Learning to Generate Realistic LiDAR Point Clouds

Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang

University of Illinois at Urbana-Champaign, IL, USA
{vlasz2,xiyuez2,shenlong}@illinois.edu

Abstract. We present LiDARGen, a novel, effective, and controllable generative model that produces realistic LiDAR point cloud sensory readings. Our method leverages the powerful score-matching energy-based model and formulates the point cloud generation process as a stochastic denoising process in the equirectangular view. This model allows us to sample diverse and high-quality point cloud samples with guaranteed physical feasibility and controllability. We validate the effectiveness of our method on the challenging KITTI-360 and NuScenes datasets. The quantitative and qualitative results show that our approach produces more realistic samples than other generative models. Furthermore, LiDARGen can sample point clouds conditioned on inputs without retraining. We demonstrate that our proposed generative model could be directly used to densify LiDAR point clouds. Our code is available at: https://www.zyrianov.org/lidargen/

Keywords: LiDAR generation, self-driving, diffusion models

1 Introduction

The past decade witnessed rapid progress in machine perception. Many embodied systems leverage various sensors and the power of deep learning to perceive the world better. LiDAR provides accurate 3D geometry of the surrounding environment, making it one of the most popular sensor choices for various autonomous systems, including self-driving cars [33, 38, 77], surveying drones [84], indoor robots [8], and planetary rovers [9].

Realistic and scalable LiDAR simulation suites are desirable for studying LiDAR-based perception for various reasons. First, LiDAR is an expensive sensor. A 64-beam spinning LiDAR costs over 50,000 USD [1]. Not everyone can afford one, prohibiting physical sensors from being a scalable and customizable solution for data collection in research. Furthermore, training and testing in safety-critical situations is crucial for autonomy safety. However, collecting data for extreme scenarios in the real world is costly, unsafe, and even unethical. Simulations allow overcoming the above limitations by generating realistic data for long-tail events and training and testing agents at low cost.

Nevertheless, generating highly realistic and scalable LiDAR data remains an unsolved challenge. Existing approaches are either unrealistic or not scalable.

The primary paradigm for creating realistic LiDAR data is through model-based simulation. Early work on LiDAR simulation is purely physics-driven. The general idea is to mimic the time-of-flight (ToF) sensing process of LiDAR [14]. Specifically, the simulator casts rays in a 3D environment and simulates the receiver's returns by measuring the distance of the hitting surface to the sensor. The reality gap remains substantial because of the imperfect physical model and artist-designed assets. State-of-the-art simulation [44] combines physical modeling with learning components to compensate for complicated rendering artifacts. It produces high realism in both geometry and radiometric appearance. In addition, such a simulation method also gives complete controllability, allowing us to rearrange the scene layout and change viewpoint freely. However, the data-driven approach requires scanning the physical world in advance, which is expensive and not scalable. Recent approaches [5] investigated asset-free LiDAR generation using deep generative models to overcome such a limit. Nevertheless, neither controllability nor realism has yet been achieved.

In this paper, we present LiDARGen, a *realistic*, *controllable* and *asset-free* LiDAR point cloud generation framework. Following the imaging process of spinning LiDAR, we treat each LiDAR scan as an equirectangular view image, a 2.5D panoramic representation encoding information about ray angles, reflectance, and depth range. Generating LiDAR points under this representation guarantees physical feasibility. Inspired by the success of score-matching diffusion models in image generation [58], LiDARGen then learns a score function [25, 67], modeling the log-likelihood gradient given a sample in the equirectangular image space. This score function is trained on real-world LiDAR datasets. In the sampling stage, our method gradually converts an initial Gaussian random noise point into a realistic point cloud by progressively applying the score function to remove the noise via Langevin dynamics [58, 72]. Fig. 1 depicts an overview of our approach.

LiDARGen can be applied to conditional generation, such as LiDAR densification, by sampling from a posterior distribution [61]. Specifically, we leverage Bayes' rule to calculate the prior gradient based on the score-matching generative model and the likelihood gradient reflecting the conditions. The generated results are both realistic and plausible with regard to the controlled input. Notably, we also enjoy the simplicity – such a controlled generation process does not require retraining new models.

We validate the effectiveness of our method on the KITTI-360 [38] and NuScenes [6] datasets. Results demonstrate superior performance compared to other competing methods in various metrics and visual quality. We further evaluate LiDAR densification performance, demonstrating LiDARGen's potential for downstream tasks.

2 Related Work

This work studies the problem of generating realistic 3D LiDAR point clouds. It closely relates to point cloud generation and 3D deep learning. We also draw upon various efforts in energy-based generative models and LiDAR simulation.

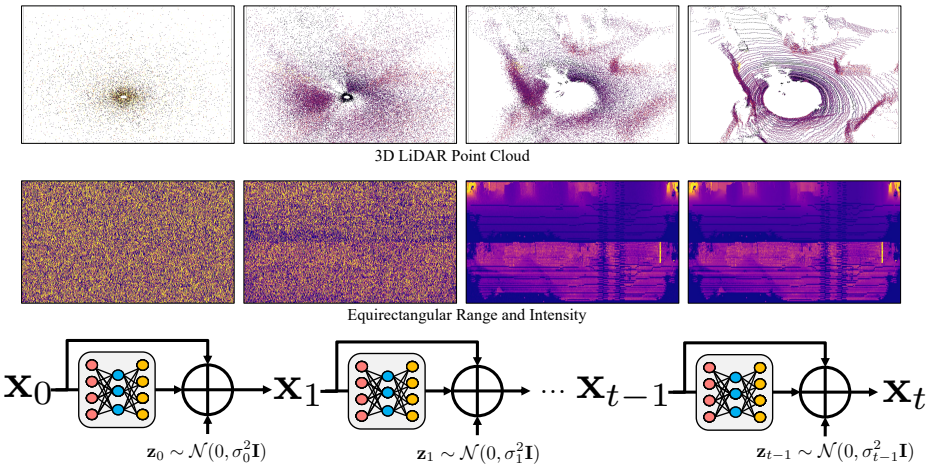


Fig. 1: Overview. We sample a LiDAR point cloud by progressively denoising the equirectangular view using a trained score function and Langevin dynamics.

2.1 Point Cloud Generation

Our task of generating LiDAR point clouds belongs to the broad category of 3D point cloud generation [2, 15, 17, 55, 64, 66, 82, 83]. Various works successfully apply deep generative models to advance this task. Representative works include varational autoencoder [15, 20, 83], generative adversarial networks [2, 35, 55, 66, 82], flow-based methods [64, 78], and diffusion processes [7, 43, 79].

Most previous methods on point cloud generation treat point clouds as fixed-size data [2, 15, 55, 66, 82, 83], restricting their practicability in handling real-world data, where the number of points varies significantly. Recent works [43, 78] start to look into point cloud generation with a diverse size. However, to achieve this, these approaches require an additional resampling step from an implicit surface distribution or an expensive auto-regressive procedure. In our work, inspired by the physics of LiDAR sensing, we explicitly generate a mask from the range image, mimicking the ray-drop patterns of LiDAR. This masking operation allows us to generate point clouds with various sizes and guaranteed physical feasibility.

Most aforementioned point cloud generation methods are developed and evaluated on clean, synthetic object shapes, such as ShapeNet [10]. Due to several unique challenges, it remains unclear whether we can directly transfer this success to LiDAR point clouds. First, LiDAR scans are generated through a time-of-flight sensing process. An ideal generator should produce a point cloud following light transport physics. Additionally, LiDAR point clouds are partial observations of a large scene, making the data highly unstructured, sparse, and non-uniform. It is therefore much more challenging to generate realistic LiDAR point clouds. Several recent works explore this direction with moderate success [5, 52]. The pioneering work [5] applies a variational autoencoder and generative adversarial network on LiDAR point clouds. Another line of work also leverages GANs but exploits a hybrid representation [52]. However, the level of realism is still limited.

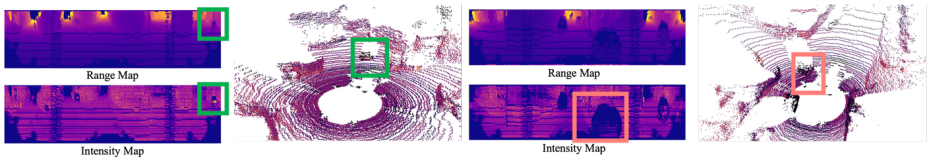


Fig. 2: LiDARGen sampling output. Left: equirectangular view. Right: 3D point cloud colored by intensity. Our method learns to generate cars with highly detailed structures, drop rays around transparent region,and produce reflectance intensity for specular objects.

2.2 Deep Learning for Point Clouds

One of the core challenges for generating a 3D point cloud is to select a good representation. This subsection briefly categorizes them according to the representation.

Voxelization methods build a dense volumetric representation in the form of a 3D grid [12, 18, 41, 42, 49, 77]. 3D convolution could then be directly applied to such representation. It is simple and straightforward. However, the dense voxel representation suffers from resolution loss and inefficient memory. Hence, researchers also improve the 3D voxel method with sparse convolutions [12, 18], hierarchical structures [81] and hybrid representations [41, 42].

Many neural networks directly learn to represent the **raw point cloud**. The pioneering work PointNet [39, 48, 50] leverages aggregation to collect context information. Other lines of research propose novel convolutional operators that can be directly applied on point clouds [24, 36, 45, 56, 62, 65, 69, 74, 76]. Graph-based methods explicitly create graph structures from the point cloud and exploit graph neural networks onto the structures [56, 68, 71, 85].

Another popular line of research models 3D data by projecting it onto 2D perspectives [11, 28, 33, 34, 46, 63, 77]. One can then directly apply 2D deep learning algorithms for 3D tasks. In these works, the depth value is often encoded in the 2D map, providing partial yet compact information about 3D. **Bird's eye view representation** [33, 77] is established through orthographically projecting the 3D point cloud along the vertical direction. **Perspective projection** obtains a 2D representation that resembles human vision. Several early works exploit perspective projection to produce images from multiple views and fuse the decision in 3D [11, 63]. The most closely related representation to us is the **equirectangular view representation**, which encodes the polar coordinate into a 2.5D image [13, 46, 73]. It provides a panoramic view of the surrounding scene that closely resembles the imaging process of LiDAR.

2.3 LiDAR Simulation

LiDAR simulation aims to produce a realistic LiDAR point cloud by mimicking the physical process of its imaging [3, 14, 23, 30, 44, 47, 57, 70, 75, 75]. Physical-based LiDAR simulation uses raycasting methods to simulate LiDAR. Ray intersections are calculated by shooting rays from the origin sensor position outwards onto a geometry surface of the environment. Most self-driving and robotics simulators

(e.g., CARLA [14] and Gazebo [30]) typically use this approach. However, physical-based approaches can suffer from a lack of realism because it requires very high-quality 3D assets (e.g., car models with material reflectance information, detailed maps with realistic foliage, etc.), and many LiDAR effects are difficult to simulate (e.g., atmospheric noise or LiDAR ray drop, which occurs when a LiDAR ray reflects off a surface and never gets a reading back). To approach this problem, research has recently focused on building data-driven LiDAR simulations [3, 23, 44, 70]. The representative work, LiDARSim [44] leverages machine learning models to learn ray-dropping patterns and exploits 3D assets that are built from modeling the urban environment. Many cut-and-paste data augmentation techniques could also be treated as a special form of LiDAR simulation, where objects are removed, inserted, or rearranged into a real LiDAR point cloud to create new ones. However, these methods build upon manipulating real-world point cloud data, restricting their scale and controllability. Furthermore, object insertion often aims to improve the performance of specific perception methods with little consideration for physical plausibility and realism [16, 19, 37].

3 Background

3.1 Energy-based Models

Given a dataset $\{\mathbf{x}_1, ...\mathbf{x}_N\}$, where each data sample $\mathbf{x}_i$ is assumed to be independently sampled from an underlying distribution, energy-based generative models aim to find a probability model in the following form that best fits the dataset: $p(\mathbf{x}) = \frac{e^{-E_\theta(\mathbf{x})}}{Z_\theta}$ where the energy function $E_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$ is a real-valued function and θ represents its learnable parameters. Z_θ is the normalization term $Z_\theta = \int e^{-E_\theta(\mathbf{x})} d\mathbf{x}$ that ensures the $p(\mathbf{x})$ to be a valid probability. Energy-based modeling is a family of expressive yet general probabilistic models that capture underlying dependencies among associated variables in high-dimensional space. Many generative models are instantiations of energy-based models, such as restricted Boltzmann machine [22], conditional random field [32], factor graphs [31] and recent deep energy-based generative models [58, 60]. Nevertheless, learning a generic energy model by maximizing log-likelihood is difficult, since computing the partition function Z_θ or estimating its gradient $\nabla_\theta \log Z_\theta$ is computationally intractable due to the integration over a high-dimensional space.

3.2 Score-based Energy Models

To alleviate this problem, researchers try to approximate the computation of the log-likelihood (or the gradient of the partition), with methods such as pseudo-likelihood [4], variational inference [29], and contrastive divergence [22]. Other methods bypass it using other learning objectives to train energy-based models, such as structured loss minimization [21]. Among them, score matching [26] recently became a popular choice thanks to its simplicity. Formally speaking, the

goal of score matching is to minimize the following objective:

$$\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\left[\text{tr}(\nabla_{\mathbf{x}}^2 \log p_\theta(\mathbf{x})) + \frac{1}{2}\|\nabla_{\mathbf{x}} \log p_\theta(\mathbf{x})\|_2^2\right] \tag{1}$$

As shown in this equation, the objective only involves the first and second-order gradient w.r.t. the $\mathbf{x}$, both of which are independent of the partition function.

Based on this, Hyvarinen [26] proposes the score-based model. It directly models the gradient of log-likelihood $\log p(\mathbf{x})$ with a parametric score function $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}^D$. However, minimizing the original score matching objective in Eq. 1 involves calculating the gradient of the Hessian of the log-likelihood $\nabla_\theta \nabla_{\mathbf{x}} s_\theta(\mathbf{x})$, which is computationally expensive. Vincent et al. [67] further reformulates the score matching energy and shows that score-based models could be more efficiently trained with the following denoising objective:

$$\frac{1}{2}\mathbb{E}_{p_{\text{data}}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma^2 I)}\left[\left\|s_\theta(\tilde{\mathbf{x}}) + \frac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma^2}\right\|_2^2\right] \tag{2}$$

where $\tilde{\mathbf{x}}$ is the Gaussian-noise perturbed sample and σ is the standard deviation.

Sampling from a score-based model $s_\theta(\mathbf{x})$ can be done with Langevin dynamics [72]. It is a Markov chain Monte Carlo (MCMC) process that can be interpreted as a noisy form of gradient ascent. For each step, Langevin dynamics sums the value of the previous step, the current gradient estimation based on the score function, and a Gaussian-distributed random noise:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\epsilon_t}{2}\nabla_{\mathbf{x}} \log p(\mathbf{x}_{t-1}) + \sqrt{\epsilon_t}\boldsymbol{z}_t \tag{3}$$

where $\boldsymbol{z}_t \sim \mathcal{N}(0, I)$ and ϵ_t is the learning rate, which is usually decreased (annealed) with a schedule [58]. When $\epsilon \to 0$ and $t \to +\infty$ Langevin dynamics is convergent to a true samples from the distribution $p(\mathbf{x})$ under certain mild conditions [58]. The denoising score-matching model and its variants have shown state-of-the-art performance in data generation [60, 79].

4 Method

Our goal is to model the underlying distribution of LiDAR point clouds in an urban driving scenario. We could then leverage such a generative model to sample new point clouds or use it for downstream MAP inference tasks. The challenge of LiDAR generation is to model the diverse structures that exist in the real world while still maintaining physical plausibility. Towards this goal, we leverage the denoising score-matching generative model [58] to model the gradient of its log-probability. Formally speaking, our training dataset consists of a list of raw LiDAR point clouds $\{(\mathbf{x}_1, r_i), ...(\mathbf{x}_N, r_N)\}$, where $\mathbf{x}_i \in \mathbb{R}^{D_i \times 3}$ represents the 3D location and $r_i \in \mathbb{R}$ is a scalar representing the reflectance intensity value for each point. Our method learns a score function $s_\theta(\mathbf{x})$ to approximate $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ with score-matching [67]. Sampling can then be conducted with Langevin dynamics,

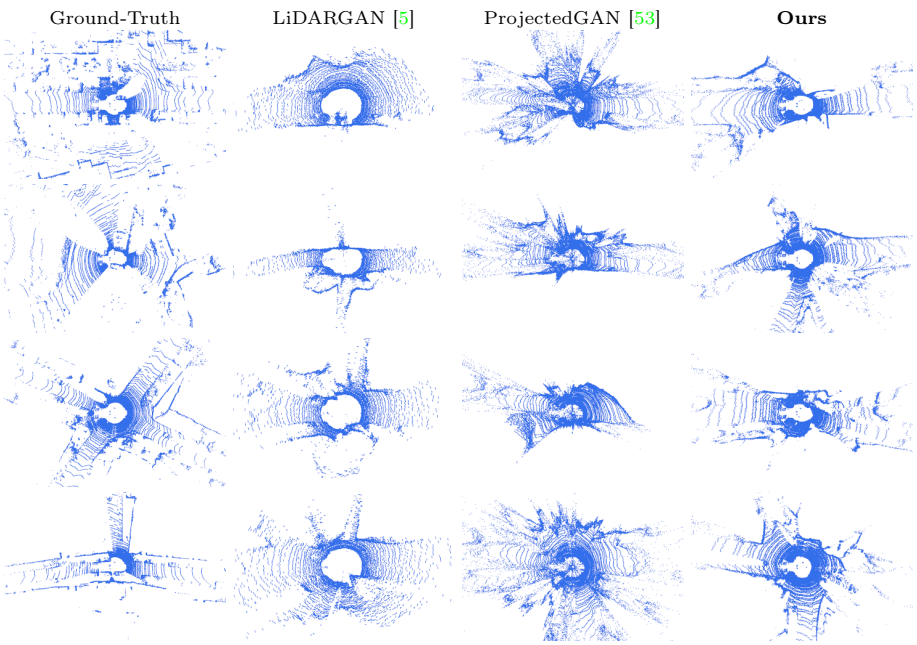


Fig. 3: Qualitative Results for LiDAR Point Cloud Generation on KITTI-360.

which gradually denoises an initial random Gaussian point cloud and returns a clean and realistic point cloud. Inspired by LiDAR's imaging process, we leverage the equirectangular representation as our underlying representation to ensure physical feasibility and develop an encoder-decoder network on top of it as the score function. Fig. 1 gives an overview of our approach.

4.1 LiDAR Generation

Input Representation Our model starts by converting the input parameterization from an unstructured point cloud sparsely distributed in euclidean space into a dense multi-channel equirectangular perspective image, with one channel representing depth and the other representing intensity. More specifically, we first convert each point from the Cartesian coordinate $\mathbf{x} = (x, y, z) \in \mathbb{R}^3$ into the spherical coordinate $\mathbf{z} \in (\theta, \phi, d)$: $d = \sqrt{x^2 + y^2 + z^2}, \theta = \arccos \frac{z}{\sqrt{x^2+y^2+z^2}}, \phi = \text{atan2}(y, x)$, where θ is the inclination, ϕ is azimuth and d is the depth range; atan2 is the standard 2-argument arctangent function taking into account the discontinuity across quadrant boundaries of $\text{atan}(y/x)$. Furthermore, we remap the depth range so that it is normalized from 0 to 1: The two-channel rectangular image is then produced through quantizing the two angles and rasterization. Concretely, for each point $\mathbf{z}_i = (\theta_i, \phi_i, d_i), r_i$: $\mathcal{I}(\lfloor \theta_i / s_\theta \rfloor, \lfloor \phi_j / s_\phi \rfloor) = \left(\frac{1}{6} \log_2(d_i + 1), \frac{1}{255} r_i\right)$. Both channels of the image are normalized to the range $(0, 1)$ and we use a logarithm mapping to esure nearby points have a higher geometry resolution. For simplicity, throughout the rest of section we will also use $\mathbf{x}$ to represent the point

cloud in its equirectangular representation. Fig. 2 demonstrates one example of the range view representation. Our input representation enjoys several benefits. Firstly, it encodes information into a dense and compact 2D map, allowing us to exploit efficient network architecture transferred from the 2D image generation domain. Secondly, due to the ray casting nature, most spinning LiDAR scans will only return the peak pulse for each beam[1]. In other words, encoding the point cloud into this representation will not lose any information, and the generated point cloud properly reflects the scanning and ray-casting nature of the sensor.

Network Architecture Our score-based network s_θ uses a **U-Net** architecture [51] following its success in image generation [27,59]. Specifically, at each step, it takes a $W \times H \times 2$ input image and outputs a $W \times H \times 2$ score map at the same size. We also make important changes suitable for our LiDAR point cloud generation task. Firstly, standard 2D images have disconnected left and right boundaries. Hence zero-padding or symmetry padding is often sufficient for dense prediction. However, equirectangular images are inherently circular. Applying standard convolutions does not take into account such constraints. To alleviate this issue, **circular convolution** [54] treats left and right boundaries as connected neighbors in its topology. Inspired by this, we substitute all the convolution and pooling layers in our network with circular versions. Second, LiDAR point clouds collected from urban driving environments have a highly structured geometry. And this geometric structure is often viewpoint-aware. For instance, the depth range of the lower beam might have a strong bias due to the ground height; the depth range of the frontal facing positions tends to be larger since the car is mostly driving forward along a straight road. To better encode this prior, our model takes the **angular coordinate as an additional input** to the convolution, similarly to CoordConv [40].

Training One of the difficulties for training denoising score matching models is the choice of a proper noise level for Eq. 2, which heavily influences the accuracy of score estimation. In practice, we find that having a noise-conditioned extension is crucial for its success. More specifically, we expand our score network $s_\theta(\mathbf{x}, \sigma_i)$ to be dependent on the current noise perturbation level σ_i. At the training stage, following the noise-conditioned score-matching model [58], we adopt a multi-scale loss function, with a re-weighting factor for the loss at each noise level:

$$\frac{1}{2L}\sum_{i=1}^{L}\sigma_i^2\mathbb{E}_{p_{data}(\mathbf{x})}\mathbb{E}_{\tilde{\mathbf{x}}\sim\mathcal{N}(\mathbf{x},\sigma_i^2 I)}\left[\left\|s_\theta(\tilde{\mathbf{x}},\sigma_i)+\frac{\tilde{\mathbf{x}}-\mathbf{x}}{\sigma_i^2}\right\|\right] \tag{4}$$

where $\tilde{\mathbf{x}}$ is the randomly perturbed noisy signal at each level, and σ_i is the standard deviation of the noise distribution.

Sampling We exploit annealed Langevin dynamics sampling [59] for our point generation task to increase sampling efficiency. Specifically, we start from the highest pretrained noise level and gradually reduce the noise level:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \gamma\frac{\sigma_i^2}{2\sigma_L^2}s_\theta(\mathbf{x}_{t-1},\sigma_i) + \gamma\frac{\sigma_i}{\sigma_L}\mathbf{z}_t \tag{5}$$

[1] some sensors return two beams for a small fraction of beams

where γ is the learning rate and σ_L is the smallest noise level. The final step of Langevin dynamic sampling gives us a clean equirectangular range image. We unproject this resulting image back into 3D Cartesian space to recover the 3D point cloud. Please refer to Fig. 1 for the full sampling procedure.

4.2 Posterior Sampling

Learning the unconditional prior distribution $p(\mathbf{x})$ of LiDAR point clouds enables many applications. In particular, we often expect our generated LiDAR point cloud to satisfy a specific property or to be consistent with certain conditions. For instance, we might want to create a LiDAR point cloud that agrees with some partial observation; or we might generate a LiDAR point cloud conditioned on its semantic layout. Conventional methods, such as GANs, often require training different conditional generative models for each task. However, thanks to the gradient-based approach used in score-based models, we could efficiently conduct the tasks mentioned above with only the pretrained unconditional generation model $p(\mathbf{x})$. Next, we will show how to achieve this in LiDARGen.

Specifically, given a pretrained generation model $p(\mathbf{x})$ and an input condition $\mathbf{y}$, we formulate the agreement between the condition $\mathbf{y}$ and the LiDAR point cloud $\mathbf{x}$ as a likelihoood function $p(\mathbf{y}|\mathbf{x})$. Our goal is to sample new point cloud that reflect the input condition $p(\mathbf{x}|\mathbf{y})$. According to the Bayes' rule we have:

$$p(\mathbf{x}|\mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x})/p(\mathbf{y}), \nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}) + \nabla_{\mathbf{x}} \log p(\mathbf{x}) \quad (6)$$

where $\nabla_{\mathbf{x}} \log p(\mathbf{x})$ is our pretrained score function $s_\theta(\mathbf{x})$. In many situations, the likelihood model has an analytical gradient or is a neural network; hence calculating the gradient with respect to $\mathbf{x}$ is straightforward. We, therefore, leverage the following Langevin dynamics to sample from the posterior distribution:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \frac{\epsilon}{2}\left(s_\theta(\mathbf{x}_{t-1}) + \nabla_{\mathbf{x}} \log p(\mathbf{y}|\mathbf{x}_{t-1})\right) + \sqrt{\epsilon}\mathbf{z}_t. \quad (7)$$

Next, we will discuss a concrete application of posterior sampling.

LiDAR Densification Spinning LiDAR used on autonomy is expensive due to its complicated mechanical design. In particular, the price of LiDAR sensors grows exponentially as the number of beams increases. Therefore, there is a practical need to produce high-beam LiDAR readings with a low-beam model. Given a low-beam LiDAR point cloud $\mathbf{y}$, our goal is to recover its high-beam version $\mathbf{x}$. Assuming that $\mathbf{m}$ is the visibility mask denoting pixels with a provided gt ray, the gradient of the posterior can be computed as follows:

$$\nabla_{\mathbf{x}} \log p(\mathbf{x}|\mathbf{y}) = \nabla_{\mathbf{x}} \left(\log p(\mathbf{x}) + \frac{\lambda}{2} ||(\mathbf{x} - \mathbf{y}) \odot \mathbf{m}||^2 \right) = s_\theta(\mathbf{x}) + \lambda \left[(\mathbf{x} - \mathbf{y}) \odot \mathbf{m} \right],$$

where $\odot$ is the Hadamard product. Intuitively, each Langevin dynamic step pushes the samples towards the direction of being both realistic and consistent with the partial observation $\mathbf{y}$.

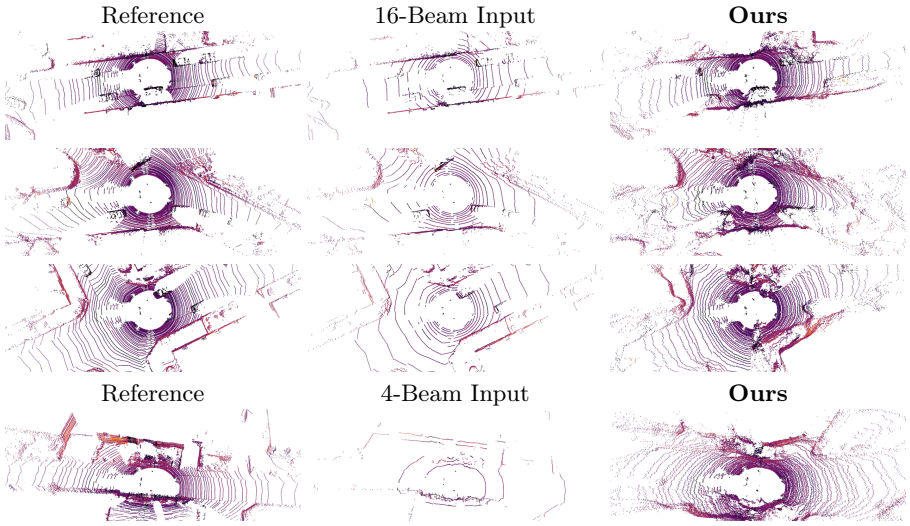


Fig. 4: Qualitative Results for Unsupervised LiDAR Densification.

5 Experiments

5.1 Experimental Setup

Datasets We train and test our model's performance on the challenging KITTI-360 [38] and **nuScenes** [6] datasets. KITTI-360 contains 81,106 LiDAR readings from 9 long sequences around the suburbs of Karlsruhe, Germany. The scenes KITTI-360 covers are diverse, consisting of driving on highways and through residential and commercial districts. We split the dataset into two parts, where the first two sequences (30,758 frames) are the testing set, and the rest are used for training and cross-validation. nuScenes contains 297,737 LiDAR sweeps in the training set and 52,423 LiDAR sweeps in the testing and cross-validation set. The LiDAR sweeps were collected in the cities of Boston and Singapore. The two datasets provide different sensors (64 and 32 beams), geographic regions (EU and NA), and content (suburbs and cities).

Metrics Quantitatively measuring generative models is known to be difficult. In our work, we leverage three different metrics for evaluation. **Maximum Mean Discrepancy (MMD)** is a non-parameteric distance between two sets of samples. It compares the distance between two sets of samples by measuring the mean squared difference of the statistics of the two. MMD could be measured through the kernel trick:

$$\text{MMD} = \frac{1}{N^2}\sum_{i}^{N}\sum_{i'}^{N} k(\mathbf{x}_i, \mathbf{x}_{i'}) - \frac{2}{NM}\sum_{i}^{N}\sum_{j}^{M} k(\mathbf{x}_i, \mathbf{x}_j) + \frac{1}{M^2}\sum_{j}^{M}\sum_{j'}^{N} k(\mathbf{x}_j, \mathbf{x}_{j'}).$$

For each point cloud we compute a 50×50 spatial histogram along the ground plane (x and y coordinates) then use a Gaussian kernel to measure the similarity

Table 1: Quantitative Results on KITTI-360 [38].**Bold** is Best; Blue is Second.

	$\text{MMD}_{\text{BEV}} \downarrow$	$\text{FID}_{\text{range}} \downarrow$	$\text{JSD}_{\text{BEV}} \downarrow$
LiDAR GAN [5]	3.06×10^{-3}	3003.8	–
LiDAR VAE [5]	1.00×10^{-3}	2261.5	0.161
Projected GAN [53]	$\mathbf{3.47 \times 10^{-4}}$	**2117.2**	**0.085**
Ours	$\mathbf{3.87 \times 10^{-4}}$	**2040.1**	**0.067**

between the two. Additionally, inspired by the FID score for image generation, we evaluate a new **Frechet Range Distance** (FRD score) on KITTI-360. It evaluates the squared Wasserstein metric between mean and the covariance of a LiDAR perception network's activations from the synthetic samples and true samples. We choose RangeNet++, which is a encoder-decoder based network for segmentation pretrained on KITTI-360. To trade-off between quality and and preserve locality, we randomly choose 4,096 activation from the feature map of its bottleneck layer to fit the Gaussian distribution. Finally, we report the **Jensen–Shannon divergence** (JSD) between the empirical distributions of two sets of point clouds. We approximate the distribution through a birds-eye view 2D histogram at a resolution 100×100 for both reference sets and generated sets.

Baselines We have 3 baseline comparisons. The first baseline is the range-view based VAE-based LiDAR generation model proposed by Caccia et al. [5]. We have added additional layers and increased the generated range image size to 1024×64. We train the models for 165 epochs until convergence. The second baseline is the GAN-based model from Caccia et al. [5]. The GAN was pre-trained at a resolution of 256×40 following the original paper[2], followed by a upsampling layer to 1024×40. The last baseline is Projected GAN [53], one of the state-of-art GAN models for image generation. We adapt ProjectedGAN into our setting and train it for 3,000 epochs. All the generated range image samples are converted to a 3D point cloud in Cartesian coordinates for quality comparison.

Implementation Details We use a UNet-like model for the score function. It takes in a 64x1024x2 (KITTI) or 32x1024x2 (Nuscenes) tensor as input and outputs the same size, denoting the gradient of log-prob. The U-Net comprises a stack of 6 down-sampling and a stack of 6 upsampling blocks, with skip connections in between. Each block has two convs. Each conv is preceded by InstanceNorm++ and an ELU activation. Number of channels is 32-64-64-64-128-128-128-128-64-64-64-32. Our model was trained with Adam optimizer with a learning rate of 1e-4. For sampling, we use a gradient update step of 2e-6 and 5 iterations per noise level. The initial σ_0 is 50, the final σ_L is 0.01, and the number of levels is 232. To train Caccia et al.'s [5] models, we used a learning rate of 1e-4. All the models are trained and tested with an Nvidia RTX A4000 GPU.

[2]we did not manage to make training converge in higher resolution

Table 2: Human Study Results on KITTI-360

Method	Percent Prefer Ours
Ours vs. VAE	97%
Ours vs. GAN	96.6%
Ours vs. ProjectedGAN	100%

5.2 KITTI-360 Evaluation

Quantitative Results Tab. 1 shows quantitative results among all the competing algorithms. From the table we could see that our method produces superior performance on the FRD score compared against other methods. In terms of MMD our approach is also ranking high. It is slightly lower than projected GAN, however both match the histogram well with an MMD score smaller than 1e-4. As we mention in metric subsection, every metric is a partial evaluation of the sampling quality, and urge the readers to consider all quantitative metrics, the human study, and the qualitative results as a holistic evaluation.

Qualitative Results Fig. 3 demonstrates some randomly selected samples from all the competing algorithms. We also list the true point cloud samples from KITTI as a reference. From the figure, we could see our approach produces significantly higher quality samples than the competing algorithms. Specifically, LiDARGAN captures the overall layout, but fails in producing high-detailed structures, such as cars, trees, sidewalks, pedestrians, etc. Projected GAN generates reasonable, detailed structures at near range, but brings significant artifacts at far range. Ours excel in terms of both realism in layout and geometry details, as well as diversity in content. Additionally, we provide a zoom-in visualization of our 3D point cloud in Fig. 2, highlighting the high quality geometric details our method could offer.

Human Study To evaluate the perceptual quality, we perform an A/B test on a team of students. Our test system shows a pair of randomly chosen images of two point cloud sampled from two different methods. Human judges then choose which one is more realistic. Participates also have access to real KITTI point clouds for reference. The raw results are shown in Tab. 2. In total, 5 participants labeled 600 image pairs. At a confidence level of 99% the two-sided test p-value is smaller than 1e-4, demonstrating statistical significance.

5.3 NuScenes Results

Fig. 5 depicts the qualitative comparison results. From this figure, we can see that our method still achieves superior results compared to both VAE [5] and projected GAN [53]. An AB test on a group of four human subjects suggests that our method is significantly favored over other competing algorithms in 89% of cases. While achieving superior human study performance, we notice that our method tends to generate point clouds that concentrate their mass closer to the viewpoint. As a result, despite superior visual quality, our MMD score at BEV is

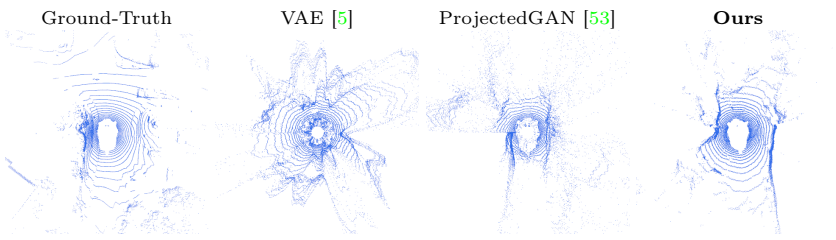


Fig. 5: Qualitative Results on the nuScenes dataset.

Table 3: LiDAR Densification.

	MAE ↓
PUNet [80]	6.88
NN	2.18
Ours	**1.23**

Table 4: Ablation Study

Coord-aware	CircConv	FRD	MMD
No	No	2422.3	7.60×10^{-4}
Yes	No	2251.1	3.94×10^{-4}
Yes	Yes	**2040.1**	$\mathbf{3.87 \times 10^{-4}}$

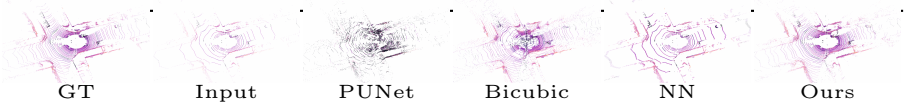


Fig. 6: Densification Results.

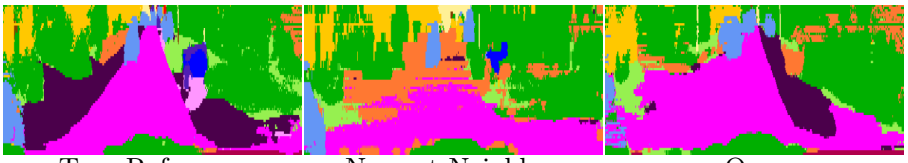
worse than VAE and Projected GAN (2e-3 vs. 1.1e-3 and 6e-5). We will leave this shrinking effect for future investigation.

5.4 Posterior Sampling

We also evaluate our LIDAR generation model on the task of LiDAR densification. More specifically, we simulate low-beam LiDAR sensor readings as our sparse input by selecting a subset of the beams from the raw 64-beam sensors. In this example, we create 4-beam and 16-beam input as shown in Fig. 4. Following the posterior sampling procedure described in Eq. 6 and Eq. 7. Fig. 4 depicts the sparse input, a dense ground-truth reference and our qualitative posterior sampling results. As shown in the figure, the resulting point cloud is realistic and reflects the input guidance.

Qualitative Comparison We compare PUNet [80], bicubic interpolation, and nearest neighbor interpolation with ours on KITTI-360. Quantitative results are shown in Tab. 3. Qualitative results are shown in Fig. 6. PUNet is B/W as it does not upsample intensity. Our results suggest that the proposed densification method is superior to both learned and interpolation approaches.

Downstream Applications We run RangeNet++ semantic seg on densified point cloud without fine-tuning. (Fig. 7). Applying LiDARGen to densify a sparse (16 Beam) LiDAR helps RangeNet++ create cleaner results (e.g., the road) and recover lost details (e.g., the cars in the distance). Our method also achieves better quantitative results compared to nearest-neighbour up-sampling. Per-point accuracy is 0.546 (NN) and 0.608 (ours). IOU is 0.394 (NN) and 0.449 (ours).



True Reference Nearest-Neighbor Ours

Fig. 7: RangeNet++ segmentation on densified LiDAR.

5.5 Discussions

Ablation Studies We conduct ablation studies to justify the design choice of our algorithms. We compare the same score function model in three different settings (w/o circular conv and w/o coordinate-encoding). As shown in Tab. 4, both help improve the performance in terms of FRD and MMD. For more information and qualitative comparison, please refer to the supplementary material.

Limitations Despite producing superior performance and flexibility, LiDARGen still has several limitations. First, sampling efficiency is one of the major drawback – LiDARGen takes approximately 12min to sample 36 LiDAR samples in a batch. We leave it as future work and anticipate that leveraging recent acceleration techniques for diffusion-based models is a promising direction to alleviate this issue. In addition, our approach cannot yet pass the Turing test for experienced LiDAR perception researchers. There are a few artifacts: our samples have degraded geometric details at far range and tended to have less straight walls than real samples. We plan to explore multi-modal networks (e.g. hybrid equirectangular view and bird's eye view) in the future.

6 Conclusion

We propose LiDARGen, a score-based approach for LiDAR point cloud generation. Our method samples a realistic point cloud by progressively denoising a noisy input. We demonstrate that our unconditional generation model could be directly applied for various conditional generation tasks through posterior sampling. A human study and perceptional similarity evaluation on the challenging KITTI-360 dataset validates the effectiveness of our method. We hope this approach will open up the research to provide easy access to realistic LiDAR sensory data directly from machine learning. We also expect to explore potential applications of LiDARGen in 3D environment generation and self-driving.

7 Acknowledgement

The authors thank Wei-Chiu Ma and Zhijian Liu for their feedback on early drafts and all the participants in the human perceptual quality study. The project is partially funded by the Illinois Smart Transportation Initiative STII-21-07. We also thank Nvidia for the Academic Hardware Grant.

References

1. Google's waymo invests in lidar technology, cuts costs by 90 percent. https://arstechnica.com/cars/2017/01/googles-waymo-invests-in-lidar-technology-cuts-costs-by-90-percent/, accessed: 2012-03-07 1
2. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: ICML (2018) 3
3. Amini, A., Wang, T.H., Gilitschenski, I., Schwarting, W., Liu, Z., Han, S., Karaman, S., Rus, D.: Vista 2.0: An open, data-driven simulator for multimodal sensing and policy learning for autonomous vehicles. arXiv preprint arXiv:2111.12083 (2021) 4, 5
4. Besag, J.: Statistical analysis of non-lattice data. Journal of the Royal Statistical Society: Series D (The Statistician) **24**(3), 179–195 (1975) 5
5. Caccia, L., van Hoof, H., Courville, A.C., Pineau, J.: Deep generative modeling of lidar data. IROS pp. 5034–5040 (2019) 2, 3, 7, 11, 12, 13
6. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019) 2, 10
7. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., Hariharan, B.: Learning gradient fields for shape generation. In: European Conference on Computer Vision. pp. 364–381. Springer (2020) 3
8. Cao, C., Zhu, H., Choset, H., Zhang, J.: Tare: A hierarchical framework for efficiently exploring complex 3d environments. In: Robotics: Science and Systems Conference (RSS), Virtual (2021) 1
9. Carle, P.J., Furgale, P.T., Barfoot, T.D.: Long-range rover localization by matching lidar scans to orbital elevation maps. Journal of Field Robotics **27**(3), 344–370 (2010) 1
10. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015) 3
11. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: CVPR (2017) 4
12. Choy, C.B., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In: ECCV (2016) 4
13. Cohen, T.S., Geiger, M., Köhler, J., Welling, M.: Spherical cnns. arXiv preprint arXiv:1801.10130 (2018) 4
14. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017) 2, 4, 5
15. Fan, H., Su, H., Guibas, L.J.: A point set generation network for 3d object reconstruction from a single image. In: CVPR (2017) 3
16. Fang, J., Zhou, D., Yan, F., Zhao, T., Zhang, F., Ma, Y., Wang, L., Yang, R.: Augmented lidar simulator for autonomous driving. IEEE Robotics and Automation Letters **5**(2), 1931–1938 (2020) 5
17. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 103–118 (2018) 3

18. Graham, B., Engelcke, M., Van Der Maaten, L.: 3d semantic segmentation with submanifold sparse convolutional networks. In: CVPR (2018) 4
19. Gusmão, G.F., Barbosa, C.R.H., Raposo, A.B.: Development and validation of lidar sensor simulators based on parallel raycasting. Sensors **20**(24), 7186 (2020) 5
20. Han, Z., Wang, X., Liu, Y.S., Zwicker, M.: Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). pp. 10441–10450. IEEE (2019) 3
21. Hazan, T., Keshet, J., McAllester, D.: Direct loss minimization for structured prediction. Advances in neural information processing systems **23** (2010) 5
22. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. science **313**(5786), 504–507 (2006) 5
23. Hu, J.S., Waslander, S.L.: Pattern-aware data augmentation for lidar 3d object detection. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 2703–2710. IEEE (2021) 4, 5
24. Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: Randla-net: Efficient semantic segmentation of large-scale point clouds. In: CVPR (2020) 4
25. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**, 695–709 (2005) 2
26. Hyvärinen, A.: Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research **6**(Apr), 695–709 (2005) 5, 6
27. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1125–1134 (2017) 8
28. Kanezaki, A., Matsushita, Y., Nishida, Y.: Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In: CVPR (2018) 4
29. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014) 5
30. Koenig, N., Howard, A.: 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). vol. 3, pp. 2149–2154. IEEE (2004) 4, 5
31. Kschischang, F.R., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. IEEE Transactions on information theory **47**(2), 498–519 (2001) 5
32. Lafferty, J.D., McCallum, A., Pereira, F.C.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: ICML (2001) 5
33. Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: CVPR (2019) 1, 4
34. Li, B., Zhang, T., Xia, T.: Vehicle detection from 3d lidar using fully convolutional network. In: RSS (2016) 4
35. Li, C.L., Zaheer, M., Zhang, Y., Poczos, B., Salakhutdinov, R.: Point cloud gan. arXiv preprint arXiv:1810.05795 (2018) 3
36. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on $\mathcal{X}$-transformed points. In: NIPS (2018) 4
37. Li, Y., Wen, C., Juefei-Xu, F., Feng, C.: Fooling lidar perception via adversarial trajectory perturbation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7898–7907 (2021) 5
38. Liao, Y., Xie, J., Geiger, A.: KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. arXiv preprint arXiv:2109.13410 (2021) 1, 2, 10, 11

39. Lin, Z., Feng, M., Santos, C.N.d., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding. In: ICLR (2017) 4
40. Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., Yosinski, J.: An intriguing failing of convolutional neural networks and the coordconv solution. Advances in neural information processing systems **31** (2018) 8
41. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel CNN for efficient 3d deep learning. CoRR **abs/1907.03739** (2019) 4
42. Liu, Z., Tang, H., Zhao, S., Shao, K., Han, S.: Pvnas: 3d neural architecture search with point-voxel convolution. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021) 4
43. Luo, S., Hu, W.: Diffusion probabilistic models for 3d point cloud generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2021) 3
44. Manivasagam, S., Wang, S., Wong, K., Zeng, W., Sazanovich, M., Tan, S., Yang, B., Ma, W.C., Urtasun, R.: 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) 2, 4, 5
45. Mao, J., Wang, X., Li, H.: Interpolated convolutional networks for 3d point cloud understanding. In: ICCV (2019) 4
46. Milioto, A., Vizzo, I., Behley, J., Stachniss, C.: Rangenet++: Fast and accurate lidar semantic segmentation. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4213–4220. IEEE (2019) 4
47. Nakashima, K., Kurazume, R.: Learning to drop points for lidar scan synthesis. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 222–229. IEEE (2021) 4
48. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: CVPR (2017) 4
49. Qi, C.R., Su, H., Nießner, M., Dai, A., Yan, M., Guibas, L.: Volumetric and multi-view cnns for object classification on 3d data. In: CVPR (2016) 4
50. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In: NeurIPS (2017) 4
51. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention. pp. 234–241. Springer (2015) 8
52. Sallab, A.E., Sobh, I., Zahran, M., Essam, N.: Lidar sensor modeling and data augmentation with gans for autonomous driving. arXiv preprint arXiv:1905.07290 (2019) 3
53. Sauer, A., Chitta, K., Müller, J., Geiger, A.: Projected gans converge faster. In: Advances in Neural Information Processing Systems (NeurIPS) (2021) 7, 11, 12, 13
54. Schubert, S., Neubert, P., Pöschmann, J., Protzel, P.: Circular convolutional neural networks for panoramic images and laser data. 2019 IEEE Intelligent Vehicles Symposium (IV) pp. 653–660 (2019) 8
55. Shu, D.W., Park, S.W., Kwon, J.: 3d point cloud generative adversarial network based on tree structured graph convolutions. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 3859–3868 (2019) 3
56. Simonovsky, M., Komodakis, N.: Dynamic edge-conditioned filters in convolutional neural networks on graphs. In: CVPR (2017) 4
57. Sobczak, Ł., Filus, K., Domański, A., Domańska, J.: Lidar point cloud generation for slam algorithm evaluation. Sensors **21**(10), 3313 (2021) 4

58. Song, Y., Ermon, S.: Generative modeling by estimating gradients of the data distribution. In: Advances in Neural Information Processing Systems. pp. 11895–11907 (2019) 2, 5, 6, 8
59. Song, Y., Ermon, S.: Improved techniques for training score-based generative models. In: Advances in Neural Information Processing Systems (NeurIPS) (2020) 8
60. Song, Y., Garg, S., Shi, J., Ermon, S.: Sliced score matching: A scalable approach to density and score estimation. arXiv preprint arXiv:1905.07088 (2019) 5, 6
61. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., Poole, B.: Score-based generative modeling through stochastic differential equations. In: 9th International Conference on Learning Representations (ICLR) (2021) 2
62. Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H., Kautz, J.: Splatnet: Sparse lattice networks for point cloud processing. In: CVPR (2018) 4
63. Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E.G.: Multi-view convolutional neural networks for 3d shape recognition. In: ICCV (2015) 4
64. Sun, Y., Wang, Y., Liu, Z., Siegel, J.E., Sarma, S.E.: Pointgrow: Autoregressively learned point cloud generation with self-attention. arXiv preprint arXiv:1810.05591 (2018) 3
65. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: ICCV (2019) 4
66. Valsesia, D., Fracastoro, G., Magli, E.: Learning localized generative models for 3d point clouds via graph convolution (2018) 3
67. Vincent, P.: A connection between score matching and denoising autoencoders. Neural computation **23**(7), 1661–1674 (2011) 2, 6
68. Wang, C., Samari, B., Siddiqi, K.: Local spectral graph convolution for point set feature learning. In: ECCV (2018) 4
69. Wang, S., Suo, S., Ma, W.C., Pokrovsky, A., Urtasun, R.: Deep parametric continuous convolutional neural networks. In: CVPR (2018) 4
70. Wang, T.H., Amini, A., Schwarting, W., Gilitschenski, I., Karaman, S., Rus, D.: Learning interactive driving policies via data-driven simulation. arXiv preprint arXiv:2111.12137 (2021) 4, 5
71. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. TOG (2019) 4
72. Welling, M., Teh, Y.W.: Bayesian learning via stochastic gradient langevin dynamics. In: Proceedings of the 28th international conference on machine learning (ICML-11). pp. 681–688 (2011) 2, 6
73. Wu, B., Wan, A., Yue, X., Keutzer, K.: Squeezeseg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud. CoRR **abs/1710.07368** (2017) 4
74. Wu, W., Qi, Z., Fuxin, L.: Pointconv: Deep convolutional networks on 3d point clouds. In: CVPR (2019) 4
75. Xiao, A., Huang, J., Guan, D., Zhan, F., Lu, S.: Synlidar: Learning from synthetic lidar sequential point cloud for semantic segmentation. arXiv preprint arXiv:2107.05399 (2021) 4
76. Xu, Y., Fan, T., Xu, M., Zeng, L., Qiao, Y.: Spidercnn: Deep learning on point sets with parameterized convolutional filters. In: ECCV (2018) 4
77. Yang, B., Luo, W., Urtasun, R.: Pixor: Real-time 3d object detection from point clouds. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7652–7660 (2018) 1, 4
78. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 4541–4550 (2019) 3

79. Yang, M., Dai, B., Dai, H., Schuurmans, D.: Energy-based processes for exchangeable data. In: International Conference on Machine Learning. pp. 10681–10692. PMLR (2020) 3, 6
80. Yu, L., Li, X., Fu, C.W., Cohen-Or, D., Heng, P.A.: Pu-net: Point cloud upsampling network. In: CVPR (2018) 13
81. Yuan, Y., Wang, J.: Ocnet: Object context network for scene parsing. arXiv:1809.00916 (2018) 4
82. Zamorski, M., Zieba, M., Nowak, R., Stokowiec, W., Trzcinski, T.: Adversarial autoencoders for generating 3d point clouds. arXiv preprint arXiv:1811.07605 **2** (2018) 3
83. Zamorski, M., Zieba, M., Nowak, R., Stokowiec, W., Trzciński, T.: Adversarial autoencoders for generating 3d point clouds. arXiv preprint arXiv:1811.07605 (2018) 3
84. Zhang, J., Singh, S.: Loam: Lidar odometry and mapping in real-time. In: Robotics: Science and Systems. vol. 2, pp. 1–9. Berkeley, CA (2014) 1
85. Zhao, H., Jiang, L., Fu, C.W., Jia, J.: PointWeb: Enhancing local neighborhood features for point cloud processing. In: CVPR (2019) 4

Learning to Generate Realistic LiDAR Point Clouds – Supplementary Material

Vlas Zyrianov, Xiyue Zhu, and Shenlong Wang

University of Illinois at Urbana-Champaign, IL, USA
{vlasz2,xiyuez2,shenlong}@illinois.edu

Abstract. In the supplementary material, we first provide more qualitative and quantitative results and additional ablation analysis in Sec. 1. In addition, we report experimental results on the challenging NuScenes dataset in Sec. 2. Finally, we provide additional quantitative and qualitative results for posterior sampling in Sec. 3. The supplementary video "*LiDARGen-intro.mp4*" briefly introduces our method, demonstrates the diffusion process in detail on KITTI-360, and compares qualitative results with other methods.

Keywords: 3D generation, self-driving, generative models

1 Additional Analysis

Qualitative Ablation Study We conduct ablation studies to justify the design choice of our algorithm. We compare the same score function model in three different settings: with circular convolution and without coordinate encoding, with circular convolution and without coordinate encoding, and our final model, which uses circular convolution and a coordinate encoding. Qualitative results are shown in Fig. 2.

Without **circular convolution**, a discontinuity appears in the point cloud representation horizontally, starting from the origin. This discontinuity is most clearly seen in the sixth row of the second column in Fig. 2. This discontinuity is caused by the left and right edges of the range image lacking a receptive field between each other when using normal convolutions. To address this issue, we use Circular Convolutions. Qualitatively, this discontinuity is fixed with this change.

With the help of **coordinate encoding**, our approach generates more straight road layouts that appropriately reflect the real-world layout distribution in the urban driving environment.

Comparison to Point-based Backbone We also compare our model against the point-based score-matching model proposed in ShapeGF [3]. The original ShapeGF model was trained and tested in ShapeNet. We adapt their model on LiDAR generation by changing the point cloud size to be 50k points and changing the noise level schedule by setting the number of noise steps to be 15 and end noise sigma to be 0.001. We train its stage 1 autoencoding and stage 2 GAN model from scratch on the KITTI training set until the validation loss

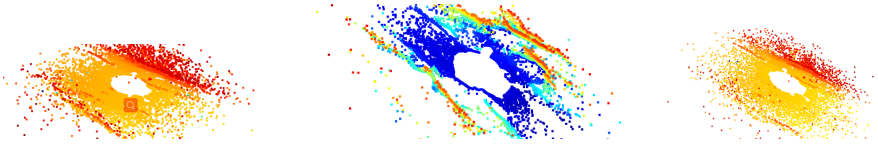


Fig. 1: Qualitative Results for ShapeGF [3] on KITTI.

converges. As shown in Fig. 1, ShapeGF cannot provide physically feasible results like equirectangular-based approaches. Further, despite working very well for ShapeNet-like objects, we find it has a strong mode collapse issue on complicated urban driving scenes.

2 nuScenes

Datasets The proposed LiDAR generation model is applicable across different datasets, geolocations and LiDAR sensors. To demonstrate this, we train and test our model on the **nuScenes dataset** [2]. nuScenes contains 297,737 LiDAR sweeps in the training set and 52,423 LiDAR sweeps in the testing and cross-validation set. The LiDAR sweeps were collected in the cities of Boston and Singapore. These locations present readings that are uniquely different compared to readings from the KITTI dataset [4], which have mostly been collected in the suburbs of Karlsruhe, Germany. In addition to the different environment, the LiDAR sensor used in nuScenes is different. The data was collected with a Velodyne HDL32E, which has 32 Beams and a $+10°$ to $-30°$ vertical Field of View. Compared to the sensor used in the KITTI dataset (Velodyne HDL-64E), the one used in nuScenes has lower vertical resolution, however has a higher vertical field of view.

Implementation Details We encode the raw nuScenes point cloud into an equirectangular view. Specifically, our range image resolution is set to be 32×1024, tailored for nuScenes LiDAR sensor's spatial resolution. And our Cartesian-to-range encoding is changed to the following: $\mathbf{z}_i = (\theta_i, \phi_i, d_i), r_i \colon \mathcal{I}(\lfloor \theta_i / s_\theta \rfloor, \lfloor \phi_j / s_\phi \rfloor) = \left(\frac{1}{6.5}\log_2(d_i + 1), \frac{1}{31} r_i\right)$, ensuring the full range to be normalized to $[0, 1]$,

Baselines Following the main paper's experiments on KITTI-360. We also leverage two baselines for comparison. The first is ProjectedGAN [5]. This was the second-best performing model in the KITTI evaluation, so we include it in this evaluation too. The second baseline is Caccia et al.'s LiDAR VAE [1]. All models were trained with the same settings as for KITTI. Note that the GAN model described in Caccia et al. [1] does not converge after our hyper-parameter tuning, hence we omit it from this study.

Experimental Results Fig. 3 depicts the qualitative comparison results. From this figure, we can see that our method still achieves superior results compared to both VAE [1] and projected GAN [5]. An AB test on a group of four human sub-

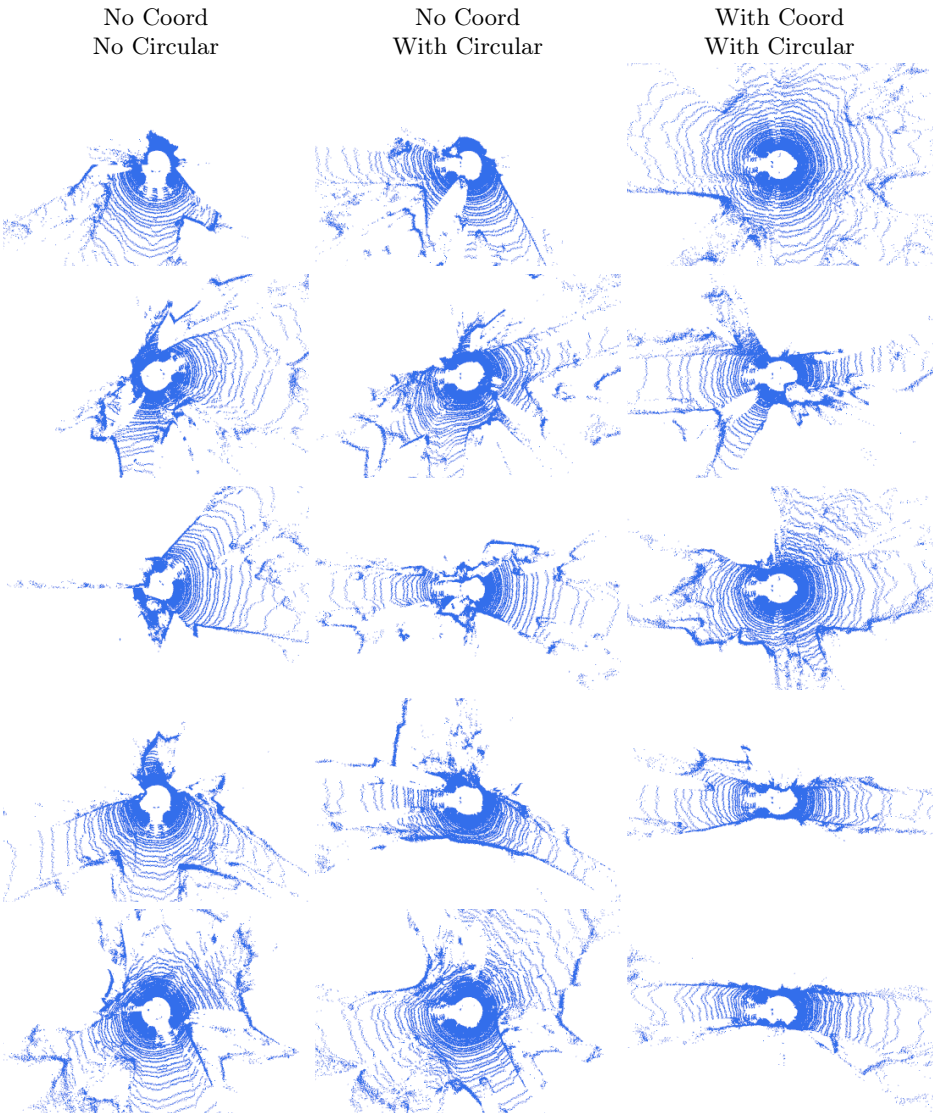


Fig. 2: Qualitative Ablation Comparison. Circular convolutions prevents a discontinuity that happens on a vector that starts at the origin and points left. The coordinate encoding encourages the network to generate straighter and more realistic roads.

jects suggests that our method is still significantly favored over other competing algorithms in 89% of cases.

While achieving superior human performance, we notice that our current nuScenes model has a noticeable weakness on the nuScenes dataset. In particular,

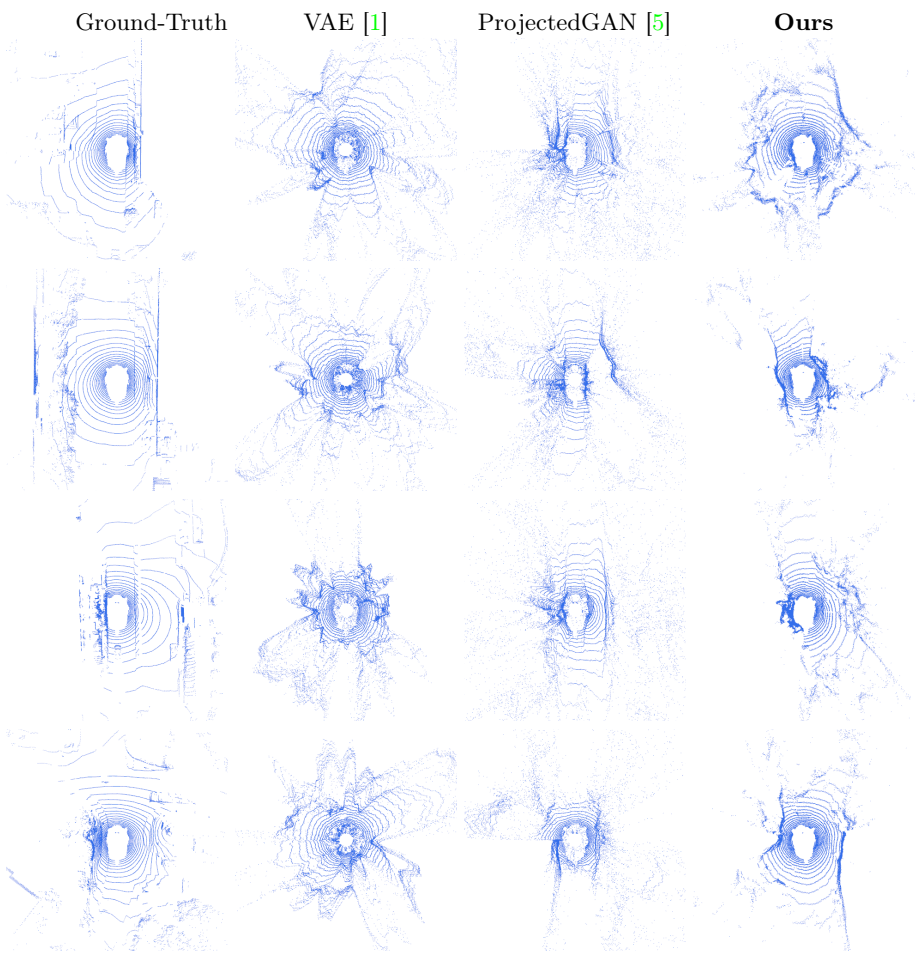


Fig. 3: Qualitative Results on the nuScenes dataset.

our method tends to generate point clouds that concentrate their mass closer to the viewpoint. As a result, despite superior visual quality, our MMD score at BEV is worse than VAE and Projected GAN (2e-3 vs. 1.1e-3 and 6e-5). In the nuScenes experiment, we directly adopt the same starting and ending noise level (150 and 0.01) as in KITTI, which might be too large for nuScenes. We believe better-tuned noise parameters will resolve this issue.

3 Additional Posterior Sampling Results

Densification We demonstrate additional densification results in Fig. 5 and Fig. 6. From the figure, we can see our produced diversified point clouds are both highly-realistic and have high fidelity and consistency to the input.

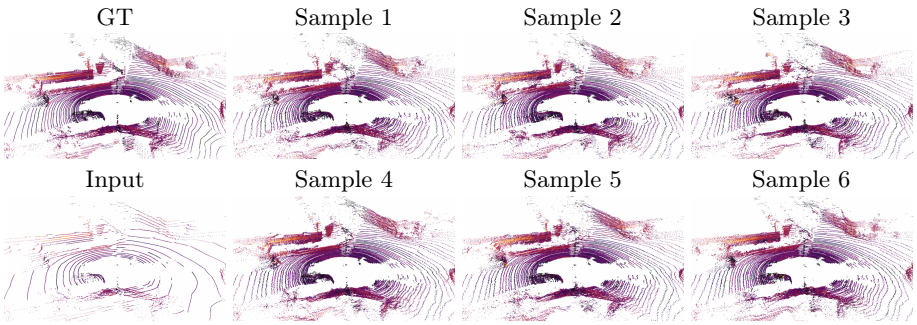


Fig. 4: Multiple samples of posterior sampling conditioned on the same sparse input. Notice the diversity of the shapes and intensity values of the car on the left, as well as the structure of the wall.

Fig 4 provides additional results demonstrating multiple samples given the same sparse point input. The figure shows that our posterior sampling approach produces multiple plausible resulting point clouds, further demonstrating the advantage of tackling such a task in a probabilistic fashion.

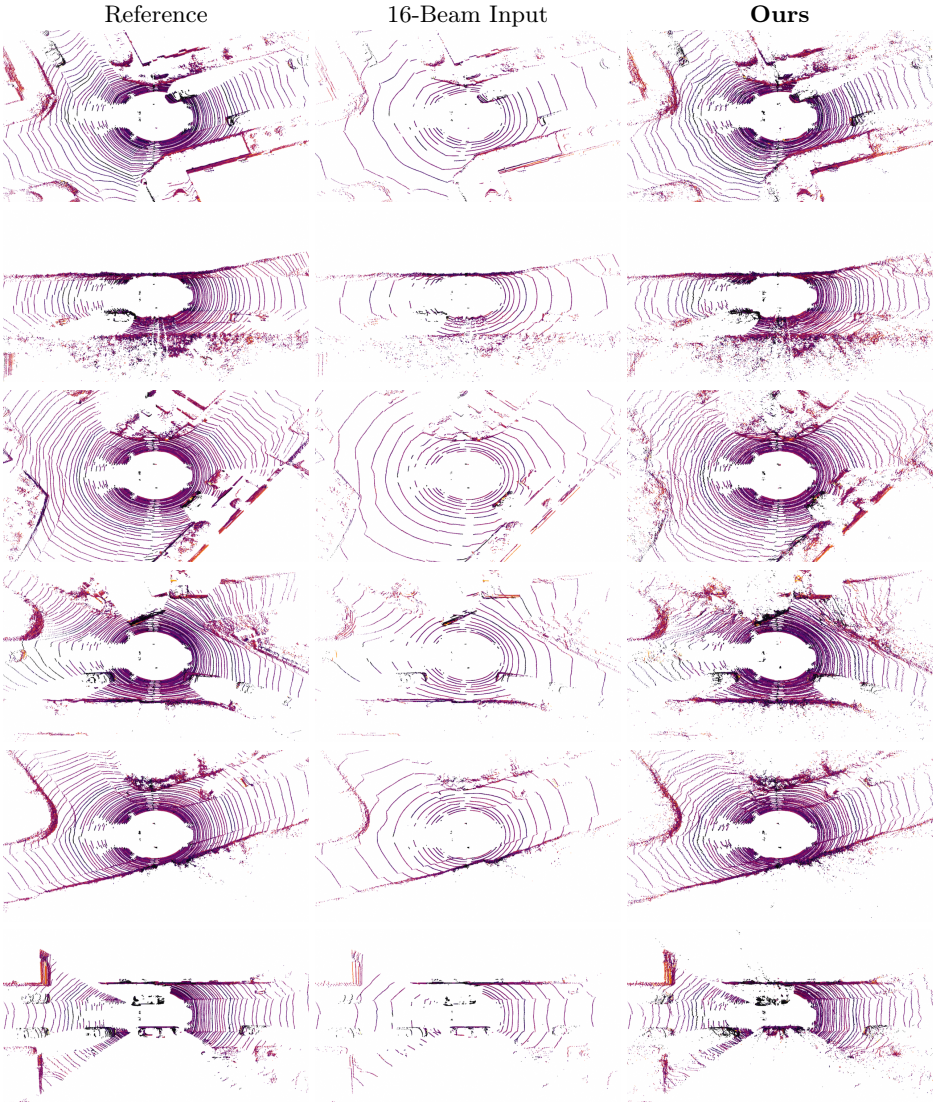


Fig. 5: Additional Results for Unsupervised LiDAR Densification.

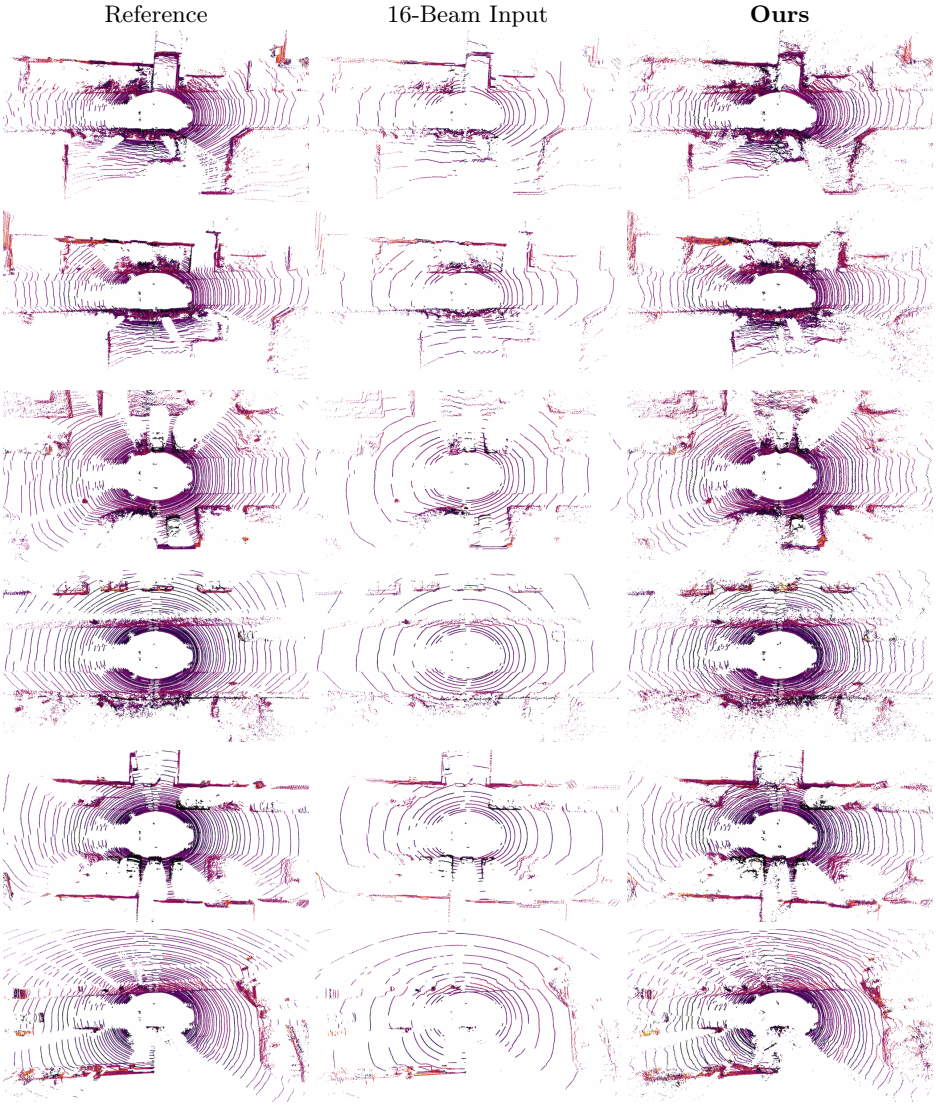


Fig. 6: Additional Results for Unsupervised LiDAR Densification (continued).

References

1. Caccia, L., van Hoof, H., Courville, A.C., Pineau, J.: Deep generative modeling of lidar data. IROS pp. 5034–5040 (2019) 2, 4
2. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. arXiv preprint arXiv:1903.11027 (2019) 2
3. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., Hariharan, B.: Learning gradient fields for shape generation. In: European Conference on Computer Vision. pp. 364–381. Springer (2020) 1, 2
4. Liao, Y., Xie, J., Geiger, A.: KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2d and 3d. arXiv preprint arXiv:2109.13410 (2021) 2
5. Sauer, A., Chitta, K., Müller, J., Geiger, A.: Projected gans converge faster. In: Advances in Neural Information Processing Systems (NeurIPS) (2021) 2, 4