A. Further Explanation of Concepts

A.1. Range Images and Point Clouds Conversion

In Section 3.1, we introduced range images as the modality for both input and output within LiDAR Diffusion Models (LiDMs). Subsequently, in Section 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:

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

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:

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

$$\text{pitch} = (1 - b) \times (\text{fov}_{up} - \text{fov}_{down}) + \text{fov}_{down}, \tag{3}$$

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 [1] and adopted by some recent works [20, 22].

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:

$$\text{JSD}\left(P_S \| P_R\right) = \frac{1}{2} D_{KL}\left(P_R \| M\right) + \frac{1}{2} D_{KL}\left(P_S \| M\right), \tag{4}$$

where $M = \frac{1}{2}(P_R + P_S)$ and D_{KL} is KL divergence [10]. 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:

$$\text{MMD}\left(P_S \| P_R\right) = \frac{1}{|P_R|} \sum_{Y \in P_R} \min_{X \in P_S} D_{CD}(X, Y). \tag{5}$$

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) [8] and Inception Score (IS) [17] 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:

$$a_{final} = \Theta_{\mathcal{D}}^{L}([x, \Theta_{\mathcal{E}}^{1}(x)]), \tag{6}$$

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,

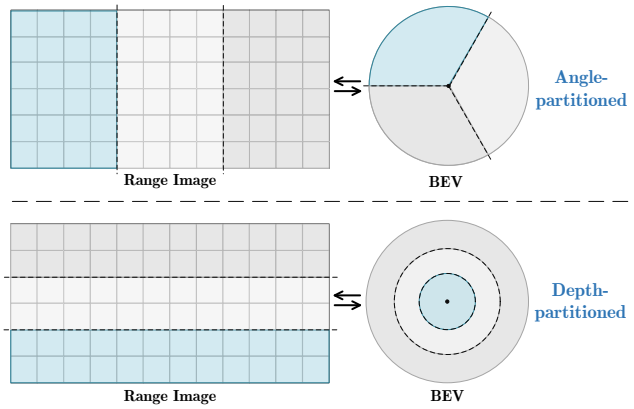


Figure 1. Example for two manners of partition-based aggregation on range images and bird's eye view (BEV). Angle-partitioned aggregation performs average on partitions of several columns on range images and of a sector on BEVs, while depth-partitioned aggregation performs average on partitions of several 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 Figure 1. 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 Figure 1 *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 Figure 1 *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++ [12] 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 [13] 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 [22].
- **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 [4], to extract features from the sparse volumes converted from range images. We utilize a public implementation with the pretrained weights[2] based on torchsparse [19]. 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 [18] 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.

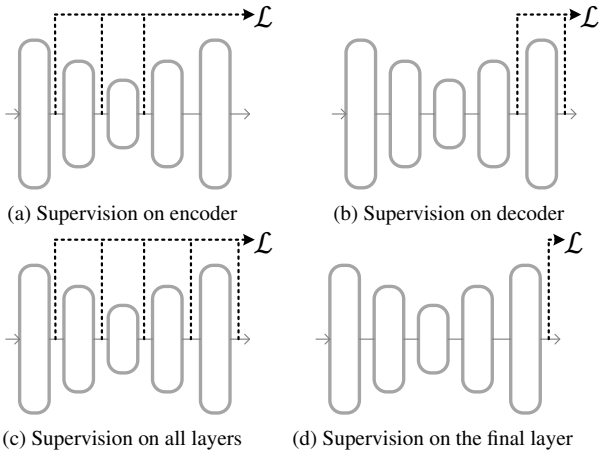
A.4. Details of Training

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 2. Different types of feature extraction for the computation of perceptual loss.

[7] introduced the output of different image processing stages in a pretrained VGG network [15] 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 [2, 9], neural style transfer [7], and image synthesis [3, 5]. Some recent popular generators [6, 14] are trained in a perceptual space based on LPIPS [21], 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 [3].

We utilize pretrained segmentation-based networks (*e.g.*, RangeNet++ [12]) 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 Figure 2.

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:

$$\mathcal{L}_{rec}(x) = \mathbb{E}_x[\|x - \hat{x}\| + \lambda_1 \|p - \hat{p}\|_2^2 + \lambda_2 \|m - \hat{m}\|_2^2], \tag{7}$$

$$\mathcal{L}_{\text{GAN}}(x) = \mathbb{E}_x[\log \mathcal{D}([x, p, m]) + \log(1 - \mathcal{D}([\hat{x}, \hat{p}, \hat{m}]))], \tag{8}$$

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 1, 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 [11] 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 [1, 20], we define CD and EMD as follows:

$$\text{CD}(P_x, P_{\hat{x}}) = \sum_{p \in P_x} \min_{\hat{p} \in P_{\hat{x}}} \|p - \hat{p}\|_2^2 + \sum_{\hat{p} \in P_{\hat{x}}} \min_{p \in P_x} \|p - \hat{p}\|_2^2, \tag{9}$$

$$\text{EMD}(P_x, P_{\hat{x}}) = \min_{\phi: P_x \to P_{\hat{x}}} \sum_{p \in P_x} \|p - \phi(p)\|_2, \tag{10}$$

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 [14], 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 1, 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 1. Performance of autoencoders in different downsampling factors f_c and f_p *after 40k training steps* on the KITTI-360 *val* [11]. 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 2. Performance of LiDMs with autoencoders in different downsampling factors f_c and f_p *after 10k training steps* on the KITTI-360 *val* [11]. 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 1. 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 2. We train each DM with 10k steps and adopt the same experimental setup in Sec. B.1. We apply

50 sampling DDIM [16] steps to each model and generate 5,000 samples for evaluation.

B.2.2 Analysis and Discussion

Through the experimental results in Table 2, 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 1, 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. 3, 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 [11], 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.

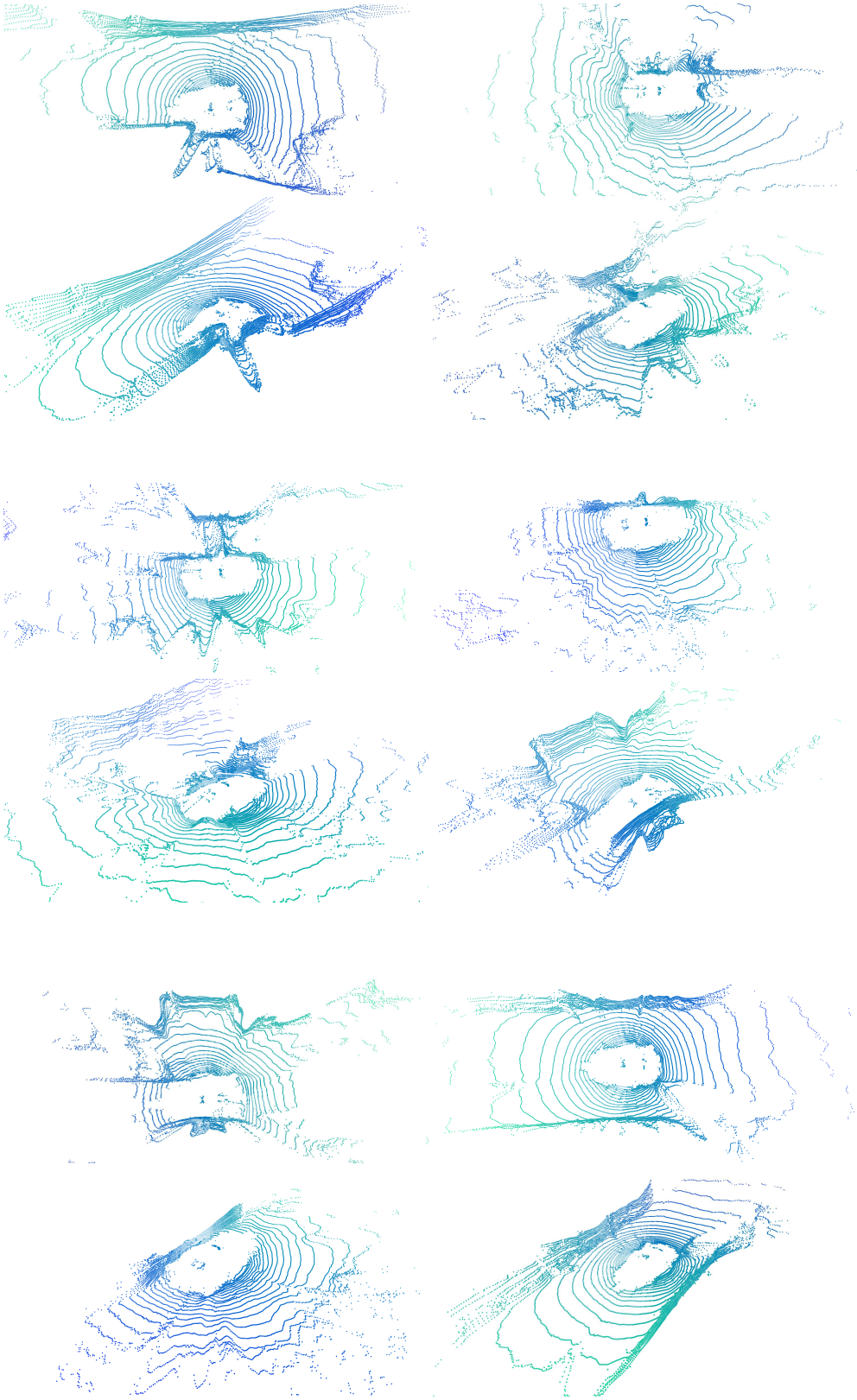


Figure 3. Unconditional samples on 32-beam scenario.

References

[1] 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. 1, 3

[2] Joan Bruna, Pablo Sprechmann, and Yann LeCun. Super-resolution with deep convolutional sufficient statistics. *arXiv preprint arXiv:1511.05666*, 2015. 3

[3] 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. 3

[4] 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. 2

[5] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. *Advances in neural information processing systems*, 29, 2016. 3

[6] 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. 3

[7] 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. 3

[8] 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. 1

[9] 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. 3

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

[11] 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. 3, 4, 5

[12] 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, 3

[13] Joseph Redmon. Darknet: Open source neural networks in c, 2013. 2

[14] 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. 3

[15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

[16] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 5

[17] 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. 1

[18] 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. 2

[19] 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. 2

[20] 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. 1, 3

[21] 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. 3

[22] 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. 1, 2