} = \mathcal{D}(z) = \mathcal{D}(\mathcal{E}(x))$. For the diffusion process, following standard practice [51], we denote an equally weighted sequence of unconditional denoising auto-encoders as $\epsilon_\theta(z_t, t)$, with timestamp $t = 1 \dots T$.

3.3. Towards LiDAR-Realistic Generation

Pattern Realism The presence of curves is a common pattern in LiDAR scenes. A curve $c_i = [p_1, ..., p_{n_i}]$ is represented by a sequence of n_i points where consecutive pairs are connected by a polyline. Furthermore, as defined in [59], a LiDAR point cloud is equivalent to a set of curves, namely a curve cloud $C = \{c_1, ..., c_m\}$. While we cannot directly apply the concept of curve cloud to range images, we can still explore curve-like structures.

Given that each beam of a LiDAR sensor sequentially captures 3D points by scanning the scene horizontally, we can safely assume that each curve is stored in only one row of a range image, and is represented as a continuous segment of pixels. Thus, we design our auto-encoders through curve-wise compression, which results in *horizontally* downsampled range images, by a factor $f_c := 2^\eta$, where $\eta \in \mathbb{N}$. To achieve curve-wise compression, we implement with these details: 1) the kernel size of convolutions inside each curve-wise residual block is 1×4, instead of the traditional 3×3 used on images; 2) each downsampling or upsampling layer is applied only horizontally; 3) the padding is circular considering the two sides of an range image to be connected end to end. Through this implementation, we effectively preserve curve-like structures in a perceptually equivalent space.

Geometry Realism Preserving scene-level geometry is another key aspect in LiDAR-realistic generation. To achieve this, our model should possess the capability to clearly distinguish between objects and the background, demonstrating sensitivity to the contours of 3D geometry. However, the conversion of point clouds to range images (as described in Sec. 3.2) may lead to a loss of geometry. Thus, we introduce a novel point-wise coordinate supervision to enhance the understanding of autoencoders in 3D space. Point clouds are informative to describe the geometry of LiDAR scenes through coordinates, but due to irregularity, we cannot directly apply point cloud distance loss functions (*e.g.*, Chamfer Distance [16]) to the training of autoencoders. For this purpose, we design a simple manner by supervising the converted coordinate of each pixel between the input and output range images. Note that, the converted coordinate-based images are in the shape of $H \times W \times 3$. Point-wise coordinate supervision includes both a pixelwise 3D distance loss and an adversarial objective on the coordinate-based images.

Object Realism Achieving object realism is challenging, but an important aspect of recovering reasonable and complete shapes. While curve compression effectively captures patterns in LiDAR scenes, it suffers from a restricted recep-

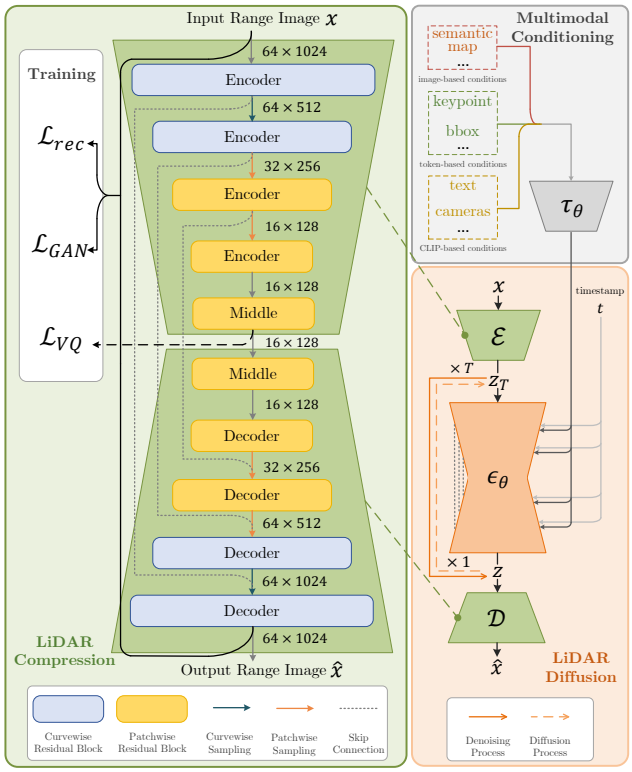


Figure 2. An overview of LiDMs on 64-beam data, which includes three parts: LiDAR compression (*cf*. Sec. 3.3 & 3.5), Multimodal Conditioning (*cf*. Sec. 3.4), and LiDAR Diffusion (*cf*. Sec. 3.5).

tive field when capturing the complete context of 3D objects, particularly for visually larger objects in range images (*i.e*., objects near the ego-center). Thus, we introduce patch-wise blocks in the intermediate layers of autoencoders for patch-wise encoding, which is qualitatively effective as a way to improve the synthesis quality of objects. We define the factor of downsampling during patch-wise encoding as $f_p = 2^\mu$, $\mu \in \mathbb{N}$, and thus $h = H/f_p$, $w = W/(f_c \times f_p)$.

3.4. Multimodal Conditioning

Previous works [51, 72] have shown DMs' capability to model conditional distributions. Utilizing our LiDMs, we further introduce multimodal conditioning to realize the significant potential for downstream tasks within the domain of autonomous driving. Typically, two types of conditions can serve as inputs in LiDAR scenes: image-based conditions (*e.g*., semantic maps), and token-based conditions (*e.g*., bounding boxes, keypoints). We approach the applications of image-based conditioning as image-to-image translation tasks [25], while we employ the cross-attention mechanism to handle token-based conditions, aligning with a widely adopted practice [51]. To broaden the scope of conditional LiDAR generation, we introduce **Camera-to-LiDAR** task by extracting the global features of each view using a pretrained latent space provided by CLIP [47]. Due to the spatial mismatch between multiple camera views and a LiDAR point cloud, we cannot directly treat Camera-to-LiDAR generation as another image-to-image translation task. Therefore, for a LiDAR scene, in contrast to tasks employing the entire image-based condition, we guide LiDMs with a condition representation formed by concatenating global features of all camera views.

Most recently, the paradigm of contrastive image-text pretraining (*e.g*., CLIP [47]) has demonstrated remarkable progress in zero-shot learning [48]. CLIP enables zero-shot understanding across diverse generation tasks of text-to-image [49], text-to-video [68], *etc*. To explore the potential applications of language-guided autonomous driving [26] for LiDAR generation, we leverage the text-image latent space of CLIP to encode descriptive prompts for LiDMs. Consequently, we can seamlessly transition the Camera-to-LiDAR task into a novel **Text-to-LiDAR** generation.

3.5. Training Objectives

LiDAR Compression Through curve-wise compression, point-wise coordinate supervision and patch-wise encoding, we design autoencoders to compress range images. To train these autoencoders, we adopt a set of objectives, including a pixelwise L_1 reconstruction objective L_{rec}, a curve-based adversarial objective adapted from [25] L_{GAN} and a vector quantization regularization [64] L_{VQ}. Specifically, we compute both L_{rec} and L_{GAN} with both range and coordinate-based images as input. We compute L_{rec} in the parts of pixelwise L_1 loss with range images and pixelwise coordinate distance loss with coordinate-based images:

$$\mathcal{L}_{rec}\left(x\right) = \mathbb{E}_x[\|x - \hat{x}\| + \lambda\|p - \hat{p}\|_2^2], \tag{4}$$

where λ is a scale factor for the supervision of coordinate-based images. Additionally, we compute L_{GAN} by feeding both range images and coordinate-based images into our CurveGAN, a variant of PatchGAN [25] by applying curve-wise compression in the first stage. We define the adversarial objective as:

$$\mathcal{L}_{\text{GAN}}(x) = \mathbb{E}_x[\log \mathcal{D}([x, p]) + \log(1 - \mathcal{D}([\hat{x}, \hat{p}]))], \tag{5}$$

where $[\dots]$ means concatenation operation. Besides, we utilize the loss of vector quantization [64] L_{VQ} to learn a codebook of range image constituents. We incorporate the vector quantization layer in the decoder, following the implementation of [15]. Overall, the complete training objective of autoencoders is:

$$L_{\text{AE}} = \mathcal{L}_{rec} + \mathcal{L}_{\text{GAN}} + \mathcal{L}_{\text{VQ}}. \tag{6}$$

LiDAR Diffusion As probabilistic models, DMs [57] aim to comprehend a data distribution through a progressive de-

noising process on variables sampled from a Gaussian distribution. In image synthesis, previous works have employed DMs either in the pixel space [11, 22] or a latent space [51]. In this paper, we enable DMs to leverage a low-dimensional latent space created by our autoencoders. This space adequately preserves LiDAR patterns and the geometry of scenes and objects while maintaining computational efficiency. Furthermore, it empowers DMs to focus on the semantic information of large-scale LiDAR scenes. We define the objective of our *unconditional* LiDMs:

$$L_{\text{LiDM}} = \mathbb{E}_{\mathcal{E}(x),\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t\right)\|_2^2\right], \quad (7)$$

where $\epsilon_\theta\left(z_t, t\right)$ is a UNet [52] backbone with timestamp conditioning. For inference, we obtain sampled range images by decoding the denoised latent code z with $\mathcal{D}$. Similarly, given an input condition y, we define the objective of our *conditional* LiDMs as:

$$L_{\text{cLiDM}} = \mathbb{E}_{\mathcal{E}(x),y,\epsilon\sim\mathcal{N}(0,1),t}\left[\|\epsilon - \epsilon_\theta\left(z_t, t, \tau_\theta(y)\right)\|_2^2\right], \quad (8)$$

where τ_θ is a condition encoder responsible for converting y into an accessible representation for the UNet backbone.

3.6. Evaluation Metrics

To evaluate LiDAR generative models, previous works on LiDAR scene generation [7, 75] commonly utilize statistical metrics as proposed in [3]. Nonetheless, they may encounter difficulties to quantitatively measure the synthesis quality at a perceptual level. Thus, we design several perceptual level metrics for LiDAR generative models.

Perceptual evaluation (*e.g.*, FID [21]) stands as a prevalent manner to measure image synthesis quality across a spectrum of popular image-based generators [11, 22, 28–30, 51, 53]. Typically, these prior works compute the Fréchet distance [18] between the data distributions of of real data and generated samples within a perceptual space defined by a pretrained classifier (*e.g.*, Inception model [60]). To enhance comprehension of the performance of LiDAR generative models at a perceptual level, we augment the perceptual evaluation with three Fréchet-distance-based perceptual metrics: Fréchet Range Image Distance (FRID), Fréchet Sparse Volume Distance (FSVD), and Fréchet Point-based Volume Distance (FPVD).

In the absence of a pretrained classifier specifically tailored for LiDAR scenes, we adopt segmentation-based pretrained models for our evaluations. Specifically, we calculate FRID, FSVD, FPVD through three simple methods, which include RangeNet++ [44], MinkowskiNet [10], and SPVCNN [61], respectively. Different from FID, which is computed relying on the global feature of each input image in the final stage, our proposed metrics are founded on the average of the output features of each LiDAR scene in the intermediate stage. With the computed features of samples and the real-world data, we enable the performance comparisons in perceptual space between previous LiDAR generators and our LiDMs. *cf*. the supplement for details.

Method	Perceptual			Statistical	
	FRID ↓	FSVD ↓	FPVD ↓	JSD ↓	MMD ↓ (10^{-4})
Noise	3277	497.1	336.2	0.360	32.09
LiDARGAN [7]	1222	183.4	168.1	0.272	4.74
LiDARVAE [7]	199.1	129.9	105.8	0.237	7.07
ProjectedGAN [54]	149.7	44.7	33.4	0.188	2.88
UltraLiDAR [67]	370.0	72.1	66.6	0.747	17.12
LiDARGen [75] (1160s)	129.1	39.2	33.4	**0.188**	**2.88**
LiDARGen [75] (50s)	2051	480.6	400.7	0.506	9.91
LDM [51] (50s)†	199.5	70.7	61.9	0.236	5.06
LiDM (ours, 50s)†	158.8	53.7	42.7	0.213	4.46
Δ *Improv.*	20.4%	24.0%	31.0%	9.7%	11.9%
LiDM (ours, 50s)	**125.1**	**38.8**	**29.0**	0.211	3.84
Δ *Improv.*	37.3%	45.1%	53.2%	10.6%	24.1%

Table 1. Comparison of *unconditional* LiDAR scene generation with recent state-of-the-art methods. We conduct experiments on 64-beam (*i.e.*, KITTI-360 [37]) data."↓" indicates that lower values are better. N-s refers to N sampling steps during inference. Δ *Improv.* is the relative improvement of our method compared to the baseline of Latent Diffusion (LDM) [51], with the same number of diffusion steps. Note that, the diffusion process of LiDARGen [75] has 232 levels and 5 iterations in each level (*i.e.*, 1160 steps in total). denotes baseline results, while / denotes our results. †: Exactly the same settings except the architecture of autoencoders. We evaluate each method with 2000 randomly generated samples.

4. Experiments

4.1. Experimental Settings

We train and evaluate our models in the LiDAR scenarios of 32-beam data from nuScenes [17], gathered around the suburbs in Germany, and 64-beam data from KITTI-360 [37], collected inside the cities. For the 32-beam scenario, we train autoencoders on the full training dataset, containing 286,816 samples, and validate with 10,921 samples. For the 64-beam scenario, we train autoencoders on 63,315 samples of 9 sequences (including training and test set) and evaluate on 1,031 samples of one sequence. Different from autoencoders, we train and validate LiDMs on the widely adopted subsets of both datasets, which provide various conditions, including bounding boxes, views of multiple cameras or perspective views. Specifically, we adopt SemanticKITTI [5] for Semantic-Map-to-LiDAR task.

Our LiDMs process range images with dimensions of 32×1024 for 32-beam data and 64×1024 for 64-beam data. The pixel values of the range images are computed through binary logarithm followed by scaling. Training is conducted on eight NVIDIA RTX 3090, each with 24GB of GPU memory, and one of them is utilized for inference.

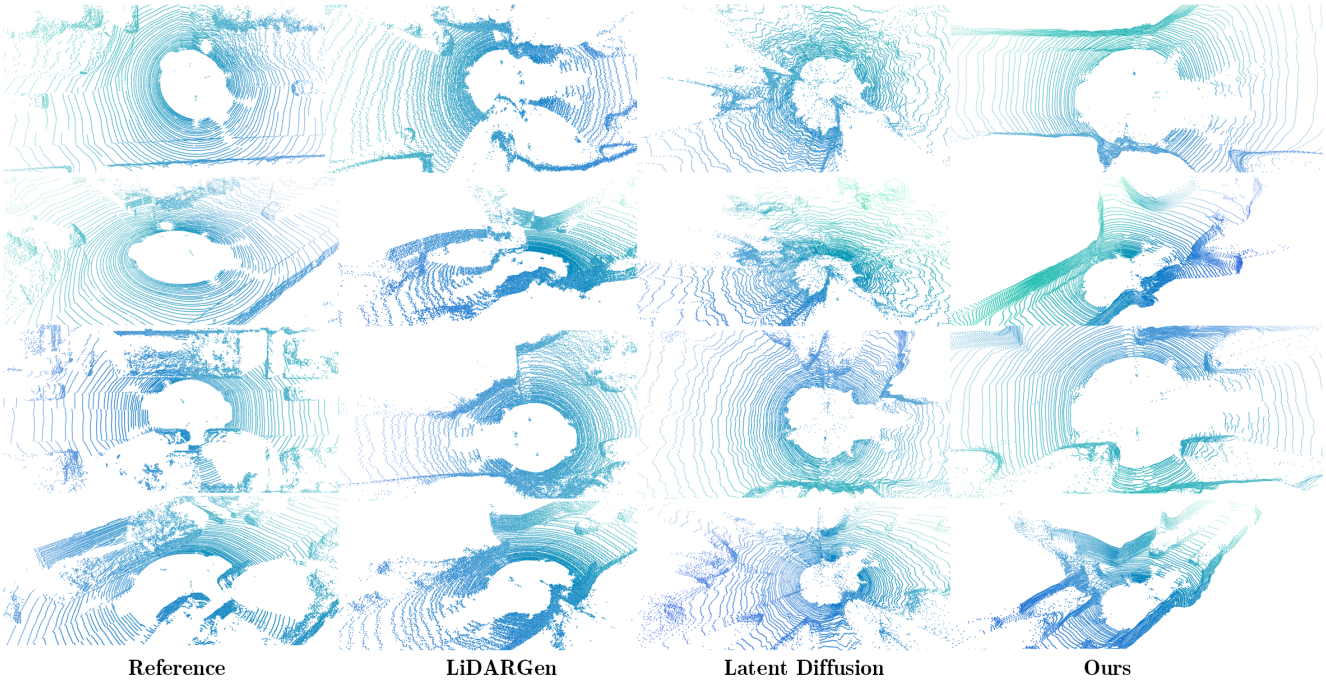


Figure 3. Samples from LiDARGen [75], Latent Diffusion [51], and our LiDMs on 64-beam scenario.

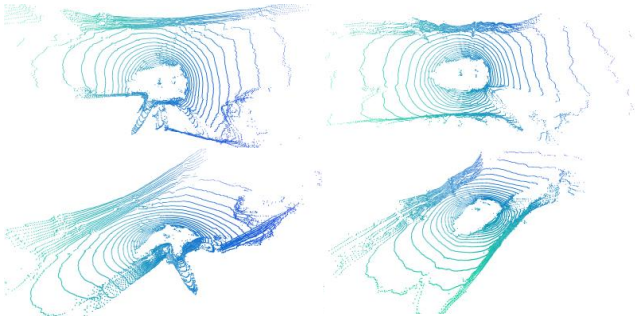


Figure 4. Samples from our LiDMs on 32-beam scenario.

4.2. Unconditional LiDAR Diffusion

To verify the effectiveness of our method, we train unconditional LiDMs on 64-beam data from KITTI-360 [37]. Following the practice of FID [21], we generate 2000 samples for the evaluation of unconditional generation. We evaluate the performance of each method using FRID (range image), FSVD (sparse volume), and FPVD (point-based volume), as described in Sec. 3.6. Together, these three metrics enable the perceptual evaluation to assess the quality of LiDAR scenes generated with different methods.

As shown in Table 1, with a very limited number of sampling steps (*i.e.*, 50), we establish a new state of the art for almost all considered metrics. Specifically, within 50 sampling steps, our approach outperforms the previous state-of-the-art method (*i.e.*, LiDARGen [75]) by a large margin for all considered metrics. Additionally, with only 50 evaluation steps, our method performs competitive with LiDARGen with a longer diffusion process of 1160 steps. LiDM reports $9.7\% \sim 31.0\%$ improvement over baseline model LDM [51] after replacing only the autoencoder. With all the techniques mentioned in Sec. 3.3, LiDM further improves by $10.6\% \sim 53.2\%$ over LDM. For a qualitative comparison, in Fig. 3 we provide examples generated with each model, alongside reference point clouds. We further provide some 32-beam samples for example in Fig. 4.

4.3. Conditional LiDAR Diffusion

To further exploit the potential of LiDMs, we implement several variations of conditional LiDAR scene generation, including Semantic-Map-to-LiDAR and Camera-to-LiDAR. For a quantitative analysis, we compare LiDMs to LiDARGen [75] and to our baseline Latent Diffusion [51], with results reported in Table 2.

Semantic-Map-to-LiDAR Transforming semantic maps into RGB images is a typical image-to-image translation task [25]. However, it remains underexplored in the context of LiDAR scene generation. As shown in our reported results, LiDM outperforms Latent Diffusion and LiDARGen by a substantial margin. Additionally, conditioning LiDMs with semantic maps leads to significant improvements relative to unconditional generation. We argue that having access to such data enhances the understanding of LiDMs at a semantic level, which facilitates the generation of LiDAR-realistic scenes. Fig. 5 further illustrates the effectiveness of semantic-map-based conditioning with LiDMs.

Method	Semantic-Map-to-LiDAR [5]					Camera-to-LiDAR [37]				
	FRID ↓	FSVD ↓	FPVD ↓	JSD ↓	MMD ↓ (10^{-4})	FRID ↓	FSVD ↓	FPVD ↓	JSD ↓	MMD ↓ (10^{-4})
LiDARGen [75]	42.5	31.7	30.1	0.130	5.18	-	-	-	-	-
Latent Diffusion [51]	24.0	21.3	20.3	0.088	3.73	50.2	35.9	26.5	0.256	3.80
LiDAR Diffusion (ours)	**22.9**	**20.2**	**17.7**	**0.072**	**3.16**	**44.9**	**32.5**	**25.8**	**0.205**	**3.69**

Table 2. Comparison of *conditional* LiDAR scene generation with recent state-of-the-art methods. We conduct Semantic-Map-to-LiDAR experiments on SemanticKITTI [5] and Camera-to-LiDAR on KITTI-360 [37]. "↓" indicates that lower values are better. We implement Semantic-Map-to-LiDAR on LiDARGen through the concatenation operation. Camera-to-LiDAR on LiDARGen is not viable through concatenation, and hence we do not report results in this setting.

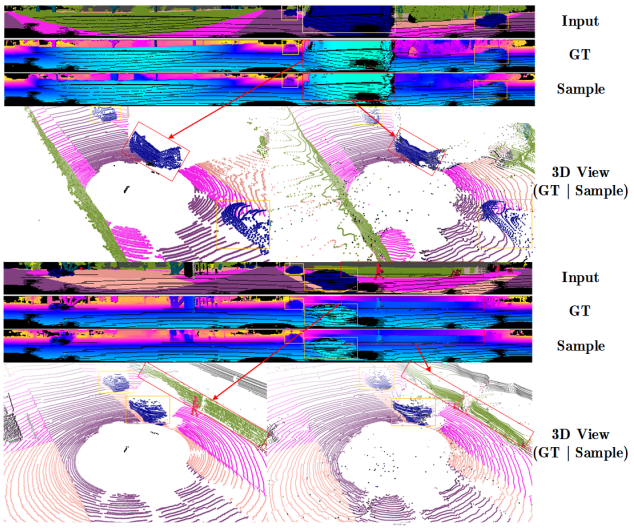


Figure 5. Samples from our LiDM for Semantic-Map-to-LiDAR generation on SemanticKITTI [5].

Camera-to-LiDAR Camera views are commonplace in the context of autonomous driving. To explore the relationship and complementarity between cameras and LiDAR sensor data, we implement Camera-to-LiDAR generation on KITTI-360 [37]. In this setting, LiDMs outperform LiDARGen [75] by over 36% among all metrics, while also successfully capturing semantic information from camera views. In Fig. 6, on the top, we see LiDM generating smooth ground from an input image containing a road without any objects. Similarly, on the bottom we see LiDM extracting the semantic information about the presence of a car and generating it on the synthesized LiDAR scene, although it still struggles with its precise scale in 3D space.

4.4. Zero-shot Text-to-LiDAR Generation

Text-to-image learning has become very popular recently due to the introduction of contract language-vision pretraining paradigm [47]. To facilitate LiDAR generation with language-guided conditioning, we introduce zero-shot Text-to-LiDAR generation based on a pretrained Camera-to-LiDAR LiDM, which is transformable to the task of

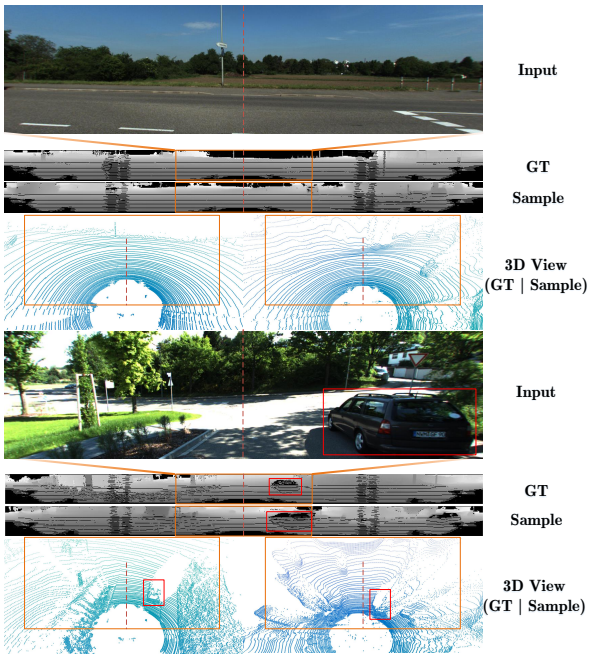


Figure 6. LiDM samples for conditional Camera-to-LiDAR generation on KITTI-360 [37]. The orange box indicates the area covered by the input image. For each scene, KITTI-360 provides one perspective, which cover only a part of the scene. Thus, LiDM performs conditional generation for the camera-covered region and unconditional generation for the remaining unobserved regions.

Text-to-LiDAR. Through provided prompts, LiDM can hallucinate possible scenes related to the input prompts. Fig. 7 shows some evidence to this argument. However, Text-to-LiDAR LiDM still struggles to generate scenes when presented with complex prompts, primarily due to constraints imposed by the limited amount of available training data.

4.5. Study on LiDAR-Realistic Generation

We explore our designed autoencoders for LiDAR compression and ablate on our proposed point-wise coordinate supervision. To analyze the behavior of LiDAR compression in terms of curve-wise and patch-wise encoding, we conduct experiments on various scale factors. The results in Fig. 8 show that curve-wise encoding generally performs

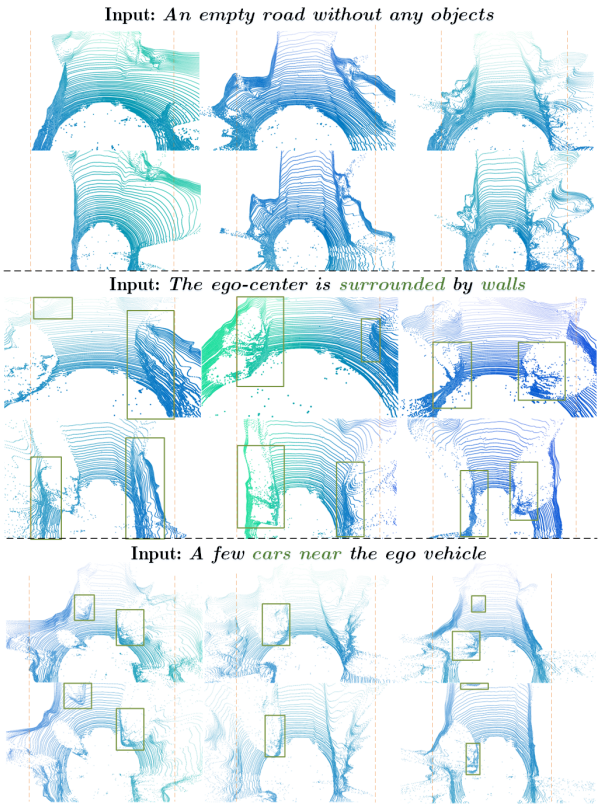


Figure 7. LiDM samples for zero-shot Text-to-LiDAR generation on 64-beam scenario. The areas enclosed by orange dotted lines indicate those influenced by the conditioning, and green boxes highlight objects potentially associated with the prompts.

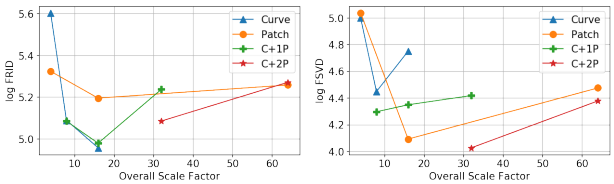


Figure 8. Overall scale factor ($f_c \times f_p$) vs sampling quality (FRID & FSVD). We compare different scales of curve-wise encoding (Curve), patch-wise encoding (Patch), and curve-wise encoding with one (C+1P) or two (C+2P) stages of patch-wise encoding on KITTI-360 [37].

better than patch-wise encoding. However, by introducing one stage of patch-wise encoding, we allow the autoencoders to further compress range images while maintaining competitive performance. To balance between performance and compression rate, we chose $f_c = 2$ and $f_p = 4$ as the default settings for our experiments.

Additionally, we study the effectiveness of our proposed point-wise coordinate supervision. The visualization in Fig. 9 illustrates that point-wise coordinate supervision aids autoencoders in preserving scene-level geometry by reconstructing sharper boundaries in 3D space.

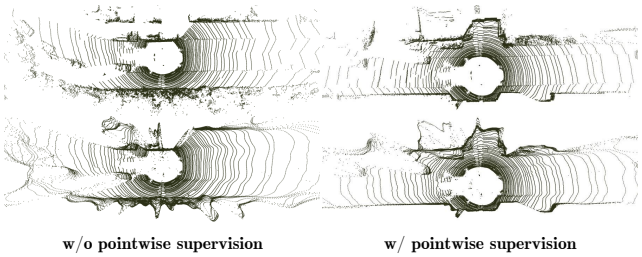


Figure 9. Samples from LiDM with or without point-wise supervision, as proposed in Sec. 3.3.

Method	Diffusion Size	Throughput↑	Infer.Speed↑
LiDARGen [75]	64×1024	0.015	17.5
LiDM (ours)	16×128	**1.603**	**80.2**

Table 3. Efficiency of LiDARGen and our LiDMs on the 64-beam scenario. We compute the number of generated samples per second of each model as throughput. *Infer. Speed* is the number of inference passes (*i.e.*, one pass representes a diffusion step) per second. We test both models in one NVIDIA RTX 3090 and adjust batch size to make full use of 24GB GPU memory.

4.6. Efficiency Analysis

Efficiency is particularly important for LiDAR generative models, specially when considering adoption to downstream tasks. To investigate this aspect of LiDMs, we provide an overhead comparison between LiDMs and the previous state-of-the-art point-based DM, LiDARGen [75], in terms of throughput (*samples/sec*), inference speed (*steps/sec*). As shown in Table 3, the throughput and inference speed of LiDM is around ×107 and ×4.6 faster than LiDARGen [75], respectively, which shows the superiority of our method in terms of efficiency.

5. Limitations

Even though LiDM establishes a new state of the art, its generated samples still have a visual gap relative to real-world LiDAR data. As shown in previous work [51], diffusion models should be powerful enough to capture the semantic information of input data, and therefore we argue that autoencoders are the key to failures when recovering most details of scenes. Our contributions are a step towards LiDAR-realistic autoencoders, however further work is still required. For example, autoencoders may reconstruct blurry boundaries between objects and the background, which though imperceptible on range images, may lead to visually unreasonable objects in 3D space.

6. Conclusion

We propose LiDAR Diffusion Models (LiDMs), a *general-conditioning* framework for LiDAR scene generation. With a focus on preserving curve-like patterns as well as scene-

level and object-level geometry, we design an efficient latent space for DMs to achieve LiDAR-realistic generation This design empowers our LiDMs to achieve competitive performance in unconditional generation and state of the art in conditional generation under 64-beam scenario, and enables the controllability of LiDMs with diverse conditions, including semantic maps, camera views, and text prompts. To the best of our knowledge, ours is the first method to successfully introduce conditioning to LiDAR generation.

References

[1] Midjourney. https://www.midjourney.com. 1, 2

[2] Stable diffusion. https://github.com/CompVis/stable-diffusion. 1, 2

[3] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, pages 40–49. PMLR, 2018. 3, 5, 12, 14

[4] Alexander Amini, Tsun-Hsuan Wang, Igor Gilitschenski, Wilko Schwarting, Zhijian Liu, Song Han, Sertac Karaman, and Daniela Rus. Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2419–2426. IEEE, 2022. 2

[5] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9297–9307, 2019. 5, 7

[6] Joan Bruna, Pablo Sprechmann, and Yann LeCun. Super-resolution with deep convolutional sufficient statistics. *arXiv preprint arXiv:1511.05666*, 2015. 14

[7] 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)*, pages 5034–5040. IEEE, 2019. 2, 3, 5

[8] Qifeng Chen and Vladlen Koltun. Photographic image synthesis with cascaded refinement networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1511–1520, 2017. 14

[9] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tuyakov, Alex Schwing, and Liangyan Gui. SDFusion: Multimodal 3d shape completion, reconstruction, and generation. *arXiv*, 2022. 2

[10] 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*, pages 3075–3084, 2019. 5, 13

[11] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021. 2, 5

[12] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29, 2016. 14

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 2

[14] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion, 2023. 2

[15] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021. 4, 14

[16] 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*, pages 605–613, 2017. 3

[17] Whye Kit Fong, Rohit Mohan, Juana Valeria Hurtado, Lubing Zhou, Holger Caesar, Oscar Beijbom, and Abhinav Valada. Panoptic nuscenes: A large-scale benchmark for lidar panoptic segmentation and tracking. *IEEE Robotics and Automation Letters*, 7(2):3795–3802, 2022. 5

[18] Maurice Fréchet. Sur la distance de deux lois de probabilité. In *Annales de l'ISUP*, pages 183–198, 1957. 5

[19] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2414–2423, 2016. 14

[20] 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. 2

[21] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 5, 6, 12

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. 2, 5

[23] Jingyu Hu, Ka-Hei Hui, Zhengzhe Liu, Ruihui Li, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation, inversion, and manipulation. *arXiv preprint arXiv:2302.00190*, 2023. 2

[24] Jordan SK Hu and Steven L Waslander. Pattern-aware data augmentation for lidar 3d object detection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 2703–2710. IEEE, 2021. 2

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017. 4, 6

[26] Kanishk Jain, Varun Chhangani, Amogh Tiwari, K Madhava Krishna, and Vineet Gandhi. Ground then navigate: Language-guided navigation in dynamic scenes. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4113–4120. IEEE, 2023. 4

[27] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 694–711. Springer, 2016. 14

[28] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 5

[29] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[30] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 5

[31] Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. 2

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ international conference on intelligent robots and systems (IROS)(IEEE Cat. No. 04CH37566)*, pages 2149–2154. IEEE, 2004. 2

[33] Solomon Kullback and Richard A Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1): 79–86, 1951. 12

[34] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. *arXiv preprint arXiv:1608.07916*, 2016. 2

[35] Muheng Li, Yueqi Duan, Jie Zhou, and Jiwen Lu. Diffusion-sdf: Text-to-shape via voxelized diffusion. *arXiv preprint arXiv:2212.03293*, 2022. 2

[36] Yuhan Li, Yishun Dou, Xuanhong Chen, Bingbing Ni, Yilin Sun, Yutian Liu, and Fuzhen Wang. 3dqd: Generalized deep 3d shape prior via part-discretized diffusion process. *arXiv preprint arXiv:2303.10406*, 2023. 2

[37] 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. 5, 6, 7, 8, 14, 15, 16

[38] Zhen Liu, Yao Feng, Michael J. Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. In *International Conference on Learning Representations*, 2023. 2

[39] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2837–2845, 2021. 2

[40] Zhaoyang Lyu, Jinyi Wang, Yuwei An, Ya Zhang, Dahua Lin, and Bo Dai. Controllable mesh generation through sparse latent point diffusion models. *arXiv preprint arXiv:2303.07938*, 2023. 2

[41] 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*, pages 11167–11176, 2020. 2, 3

[42] Luke Melas-Kyriazi, Christian Rupprecht, and Andrea Vedaldi. Pc2: Projection-conditioned point cloud diffusion for single-image 3d reconstruction. In *Arxiv*, 2023. 2

[43] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12677–12686, 2019. 2

[44] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 4213–4220. IEEE, 2019. 2, 5, 13, 14

[45] Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021. 2

[46] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2

[47] 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*, pages 8748–8763. PMLR, 2021. 4, 7

[48] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021. 4

[49] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1 (2):3, 2022. 2, 4

[50] Joseph Redmon. Darknet: Open source neural networks in c, 2013. 13

[51] 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*, pages 10684–10695, 2022. 1, 2, 3, 4, 5, 6, 7, 8, 14

[52] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015. 5

[53] 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. 5

[54] Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34:17480–17492, 2021. 5

[55] J Ryan Shue, Eric Ryan Chan, Ryan Po, Zachary Ankner, Jiajun Wu, and Gordon Wetzstein. 3d neural field generation using triplane diffusion. *arXiv preprint arXiv:2211.16677*, 2022. 2

[56] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 14

[57] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. PMLR, 2015. 2, 4

[58] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 16

[59] Colton Stearns, Jiateng Liu, Davis Rempe, Despoina Paschalidou, Jeong Joon Park, Sebastien Mascha, and Leonidas J Guibas. Curvecloudnet: Processing point clouds with 1d structure. *arXiv preprint arXiv:2303.12050*, 2023. 2, 3

[60] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016. 5, 12

[61] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3d architectures with sparse point-voxel convolution. In *European conference on computer vision*, pages 685–702. Springer, 2020. 5, 13

[62] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *Conference on Machine Learning and Systems (MLSys)*, 2022. 13

[63] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Gecco: Geometrically-conditioned point diffusion models. *arXiv preprint arXiv:2303.05916*, 2023. 2

[64] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 4

[65] Tsun-Hsuan Wang, Alexander Amini, Wilko Schwarting, Igor Gilitschenski, Sertac Karaman, and Daniela Rus. Learning interactive driving policies via data-driven simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 7745–7752. IEEE, 2022. 2

[66] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1887–1893. IEEE, 2018. 2

[67] Yuwen Xiong, Wei-Chiu Ma, Jingkang Wang, and Raquel Urtasun. Learning compact representations for lidar completion and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1074–1083, 2023. 2, 5

[68] Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. *arXiv preprint arXiv:2109.14084*, 2021. 4

[69] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4541–4550, 2019. 3, 12, 14

[70] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 206–215, 2018. 3

[71] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 2

[72] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1, 2, 4

[73] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 14

[74] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5826–5835, 2021. 2

[75] Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision*, pages 17–35. Springer, 2022. 2, 3, 5, 6, 7, 8, 12, 13

A. Further Explanation of Concepts

A.1. Range Images and Point Clouds Conversion

In Sec. 3.1, we introduced range images as the modality for both input and output within LiDAR Diffusion Models (LiDMs). Subsequently, in Sec. 3.2, we provided a concise overview of the conversion process from range images to point clouds. This section extends our implementation discourse by delving into more comprehensive details.

The depth values are logarithmically scaled. To covert the pixel value v back into depth value, we define:

$$\text{depth} = 2^{\omega \times v} - 1, \tag{9}$$

where ω is a predefined scale factor. Given the normalized location (a, b) of pixel x, where $a, b \in [0, 1]$, we can compute its yaw and pitch through:

$$\text{yaw} = (2a - 1) \times \pi, \tag{10}$$

$$\text{pitch} = (1 - b) \times (\text{fov}_{up} - \text{fov}_{down}) + \text{fov}_{down}, \tag{11}$$

where fov_{up} and fov_{down} are specified based on the sensor settings of different datasets. Through the above computation, we can obtain the 3D coordinate p of the pixel x. Likewise, we implement the conversion from a point cloud to a range image by performing the inverse calculation. For the 32-beam scenario, $\text{fov}_{up} = 10°$, $\text{fov}_{down} = -30°$, $\omega = 5.53$. For the 64-beam scenario, $\text{fov}_{up} = 3°$, $\text{fov}_{down} = -25°$, $\omega = 5.84$.

The transition from range images to point clouds is characterized by a lossless conversion. Conversely, when converting from point clouds to range images, occlusions commonly emerge. This occurrence is intricately tied to the resolution of range images. At a lower predefined resolution, multiple neighboring points tend to converge within a single pixel of a range image. In contrast, with a higher resolution, the incidence of missing pixels markedly rises, resulting in sparser range images. Hence, it is important to appropriately define resolutions in diverse scenarios to encompass more points with little geometric loss and to main a high density of range images. In the context of a 32-beam scenario, we set $H = 32$ and $W = 1024$, while in the 64-beam scenario, we set $H = 64$ and $W = 1024$.

A.2. Statistical Evaluation Metrics

In this paper, we adopt common statistical metrics, Jensen-Shannon Divergence (JSD) and Minimum Matching Distance (MMD), for evaluation introduced in [3] and adopted by some recent works [69, 75].

Jensen-Shannon Divergence (JSD) measures the degree to which point clouds of synthesized set S tend to occupy the similar locations as those of reference set R. It can be defined as follows:

$$\text{JSD}\left(P_S \| P_R\right) = \frac{1}{2} D_{KL}\left(P_R \| M\right) + \frac{1}{2} D_{KL}\left(P_S \| M\right), \tag{12}$$

where $M = \frac{1}{2}(P_R + P_S)$ and D_{KL} is KL divergence [33]. In this paper, we compute JSD after discretizing each LiDAR point cloud into 2000^2 voxels in the form of Birds' Eye View (BEV), with width and length of each voxel 0.05.

Minimum Matching Distance (MMD) matches each LiDAR point cloud of reference set R to the one in synthesized set S with minimum distance and averages all distances in the matching. It indicates the fidelity of S with respect to R. We define MMD as follows:

$$\text{MMD}\left(P_S \| P_R\right) = \frac{1}{|P_R|} \sum_{Y \in P_R} \min_{X \in P_S} D_{CD}(X, Y). \tag{13}$$

Considering efficiency, we choose Chamfer Distance (CD) instead of Earth Mover's Distance (EMD) to represent the distance of two LiDAR point clouds. Both are defined in Sec. B.1.1. Different from JSD, the computation of MMD requires traversing all reference samples for each synthesized sample, which results in larger amounts of computation. To guarantee its efficiency, we adopt a larger voxel size of 0.5 to voxelize each point cloud into 200^2 BEV.

A.3. Perceptual Evaluation Metrics

A.3.1 Background

In general, distinguished from statistical evaluation metrics, perceptual metrics describe the performance of generative models through a perceptual space provided by pretrained models. In light of the incompatibility of classification-based models in the context of LiDAR scenes, we opt for segmentation-based pretrained models to delineate the perceptual metrics proposed in this paper.

Similar to the widely adopted perceptual metrics Fréchet Image Distance (FID) [21] and Inception Score (IS) [60] in image synthesis, we compute the results of our proposed perceptual metrics in the final stage. Given a trained UNet-like model Θ consisting of an encoder $\Theta_{\mathcal{E}}$ with L layers and a decoder $\Theta_{\mathcal{D}}$ with L layers, the output activation of a pixel (before dropout) from the final stage can be defined as:

$$a_{final} = \Theta_{\mathcal{D}}^{L}([x, \Theta_{\mathcal{E}}^{1}(x)]), \tag{14}$$

where $a_{final} \in \mathbb{R}^{H \times W \times C}$.

A.3.2 Aggregation Manners

Unlike classification-based network, the output of segmentation-based networks is a map of activations. Therefore, we cannot directly obtain the global feature of the input. In this paper, we provide two possible manners,

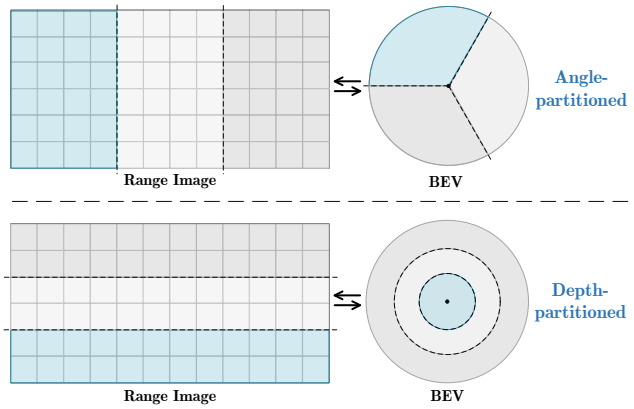


Figure 10. Example for two manners of partition-based aggregation on range images and bird's eye view (BEV). Angle-partitioned aggregation performs average on partitions of several columns on range images and of a sector on BEVs, while depth-partitioned aggregation performs average on partitions of several rows on range images and of a ring on BEVs. In this paper, we adopt depth-partitioned aggregation by default for its rolling-operation-invariant ability.

angle-partitioned aggregation and *depth*-partitioned aggregation, to approximately represent the global feature given the output map of activations of an input range image. An illustration is shown in Fig. 10. To obtain the global feature of one LiDAR point cloud, we uniformly divide it into P parts and concatenate all of them after average pooling on each part, resulting in a vector with $P \times C$ channels.

As shown in Fig. 10 *Above*, angle-partitioned aggregation partitions each LiDAR point cloud into P sectors by yaw angle. Since x-coordinate of each pixel on a range image is defined through linear transformation of the yaw angle of a point (*cf*., Sec. A.1), the range image is partitioned into P regions, and each is represented by W/P columns. Each sector has an equivalent region in the range image.

Similarly, in Fig. 10 *Below*, depth-partitioned aggregation splits a point cloud into P rings in the BEV level and P regions represented by H/P rows in the range-image level. Note that, different from angle-partitioned aggregation, the divided ring of the point cloud and its corresponding region of the range image are not equivalent in each pair.

Since the LiDAR point clouds are density-varying with depth, depth-partitioned aggregation is density-aware. Contrarily, the partitions by angle ignore the depth and each represent a sub-LiDAR-point-cloud. In this paper, we default to the utilization of depth-partitioned aggregation, as it effectively avoids the variability from rolling operation associated with angle-partitioned aggregation.

A.3.3 Implementation Details

In this paper, we propose three perceptual metrics: Fréchet Range Image Distance (FRID), Fréchet Sparse Volume Distance (FSVD), and Fréchet Point-based Volume Distance (FPVD) and set the number of partitions $P = 16$ by default. For each proposed perceptual metric, we further provide its details as follows:

- **FRID:** RangeNet++ [44] is a range-image-based method to predict per-pixel semantic labels. It adopts various image-based UNet for training. In this paper, we adopt a DarkNet21-based [50] model trained through the official implementation[1]. With the trained model, we can easily obtain the output in the final stage, which is in the shape of $64 \times 1024 \times 32$. We derive a global feature vector of each range image with 512 channels followed by a spatial averaging pooling. It is noteworthy that our proposed FRID effectively addresses the issue of result instability arising from random sampling, as indicated by the FRD score introduced in [75].
- **FSVD:** Sparse volumes are a prevalent 3D modality in LiDAR scenes. Unlike range images, volumes can directly represent 3D shapes without projection. To compute FSVD, we adopt a simple backbone, MinkowskiNet [10], to extract features from the sparse volumes converted from range images. We utilize a public implementation with the pretrained weights[2] based on torchsparse [62]. We calculate the average of all active (*i.e*., non-empty) voxel features for each partition in the final stage, resulting in a 1536-channel vector.
- **FPVD:** Leveraging the support of point clouds, the hybrid of point clouds and sparse volumes preserves a richer set of geometric information compared to utilizing sparse volumes alone. In the calculation of FPVD, we employ SPVCNN [61] as the backbone, utilizing the public implementation as in FSVD. The computational process of FPVD is the same as FSVD, with the output of 1536-channel global features.

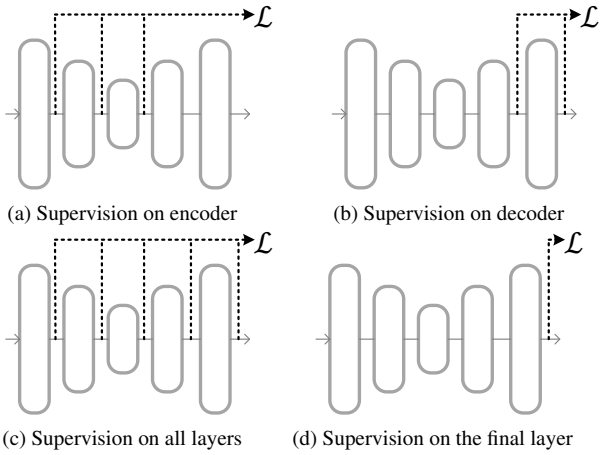
A.4. Details of Training

A.4.1 Perceptual Loss for LiDAR Compression

The regularization of models based on the pixel-wise depth of the synthesized range image and the ground-truth range image has the potential to disproportionately penalize outputs that are, in fact, LiDAR-realistic. For instance, given the same 2D projected shapes and scales in range images, generating cars farther away from the ego-center than closer to the ego-center induces a high loss. To mitigate this challenge, we leverage the success demonstrated by perceptual loss in the domain of image synthesis.

[1] https://github.com/PRBonn/lidar-bonnetal

[2] Implementation from https://github.com/yanx27/2DPASS



(a) Supervision on encoder (b) Supervision on decoder

(c) Supervision on all layers (d) Supervision on the final layer

Figure 11. Different types of feature extraction for the computation of perceptual loss.

[19] introduced the output of different image processing stages in a pretrained VGG network [56] as the "content representation". This idea was subsequently evolved as a perceptual regularization to learn both fine-grained details by matching lower-layer activations and global part arrangement by matching higher-layer activations. This regularization is widely adopted in various image tasks, including image super-resolution [6, 27], neural style transfer [19], and image synthesis [8, 12]. Some recent popular generators [15, 51] are trained in a perceptual space based on LPIPS [73], a common learned perceptual metric to evaluate image synthesis performance. Unfortunately, this "learned" metric is not available in the context of LiDAR scenes, and thus in this paper, we design perceptual loss by matching the output activations of stages with different scales, similar to the perceptual loss introduced in [8].

We utilize pretrained segmentation-based networks (*e.g.*, RangeNet++ [44]) to extract features as the preparation of feature matching. We explore four variants of perceptual loss based on different types of feature extraction. An illustration is shown in Fig. 11.

A.4.2 Mask Prediction Loss in LiDAR Compression

Though range images are dense representation for LiDAR scans, they contain a large number of invalid pixels. To distinguish them from valid pixels, our autoencoder outputs a binary mask along with the resulting range image. To this end, we apply mask prediction loss to our reconstruction loss and adversarial loss as follows:

$$\mathcal{L}_{rec}(x) = \mathbb{E}_x[\|x - \hat{x}\| + \lambda_1 \|p - \hat{p}\|_2^2 + \lambda_2 \|m - \hat{m}\|_2^2], \tag{15}$$

$$\mathcal{L}_{\text{GAN}}(x) = \mathbb{E}_x[\log \mathcal{D}([x, p, m]) + \log(1 - \mathcal{D}([\hat{x}, \hat{p}, \hat{m}]))], \tag{16}$$

where m is the mask value corresponding to x.

B. Additional Experimental Results

B.1. Design of Autoencoders for LiDAR Compression

B.1.1 Settings

To study on the behavior of LiDAR compression with different manners and ratios of downsampling, we provide our comprehensive studies on the design of autoencoders for LiDAR compression. In Table 4, we conduct experiments on various downsampling factors f_c and f_p for curve-wise and/or patch-wise encoding, respectively. To set up a comparable test field, we fix computational resources to four NVIDIA RTX 3090 GPUs and training steps to 40k steps for all listed experiments. We train autoencoders on KITTI-360 [37] and evaluate the quality of reconstruction in autoencoding process with perceptual metrics of reconstruction, *i.e.*, R-FRID, R-FSVD, R-FPVD, and statistical metrics, *i.e.*, Chamfer Distance (CD), Earth Mover's Distance (EMD). Following prior works [3, 69], we define CD and EMD as follows:

$$\text{CD}(P_x, P_{\hat{x}}) = \sum_{p \in P_x} \min_{\hat{p} \in P_{\hat{x}}} \|p - \hat{p}\|_2^2 + \sum_{\hat{p} \in P_{\hat{x}}} \min_{p \in P_x} \|p - \hat{p}\|_2^2, \tag{17}$$

$$\text{EMD}(P_x, P_{\hat{x}}) = \min_{\phi: P_x \to P_{\hat{x}}} \sum_{p \in P_x} \|p - \phi(p)\|_2, \tag{18}$$

where P_x and $P_{\hat{x}}$ are the ground-truth and reconstructed point clouds and ϕ is a bijection between them. Note that, P_x and $P_{\hat{x}}$ are the point clouds projected back from range images x and $\hat{x}$ instead of the raw point clouds.

B.1.2 Analysis and Discussion

As shown in [51], performance of image compression is highly related with the synthesis quality of DMs. By analyzing the results of either curve-wise or patch-wise encoding in Table 4, we conclude several valuable clues for the design of autoencoders: for curve-wise encoding (**Curve**), (i) as indicated by metrics R-FSVD and R-FPVD, the quality in the point-cloud level decreases with f_c increasing, and (ii) when $f_c \in \{4, 8, 16\}$, curve-wise encoding strikes better perceptually faithful results, while for patch-wise encoding (**Patch**), (i) when $f_p = 4$, with the same overall scale factor f, patch-wise encoding results in comparable reconstructed results of curve-wise encoding with $f_c = 16$, and (ii) when $f = 4$, curve-wise encoding outperforms patch-wise encoding by a large margin in both point-cloud and range-image level.

Curve-wise and patch-wise encoding can be complementary: curve-wise encoding learns within horizontal receptive fields to capture the curve-like structures existing in

	f_c	f_p	c	$\lvert\mathcal{Z}\rvert$	Overall Scale f	Encoded Size	R-FRID ↓	R-FSVD ↓	R-FPVD ↓	CD ↓	EMD ↓	#Params (M)
Curve	4	1	2	4096	4	$64 \times 256 \times 2$	**0.2**	**12.9**	**13.8**	**0.069**	**0.151**	9.52
	8	1	3	8192	8	$64 \times 128 \times 3$	0.9	21.2	17.4	0.141	0.230	10.76
	16	1	4	16384	16	$64 \times 64 \times 4$	2.8	31.1	23.9	0.220	0.265	12.43
	32	1	8	16384	32	$64 \times 32 \times 8$	16.4	49.0	38.5	0.438	0.344	13.72
	64	1	16	16384	64	$64 \times 16 \times 16$	34.1	98.4	83.7	0.796	0.437	20.06
Patch	1	2	2	4096	4	$32 \times 512 \times 2$	1.5	25.0	23.8	**0.096**	**0.178**	2.87
	1	4	4	16384	16	$16 \times 256 \times 4$	**0.6**	**15.4**	**15.8**	0.142	0.233	12.45
	1	8	16	16384	64	$8 \times 128 \times 16$	17.7	35.7	33.1	0.384	0.327	15.78
	1	16	64	16384	256	$4 \times 64 \times 64$	37.1	68.7	63.9	0.699	0.416	16.25
Hybrid ($8 \leq f \leq 64$)	2	2	3	8192	8	$32 \times 256 \times 3$	**0.4**	**11.2**	**12.2**	**0.094**	**0.199**	13.09
	4	2	4	16384	16	$32 \times 128 \times 4$	3.9	19.6	16.6	0.197	0.236	14.35
	8	2	8	16384	32	$32 \times 64 \times 8$	8.0	25.3	20.2	0.277	0.294	16.06
	16	2	16	16384	64	$32 \times 32 \times 16$	21.5	54.2	44.6	0.491	0.371	17.44
	2	4	8	16384	32	$16 \times 128 \times 8$	2.5	16.9	15.8	0.205	0.273	15.07
	4	4	16	16384	64	$16 \times 64 \times 16$	13.8	29.5	25.4	0.341	0.317	16.86

Table 4. Performance of autoencoders in different downsampling factors f_c and f_p *after 40k training steps* on the KITTI-360 *val* [37]. f_c is the curve-wise encoding factor, and f_p is the patch-wise encoding factor. $f = f_c \times f_p^2$ is the overall scaling factor. Encoded size ($h \times w \times c$) is the output after encoding, where $h = H/f_p$ and $w = W/(f_c \times f_p)$. We evaluate the reconstruction quality of the trained autoencoders through reconstruction-based perceptual metrics (*i.e.*, R-FRID, R-FSVD, R-FPVD) and statistical pairwise metrics (*i.e.*, CD, EMD). For comparison of *each* encoding manner, **bold** means the best in one metric, and underline means the second best.

	f_c	f_p	c	$\lvert\mathcal{Z}\rvert$	Overall Scale f	Encoded Size	FRID ↓	FSVD ↓	FPVD ↓	JSD ↓	MMD ($\times 10^{-4}$) ↓	#Params (M)
Curve	4	1	2	4096	4	$64 \times 256 \times 2$	271	148	118	0.262	5.33	9.5+36*
	8	1	3	8192	8	$64 \times 128 \times 3$	162	**85**	**68**	0.234	**5.03**	10.8+258
	16	1	4	16384	16	$64 \times 64 \times 4$	**142**	116	106	**0.232**	5.15	11.1+258
Patch	1	2	2	4096	4	$32 \times 512 \times 2$	205	154	132	0.248	6.15	2.9+36*
	1	4	4	16384	16	$16 \times 256 \times 4$	**180**	**60**	**55**	**0.230**	**5.34**	12.5+258
	1	8	16	16384	64	$8 \times 128 \times 16$	192	88	78	0.243	5.14	15.8+258
Hybrid ($8 \leq f \leq 64$)	2	2	3	8192	8	$32 \times 256 \times 3$	161	73	63	0.228	5.44	13.1+258
	2	2	3	16384	8	$32 \times 256 \times 3$	165	76	65	0.231	5.28	13.1+258
	4	2	4	16384	16	$32 \times 128 \times 4$	**145**	77	68	**0.222**	5.10	14.4+258
	8	2	8	16384	32	$32 \times 64 \times 8$	188	83	71	0.228	5.33	16.1+258
	2	4	8	16384	32	$16 \times 128 \times 8$	162	**56**	**49**	0.228	**4.82**	15.1+258
	4	4	16	16384	64	$16 \times 64 \times 16$	195	80	70	0.240	5.84	16.9+258

Table 5. Performance of LiDMs with autoencoders in different downsampling factors f_c and f_p *after 10k training steps* on the KITTI-360 *val* [37]. f_c is the curve-wise encoding factor, and f_p is the patch-wise encoding factor. $f = f_c \times f_p^2$ is the overall scaling factor. Encoded size ($h \times w \times c$) is the output after encoding, where $h = H/f_p$ and $w = W/(f_c \times f_p)$. We evaluate the synthesis quality of the trained LiDMs through perceptual metrics (*i.e.*, FRID, FSVD, FPVD) and statistical metrics (*i.e.*, JSD, MMD). For comparison of *each* encoding manner, **bold** means the best in one metric. We present the number of parameters (#Params) with blue for the autoencoder part and red is for the diffusion model part. $*$: Modification on the number of basic channels for appropriate GPU memory cost.

range images, and patch-wise encoding after curve-wise encoding vertically extends the receptive fields to learn object-level information. Following this nature, we design autoencoders to compress range images through both curve-wise and patch-wise encoding as hybrid encoding.

Based on the aforementioned analysis, we conduct studies on hybrid encoding (**Hybrid**) with diverse settings, keeping overall scale $8 \leq f \leq 64$. The results are listed in Table 4. Considering both performance and efficiency (overall scale), we select two settings: (a) $f_c = 2$, $f_p = 2$, and (b) $f_c = 2$, $f_p = 4$. We indicate that Model (a) outperforms all settings of curve-wise and patch-wise encoding when $f \geq 8$, and Model (b) achieves both competitive performance and high compression rate.

B.2. Design of LiDAR Diffusion Models

B.2.1 Settings

In Sec. B.1, we report the performance of autoencoders with different encoding manners and scaling factors. Although the reconstruction performance on validation set and the synthesis quality of DMs show a strong relation, the two are not always *positively* correlated. Thus, to further explore the behavior of LiDAR compression, we conduct experiments to train DMs with the trained autoencoders. The results are reported in Table 5. We train each DM with 10k steps and adopt the same experimental setup in Sec. B.1. We apply

50 sampling DDIM [58] steps to each model and generate 5,000 samples for evaluation.

B.2.2 Analysis and Discussion

Through the experimental results in Table 5, we conclude a similar fact as in Sec. B.1 that hybrid encoding generally performs much better than curve-wise or patch-wise encoding with the same compression rate. However, though the reconstruction performance of Model (a) ($f_c = 2$, $f_p = 2$) performs the best among all settings in Table 4, Model (b) ($f_c = 2$, $f_p = 4$) generates samples with better synthesis quality under an attractive compression rate. Therefore, in this paper, we adopt Model (b) ($f_c = 2$, $f_p = 4$) as the default autoencoder for LiDAR compression.

C. Additional Qualitative Results

C.1. 32-Beam Unconditional LiDAR Generation

In Fig. 12, we visualize the results unconditional LiDM on 32-beam data. 32-beam results appear sparser and noisier than the 64-beam results. We argue that this is highly related to the density and quality of the collected LiDAR point clouds. The 64-beam dataset, KITTI-360 [37], provides point clouds with denser foreground objects and clear boundaries between objects and backgrounds (*e.g.*, walls, roads). LiDMs benefit from the dense data, and thus can recognize objects more easily and learn from the geometry of complex 3D scenes.

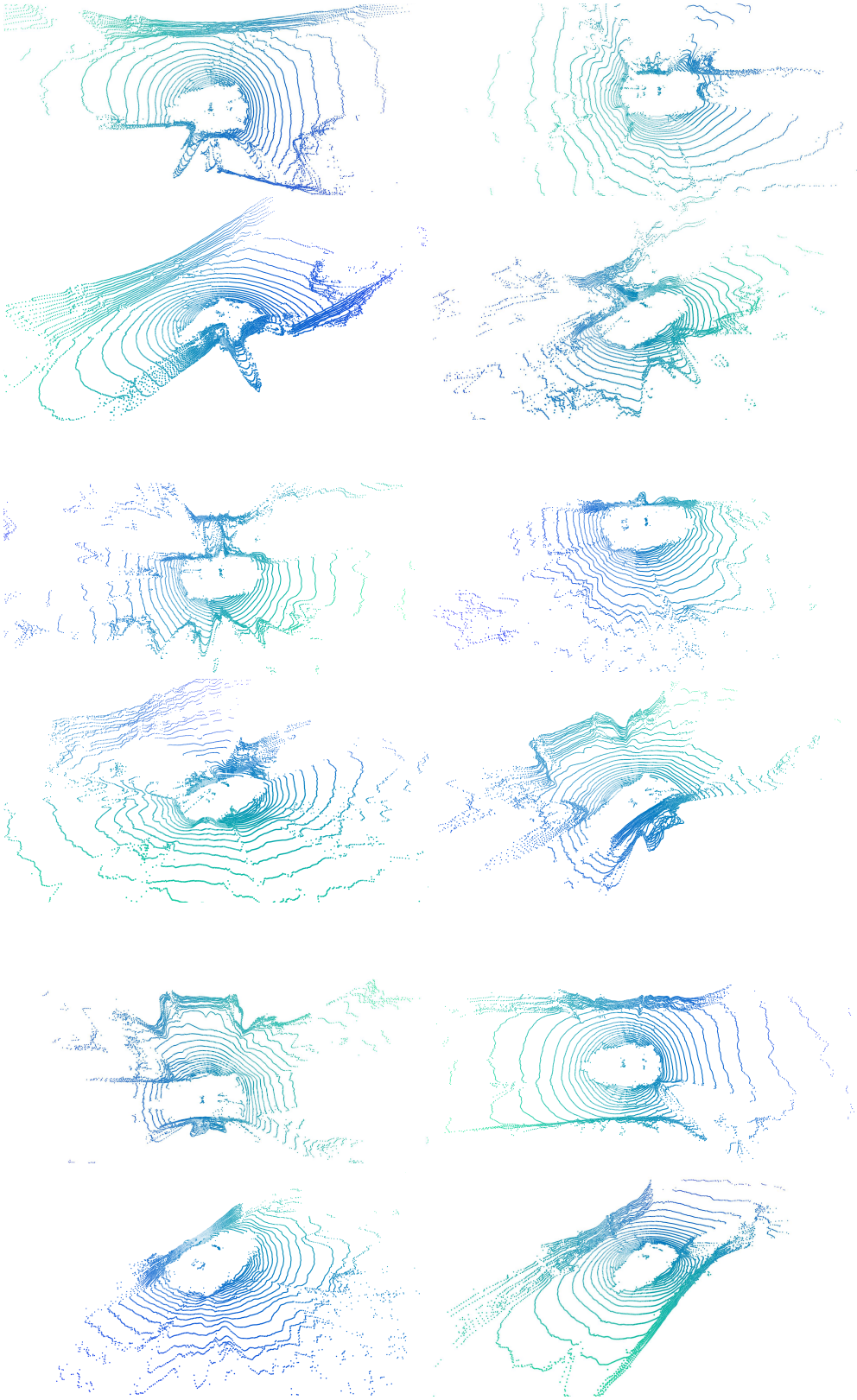


Figure 12. Unconditional samples on 32-beam scenario.