Fast Point Cloud Generation with Straight Flows

Lemeng Wu[1], Dilin Wang[2], Chengyue Gong[1], Xingchao Liu[1], Yunyang Xiong[2],
Rakesh Ranjan[2], Raghuraman Krishnamoorthi[2], Vikas Chandra[2], Qiang Liu[1]
[1] University of Texas at Austin [2] Meta
{lmwu, cygong, xcliu, lqiang}@cs.utexas.edu, {wdilin, rakeshr, raghuraman, vchandra}@fb.com

Abstract

Diffusion models have emerged as a powerful tool for point cloud generation. A key component that drives the impressive performance for generating high-quality samples from noise is iteratively denoise for thousands of steps. While beneficial, the complexity of learning steps has limited its applications to many 3D real-world. To address this limitation, we propose Point Straight Flow (PSF), a model that exhibits impressive performance using one step. Our idea is based on the reformulation of the standard diffusion model, which optimizes the curvy learning trajectory into a straight path. Further, we develop a distillation strategy to shorten the straight path into one step without a performance loss, enabling applications to 3D real-world with latency constraints. We perform evaluations on multiple 3D tasks and find that our PSF performs comparably to the standard diffusion model, outperforming other efficient 3D point cloud generation methods. On real-world applications such as point cloud completion and training-free text-guided generation in a low-latency setup, PSF performs favorably.

1. Introduction

3D point cloud generation has many real-world applications across vision and robotics, including self-driving and virtual reality. A lot of efforts have been devoted to realistic 3D point cloud generation, such as VAE [14], GAN [1, 36], Normalizing Flow [13, 16, 43] and score-based method [5, 27, 47, 50], and diffusion model [27, 47, 50]. Among them, diffusion models gain increasing popularity for generating realistic and diverse shapes by separating the distribution map learning from a noise distribution to a meaningful shape distribution into thousands of steps.

Despite the foregoing advantages, the transport trajectory learning from a noise distribution to a meaningful shape distribution also turns out to be a major efficiency bottleneck during inference since a diffusion model requires thousands of generative steps to produce high-quality and diverse shapes [11, 37, 50]. As a result, it leads to high computation costs for generating meaningful point cloud in practice. Notice that the learning transport trajectory follows the simulation process of solving stochastic differentiable equation (SDE). A trained neural SDE can have different distribution mappings at each step, which makes the acceleration challenging even with an advanced ordinary differentiable equation (ODE) solver.

To address this challenge, several recent works have proposed strategies that avoid using thousands of steps for the meaningful 3D point cloud generation. For example, [26,35] suggest distilling the high-quality 3D point cloud generator, DDIM model [37], into a few-step or one-step generator. While the computation cost is reduced by applying distillation to shorten the DDIM trajectory into one-step or few-step generator. The distillation process learns a direct mapping between the initial state and the final state of DDIM, which needs to compress hundreds of irregular steps into one-step. Empirically it leads to an obvious performance drop. Further, these distillation techniques are mainly developed for generating images with the grid structure, which is unsuitable for applying to point cloud generation since the point cloud is an unordered set of points with irregular structures.

In this paper, we propose using one-step to generate 3D point clouds. Our method, Point Straight Flow (PSF), learns a straight generative transport trajectory from a noisy point cloud to a desired 3D shape for acceleration. This is achieved by passing the neural flow model once to estimate the transport trajectory. Specifically, we first formulate an ODE transport flow as the initial 3D generator with a simpler trajectory compared with the diffusion model formulated in SDE. Then we optimize the transport flow cost for the initial flow model to significantly straighten the learning trajectory while maintaining the model's performance by adopting the ideas from recent works [20, 22]. This leads to a straight flow by optimizing the curvy learning trajectory into a straight path. Lastly, with the straight transport trajectory, we further design a distillation technique to shorten the path into one-step for 3D point cloud generation.

To evaluate our method, we undertake an extensive set of experiments on 3D point cloud tasks. We first verify

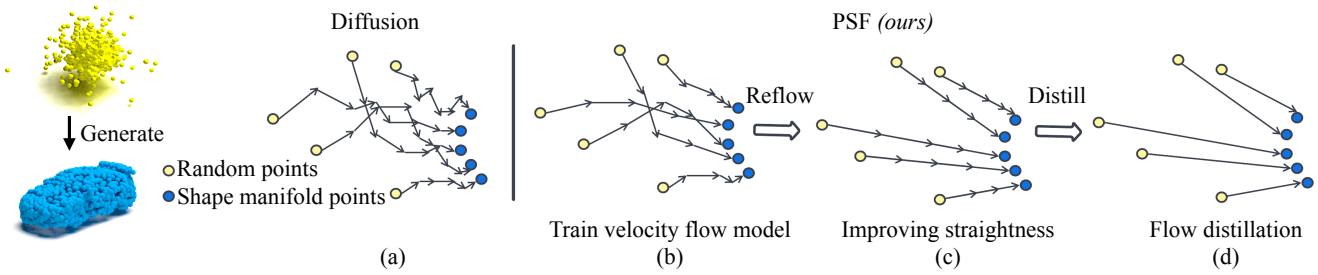

Figure 1. Trajectories during the generate process for the point cloud generation. (a) The SDE (PVD) trajectory involves random noise in each step and thus gives a curvy trajectory. (b) The PSF initial flow model removes the random noise term and gets a simulation procedure trajectories with smaller transport cost. (c) By utilizing the *reflow* process on the initial flow model, we reduce the transport cost. As a result, the trajectories are becoming straightened and easy to simulate with one step. (d) The straight path leads to a small time-discrimination error during the simulation, which makes the model easy to distill into one-step.

that our one-step PSF can generate high-quality point cloud shapes, performing favorably relative to the diffusion-based point cloud generator PVD [50] with a more than 700× faster sampling on the unconditional 3D shape generation task. Further, we demonstrate it is highly important to learn straight generative transport trajectory for faster sampling by comparing distillation baselines, including DDIM that are difficult to generate shapes even with many more generative steps. Finally, we perform evaluations on 3D real-world applications, including point cloud completion and training-free text-guided generation, to show the flexibility of our one-step PSF generator.

- For the first time, we demonstrate that neural flow trained with one step can generate high-quality 3D point clouds by applying distillation strategy.
- We propose a novel 3D point cloud generative model, Point Straight Flow (PSF). Our proposed PSF optimizes the curvy transport trajectory between noisy samples and meaningful point cloud shapes as a straight path.
- We show our PSF can generate 3D shapes with high-efficiency on standard benchmarks such as unconditional shape generation and point cloud completion. We also successfully extend PSF to real-world 3D applications including large-scene completion and training-free text-guided generation.

2. Related Works

2.1. Generative model with transport flow

The generative model can be treated as transporting a distribution to another. Previous works like VAE [15] and GAN [10] build this transport in one-step using neural networks. However, as the data capacity and complexity increase, this one-step mapping takes a lot of effort to train. Recently, the community tries to relieve this issue by decomposing the one-step mapping into several steps in ODE [6,31,37] or SDE [11, 38–40] fashion. Among these works, denoising diffusion probabilistic models (DDPM) [11] demonstrate the power and flexibility to generate high-quality samples on large-scale image benchmarks and other domains [7, 12, 30, 32, 33, 44], which makes the diffusion model become a mainstream to learn the transport flow.

2.2. Fast sampling for transport model

Despite the huge success of DDPM, a major issue of this model is that it requires thousands of steps to generate high-quality desired samples. Previous works propose multiple strategies to decrease the simulation steps and accelerate the DDPM to learn transport process. DDIM [37] formulates the sampling trajectory process as an ODE. FastDPM [17] bridges the connection between the discrete and continuous time step. These two methods can help reduce the learning trajectory to hundreds of steps. Beyond this, to further compress the generation process to few-step simulation, [26, 35] apply knowledge distillation to learn a few mappings to recover the multiple DDIM steps. However, these methods are hard to maintain a good performance with a single step or even more steps. Recent ODE transport works [2, 20–22] try to build the ODE transport with a reduced cost for a faster simulation compared with DDIM. Motivated by the above ideas, we optimize the transport cost first to learn the transport trajectory with one step in our PSF model.

2.3. 3D generative model

Generating 3D objects enables various applications. Previous literature focus on using VAE [3, 14, 19], GAN [1, 8, 18, 41] and Normalizing Flow [13, 16, 24, 42, 43] to generate the 3D object in point cloud, mesh and voxel representations. Recently, more dedicated shapes can be generated using the score-based or diffusion models [5, 27, 28, 45, 47, 49, 50]. While the diffusion model in the point cloud generation obtains state-of-the-art results, it is computationally expensive to generate even one high-quality shape. This makes the diffusion generative model difficult to apply to real-world

3D applications. Our PSF boosts the speed under 0.1s while maintaining the performance compared with diffusion-based methods.

3. Point Straight Flow

We now introduce our method, Point Straight Flow (PSF), one-step transport flow for 3D point cloud generation. Generating point clouds with transport flow can be viewed as transporting noise point clouds to target data point clouds by following a learned trajectory. To consolidate the transport steps as few as one, we propose a three-stage training pipeline as illustrated in Figure 1: 1) Our first step is to learn a neural velocity flow network. We construct an ODE process with the shortest transport path and utilize a neural network to fit this process. At the end of this step, one can start with random Gaussian noises and then iteratively apply the ODE process from the learned network to generate samples. However, the training object does not exactly match the generative process. Thus, the trajectory could still be curvy. 2) Our second step is to optimize the trajectory's straightness learned at step 1 via *reflow* adapted from [22]. This *reflow* stage encourages the velocity network to flow straightly while performing sample generation. This allows us to combine multiple updates into one easily. 3) We further distill the neural network with the objective such that one step with a fixed large step size (Figure 1 (c)) makes the same update as the iteration of multiple small updates as in Figure 1 (b).

Training initial velocity flow network Our goal is to build a transport flow to push the point clouds from a Gaussian distribution to the point cloud data distribution.

Specifically, assume the point cloud in the xyz-coordinates and denote $X_0 \in \mathbb{R}^{M\times 3}$ as Gaussian noise and $X_1 \in \mathbb{R}^{M\times 3}$ as real data samples. We denote v_θ as the velocity field network by the following ODE process,

$$\underbrace{\mathrm{d}X_t}_{\text{drift}} = \underbrace{v_\theta(X_t, t)}_{\text{velocity}} \underbrace{\mathrm{d}t}_{\text{time interval}}, \quad \text{with } t \in [0, 1]. \tag{1}$$

Here X_t is the intermediate point cloud states at time t and the velocity filed $v_\theta : \mathbb{R}^{M\times 3} \to \mathbb{R}^{M\times 3}$ is a neural network with θ as its parameters.

Given the intermediate point cloud X_t, v_θ defines a velocity field that moves X_t further towards true data X_1. Intuitively, the optimal direction at any time t is $X_1 - X_0$. Thus, we can encourage our velocity field to directly follow the optimal ODE process $\mathrm{d}X_t = (X_1 - X_0)\mathrm{d}t$ by optimizing

$$\min_\theta \int_0^1 \mathbb{E}\Big[\|(v_\theta(X_t, t) - (X_1 - X_0)\|^2\Big]\mathrm{d}t, \quad \text{where } X_t = tX_1 + (1-t)X_0 \quad t \in [0, 1]. \tag{2}$$

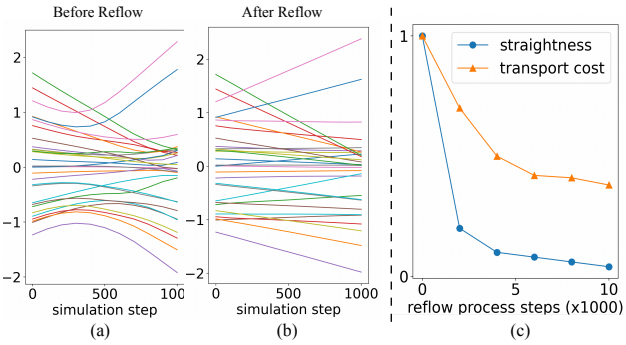

Figure 2. We visualize the impact of *reflow*. (a) and (b) shows the generation trajectory before and after *reflow*. (c) shows the transport and straightness changes during the *reflow* finetune. We use value 1 to represent the initial value before the *reflow* and 0 as the ideal value that represents the optimal transport or strict straightness.

Empirically, we do not optimize the loss in Eqn. 2 with the integration on $t \in [0, 1]$ directly. Instead, for each data sample X_1, we randomly draw a X_0 from Gaussian noise, a t from [0, 1] and minimize the following equivalent loss,

$$\min_\theta \mathbb{E}\Big[\|(v_\theta(X_t, t) - (X_1 - X_0)\|^2\Big]. \quad t \sim \mathcal{U}(0, 1). \tag{3}$$

After the neural velocity field is well-trained, samples can be generated by discretizing the ODE process with Euler solver in Eqn. 1 into N steps (e.g., $N = 1000$) as the following,

$$X'_{(\hat{t}+1)/N} \longleftarrow X'_{\hat{t}/N} + \frac{1}{N} v_\theta(X'_{\hat{t}/N}, \frac{\hat{t}}{N}), \tag{4}$$

the integer time step $\hat{t}$ is defined as $\hat{t} \in \{0, 1, \cdots, N-1\}$. Here X'_1 denotes our generated samples and $X'_0 = X_0$. Intuitively, the solver will be more accurate with a large N.

Improving straightness of the flow In the previous stage, network v_θ is trained on data pairs (X_0, X_1), with X_1 as ground truth data points. We empirically found the trajectory learned in this way is still curvy during the generative process. Therefore, a large number N is necessary for accurate approximation of Eqn. 1 with Euler solver as in Eqn. 4.

Specifically, in Figure 2 (a), we show an example of unconditional point cloud generation and plot the movement trajectories of 30 randomly sampled points with a randomly picked dimension on x/y/z. The update of each point is guided by Eqn. 4 with $N = 1000$. See Section 4.1 for more detailed discussions on experiment settings.

Intuitively, one would prefer a straight trajectory, which means a smaller N needs to give a similar output as a bigger N, namely, generating samples with fewer updates. To this

end, we sample a set of Gaussian noise X_0' and generate its corresponding samples X_1' following Eqn. 4 with v_θ fixed. After that, one can finetune the v_θ using the sampled pairs (X_0', X_1') follow Eqn. 2 by replacing the (X_0, X_1). This is referred to as the *reflow* procedure in [22]. Theoretically, the sampled fixed pairs (X_0', X_1') always provide a lower transport cost than pairs (X_0, X_1). Thus training on the (X_0', X_1') can reduce the transport cost and straighten the trajectory.

In Figure 2 (b), we show the resulting trajectories after applying *reflow* procedure by still taking $N = 1000$. As we can see from this figure, the trajectories are nearly straight for all the points. In Figure 2 (c), we quantitatively show the change of the transport cost and straightness after the *reflow* procedure. The transport cost is defined as l_2 transport cost and the straightness of trajectories can be written as the following,

$$\text{Straightness} = \frac{1}{N}\sum_{\hat{t}=0}^{N-1}\left[\parallel (X_1' - X_0') - v_\theta(X_{\hat{t}/N}, \hat{t}/N) \parallel^2\right]. \tag{5}$$

A straight trajectory between X_0' and X_1' implies a constant velocity $v_\theta(X_{\hat{t}/N}, \hat{t}/N) = X_1' - X_0'$ at every time $\hat{t}$ step. And Eqn. 5 equals 0 in this case.

Flow distillation Our observation above indicates the possibility to approximate Eqn. 1 with only one discretization step if the trajectories are straight,

$$X_1' = X_0' + v_\theta(X_0', 0). \tag{6}$$

And our goal is to make sure the update in Eqn. 6 leads to samples that have similar quality as samples generated by Eqn. 4 with a large N. See Figure 1 (c) for an illustration.

For this purpose, we introduce a third *distillation stage* to summarize the refined ODE process with v_θ in the previous stage. In particular, we construct the distillation objective as follows,

$$\min_\theta \mathbb{E}\left[\text{Dist}(\underbrace{X_0' + v_\theta(X_0', 0)}_{\text{one step}}, \underbrace{X_1'}_{N \text{ step}})\right], \tag{7}$$

where $\text{Dist}(\cdot)$ is a loss function that measures the difference between two sets of point clouds. And X_1' is generated with $N = 1000$ same as the previous *reflow* step.

In the literature [26, 35], the $\text{Dist}(\cdot)$ is typically constructed as a ℓ_2 loss. However, unlike images which impose a strict alignment at each pixel location during construction, the permutation invariant property of point clouds makes ℓ_2 loss less suitable. Hence we propose the Chamfer distance as the objective for distillation. Specifically, assume X_i, X_j as two point clouds, the Chamfer distance is defined as

$$\text{CD}(X_i, X_j) = \sum_{p\in X_i}\min_{\hat{p}\in X_j}||p-\hat{p}||_2 + \sum_{\hat{p}\in X_j}\min_{p\in X_i}||p-\hat{p}||_2. \tag{8}$$

Summary We summarize the overall algorithm in Algorithm 1, and refer readers to the Appendix for the training and sampling pseudo-code. Overall, our algorithm gives a one-step point cloud generation approach. After these three training stages, one can sample a point cloud in one step starting from a random noise by following Eqn. 6.

Algorithm 1 Point Straight Flow

Input: Point cloud dataset $\mathcal{D}$, a neural velocity field v_θ with parameter θ.
1. Training initial velocity flow model: randomly sample $X_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and $X_1 \sim \mathcal{D}$, and train v_θ follows the objective function Eqn. 3 to convergence.
2. Improving straightness via *reflow*: Sample a set of point cloud pairs, with $X_0' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and X_1' generated with Eqn. 4. Use (X_0', X_1') as training data to finetune v_θ by still minimizing Eqn. 3.
3. Flow distillation: Use pairs (X_0', X_1') to finetune v_θ with the distillation loss Eqn. 7 into one-step generator as the final PSF model.
Sampling (output): Randomly sample from $X_0' \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and output the desired point cloud X_1' with $X_1' = X_0' + v_\theta(X_0', 0)$.

4. Experiment

We empirically demonstrate the effectiveness and efficiency of our method on three tasks, including unconditional point cloud generation, training-free text-guided point cloud generation and point cloud completion. Compared with the prior-art diffusion-based point cloud generalization method PVD [50], our method reduces the GPU inference time by 75× while still producing realistic samples at a similar quality level.

General training settings We adopt the PVCNN [25] styled U-Net [34] proposed in PVD [50] as our velocity filed model v_θ. As we summarize in Algorithm 1, there are three phases, including 1) training the velocity flow model, 2) improving straightness via *reflow*, and 3) flow distillation. For step 1), we mainly follow the setting in DDPM [11] and use a batch size of 256 and a learning rate of $2e^{-4}$. We train the model for 200k steps and apply an exponential moving average (EMA) at a rate of 0.9999. For step 2), we randomly sample 50k data pairs (X_0', X_1') using the pretrained network v_θ and fine-tune it for 10k steps by minimizing Eqn. 2. Here

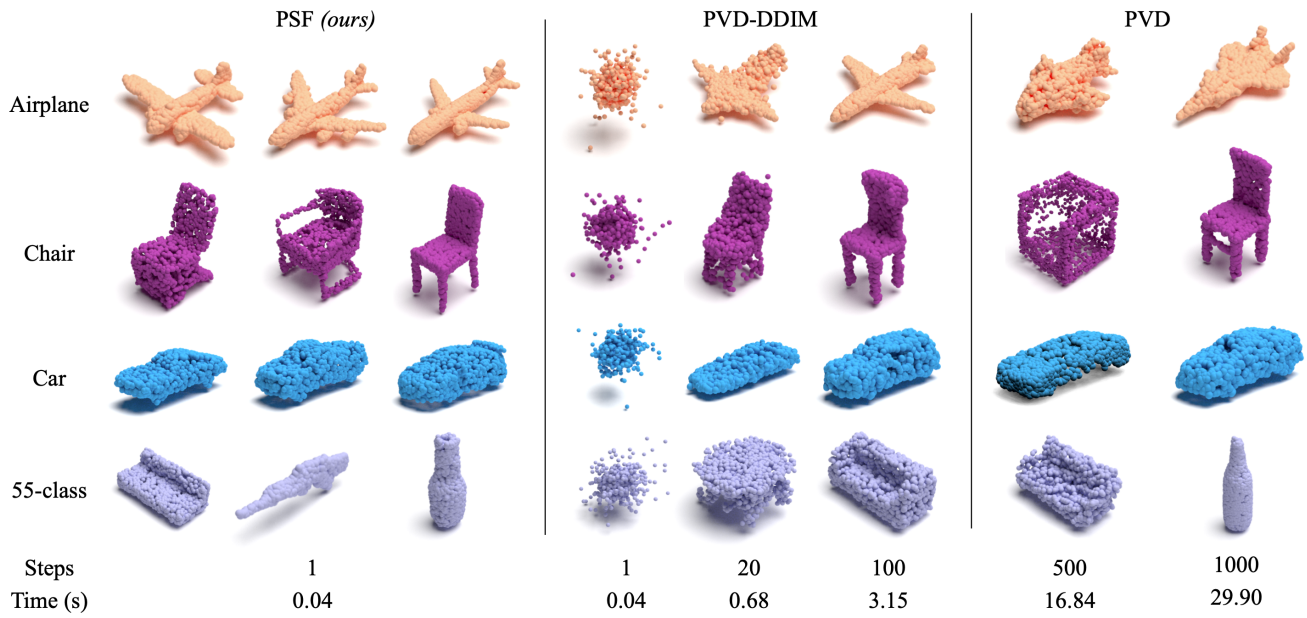

Figure 3. Visualization for the unconditional generation for Airplane, Chair, Car and 55-class. We show PSF can generate high-quality point cloud in one-step with only 0.04 seconds.

X_1' is sampled by setting $N = 1000$ in Eqn. 4 for the best quality. We use a fixed learning rate of $2e^{-5}$ in this step. For step 3), we use the samples (X_0', X_1') generated in step 2) and finetune v_θ for another 10k steps with learning rate $2e^{-5}$.

4.1. Unconditional point cloud generation

We apply our method PSF for unconditional 3D point cloud generation. Compared with existing solutions, our method runs as fast as one-shot VAE-based (e.g.,SetVAE [14]) or GAN-based (e.g., 1-GAN [1]) approaches, in the meantime, our method generates samples as high quality as computation expensive diffusion-based methods like PVD. Additionally, we also compare PSF with a fast-sampling method DDIM [37] combined with PVD, we denote this approach as PVD-DDIM. Compared with PVD-DDIM, our method achieves much better sample quality as well as sampling efficiency.

Settings We use the same training and testing data split by following PointFlow [43] and PVD [50]. We choose one nearest neighbor accuracy (1-NNA) with Chamfer Distance (CD) and Earth Movement Distance (EMD) as our metrics to measure the sample quality per the suggestion in PVD. Please refer Appendix for other common metrics, including Matching Distance (MMD) and Coverage Score (COV). We test the sampling time of our method and baselines on Nvidia RTX 3090 GPUs with a batch size of 1. All the time comparisons are averaged on 50 random trials. For the PVD-DDIM setup, we grid search the time step N in $\{1, 20, 50, 100, 500, 1000\}$ and report the $N = 100$, which is the smallest step with 1-NNA performance degrate no larger than 10 % compared with PVD.

Results We first show qualitative comparisons in Figure 3. Our method generates samples of similar visual quality compared to the samples from the expensive PVD approach with 1000 steps. Compared with PVD-DDIM in 100 steps and PVD in 500 steps, our PSF produces better samples with clear shapes and boundaries.

Additionally, we show quantitative comparisons in Table 1. Overall, our method PSF achieves similar CD and EMD scores compared with PVD (N=1000) in all three categories. The performance is only slightly worse on the Chair and Car, while the visual quality is almost the same as demonstrated in Figure 3. Most significantly, the average sampling time of our method is only 0.04s on a modern GPU, which is more than $75\times$ faster compared with PVD-DDIM (N=100) and $700\times$ faster compared with PVD (N=1000). We provide the additional comparisons for other time-step setups for PVD, PVD-DDIM and PSF in the ablation study in Section 4.4.

Initial state linear interpolation leads to interpretable interpolation for final samples Besides sampling efficiency, we show another benefit of our method: the ability to generate interpretable point clouds by starting from interpolated noise due to the straightness of the velocity trajectory. Specifically, we first randomly draw initial point clouds $\tilde{x}_0$ and $\tilde{x}_1$ from a Gaussian distribution, then we apply the lin-

Model	Sampling Time (s)	Airplane		Chair		Car	
		CD ↓	EMD ↓	CD ↓	EMD ↓	CD ↓	EMD ↓
1-GAN [1]	**0.03**	87.30	93.95	68.58	83.84	66.49	88.78
PointFlow [43]	0.27	75.68	70.74	62.84	60.57	58.10	56.25
DPF-Net [16]	0.33	75.18	65.55	62.00	58.53	62.35	54.48
SoftFlow [13]	0.12	76.05	65.80	59.21	60.05	64.77	60.09
SetVAE [13]	**0.03**	75.31	77.65	58.76	61.48	59.66	61.48
ShapeGF [5]	0.34	80.00	76.17	68.96	65.48	63.20	56.53
DPM [27]	22.8	76.42	86.91	60.05	74.77	68.89	79.97
PVD [50] (N=1000)	29.9	**73.82**	**64.81**	**56.26**	**53.32**	**54.55**	**53.83**
PVD-DDIM [37] (N=100)	3.15	76.21	69.84	61.54	57.73	60.95	59.35
PSF *(ours)*	**0.04**	**71.11**	**61.09**	**58.92**	**54.45**	**57.19**	**56.07**

Table 1. *Performance* (1-NNA ↓) and *Sampling Time* on single class generation. The second block represents the fast simulation methods. We report the smallest step size of PVD and PVD-DDIM , which do not drop performance. The sampling time is calculated when the batch size is one.

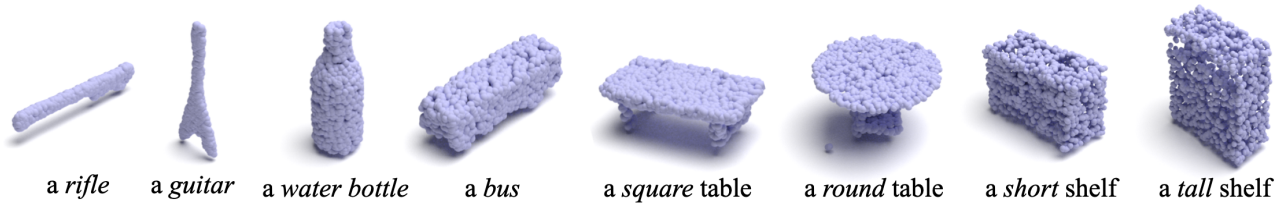

Figure 4. Training-free text-guided point cloud generation using CLIP loss. We show that our 55-class pretrained PSF can generate the correct and high-quality shapes following the text prompt.

ear interpolation between $\tilde{x}_0$ and $\tilde{x}_1$ to generate additional Gaussian noise,

$$\tilde{x}_\tau = \sqrt{(1-\tau)}\tilde{x}_0 + \sqrt{\tau}\tilde{x}_1 \quad \tau \in [0,1].$$

We apply both our method and PVD to generate samples by taking $\{\tilde{x}_\tau\}$ as the starting points. We present our findings in Figure 5. As we can see from the Figure, a smooth change in inputs leads to a smooth change between the final generated samples for our method. While PVD trained with diffusion algorithms doesn't enjoy this property.

4.2. Training-free text-guided shape generation

In this section, we extended our method for training-free (i.e., with v_θ fixed) text-guided generation [9, 23, 29]. Specifically, given a text input, training-free text-guided sample generation can be framed as finding the best initial noise X_0 such that the resulting generated sample matches the text prompt semantically.

We follow the setting in Text2Mesh [29] and FuseDream [23] for sample optimization,

$$\min_{X_0} \mathbb{E}_{\{\mathrm{Proj}_i\}} \left[S_{\text{clip}} \left(\mathrm{Proj}_i \cdot \mathrm{generator}(X_0), \ \ \text{text} \right) \right], \quad (9)$$

where $\{\mathrm{Proj}_i\}$ is a set of pre-defined projections that uniformly cover different angles and project the point cloud to 2D space. And S_{clip} is the CLIP loss that calculates the distance between image and text by a pretrained CLIP model, $\mathrm{generator}$ is a sampler that transports initial X_0 to a meaningful point cloud sample. Note that $\mathrm{generator}(\cdot)$ is fixed. During the optimization, we need to forward then backward through the $\mathrm{generator}$. Thus a multi-step iterative $\mathrm{generator}$ would largely slow down the optimization.

Settings and Results Given a text prompt, we optimize Eqn. 9 for 100 iterations. This amounts to 12 seconds for our method. In contrast, PVD takes about 15 minutes to generate a text-conditioned sample due to its thousands-step sampling.

In Figure 4, we show that both the shape and category of our generated point clouds correlated well with the provided text prompts. It can correctly match the text prompt about the object class as well as the basic properties.

4.3. Point Cloud Completion

We further apply our method for point cloud completion in both synthetic and real-world settings. We follow the PVD setup and treat the partial point cloud as the conditional input of the generative model. Our goal is to sample meaningful point clouds conditioned on a partial point cloud as input. Please refer Appendix for a detailed discussion on our settings.

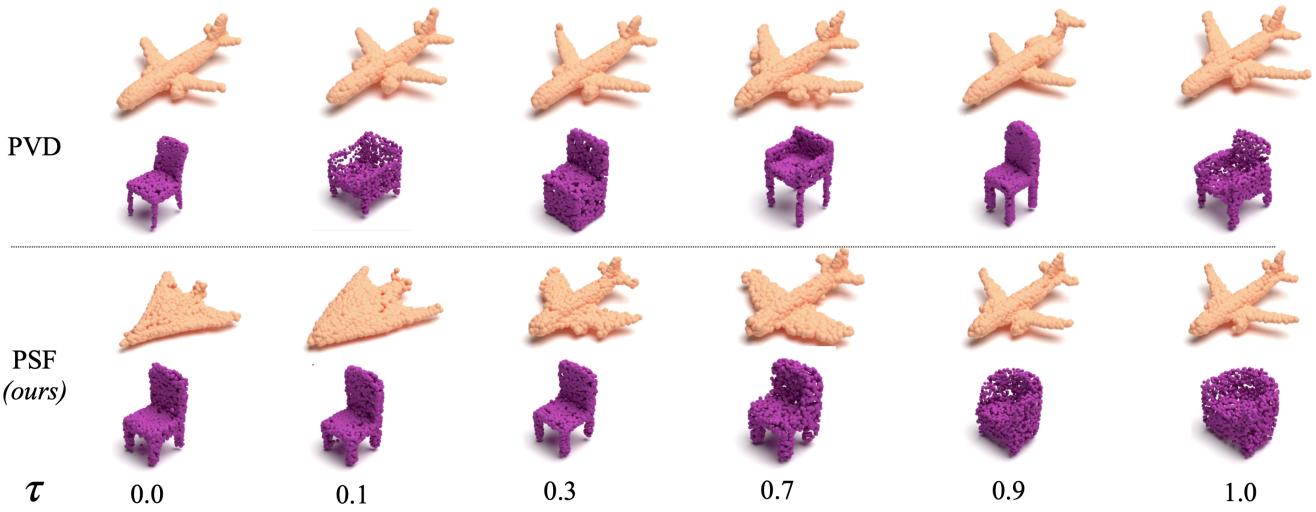

Figure 5. Interpolation of two random picked initialization. Our PSF can simulate the interpolated initial status with a continuously changing shape, while PVD simulation the shapes without a relationship.

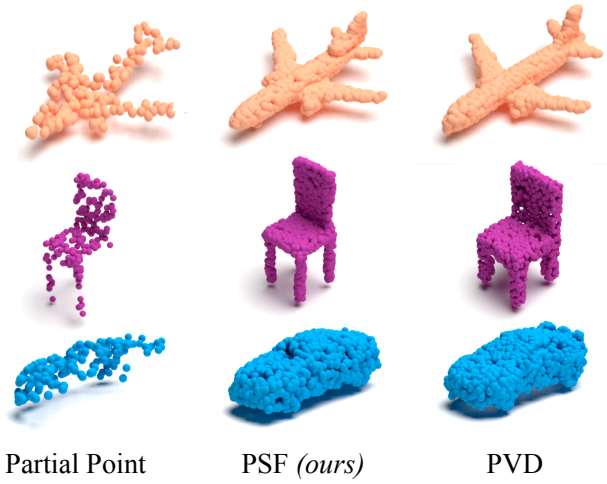

Figure 6. Point Cloud Completion visualization. We show that given the same partial point cloud, we can construct a similar quality point cloud as PVD.

Settings We follow PVD and GenRe [48] and use 20-view depth images rendered from *Chair*, *Airplane*, and *Car* as the input. We sample 200 points as the partial point cloud input for each depth image to train our PSF. We use EMD as our evaluation metrics as PVD shows that EMD is better than metric compared with CD for the shape completion task.

Results on synthetic shapes We summarize our findings in Table 2. Our method generates a completed point cloud of similar quality compared to the results from PVD while reducing the completion latency to only 0.04s. In Figure 6, we show additional qualitative comparisons with PVD. There is no quality degradation visually between our method vs. PVD, further confirming the efficacy of our method on generating high-quality 3D point clouds while being real-time.

Category	Model	Time (s) ↓	EMD ↓
Airplane	SoftFlow [13]	0.12	1.198
	PointFlow [43]	0.27	1.180
	DPF-Net [16]	0.34	1.105
	PVD (N=1000)	29.98	1.030
	PSF (*ours*)	**0.04**	**1.004**
Chair	SoftFlow [13]	0.12	3.295
	PointFlow [43]	0.27	3.649
	DPF-Net [16]	0.34	3.320
	PVD (N=1000)	29.98	2.939
	PSF (*ours*)	**0.04**	**2.937**
Car	SoftFlow [13]	0.12	2.789
	PointFlow [43]	0.27	2.851
	DPF-Net [16]	0.34	2.318
	PVD (N=1000)	29.98	**2.146**
	PSF (*ours*)	**0.04**	2.194

Table 2. Generate quality and latency between PSF and baselines. CD is multiplied by 10^3 and EMD is multiplied by 10^2.

Method	Dist	Airplane		Chair		Car	
		CD	EMD	CD	EMD	CD	EMD
PVD-DDIM	ℓ_2	85.5	83.1	82.6	79.3	80.1	77.9
PVD-DDIM	CD	79.9	73.1	72.4	68.2	66.3	65.7
PSF (*ours*)	ℓ_2	79.5	73.5	68.4	62.0	67.8	67.0
PSF (*ours*)	CD	**71.1**	**65.0**	**59.9**	**54.3**	**57.1**	**56.0**

Table 3. Ablation study on distillation loss configurations in 1-NNA (↓), the CD represents using Chamfer distance as the distance function of the distillation loss.

Transfer to Lidar point cloud completion With the fast and high-quality completion result, we are able to apply PSF completion to real-world applications that require low latency. One of the important areas that the fast completion benefits are the outdoor 3D detection for autonomous driv-

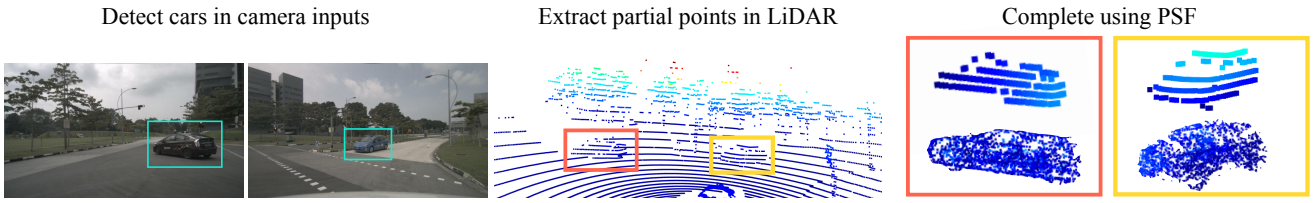

Figure 7. We adapt the PSF in real-world point cloud completion procedure in a low-latency pipeline. All images are from nuScenes [4].

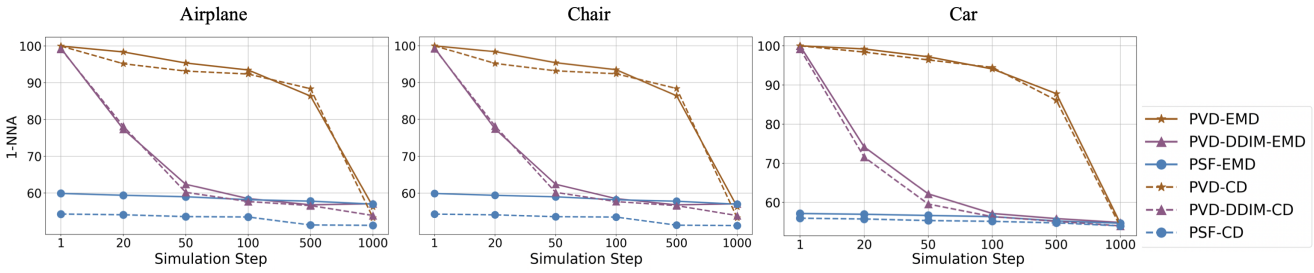

Figure 8. Ablation study on sampling in different steps for PVD, PVD-DDIM and PSF.

ing. Usually, the LiDAR scan on the car can only capture a partial view of the sparse point cloud. This makes learning a reasonable 3D detector challenging. MVP [46] shows that completing and densifying the sparse partial point cloud using nearest neighbor retrieval can lead to better 3D detection performance. To further enhance the completion and densification qualities, most of the current methods, including PVD, are too slow to meet the low latency requirement. To this end, PSF can perform a suitable role by applying this method to enhance the LiDAR scan.

To demonstrate the possibility of our method to enable point cloud completion in low latency, in Figure 7, we follow MVP and first detect the cars using the camera input with a 2D detector and locate the corresponding partial point clouds in the LiDAR scan. We then apply our model pretrained on *Car* to complete the partial point cloud. In this example, images and sparse point cloud scans are randomly sampled from nuScenes [4].

Overall, our PSF can generate the completed point clouds for multiple cars in 0.2s, compared with 2D detector and 3D detector that usually cost more than 0.8s for a single scene, our PSF completions is efficient enough for a low-latency requirement. On the contrary, PVD more than one minute to complete the point clouds.

4.4. Ablation Study

Different simulation steps We consolidate the sampling steps to one step by default. In this part, we further study the performance of the multi-step model without distillation. We follow the same training settings in Section 4.1. We only apply distillation on one-step and for other steps, we only perform reflow process. We see that reflow model with straight line already gets a well-performed model with few steps and the distillation mainly works for pushing the one-step to a similar performance.

Distillation loss Different from the image representation, which has a well-alignment pixel grid, the point cloud mapping from two distributions is randomly permutated. We study the impact of different distillation loss choices to show how Chamfer distance better deals with the point cloud representation in distillation setup and how the straightness benefits the distillation. From Table 3, we show that Chamfer distance significantly outperforms the naive ℓ_2 loss for distillation.

5. Discussion and Conclusion

In this paper, we present a fast point cloud generator, Point Straight Flow, which generates high-quality samples from noise in one step by optimizing the curvy learning transport trajectory. Extensive experiments on several standard 3D point cloud benchmarks and real-world applications, including unconditional generation, training-free text-guided generation, and point cloud completion consistently validate PSF's advantages. We also demonstrate that PSF generates a high-fidelity 3D point cloud sample much faster than PVD and the PVD-DDIM.

Our method is easy to formulate and implement, which can serve as an alternative to standard diffusion-based point cloud generator. Our preliminary work suggests that PSF has potential applications beyond complex scene completion and text-guided generation applications.

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, 2, 5, 6

[2] Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022. 2

[3] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016. 2

[4] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 8, 4

[5] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *European Conference on Computer Vision*, pages 364–381. Springer, 2020. 1, 2, 6

[6] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018. 2

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021. 2

[8] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. In *Advances In Neural Information Processing Systems*, 2022. 2

[9] Chengyue Gong, Lemeng Wu, and Qiang Liu. How to fill the optimum set? population gradient descent with harmless diversity. *arXiv preprint arXiv:2202.08376*, 2022. 6

[10] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2

[11] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 1, 2, 4

[12] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *arXiv preprint arXiv:2204.03458*, 2022. 2

[13] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. Softflow: Probabilistic framework for normalizing flow on manifolds. *Advances in Neural Information Processing Systems*, 33:16388–16397, 2020. 1, 2, 6, 7

[14] Jinwoo Kim, Jaehoon Yoo, Juho Lee, and Seunghoon Hong. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15059–15068, 2021. 1, 2, 5

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2

[16] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *European Conference on Computer Vision*, pages 694–710. Springer, 2020. 1, 2, 6, 7

[17] Zhifeng Kong and Wei Ping. On fast sampling of diffusion probabilistic models. *arXiv preprint arXiv:2106.00132*, 2021. 2

[18] Ruihui Li, Xianzhi Li, Ka-Hei Hui, and Chi-Wing Fu. Sp-gan: Sphere-guided 3d shape generation and manipulation. *ACM Transactions on Graphics (TOG)*, 40(4):1–12, 2021. 2

[19] Angela S. Lin, Lemeng Wu, and Qixing Huang Raymond J. Mooney Rodolfo Corona, Kevin Tai. Generating animated videos of human activities from natural language descriptions. In *Proceedings of the Visually Grounded Interaction and Language Workshop at NeurIPS 2018*, December 2018. 2

[20] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. 1, 2

[21] Qiang Liu. Rectified flow: A marginal preserving approach to optimal transport. *arXiv preprint arXiv:2209.14577*, 2022. 2

[22] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. 1, 2, 3, 4

[23] Xingchao Liu, Chengyue Gong, Lemeng Wu, Shujian Zhang, Hao Su, and Qiang Liu. Fusedream: Training-free text-to-image generation with improved clip+ gan space optimization. *arXiv preprint arXiv:2112.01573*, 2021. 6

[24] Xingchao Liu, Lemeng Wu, Mao Ye, and Qiang Liu. Let us build bridges: Understanding and extending diffusion generative models. *arXiv preprint arXiv:2208.14699*, 2022. 2

[25] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *Advances in Neural Information Processing Systems*, 32, 2019. 4

[26] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed. *arXiv preprint arXiv:2101.02388*, 2021. 1, 2, 4

[27] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2837–2845, 2021. 1, 2, 6

[28] Zhaoyang Lyu, Zhifeng Kong, Xudong Xu, Liang Pan, and Dahua Lin. A conditional point diffusion-refinement paradigm for 3d point cloud completion. *arXiv preprint arXiv:2112.03530*, 2021. 2

[29] Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13492–13502, 2022. 6

[30] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 2

[31] George Papamakarios, Eric T Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021. 2

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

[33] 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. 2

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 4

[35] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *arXiv preprint arXiv:2202.00512*, 2022. 1, 2, 4

[36] Dong Wook Shu, Sung Woo Park, and Junseok Kwon. 3d point cloud generative adversarial network based on tree structured graph convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 3859–3868, 2019. 1

[37] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 1, 2, 5, 6

[38] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019. 2

[39] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2020. 2

[40] Belinda Tzen and Maxim Raginsky. Theoretical guarantees for sampling and inference in generative models with latent diffusions. In *Conference on Learning Theory*, pages 3084–3114. PMLR, 2019. 2

[41] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pages 82–90, 2016. 2

[42] Lemeng Wu, Chengyue Gong, Xingchao Liu, Mao Ye, and Qiang Liu. Diffusion-based molecule generation with informative prior bridges. *arXiv preprint arXiv:2209.00865*, 2022. 2

[43] 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, 2, 5, 6, 7

[44] Ruihan Yang, Prakhar Srivastava, and Stephan Mandt. Diffusion probabilistic modeling for video generation. *arXiv preprint arXiv:2203.09481*, 2022. 2

[45] Mao Ye, Lemeng Wu, and Qiang Liu. First hitting diffusion models. *arXiv preprint arXiv:2209.01170*, 2022. 2

[46] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multimodal virtual point 3d detection. *NeurIPS*, 2021. 8

[47] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. Lion: Latent point diffusion models for 3d shape generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2

[48] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. *Advances in neural information processing systems*, 31, 2018. 7

[49] Yan Zheng, Lemeng Wu, Xingchao Liu, Zhen Chen, Qiang Liu, and Qixing Huang. Neural volumetric mesh generator. *arXiv preprint arXiv:2210.03158*, 2022. 2

[50] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5826–5835, 2021. 1, 2, 4, 5, 6