TFDM: Time-Variant Frequency-Based Point Cloud Diffusion with State Space Model
Supplementary Material

This supplementary material provides additional details, visualizations, and experimental results to support the findings presented in the main paper. The document is organized as follows:

- Sec. 1 provides the reference code implementation (including a README file) and examples of our generated point clouds, which achieve state-of-the-art performance, validating our results. (As the system does not accept .zip files, we provide a partial PDF of the code for reference.)
- Sec. 2 presents further qualitative visualizations of our method.
- Sec. 3 details the proposed 3D Spiral Serialization (3DSS) algorithm.
- Sec. 4 offers further explanations of the evaluation metrics.
- Sec. 5 provides additional background on the foundational methods used in our framework.

1. Source Code

We provide the reference code in the supplementary material and will release the full source code upon acceptance. The code is located in the `code` directory. Additionally, we include sampled point clouds generated by our model for testing purposes (see Sec. 1.6). The usage steps are as follows:

1.1. Dependency

- Python 3.8
- CUDA 11.6
- Pytorch 1.12.1 + cu116

1.2. Environment

Can simply install via:

```
conda env create -f ldm.yml
conda activate ldm
```

Listing 1. Install Environment

1.3. Compile

```
export TORCH_CUDA_ARCH_LIST="
    6.1;6.2;7.0;7.5;8.0;8.6"
cd metrics/PyTorchEMD
python setup.py install
cp build/lib.linux-x86_64-3.8/emd_cuda.cpython-36
    m-x86_64-linux-gnu.so .
```

Listing 2. Compilation Steps

1.4. Dataset

Download the ShapeNet Dataset [1]:

```
gdown https://drive.google.com/uc?id=1
    sw9gdk_igiyyt7MqALyxZhRrtPvAn0sX
unzip ShapeNetCore.v2.PC15k.zip
```

Listing 3. Download Dataset

1.5. Training & Testing

```
python train_generation.py --category chair --bs
    32 --mamba_depth 8 --latent_size 512 --order
    'hilbert'/'z'/'spiral' --use_multi_order True
python test_generation.py --category chair --
    mamba_depth 8 --latent_size 512 --order '
    hilbert'/'z'/'spiral' --use_multi_order True
    --model 'saved_checkpoint_path'
```

Listing 4. Training and Testing

1.6. Evaluation

We provide the generated point cloud set which achieve the state-of-the-art results:

```
python test.py --category car --eval_path '
    Best_Save_points/best_car_cd_cov.pth'
```

Listing 5. Evaluation

Further details can be found in README file.

2. Extra Visualization

This section presents additional qualitative results of our generative models. A supplementary video is provided to demonstrate the generation outcomes (`video.mp4`) in greater detail. A screen capture of the video is shown in Fig. 2.

As shown in Fig. 1, the diffusion model recovers finer details in the final timesteps, demonstrating its suitability for frequency analysis.

3. 3D Spiral Serialization

We provide the detailed definition and formulation of 3D Spiral Serialization (3DSS). Let the grid be of size $L \times L \times L$, with the point set $\mathcal{L} = \{0, 1, \ldots, L-1\}^3$. The geometric center is $\mathbf{c} = \left(\frac{L-1}{2}, \frac{L-1}{2}, \frac{L-1}{2}\right)$. Thus, for any point $\mathbf{x} = (x, y, z) \in \mathcal{L}$, define its Chebyshev (cube) radius as

$$r(\mathbf{x}) = \max\big(|x - c_x|,\ |y - c_y|,\ |z - c_z|\big). \quad (1)$$

where the radius-r shell is

$$S_r = \{\mathbf{x} \in \mathcal{L} \mid r(\mathbf{x}) = r\}, \qquad r = 0, 1, \ldots, \lfloor L/2 \rfloor. \quad (2)$$

Thus, for traverse all the grid, we construct a path function

$$\pi : \{0, 1, \ldots, L^3 - 1\} \ \rightarrow \ \mathcal{L} \quad (3)$$

by concatenating the ordered traversals of all shells:

$$\pi = \pi^{(0)} \,\|\, \pi^{(1)} \,\|\, \cdots \,\|\, \pi^{(R)}, \qquad R = \lfloor L/2 \rfloor, \tag{4}$$

where $\pi^{(r)}$ enumerates all points in S_r in a continuous manner.

Within each shell, we slice it by the z-coordinate. For a fixed radius r and slice z, the cross-section $_{r,z} = \{\,(x, y, z) \in S_r\,\}$, forms a 2D square ring of effective radius $m = r - |z - c_z|$. We traverse each ring using a 2D square-spiral function

$$\mathrm{Spiral}_2(u;\, m) : \{0, \ldots, (2m+1)^2 - 1\} \to \{x, y\}, \tag{5}$$

which enumerates points on the $(2m{+}1){\times}(2m{+}1)$ square in a clockwise or counter-clockwise spiral centered at (c_x, c_y).

Subsequently, the full 3D traversal for shell r is given by

$$\pi^{(r)}(k) = \big(\mathrm{Spiral}_2(u_k;\, r - |z_k - c_z|),\, z_k\big), \tag{6}$$

where z_k follows an alternating order around c_z (e.g., $c_z,\, c_z + 1,\, c_z - 1,\, c_z + 2,\, c_z - 2, \ldots$), ensuring that consecutive slices connect through a single 6-connected vertical step.

We set $\pi(0) = \mathbf{c}$. The resulting path starts at the center, visits all points in the inner cube (e.g., $3 \times 3 \times 3$ for $L = 5$), and then grows outward shell by shell until all L^3 points are traversed. By construction, consecutive points satisfy

$$\|\pi(k+1) - \pi(k)\|_1 = 1, \qquad \forall k, \tag{7}$$

ensuring a continuous 6-connected space-filling traversal of the entire 3D grid.

4. Metrics

The 1-NNA metric calculates the leave-one-out accuracy of a 1-NN classifier to evaluate point cloud generation performance, which correlates with both the quality and diversity of the generated samples. Thus, a 1-NNA score closer to 50% indicates that the distribution of generated samples is closer to the real data, signifying better performance. COV measures the number of reference point clouds matched to at least one generated shape, correlating with generation diversity; therefore, a higher COV value is desirable. These metrics provide a comprehensive evaluation by considering both the diversity and quality of the generated point clouds relative to the reference set.

- **1-NNA** (1-Nearest Neighbor) Accuracy: Measures the leave-one-out accuracy of a 1-NN classifier, reflecting both quality and diversity of generated samples. A value close to 50% indicates a better result.
- **1-NNA-Abs50** (Absolute 50-Shifted 1-NNA): Since the interpretation of 1-NNA can be ambiguous, we propose a clearer alternative for evaluation. Transforms the aforementioned 1-NNA x into $|x - 50|$, making it more sensitive to deviations from the ideal 50%; a lower score indicates an ideal generated distribution closer to real data.

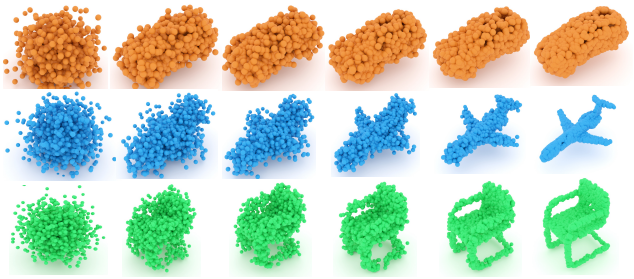

Figure 1. Illustrative examples of the reverse diffusion process demonstrating detailed information recovery at the final timesteps (left to right, timesteps progressing from T to 0).

- **COV** (Coverage): Evaluates how many reference point clouds are matched to at least one generated shape, where a higher value indicates greater diversity in generation.
- **CD** (Chamfer Distance): Measures point-wise similarity between generated and reference point clouds by computing the average nearest neighbor distance.
- **EMD** (Earth Mover's Distance): Captures the minimal cost of transforming one distribution into another, providing a global similarity measure between point clouds.

5. Preliminaries

In this section, we provide further details on the foundational concepts related to our model.

5.1. Denoising Diffusion Probabilistic Model

For given samples $x_0 \sim q(x_0)$, the diffusion model [3, 6] gradually reverses a Markovian fixed forward diffusion process:

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{8}$$

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{\alpha_t}\mathbf{x}_{t-1}, (1-\alpha_t)\mathbf{I}), \tag{9}$$

where T denotes the total time steps, and $q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the transition kernel that progressively perturbs the input according to a sequence of pre-defined variance schedules $(1-\alpha_1), \ldots, (1-\alpha_T)$.

The reverse process is parameterized as a Markovian chain $p_\theta(\mathbf{x}_{0:T})$ which is equal to $p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$,

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \sigma_t^2\mathbf{I}), \tag{10}$$

where $p(\mathbf{x}_T)$ is a standard Gaussian and $\mu_\theta(\mathbf{x}_t, t)$ is the learnable term, with σ_t^2 set as a fixed variance schedule. This term is optimized by matching the ground truth denoising step, which can be interpreted as learning the source noise ϵ_0 by minimizing $w(t)\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon_0\|_2^2$, where $w(t)$ is a weighting parameter dependent only on the timestep. The optimization objective thus becomes:

$$\mathcal{L} = \mathbb{E}_{t\sim[1,T]}\left[w(t)\|\epsilon_\theta(\mathbf{x}_t, t) - \epsilon_0\|_2^2\right] \tag{11}$$

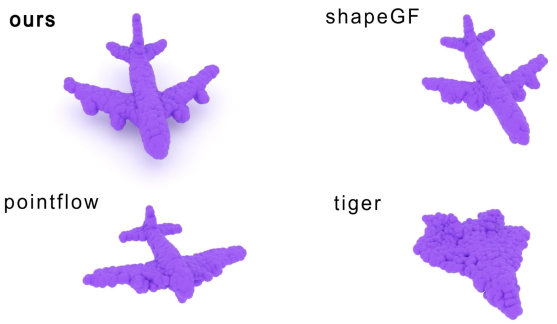

Figure 2. Screen Capture of Qualitative Results Video

After training, generation can be achieved via the inverse chain by sampling from a standard Gaussian distribution.

5.2. State Space Model

The State Space Model (SSM) [2] can be described as a continuous system that maps a 1-D function or sequence $x(t)$ to $y(t)$, mediated through an N-D latent state $h(t)$:

$$h'(t) = \mathbf{A}h(t) + \mathbf{B}x(t), \quad y(t) = \mathbf{C}h(t), \tag{12}$$

where $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$ are learnable parameters. Mamba [2] improves the SSM by relaxing the time-invariance constraint and discretizing the formulation via a timescale transformation parameter Δ. By using zero-order hold techniques, the parameters are defined as:

$$\overline{\mathbf{A}} = \exp(\Delta\mathbf{A}), \quad \overline{\mathbf{B}} = (\Delta\mathbf{A})^{-1}\left(\exp(\Delta\mathbf{A}) - \mathbf{I}\right)\Delta\mathbf{B}. \tag{13}$$

Subsequently, Eq. (13) can be discretized to compute the outputs at specific time steps:

$$h_t = \overline{\mathbf{A}}h_{t-1} + \overline{\mathbf{B}}x_t, \quad y_t = \mathbf{C}h_t. \tag{14}$$

Finally, it employs a structured global convolution to enhance computational efficiency:

$$\overline{\mathbf{K}} = (\mathbf{C}\overline{\mathbf{B}},\ \mathbf{C}\overline{\mathbf{A}\mathbf{B}},\ \ldots,\ \mathbf{C}\overline{\mathbf{A}}^{M-1}\overline{\mathbf{B}}), \quad \mathbf{y} = \mathbf{x} * \overline{\mathbf{K}}, \tag{15}$$

where M and $\overline{\mathbf{K}}$ represent the sequence length of $\mathbf{x}$ and the kernel of the global convolution, respectively.

5.3. Graph Filter

Given a graph $\mathcal{G} = (\mathcal{V}, \tilde{\mathcal{A}}_w, \tilde{\mathcal{A}}_u)$, let $\mathcal{V} = \{v_1, \ldots, v_N\}$ denote a set of N nodes, and let $\tilde{\mathcal{A}}_w, \tilde{\mathcal{A}}_u \in \mathbb{R}^{N\times N}$ represent the weighted and unweighted adjacency matrices, respectively. We refer to one-channel features on all nodes as a graph signal $\mathbf{s} \in \mathbb{R}^N$. $\tilde{\mathcal{A}}_w$ has the eigen decomposition $\tilde{\mathcal{A}}_w = V\mathbf{\Lambda}V^{-1}$, where the matrix V contains eigenvectors of $\tilde{\mathcal{A}}_w$ and $\mathbf{\Lambda}$ is the diagonal eigenvalue matrix corresponding to ordered eigenvalues $\lambda_1, \ldots, \lambda_N$.

As stated in [4], the ordered eigenvalues represent frequencies on the graph. Consider $\tilde{\mathcal{A}}_w$ as a graph shift operator and take a signal $\mathbf{s}$ to produce $\mathbf{y} = \tilde{\mathcal{A}}_w\mathbf{s}$, which implies $V^{-1}\mathbf{y} = \mathbf{\Lambda}V^{-1}\mathbf{s}$. The graph Fourier transformation of graph signals $\mathbf{s}$ and $\mathbf{y}$, denoted as $\hat{\mathbf{s}} = V^{-1}\mathbf{s}$ and $\hat{\mathbf{y}} = V^{-1}\mathbf{y}$, can be considered as the frequency contents of signals $\mathbf{s}$ and $\mathbf{y}$. Additionally, a graph filter is a polynomial in the graph shift [5]: $h(\tilde{\mathcal{A}}_w) = \sum_{l=0}^{L-1} h_l\tilde{\mathcal{A}}_w^l$, where h_l and L denote the filter coefficients and the length of the filter, respectively. This filter takes signal $\mathbf{s}$ and generates $\mathbf{y} = h(\tilde{\mathcal{A}}_w)\mathbf{s} = Vh(\mathbf{\Lambda})V^{-1}\mathbf{s}$, yielding $V^{-1}\mathbf{y} = h(\mathbf{\Lambda})V^{-1}\mathbf{s}$ and thus $\hat{\mathbf{y}} = h(\mathbf{\Lambda})\hat{\mathbf{s}}$. The diagonal matrix $h(\mathbf{\Lambda})$ is the graph frequency response of the filter $h(\tilde{\mathcal{A}}_w)$, denoted as $\hat{h}(\tilde{\mathcal{A}}_w)$, where the frequency response of λ_i is $\sum_{l=0}^{L-1} h_l\lambda_i^l$.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1

[2] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 3

[3] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Adv. Neural Inform. Process. Syst. (NeurIPS)*, 33:6840–6851, 2020. 2

[4] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José M. F. Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018. 3

[5] Aliaksei Sandryhaila and José M. F. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014. 3

[6] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Int. Conf. Mach. Learn. (ICML)*, pages 2256–2265. PMLR, 2015. 2