Adaptive Piecewise Distillation for Efficient LiDAR Data Generation

Ruibo Li, Xiaofeng Yang, Ze Yang, Jiacheng Wei, Chunyan Miao, Guosheng Lin*

Nanyang Technological University, Singapore
ruibo001@e.ntu.edu.sg, gslin@ntu.edu.sg

Abstract

LiDAR data generation has emerged as a promising solution to the high cost and limited scalability of real-world LiDAR sensing. Recent diffusion and rectified flow models have demonstrated strong capabilities in synthesizing realistic 3D point clouds; however, their iterative sampling procedures result in significant inference overhead. To address this, we focus on efficient few-step LiDAR generation for both unconditional and multi-modal conditional settings. Specifically, we propose an adaptive piecewise distillation strategy tailored for rectified flow-based LiDAR generation models, where the teacher model's flow trajectory is adaptively segmented into consecutive intervals, and the student is trained only at the start of each interval to directly predict the velocity toward its endpoint. By sequentially sampling at the start timestep of each interval, our method enables fast few-step generation. Moreover, instead of uniform partitioning, we introduce an adaptive timestep selection strategy that chooses interval boundaries with minimal initial error, thereby reducing the complexity of distillation. Experimental results show that our method achieves comparable or superior performance to state-of-the-art methods in both unconditional and multi-modal conditional LiDAR generation, using only four sampling steps.

Introduction

Light Detection and Ranging (LiDAR) technology, which generates precise 3D point clouds of the environment, has become a cornerstone of 3D perception. It supports a wide range of applications in autonomous driving (Geiger, Lenz, and Urtasun 2012; Sun et al. 2020; Liao, Xie, and Geiger 2022; Cui et al. 2024) and robotics (Shan et al. 2020; Xu et al. 2022). However, due to the high cost of LiDAR sensors, collecting high-quality point clouds remains expensive and difficult to scale. These challenges have sparked increasing interest in LiDAR data generation, which aims to synthesize realistic and semantically consistent 3D point clouds.

Recently, diffusion models have emerged as a powerful generative framework for LiDAR synthesis. By representing LiDAR scans as range images, methods such as LiDARGen (Zyrianov, Zhu, and Wang 2022), R2DM (Nakashima and Kurazume 2024), and Text2LiDAR (Wu et al. 2024) leverage diffusion models to generate LiDAR data. However, their iterative sampling procedures are highly time-consuming, limiting their practicality. To alleviate this, inspired by latent diffusion models (Rombach et al. 2022), some methods (Ran, Guizilini, and Wang 2024; Hu, Zhang, and Hu 2024) compress range images into a latent space and train diffusion models in this compact domain to accelerate sampling. While this reduces inference costs, generating point clouds still requires tens to hundreds of steps.

Rectified Flow (Liu, Gong, and Liu 2023; Liu et al. 2023) has recently been proposed as an alternative to denoising diffusion models, offering nearly linear generation trajectories. Building on this idea, R2Flow (Nakashima et al. 2025) adapts rectified flow for unconditional LiDAR generation. R2Flow improves sampling efficiency but struggles with generation quality and lacks support for conditional tasks. To address this limitation, our work investigates how to achieve fast and flexible LiDAR synthesis across both unconditional and multi-modal conditional scenarios, delivering high-quality results within just a few sampling steps.

In few-step generation, the sampling process involves only a small subset of timesteps, allowing distillation to focus on these specific steps rather than the entire trajectory, thereby reducing learning complexity. To this end, we propose a novel **adaptive piecewise distillation** method, which adaptively segments the entire time range into a sequence of consecutive intervals and trains a student network solely at the start of each interval to predict the velocity that maps it to the corresponding endpoint. The starting point is generated by adding random noise to clean data, while the endpoint is solved by a pretrained teacher network. Furthermore, instead of using uniform segmentation, we introduce an adaptive timestep selection strategy that determines interval boundaries by minimizing the initial prediction error, thereby easing the overall distillation process. As illustrated in Figure 1, our method enables student networks to generate high-quality LiDAR data with only 4 sampling steps, under both unconditional and multi-modal conditional settings.

The main contributions are listed as follows:

- We propose an efficient rectified flow-based framework, **LiDAR-Flow**, capable of generating high-quality LiDAR data in a few steps, supporting various tasks such as unconditional LiDAR generation, image-to-LiDAR syn-

*Corresponding Author

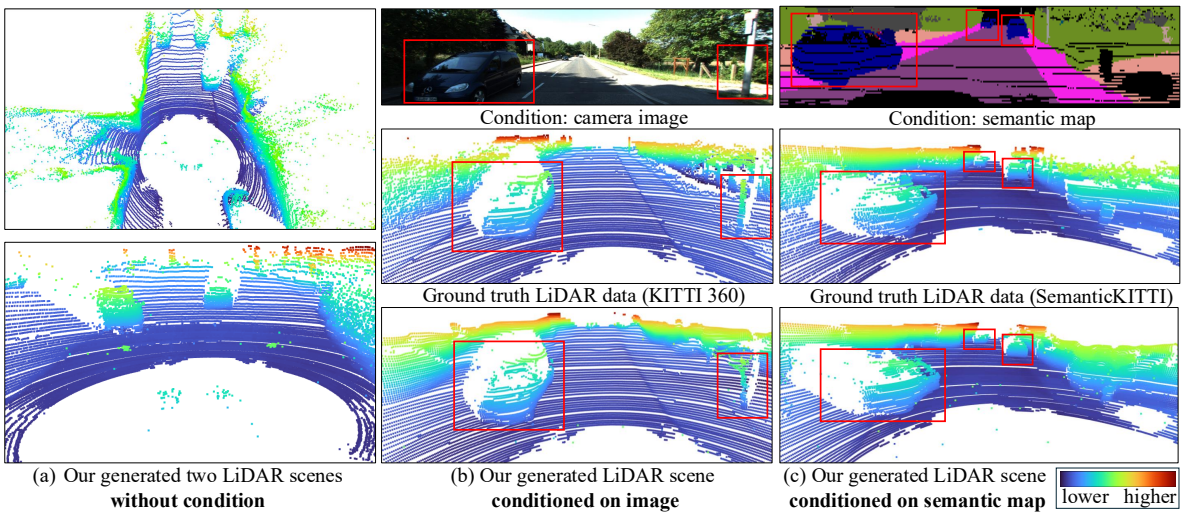

Figure 1: LiDAR scenes generated by our method under (a) unconditional, (b) image-conditioned, and (c) semantic-conditioned settings, with only 4 sampling steps. Our generated LiDAR scenes show reasonable scene layouts and plausible object structures. Color indicates the height of each 3D point (blue = lower, red = higher).

thesis, and semantic map-to-LiDAR synthesis.

- To reduce the number of sampling steps, we propose a novel distillation method for rectified flow-based LiDAR generation models, in which the time range is adaptively segmented into consecutive intervals, and the student model is supervised to learn interval-wise velocities determined by the teacher model.
- With only four sampling steps, our proposed **LiDAR-Flow** achieves comparable or superior performance to existing state-of-the-art methods in both unconditional and multi-modal conditional LiDAR data generation.

Related Work

LiDAR point cloud generation. 3D point cloud generation has received growing attention due to its importance in 3D perception and simulation. While numerous methods (Achlioptas et al. 2018; Yang et al. 2019; Kim et al. 2020; Luo and Hu 2021; Zhou, Du, and Wu 2021; Vahdat et al. 2022; Mittal et al. 2022; Wu et al. 2023) focus on object-level generation, our work targets scene-level LiDAR generation to model scene geometry and sensor-specific patterns in real-world environments.

Unlike simulation-based LiDAR generation methods (Dosovitskiy et al. 2017; Manivasagam et al. 2020; Zyrianov et al. 2024), which rely on manually designed assets, asset-free generative models learn real LiDAR data distributions directly from observations, offering a promising alternative for realistic and diverse LiDAR data. By representing LiDAR scans as range images, some approaches (Caccia et al. 2019; Sauer et al. 2021; Xiong et al. 2023) utilize Variational Autoencoders or Generative Adversarial Networks for LiDAR data generation. With the rise of diffusion models, LiDARGen (Zyrianov, Zhu, and Wang 2022) learns a score-based diffusion model to synthesize point clouds, while methods such as R2DM (Nakashima and Kurazume 2024), Text2LiDAR (Wu et al. 2024), and LiDPM (Martyniuk et al. 2025) adopt denoising diffusion probabilistic models (Ho, Jain, and Abbeel 2020). However, diffusion-based methods usually have high inference latency due to iterative sampling. To reduce computation, following latent diffusion models (Rombach et al. 2022), recent methods (Ran, Guizilini, and Wang 2024; Hu, Zhang, and Hu 2024) train diffusion models in a compact latent space derived from range images. Despite improved efficiency, generation still requires tens to hundreds of iterative steps.

Recently, R2Flow (Nakashima et al. 2025) adapts rectified flow for unconditional LiDAR generation, achieving few-step or even one-step sampling. However, it suffers from suboptimal generation quality and is limited to the unconditional setting. To this end, we focus on efficient and high-quality LiDAR generation under both unconditional and multi-modal conditional settings, aiming to maintain strong generative performance with only a few sampling steps.

Efficient diffusion/rectified flow models. Although diffusion models achieve impressive results, slow inference remains a major limitation. To overcome this, some approaches (Song, Meng, and Ermon 2020; Lu et al. 2022) use efficient solvers, while others (Salimans and Ho 2022; Song et al. 2023; Kim et al. 2023) distill pre-trained models into faster variants supporting one- or few-step sampling.

As an advanced diffusion model, RFlow (Liu, Gong, and Liu 2023; Liu et al. 2023) offers an alternative paradigm by reformulating the generative process as a learned deterministic flow. Through iterative distillation, it learns to straighten the entire generative trajectory, enabling fast sampling. Instead of directly straightening the complete trajectory, PeRFlow (Yan et al. 2024) and ProReFlow (Ke et al. 2025) divide it into temporal windows and straighten the trajectory in a piecewise manner. In contrast, we observe that few-step generation only utilizes a limited number of

timesteps during inference. This insight allows us to focus distillation specifically on those selected timesteps, rather than modeling the entire trajectory. Building on this observation, we propose an **adaptive piecewise distillation** method for LiDAR generation, which adaptively segments the full time range into a sequence of intervals and trains a student network at the beginning of each interval to directly predict the velocity toward its endpoint. This design simplifies training and improves performance at the timesteps that are actually used for few-step inference. Also, rather than uniform segmentation, we identify interval boundaries by minimizing the initial prediction error, thereby reducing the difficulty of distillation. A related approach, BOSS (Nguyen, Nguyen, and Nguyen 2023), finds the optimal timestep of a flow-based teacher model to sample a synthetic dataset, which is then used to train the student network to mimic the teacher's sampling trajectory for distillation. In contrast, our method performs distillation directly along the teacher's flow trajectories, eliminating the need for synthetic data and mitigating distribution shift. Additionally, unlike BOSS, which defines timestep selection based on the squared ℓ_2 loss, our method is more general and supports arbitrary training metrics.

Preliminaries

Rectified flow. Given a data distribution π_0 and a normal distribution π_1, rectified flow (Liu, Gong, and Liu 2023; Liu et al. 2023) transforms noise $\boldsymbol{y}_1 \sim \pi_1$ into a data sample $\boldsymbol{y}_0 \sim \pi_0$ by solving an ordinary differential equation (ODE): $\mathrm{d}\boldsymbol{y}_t = v_\theta\left(\boldsymbol{y}_t, t\right)\mathrm{d}t$. The intermediate state $\boldsymbol{y}_t$ follows a linear interpolation between data sample $\boldsymbol{y}_0$ and random noise $\boldsymbol{y}_0$, given by $\boldsymbol{y}_t = (1-t)\boldsymbol{y}_0 + t\boldsymbol{y}_1$. The velocity field $v_\theta(\cdot)$, parameterized by a neural network θ, is learned by minimizing the flow matching loss (Lipman et al. 2022):

$$\min_\theta \mathbb{E}_{\boldsymbol{y}_1\sim\pi_1,\boldsymbol{y}_0\sim\pi_0}\left[\int_1^0 \left\|(\boldsymbol{y}_1-\boldsymbol{y}_0)-v_\theta(\boldsymbol{y}_t,t)\right\|^2\mathrm{d}t\right]. \quad (1)$$

Once the velocity network v_θ is trained, new samples can be generated by integrating the ODE from $t=1$ to $t=0$: $\hat{\boldsymbol{y}}_0 = \mathsf{ODE}[v_\theta](\boldsymbol{y}_1, 1, 0) = \boldsymbol{y}_1 + \int_1^0 v_\theta(\boldsymbol{y}_t, t)\mathrm{d}t$.

To further improve sampling efficiency, rectified flow introduces a distillation mechanism, **reflow**. For each noise sample $\boldsymbol{y}_1$, it replaces the original data sample $\boldsymbol{y}_0$ with the generated sample $\hat{\boldsymbol{y}}_0$ in Eq. (1) and retrains the model using the synthetic pair $(\boldsymbol{y}_1, \hat{\boldsymbol{y}}_0)$. This operation straightens the trajectories and enables faster, more efficient generation.

Piecewise rectified flow. Unlike rectified flow, which treats the entire trajectory as a global linear interpolation, piecewise rectified flow (Yan et al. 2024) splits the path into a series of time windows, $\{[t_n, t_{n-1})\}_{n=1}^N$ and straightens it within each window to better approximate the sampling dynamics of a pretrained flow model.

For a pretrained velocity field v_θ, the starting state in each time window $[t_n, t_{n-1})$ is given by $\boldsymbol{y}_{t_n} = (1-t_n)\boldsymbol{y}_0 + t_n\boldsymbol{y}_1$, while the end state is obtained by integrating the ODE within this window, $\boldsymbol{y}_{t_{n-1}} = \mathsf{ODE}[v_\theta](\boldsymbol{y}_{t_n}, t_n, t_{n-1}) = \boldsymbol{y}_{t_n} + \int_{t_n}^{t_{n-1}} v_\theta(\boldsymbol{y}_t, t)\mathrm{d}t$. Accordingly, the piecewise rectified flow model is optimized by learning a velocity field that best reconstructs the linear transition between $\boldsymbol{y}_{t_n}$ and $\boldsymbol{y}_{t_{n-1}}$ across all time windows.

LiDAR-Flow

In this section, we introduce **LiDAR-Flow**, our few-step flow-based LiDAR generative framework, and explain its key components. We first describe how to formulate LiDAR generation in the range image space and present two types of rectified flow models: a latent-based velocity network and a pixel-based velocity network. Our framework supports various tasks, including unconditional, camera-to-LiDAR, and semantic-map-to-LiDAR generation. We then detail our **adaptive piecewise distillation**, which distills the rectified flow models to enable efficient few-step LiDAR generation.

LiDAR generation with rectified flow

Representing LiDAR data by range images. A 3D point (x, y, z) from LiDAR data can be mapped onto a 2D plane using spherical projection, generating a **2D range image**. In this representation, each pixel (u, v) stores the depth (range) $r = \sqrt{x^2+y^2+z^2}$, which measures the distance from the LiDAR sensor to the point. The pixel coordinates (u, v) are derived from the azimuth angle $\theta = \arctan 2(y, x)$ and the elevation angle $\phi = \arcsin\left(\frac{z}{r}\right)$. Conversely, a 2D range image can be converted back to 3D data by inverting the spherical projection process. Therefore, this work focuses on LiDAR scene generation in the form of range images.

Architecture of rectified flow models. We employ velocity networks v_θ to generate LiDAR scenes by learning a mapping from a noise distribution to the LiDAR distribution. In our work, we explore two different architectures: latent-based velocity networks and pixel-based velocity networks.

For **latent-based** velocity networks, we follow prior work (Ran, Guizilini, and Wang 2024; Hu, Zhang, and Hu 2024) to compress range images into a latent space using a variational auto-encoder (VAE), and subsequently train a velocity network v_θ within this latent space. Specifically, we adopt the VAE model in LiDM (Ran, Guizilini, and Wang 2024) to compress range images, and construct the velocity network v_θ using the U-Net structure employed in LDM (Rombach et al. 2022) and LiDM (Ran, Guizilini, and Wang 2024). Compared to the U-Net backbone in LiDM, our velocity network v_θ integrates positional embeddings into latent features, providing explicit spatial information.

For **pixel-based** velocity networks, we employ the R2Flow model introduced in (Nakashima et al. 2025) as our velocity network v_θ.

Training. Given a range image, we compress it into a latent feature $\boldsymbol{y}_0$ using a VAE for the latent-based velocity network, while for the pixel-based velocity network, we directly use the range image as the feature $\boldsymbol{y}_0$. The flow matching loss (Eq. (1)) for unconditional generation is:

$$\min_\theta \mathbb{E}_{\boldsymbol{y}_0,\boldsymbol{y}_1\sim\mathcal{N}(0,1),t}\left[\left\|(\boldsymbol{y}_1-\boldsymbol{y}_0)-v_\theta(\boldsymbol{y}_t,t)\right\|^2\right], \quad (2)$$

where the time t is sampled from 1 to 0, $\boldsymbol{y}_1$ follows a standard normal distribution, and the model input is given by $\boldsymbol{y}_t = (1-t)\boldsymbol{y}_0 + t\boldsymbol{y}_1$.

For conditional generation, given a conditioning input $\boldsymbol{c}$ paired with the latent feature $\boldsymbol{y}_0$, the flow matching loss is:

$$\min_\theta \mathbb{E}_{\boldsymbol{y}_0,\boldsymbol{c},\boldsymbol{y}_1\sim\mathcal{N}(0,1),t}\left[\left\|(\boldsymbol{y}_1-\boldsymbol{y}_0)-v_\theta(\boldsymbol{y}_t,\boldsymbol{c},t)\right\|^2\right]. \quad (3)$$

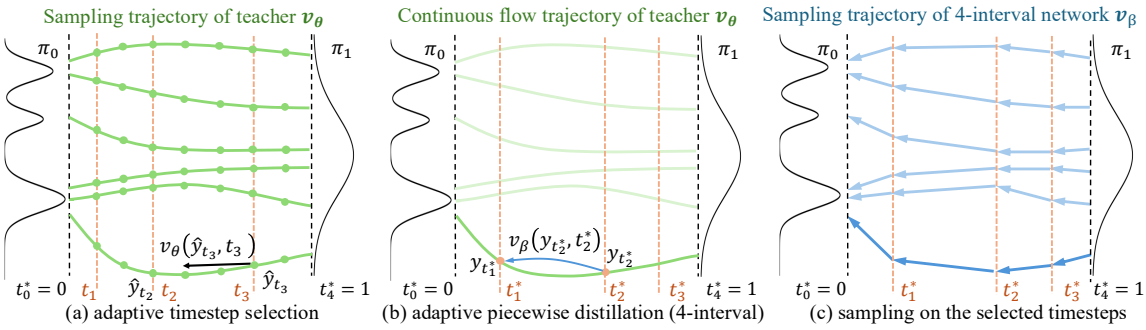

Figure 2: Illustration of our adaptive piecewise distillation. (a) Adaptive timestep selection: For a given teacher network v_θ, we identify a set of optimal timesteps $\{t_1^*, t_2^*, t_3^*\}$, such that the predicted velocities best approximate the piecewise evolution of the trajectory sampled from the teacher network. (b) Adaptive piecewise distillation (4-interval): Using the selected timesteps, a student network v_β is trained to approximate the piecewise velocity of the teacher trajectory over each interval. (C) Few-step sampling: The student network directly applies the selected timesteps as sampling points for few-step generation.

Sampling. After training v_θ, we generate a new sample by solving the ODE from $t = 1$ to 0 using the Euler method:

$$\hat{\boldsymbol{y}}_{t_{s-1}} = \hat{\boldsymbol{y}}_{t_s} + (t_{s-1} - t_s) v_\theta(\hat{\boldsymbol{y}}_{t_s}, t_s), \tag{4}$$

where $1 = t_S > \cdots > t_1 > t_0 = 0$ and S denotes the number of inference steps.

Adaptive piecewise distillation

Although the velocity networks v_θ exhibit nearly straight ODE trajectories, they still require tens of inference steps to produce realistic samples.

In few-step generation, only a limited number of timesteps are used during inference, allowing us to focus on distilling information at these specific timesteps. To further accelerate sampling, we propose a novel **adaptive piecewise distillation** method that adaptively partitions the ODE trajectory of v_θ into N consecutive time intervals $\{[t_i^*, t_{i-1}^*)\}_{i=1}^N$. We then train a student velocity network v_β to directly estimate the velocity from the start to the end of each interval. Unlike PeRFlow (Yan et al. 2024), which straightens the full trajectory within each interval, our method focuses only on these selected timesteps, simplifying the training of the student model and improving performance at the chosen times. Furthermore, instead of uniformly partitioning the ODE trajectory, we apply an **adaptive timestep selection** strategy that determines interval boundaries by minimizing the initial prediction error, thereby further reducing the difficulty of distillation. The overview of our method is presented in Fig. 2.

Adaptive timestep selection. To reduce the difficulty of distillation, we adaptively select a set of suitable timesteps $\{t_i^*\}_{i=0}^N$ for each teacher velocity network v_θ.

Specifically, we uniformly divide the time range $[1, 0]$ into M steps using a small constant step size $\frac{1}{M}$, resulting in a set of candidate timesteps: $\{1, \frac{M-1}{M}, \frac{M-2}{M}, ..., \frac{1}{M}, 0\}$. We then solve the ODE defined by the teacher network v_θ from $t = 1$ to 0 over these candidate timesteps using the Euler method (Eq. (4)). The intermediate state at any timestep t_i is given by: $\hat{\boldsymbol{y}}_{t_i} = \mathsf{ODE}[v_\theta](\hat{\boldsymbol{y}}_1, 1, t_i)$, where $\hat{y}_1 \sim \mathcal{N}(0, 1)$ denotes the initial noisy input.

For this ODE, we aim to identify a sequence of timesteps $\{t_i\}_{i=0}^N$ at which the predicted velocity of the teacher model v_θ at each starting point, $v_\theta(\hat{\boldsymbol{y}}_{t_i}, t_i)$, closely approximates the true dynamics from $\hat{\boldsymbol{y}}_{t_i}$ to $\hat{\boldsymbol{y}}_{t_{i-1}}$. Therefore, the objective of timestep selection for this ODE can be formulated as:

$$\begin{aligned} \min_{\{t_i\}_{i=1}^{N-1}} \quad & \sum_{i=1}^{N} m\left(\hat{\boldsymbol{y}}_{t_{i-1}} - \hat{\boldsymbol{y}}_{t_i},\ (t_{i-1} - t_i) \cdot v_\theta(\hat{\boldsymbol{y}}_{t_i}, t_i)\right) \\ \text{s.t.} \quad & 0 = t_0 < t_1 < \cdots < t_{N-1} < t_N = 1, \end{aligned} \tag{5}$$

where $m(\cdot, \cdot)$ is a training metric, typically the squared ℓ_2 distance, and N is the number of intervals. Here, we split the time into 4 intervals, i.e., $N = 4$. We solve this timestep selection problem efficiently using dynamic programming. More details can be found in the supplementary material.

Although Eq. (5) defines the objective for a single ODE trajectory, our goal is to find a set of timesteps that are shared across multiple trajectories of the teacher network v_θ. Therefore, in practice, we generate a batch of ODE trajectories by repeatedly sampling initial state $\hat{\boldsymbol{y}}_1 \sim \mathcal{N}(0, 1)$ and solving the ODE defined by the teacher model v_θ (Eq. (4)). We then optimize a single set of shared timesteps $\{t_i^*\}_{i=0}^N$ that minimizes the total loss aggregated over all sampled trajectories.

Fig. 3 presents an example of the distributions of selected timesteps $\{t_1, t_2, t_3\}$ for individual ODE trajectories, sampled by the **pixel-based** network, as well as the optimal timesteps $\{t_1^*, t_2^*, t_3^*\}$ shared across all sampled trajectories. For this network, we use the pseudo-Huber loss as the training metric $m(\cdot, \cdot)$. As shown in this figure, the selected timesteps for different trajectories tend to concentrate within narrow regions, indicating a consistent preference across samples. Moreover, the optimal timesteps do not necessarily align with uniformly spaced ones, highlighting the benefit of adaptive selection over uniform discretization.

Adaptive piecewise distillation. After determining the optimal timesteps $\{t_1^*, t_2^*, t_3^*\}$ for the teacher network v_θ, we train a student velocity network v_β to directly estimate the velocity over each interval. For a given interval $\{[t_i^*, t_{i-1}^*)\}_{i=1}^N$, the starting point is computed via linear interpolation: $\boldsymbol{y}_{t_i^*} = (1 - t_i^*)\boldsymbol{y}_0 + t_i^* \boldsymbol{y}_1$. The corresponding endpoint $\boldsymbol{y}_{t_{i-1}^*}$ is obtained by solving the ODE defined by

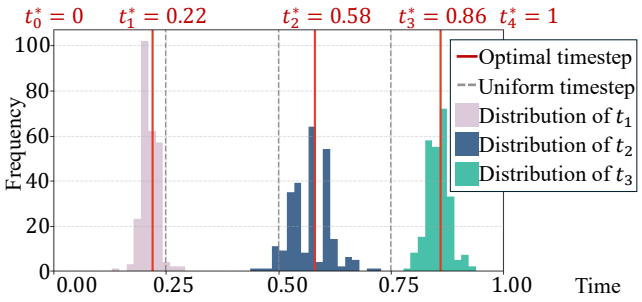

Figure 3: Histogram of selected timesteps $\{t_1, t_2, t_3\}$ across individual ODE trajectories sampled by the pixel-based network, along with the optimal timesteps $\{t_1^*, t_2^*, t_3^*\} = \{0.22, 0.58, 0.86\}$ shared by all sampled trajectories. The selected timesteps for different trajectories cluster within narrow regions, and the optimal timesteps do not align with uniformly spaced values (i.e., $\{0.25, 0.50, 0.75\}$), highlighting the benefit of adaptive timestep selection.

Algorithm 1: Adaptive piecewise distillation.

Input: Dataset $\mathcal{D}$; number of intervals N; Teacher $v_\theta(\cdot)$;
Output: Student network $v_\beta^*(\cdot)$;
Procedure:

Generate optimal timesteps $\{t_i^*\}_{i=0}^N$ for teacher network v_θ by adaptive timestep selection;
Initialize student model, $v_\beta(\cdot) \leftarrow v_\theta(\cdot)$;
while not converged **do**
  Sample latent feature $\boldsymbol{y}_0 \sim \mathcal{D}$, noise $\boldsymbol{y}_1 \sim \mathcal{N}(0, 1)$;
  Randomly choose $i \in \{1, \ldots, N\}$, construct the intermediate noisy state at the starting point of the i-th interval: $\boldsymbol{y}_{t_i^*} \leftarrow (1 - t_i^*)\boldsymbol{y}_0 + t_i^* \boldsymbol{y}_1$;
  Compute the state of the endpoint via Euler method in Eq. (4): $\boldsymbol{y}_{t_{i-1}^*} \leftarrow \mathsf{ODE}[v_\theta](\boldsymbol{y}_{t_i^*}, t_i^*, t_{i-1}^*)$;
  Compute loss: $L \leftarrow m(\frac{\boldsymbol{y}_{t_{i-1}^*} - \boldsymbol{y}_{t_i^*}}{t_{i-1}^* - t_i^*}, v_\beta(\boldsymbol{y}_{t_i^*}, t_i^*))$;
  Update student network $v_\beta(\cdot)$ using $\nabla_\beta L$;
end while
Return final student network, $v_\beta^*(\cdot) \leftarrow v_\beta(\cdot)$.

the teacher network: $\mathsf{ODE}[v_\theta](\boldsymbol{y}_{t_i^*}, t_i^*, t_{i-1}^*)$. The objective of our adaptive piecewise distillation is then formulated as:

$$\min_{\beta} \quad \sum_{n=1}^{N} \mathbb{E}_{\boldsymbol{y}_{t_i^*} \sim \pi_n} m\left(\frac{\boldsymbol{y}_{t_{i-1}^*} - \boldsymbol{y}_{t_i^*}}{t_{i-1}^* - t_i^*}, v_\beta(\boldsymbol{y}_{t_i^*}, t_i^*)\right) \tag{6}$$
$$\text{s.t.} \quad \boldsymbol{y}_{t_{i-1}^*} = \mathsf{ODE}[v_\theta](\boldsymbol{y}_{t_i^*}, t_i^*, t_{i-1}^*),$$

where $m(\cdot, \cdot)$ denotes the training metric, and ODE is approximated using the Euler method in Eq. (4). The distillation procedure is outlined in Algorithm 1. Once training is completed, we obtain the distilled student model $v_\beta^*(\cdot)$, which enables few-step generation by directly applying the selected timesteps $\{t_i^*\}_{i=1}^N$ as sampling points.

Experiment

We evaluate our LiDAR generative framework, **LiDAR-Flow**, by comparing it with state-of-the-art methods across three tasks: unconditional LiDAR generation, camera-to-LiDAR generation, and semantic-map-to-LiDAR generation. Subsequently, we conduct ablation studies to assess the effectiveness of our **adaptive piecewise distillation**.

Datasets. We conduct experiments on KITTI-360 (Liao, Xie, and Geiger 2022) for both unconditional and camera-to-LiDAR generation. For fair comparison with LiDM (Ran, Guizilini, and Wang 2024) and R2Flow (Nakashima et al. 2025), we follow their respective data splits: LiDM uses 8 training sequences and 1 test sequence; R2Flow uses the first two sequences for testing and the rest for training. For semantic-map-to-LiDAR generation, we use SemanticKITTI (Behley et al. 2019) with the same dataset partitioning as LiDM. Both datasets contain point clouds captured by 64-beam LiDAR sensors.

Implementation Details. We apply our distillation method to both pixel-based and latent-based models. Specifically, we randomly generate 256 samples and uniformly divide the time into 100 steps for adaptive timestep selection.

For the pixel-based model, we directly adopt the 1-RF model from R2Flow (Nakashima et al. 2025) as the teacher. During distillation, the time range is divided into 4 intervals with intermediate timesteps $\{0.22, 0.58, 0.86\}$, and the ODE in Eq.(6) is solved using a 20-step Euler method. To maintain consistency with R2Flow, we employ the pseudo-Huber loss as the training metric $m(\cdot, \cdot)$ in Eq.(6).

For the latent-based model, we use the same pretrained VAE from LiDM (Ran, Guizilini, and Wang 2024) and follow its U-Net backbone to design our velocity model, ensuring a fair comparison. Compared to the U-Net in LiDM, our model integrates positional embeddings into the latent features. During distillation, we use different sets of intermediate timesteps for each task, which are determined via adaptive timestep selection: $\{0.28, 0.57, 0.82\}$ for unconditional generation, $\{0.22, 0.48, 0.74\}$ for camera-to-LiDAR, and $\{0.08, 0.20, 0.42\}$ for semantic-map-to-LiDAR. In Eq. (6), the ODE is solved using a 10-step Euler method, and the squared ℓ_2 distance is used as the training metric $m(\cdot, \cdot)$. Using unconditional generation as an example, our method takes about 100 hours to train the latent-based teacher on two RTX 6000 Ada GPUs and about 30 hours for distillation.

Metrics. For a fair comparison with LiDM and R2Flow, we evaluate generation performance using the same five Fréchet-distance-based perceptual metrics: Fréchet Range Image Distance (FRID), Fréchet Sparse Volume Distance (FSVD), Fréchet Point-based Volume Distance (FPVD), Fréchet Range Distance (FRD), and Fréchet Point Cloud Distance (FPD). In addition, we assess performance with two statistical metrics: Jensen-Shannon Divergence (JSD) and Minimum Matching Distance (MMD). For clarity, all MMD scores are scaled by 10^4 when reported. Time evaluations are performed on a single RTX 4090 GPU.

Comparison with state-of-the-art methods

Unconditional LiDAR generation (Range + Reflectance). Using the 1-RF model from R2Flow (Nakashima et al. 2025) as the teacher, we train a pixel-based student model with the same architecture using our adaptive piecewise distillation method. Following the evaluation protocol of R2Flow, our

Method	Step↓	FRD ↓	FRID ↓	FPD ↓	FPVD ↓	FSVD ↓	JSD ↓	MMD ↓
R2Flow (1-RF)	50	141.86	4.57	10.88	14.86	13.97	0.029	0.45
few-step generation								
R2Flow (1-RF)	**4**	699.94	75.20	734.47	54.88	63.95	0.155	8.52
R2Flow (2-RF)	**4**	221.87	24.33	13.06	45.80	50.21	0.027	0.35
R2Flow (2-RF + 4-TD)	**4**	187.10	15.92	10.92	38.64	40.75	**0.024**	**0.31**
Our LiDAR-Flow	**4**	**165.31**	**12.41**	**9.46**	**32.09**	**33.43**	0.025	0.34

Table 1: Quantitative results for unconditional LiDAR generation (producing range and reflectance) on KITTI-360. All models share the same architecture. 1-RF: model trained via flow matching; 2-RF: trained via reflow (Liu, Gong, and Liu 2023) using 1-RF as teacher; 2-RF + 4-TD: distilled with timestep distillation (Liu, Gong, and Liu 2023) using 2-RF as teacher. Our LiDAR-Flow, using 1-RF as teacher, achieves state-of-the-art performance across most metrics. Bold: best, underline: top-2 results.

Method	Step	FRID	FSVD	FPVD	JSD	MMD
VAE/GAN methods:						
LiDARVAE	1	199.1	129.9	105.8	0.237	7.07
ProjectedGAN	1	149.7	44.7	33.4	0.188	2.88
UltraLiDAR	1	370.0	72.1	66.6	0.747	17.12
Diffusion/Flow methods:						
LiDARGen	1160	129.1	39.2	33.4	0.188	**0.88**
LiDM	50	**125.1**	38.8	29.0	0.211	3.84
few-step generation						
LiDM	**4**	305.5	98.8	87.1	0.242	12.68
Our LiDAR-Flow	**4**	144.0	**34.9**	**23.9**	**0.155**	3.50

Table 2: Quantitative results for unconditional LiDAR generation (producing range image) on the KITTI-360 dataset. Our method, LiDAR-Flow, achieves competitive performance while using only 4 sampling steps.

Method	Step	FRID	FSVD	FPVD	JSD	MMD
LiDM	50	44.9	32.6	25.8	0.205	3.69
LiDM†	50	39.7	32.7	24.0	0.202	3.53
few-step generation						
LiDM†	**4**	63.4	51.0	38.9	0.207	4.37
Our LiDAR-Flow	**4**	**27.0**	**24.5**	**17.9**	**0.123**	**3.41**

Table 3: Quantitative results for camera-image-to-LiDAR generation on the KITTI-360 dataset. Our method, LiDAR-Flow, achieves the best performance with 4 sampling steps. We report both the results presented in the LiDM paper and those of the officially released higher-performing model†.

student model generates 10,000 samples, each containing both range and reflectance (i.e., laser intensity) information.

The comparison between our model and the models proposed in R2Flow is shown in Table 1. Specifically, starting from the 1-RF model as the teacher, R2Flow applies the reflow operation (Liu, Gong, and Liu 2023) for refinement, resulting in an improved model referred to as 2-RF. Subsequently, timestep distillation (Liu, Gong, and Liu 2023) is applied to 2-RF, yielding a distilled model with 4 sampling steps, denoted as 2-RF + 4-TD. As shown in Table 1, under the same setting of 4 sampling steps, our model significantly outperforms both 2-RF and 2-RF + 4-TD on all five perceptual metrics, while achieving comparable performance on the two statistical metrics. These results demonstrate that our model achieves state-of-the-art performance in unconditional LiDAR generation, and our adaptive piecewise distillation outperforms both the reflow operation and timestep distillation used in R2Flow. Our model generates a LiDAR scene in about 55 ms, as shown in Figure 4(a).

Unconditional LiDAR generation (Range only). Following the same evaluation strategy in LiDM (Ran, Guizilini, and Wang 2024), we generate 2,000 samples for evaluation. As shown in Table 2, our method matches the performance of state-of-the-art diffusion and flow-based models (e.g., LiDM and LiDARGen (Zyrianov, Zhu, and Wang 2022)) with only 4 sampling steps, indicating a significant improvement in efficiency. Notably, our model substantially outperforms the 4-step variant of LiDM across all metrics, demonstrating the superiority of our adaptive piecewise distillation method in low-step regimes. A sample result is presented in Figure 4(b), generated in around 68 ms.

Camera-image-to-LiDAR generation. Following LiDM, we conduct camera-image-to-LiDAR generation on the KITTI-360 dataset. As shown in Table 3, our method, LiDAR-Flow, achieves the best performance across all evaluation metrics while using only 4 sampling steps. This demonstrates that our approach is also well-suited for conditional LiDAR generation tasks. Generation takes about 156 ms for this task, illustrated in Figure 4(c).

Semantic-map-to-LiDAR generation. Following LiDM, we conduct semantic-map-to-LiDAR generation on SemanticKITTI. As shown in Table 5, our LiDAR-Flow surpasses the 4-step LiDM on all metrics and is comparable to the 50-step LiDM. A LiDAR scene is produced in about 83 ms, as demonstrated in Figure 4(d). Inference time is summarized in Table 6. More qualitative results can be found in the supplementary material.

Ablation study for adaptive piecewise distillation

We conduct ablation studies to evaluate the effectiveness of our adaptive piecewise distillation on unconditional and camera-to-LiDAR generation. In **adaptive piecewise distillation**, we adaptively split the time range into a sequence of intervals and train the student model only at the beginning of each interval to predict the velocity toward its endpoint. Unlike prior work that seeks to refine the entire generation trajectory, our approach focuses on learning velocities at the adaptively selected sampling times, which simplifies the distillation and improves performance at the chosen times.

Training method	Step ↓	Uncond. (Range+Ref.)			Uncond. (Range)		Camera-to-LiDAR	
		FRD↓	FRID↓	MMD↓	FRID↓	MMD↓	FRID↓	MMD↓
Flow matching loss (teacher model)	50	141.86	4.57	0.45	141.3	3.48	25.5	3.31
	4	699.90	75.11	8.54	238.6	9.55	35.0	3.57
few-step generation								
Reflow operation	8	183.08	16.04	**0.31**	161.1	3.82	28.6	3.41
	4	221.87	24.33	0.35	164.9	3.80	29.3	3.43
Piecewise reflow operation	8	193.13	**10.92**	0.81	147.5	3.68	27.3	3.41
	4	357.00	34.47	1.41	152.4	3.71	28.4	3.45
Our uniform piecewise distillation	**4**	178.31	14.45	0.38	145.7	3.60	27.1	**3.40**
Our adaptive piecewise distillation	**4**	**165.31**	12.41	0.34	**144.0**	**3.50**	**27.0**	3.41

Table 4: Ablation study for adaptive piecewise distillation. Under the same 4-step sampling setting, our adaptive piecewise distillation outperforms competing strategies on most evaluation metrics, demonstrating the effectiveness of both the piecewise distillation and adaptive timestep selection for both unconditional and conditional LiDAR generation.

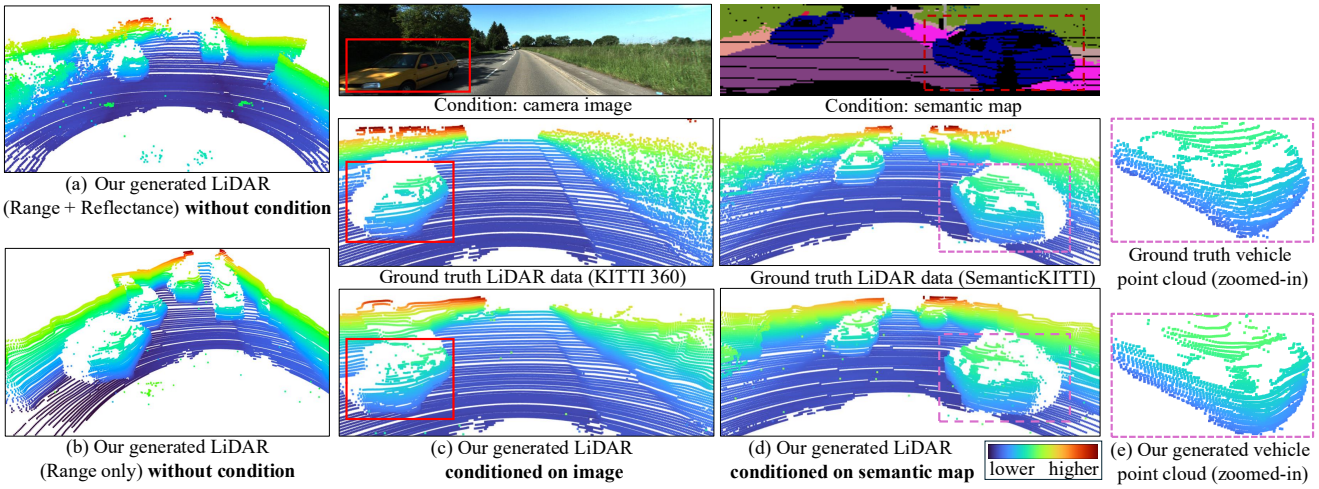

Figure 4: Qualitative results of our method for both unconditional and multi-modal conditional LiDAR generation, using only 4 sampling steps. Our method generates realistic LiDAR data with coherent scene layout and object geometry.

Method	Step	FRID	FSVD	FPVD	JSD	MMD
LiDM	50	22.9	20.2	17.7	0.072	3.16
LiDM†	50	**12.3**	**19.1**	**16.5**	**0.067**	**1.60**
few-step generation						
LiDM†	**4**	24.8	31.5	28.7	0.073	1.97
Our LiDAR-Flow	**4**	13.9	20.0	16.7	0.074	1.66

Table 5: Quantitative results for semantic-map-to-LiDAR generation on the SemanticKITTI. Our method, LiDAR-Flow, delivers strong results under a 4-step sampling setting.

We compare our method against two existing distillation strategies: (1) **Reflow operation** (Liu, Gong, and Liu 2023), which trains a new model using trajectories sampled from a previous model to straighten the entire generation path; (2) **Piecewise Reflow operation** (Yan et al. 2024), which splits the trajectory into multiple intervals and applies Reflow within each, aiming to refine the path in a piecewise manner. Specifically, in the Unconditional (range) and Camera-to-LiDAR generation tasks, we generate synthetic training data online for the Reflow operation. In addition, we include a variant of our method, referred to as **uniform piecewise distillation**, in which the adaptive timestep selection is removed and the time intervals are uniformly divided.

Task	Uncond. (R)	Uncond. (R+R)	C-to-LiDAR	S-to-LiDAR
Time	∼ 68ms	∼ 55ms	∼ 156ms	∼ 83ms

Table 6: Inference time per sample (4 steps).

As shown in Table 4, under the same 4-step sampling, our **uniform piecewise distillation** outperforms all competing strategies on most evaluation metrics across the three tasks, demonstrating the effectiveness of our piecewise distillation strategy. Also, our **adaptive piecewise distillation** achieves additional improvements, demonstrating that adaptively selecting timesteps leads to more effective distillation.

Conclusions

We propose an adaptive piecewise distillation method for efficient LiDAR generation under unconditional and conditional settings, achieving realistic results with 4 steps. In future work, we will explore dynamic LiDAR data generation.

Acknowledgments

This research is supported by the RIE2025 Industry Alignment Fund – Industry Collaboration Projects (IAF-ICP) (Award I2301E0026), administered by A*STAR, as well as supported by Alibaba Group and NTU Singapore through Alibaba-NTU Global e-Sustainability CorpLab (ANGEL).

References

Achlioptas, P.; Diamanti, O.; Mitliagkas, I.; and Guibas, L. 2018. Learning representations and generative models for 3d point clouds. In *International conference on machine learning*, 40–49. PMLR.

Behley, J.; Garbade, M.; Milioto, A.; Quenzel, J.; Behnke, S.; Stachniss, C.; and Gall, J. 2019. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9297–9307.

Caccia, L.; Van Hoof, H.; Courville, A.; and Pineau, J. 2019. Deep generative modeling of lidar data. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5034–5040. IEEE.

Cui, C.; Ma, Y.; Cao, X.; Ye, W.; Zhou, Y.; Liang, K.; Chen, J.; Lu, J.; Yang, Z.; Liao, K.-D.; et al. 2024. A survey on multimodal large language models for autonomous driving. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 958–979.

Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; and Koltun, V. 2017. CARLA: An open urban driving simulator. In *Conference on robot learning*, 1–16. PMLR.

Geiger, A.; Lenz, P.; and Urtasun, R. 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, 3354–3361. IEEE.

Ho, J.; Jain, A.; and Abbeel, P. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33: 6840–6851.

Hu, Q.; Zhang, Z.; and Hu, W. 2024. Rangeldm: Fast realistic lidar point cloud generation. In *European Conference on Computer Vision*, 115–135. Springer.

Ke, L.; Xu, H.; Ning, X.; Li, Y.; Li, J.; Li, H.; Lin, Y.; Jiang, D.; Yang, Y.; and Zhang, L. 2025. ProReflow: Progressive Reflow with Decomposed Velocity. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 28029–28038.

Kim, D.; Lai, C.-H.; Liao, W.-H.; Murata, N.; Takida, Y.; Uesaka, T.; He, Y.; Mitsufuji, Y.; and Ermon, S. 2023. Consistency trajectory models: Learning probability flow ode trajectory of diffusion. *arXiv preprint arXiv:2310.02279*.

Kim, H.; Lee, H.; Kang, W. H.; Lee, J. Y.; and Kim, N. S. 2020. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems*, 33: 16388–16397.

Liao, Y.; Xie, J.; and Geiger, A. 2022. 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.

Lipman, Y.; Chen, R. T.; Ben-Hamu, H.; Nickel, M.; and Le, M. 2022. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*.

Liu, X.; Gong, C.; and Liu, Q. 2023. Flow Straight and Fast: Learning to Generate and Transfer Data with Rectified Flow. In *The Eleventh International Conference on Learning Representations (ICLR)*.

Liu, X.; Zhang, X.; Ma, J.; Peng, J.; et al. 2023. Instaflow: One step is enough for high-quality diffusion-based text-to-image generation. In *The Twelfth International Conference on Learning Representations*.

Lu, C.; Zhou, Y.; Bao, F.; Chen, J.; Li, C.; and Zhu, J. 2022. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. *Advances in neural information processing systems*, 35: 5775–5787.

Luo, S.; and Hu, W. 2021. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2837–2845.

Manivasagam, S.; Wang, S.; Wong, K.; Zeng, W.; Sazanovich, M.; Tan, S.; Yang, B.; Ma, W.-C.; and Urtasun, R. 2020. Lidarsim: Realistic lidar simulation by leveraging the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11167–11176.

Martyniuk, T.; Puy, G.; Boulch, A.; Marlet, R.; and de Charette, R. 2025. LiDPM: Rethinking Point Diffusion for Lidar Scene Completion. In *2025 IEEE Intelligent Vehicles Symposium (IV)*, 555–560.

Mittal, P.; Cheng, Y.-C.; Singh, M.; and Tulsiani, S. 2022. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 306–315.

Nakashima, K.; and Kurazume, R. 2024. LiDAR Data Synthesis with Denoising Diffusion Probabilistic Models. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 14724–14731.

Nakashima, K.; Liu, X.; Miyawaki, T.; Iwashita, Y.; and Kurazume, R. 2025. Fast LiDAR Data Generation with Rectified Flows. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 10057–10063.

Nguyen, B.; Nguyen, B.; and Nguyen, V. A. 2023. Bellman optimal stepsize straightening of flow-matching models. *arXiv preprint arXiv:2312.16414*.

Ran, H.; Guizilini, V.; and Wang, Y. 2024. Towards Realistic Scene Generation with LiDAR Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14738–14748.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Salimans, T.; and Ho, J. 2022. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*.

Sauer, A.; Chitta, K.; Müller, J.; and Geiger, A. 2021. Projected gans converge faster. *Advances in Neural Information Processing Systems*, 34: 17480–17492.

Shan, T.; Englot, B.; Meyers, D.; Wang, W.; Ratti, C.; and Rus, D. 2020. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 5135–5142. IEEE.

Song, J.; Meng, C.; and Ermon, S. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*.

Song, Y.; Dhariwal, P.; Chen, M.; and Sutskever, I. 2023. Consistency models. In *Proceedings of the 40th International Conference on Machine Learning*, 32211–32252.

Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. 2020. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2446–2454.

Vahdat, A.; Williams, F.; Gojcic, Z.; Litany, O.; Fidler, S.; Kreis, K.; et al. 2022. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35: 10021–10039.

Wu, L.; Wang, D.; Gong, C.; Liu, X.; Xiong, Y.; Ranjan, R.; Krishnamoorthi, R.; Chandra, V.; and Liu, Q. 2023. Fast point cloud generation with straight flows. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9445–9454.

Wu, Y.; Zhang, K.; Qian, J.; Xie, J.; and Yang, J. 2024. Text2lidar: Text-guided lidar point cloud generation via equirectangular transformer. In *European Conference on Computer Vision*, 291–310. Springer.

Xiong, Y.; Ma, W.-C.; Wang, J.; and Urtasun, R. 2023. Learning compact representations for lidar completion and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1074–1083.

Xu, W.; Cai, Y.; He, D.; Lin, J.; and Zhang, F. 2022. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4): 2053–2073.

Yan, H.; Liu, X.; Pan, J.; Liew, J. H.; Liu, Q.; and Feng, J. 2024. Perflow: Piecewise rectified flow as universal plug-and-play accelerator. *Advances in Neural Information Processing Systems*, 37: 78630–78652.

Yang, G.; Huang, X.; Hao, Z.; Liu, M.-Y.; Belongie, S.; and Hariharan, B. 2019. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 4541–4550.

Zhou, L.; Du, Y.; and Wu, J. 2021. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF international conference on computer vision*, 5826–5835.

Zyrianov, V.; Che, H.; Liu, Z.; and Wang, S. 2024. LidarDM: Generative LiDAR Simulation in a Generated World. arXiv:2404.02903.

Zyrianov, V.; Zhu, X.; and Wang, S. 2022. Learning to generate realistic lidar point clouds. In *European Conference on Computer Vision*, 17–35. Springer.