Supplementary material

Neural Wavelet-domain Diffusion for 3D Shape Generation

SIGGRAPH ASIA 2022

Contents

A. More Visual Galleries on Different Categories

Figure 1. Gallery of cabinets and airplanes generated by our method.

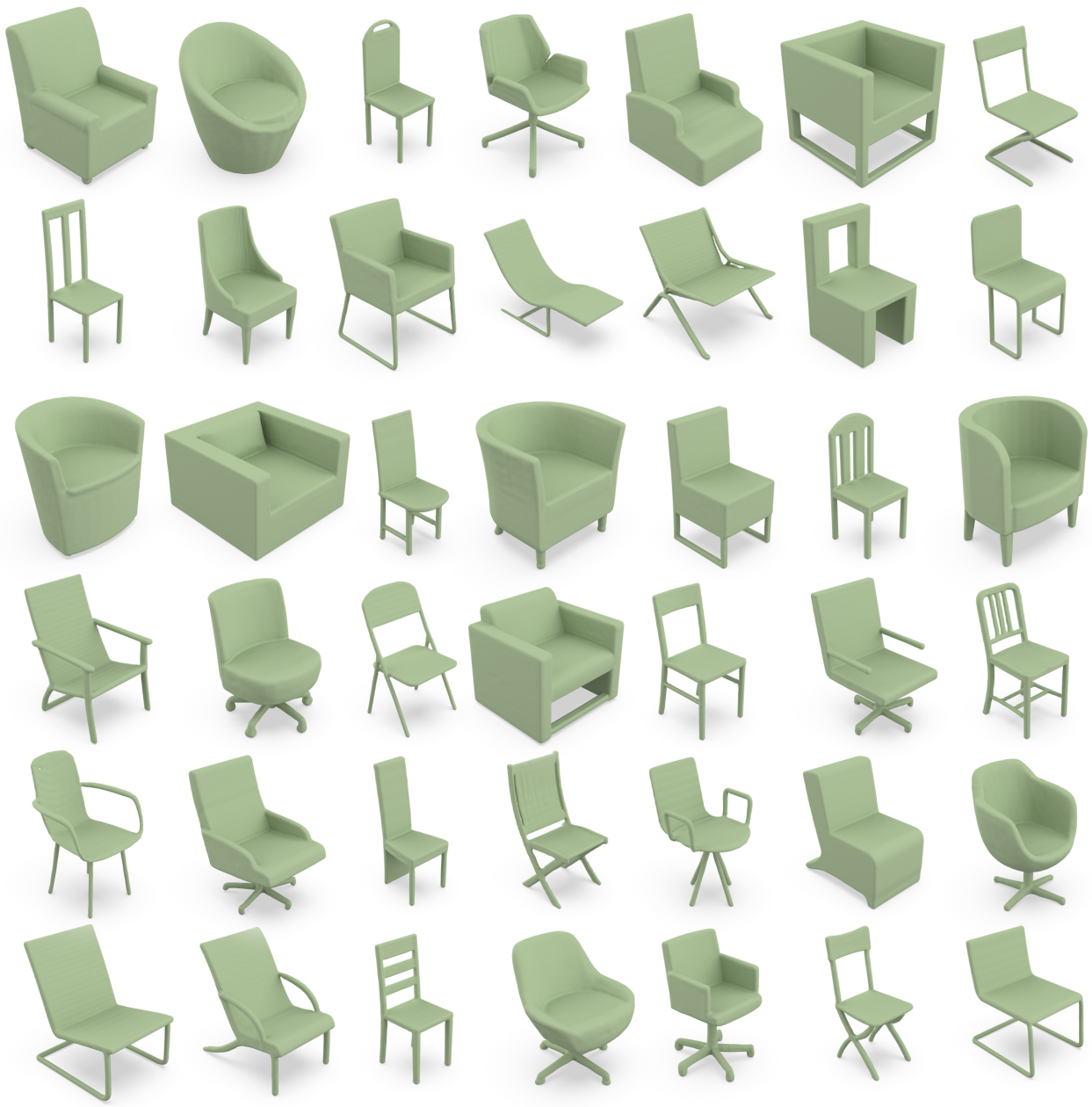

Figure 2. Gallery of chairs generated by our method.

Figure 3. Gallery of chairs generated by our method (cont.).

Figure 4. Gallery of tables generated by our method.

B. Visual Comparisons with Other Methods using Our Results for Retrieval

In this section, we provide additional visual comparisons with other methods (see also Figure 5 in main paper). For each random shape generated by our method, we find a similar shape (with similar structures and topology) generated by each of the state-of-the-art methods to make the visual comparison easier. From the retrieved results shown below, we can see that the 3D shapes generated by our method clearly exhibit finer details, higher fidelity structures, and cleaner surfaces, without obvious artifacts.

Figure 5. Visual comparisons between our method and other state-of-the-art methods. Our generated shapes exhibit finer details and cleaner surfaces without obvious artifacts.

C. Visual Comparisons on Our Shapes retrieved by IM-GAN's Shapes

As different methods likely have different statistical modes in the shape generation distribution, in addition to the visual comparisons shown earlier in Section B, we take random shapes generated by IM-GAN and find similar shapes generated by our method for visual comparisons. From the retrieved results shown below, we can also see that our method can generate shapes of similar structures as those produced by IM-GAN, while exhibiting more fine details.

Figure 6. Visual comparisons between shapes randomly generated by IM-GAN and similar shapes (retrieved from IM-GAN's) generated by our method.

D. Visual Comparisons of Random Shapes Generated by Different Methods

In this section, we showcase shapes randomly generated by our method and other state-of-the-art methods on the Chair and Airplane categories. We can see that our shapes have a larger diversity and have a better visual quality without obvious artifacts.

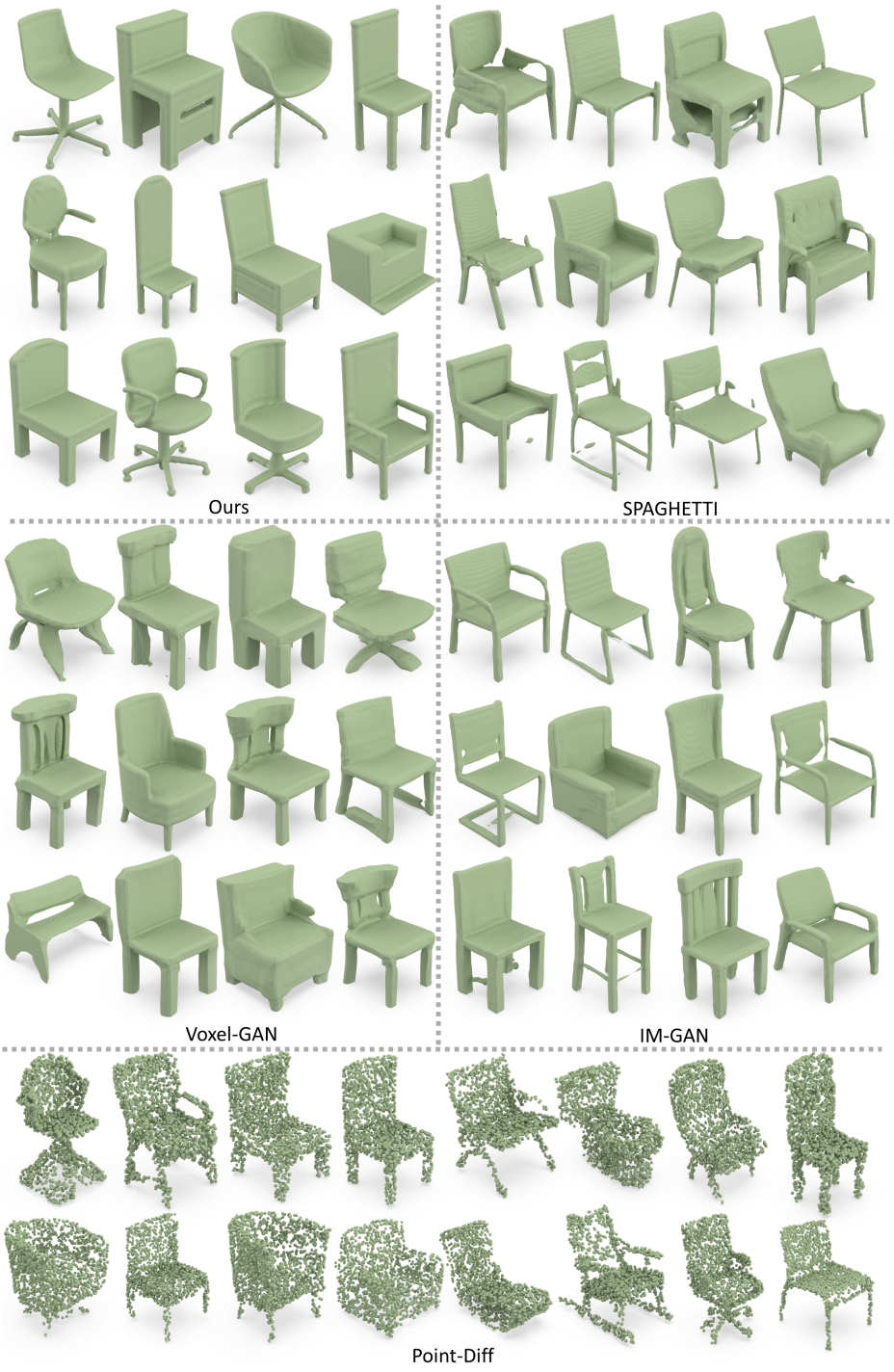

Figure 7. Visual comparisons of our randomly-generated chairs with those by SPAGHETTI Hertz et al. [2022], Voxel-GAN Kleineberg et al. [2020], IM-GAN Chen and Zhang [2019], and Point-Diff Luo and Hu [2021].

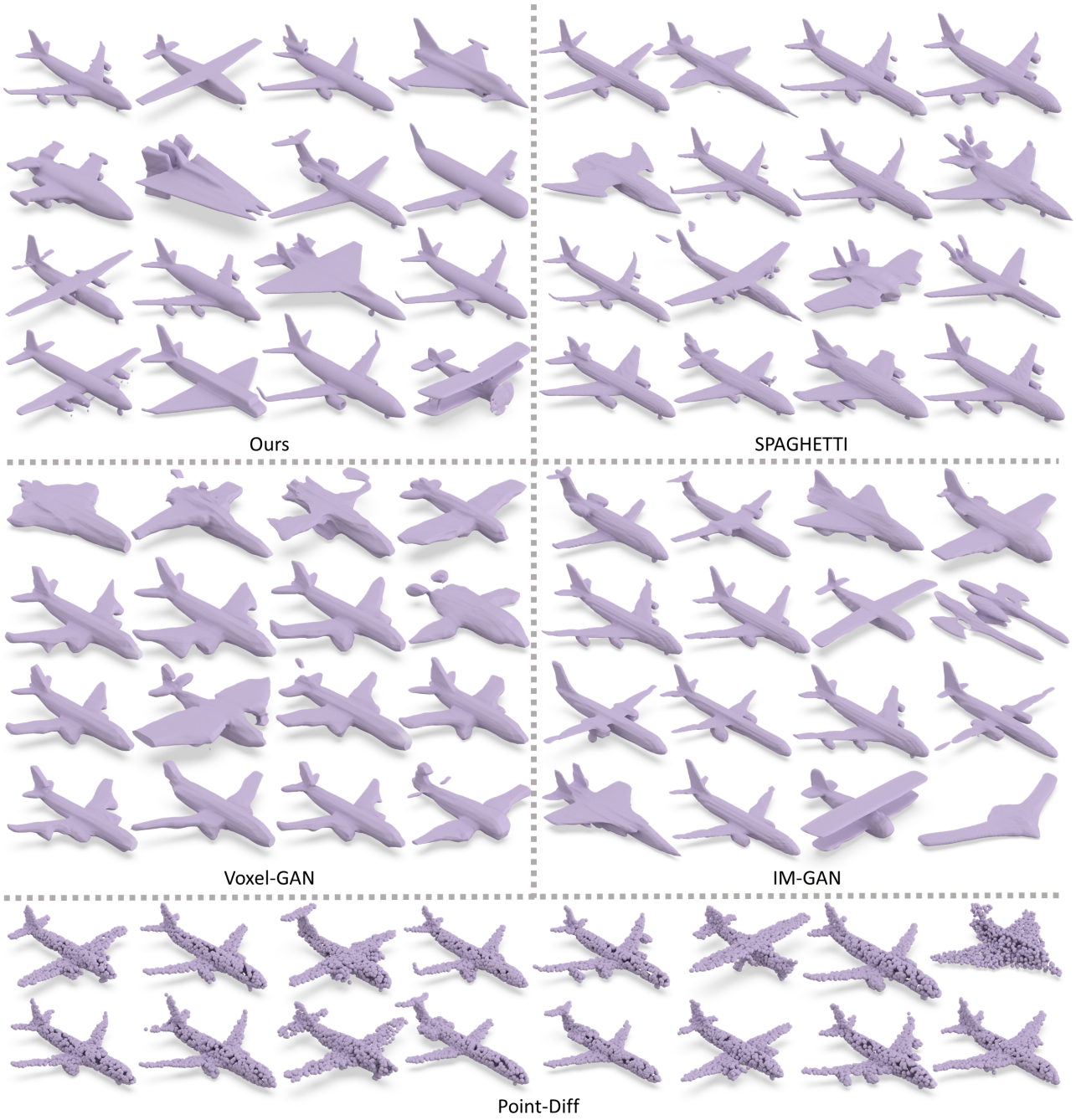

Figure 8. Visual comparisons of our randomly-generated airplanes with those by SPAGHETTI Hertz et al. [2022], Voxel-GAN Kleineberg et al. [2020], IM-GAN Chen and Zhang [2019], and Point-Diff Luo and Hu [2021].

E. More examples for Shape Novelty Analysis

Due to the page limit in the main paper, we now present more examples for the shape novelty analysis beyond Figure 6 (top) in the main paper. For each shape generated by our method, we show the top-four most similar shapes retrieved from the training set by CD and LFD. Comparing our shapes with the retrieved ones, we can see that the shapes share similar structures, showing that our method is able to generate realistic-looking structures like those in the training set. Beyond that, our shapes exhibit noticeable differences in various local structures.

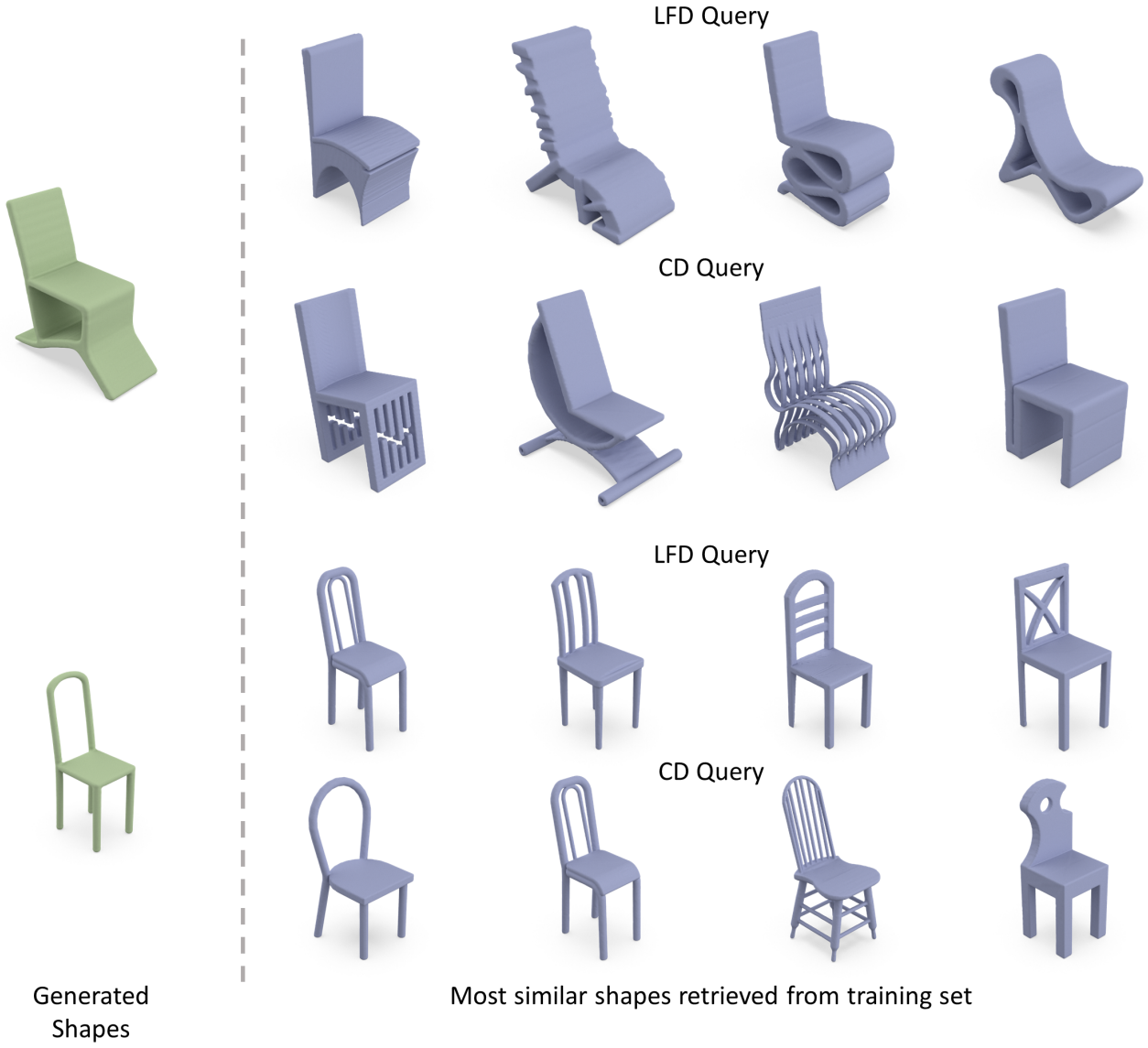

Figure 9. Shape novelty analysis. For each random shape (left) generated by our method, we retrieve top-four most similar shapes in the training set by CD and LFD.

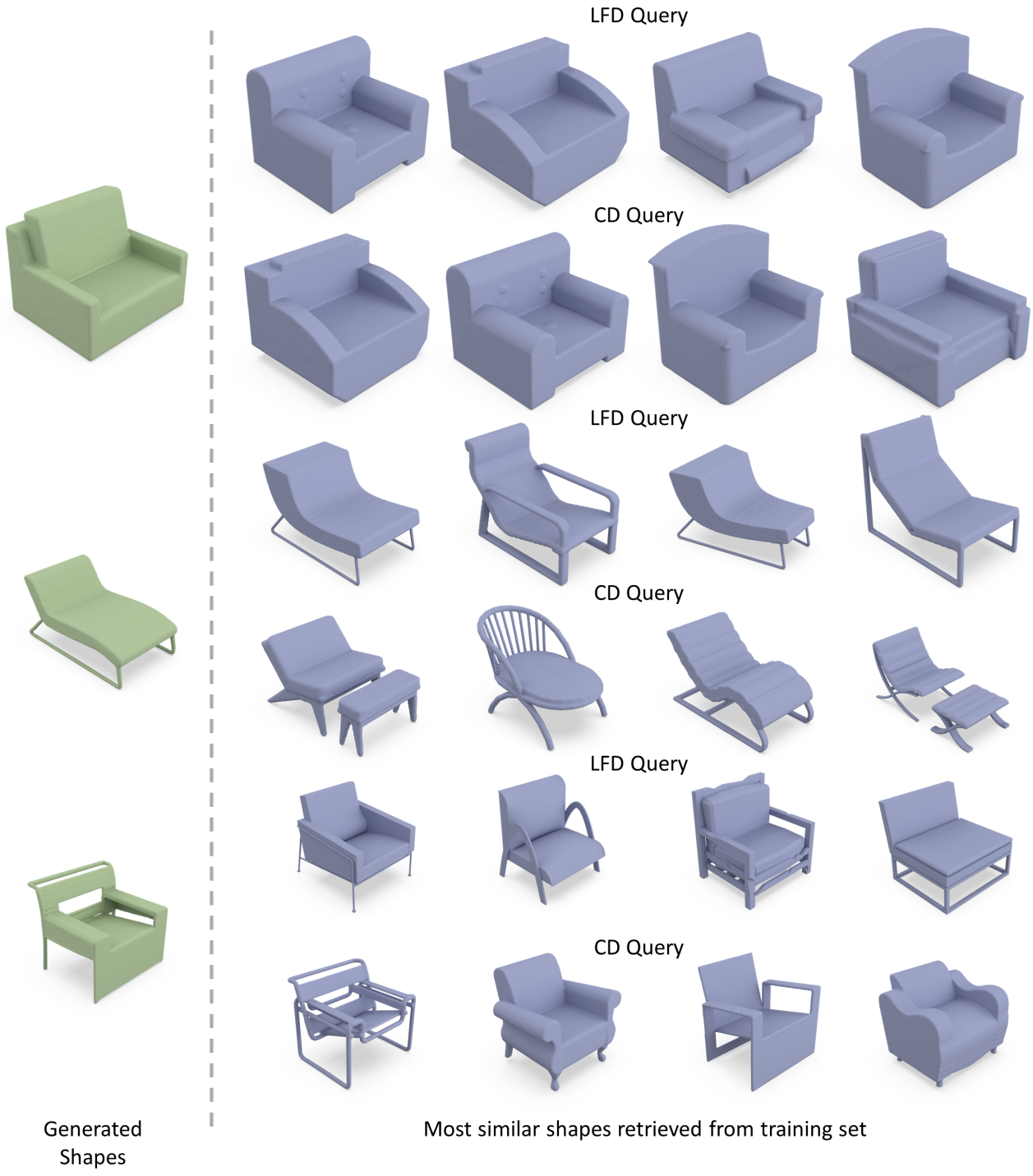

Figure 10. Shape novelty analysis. For each random shape (left) generated by our method, we retrieve top-four most similar shapes in the training set by CD and LFD (cont.).

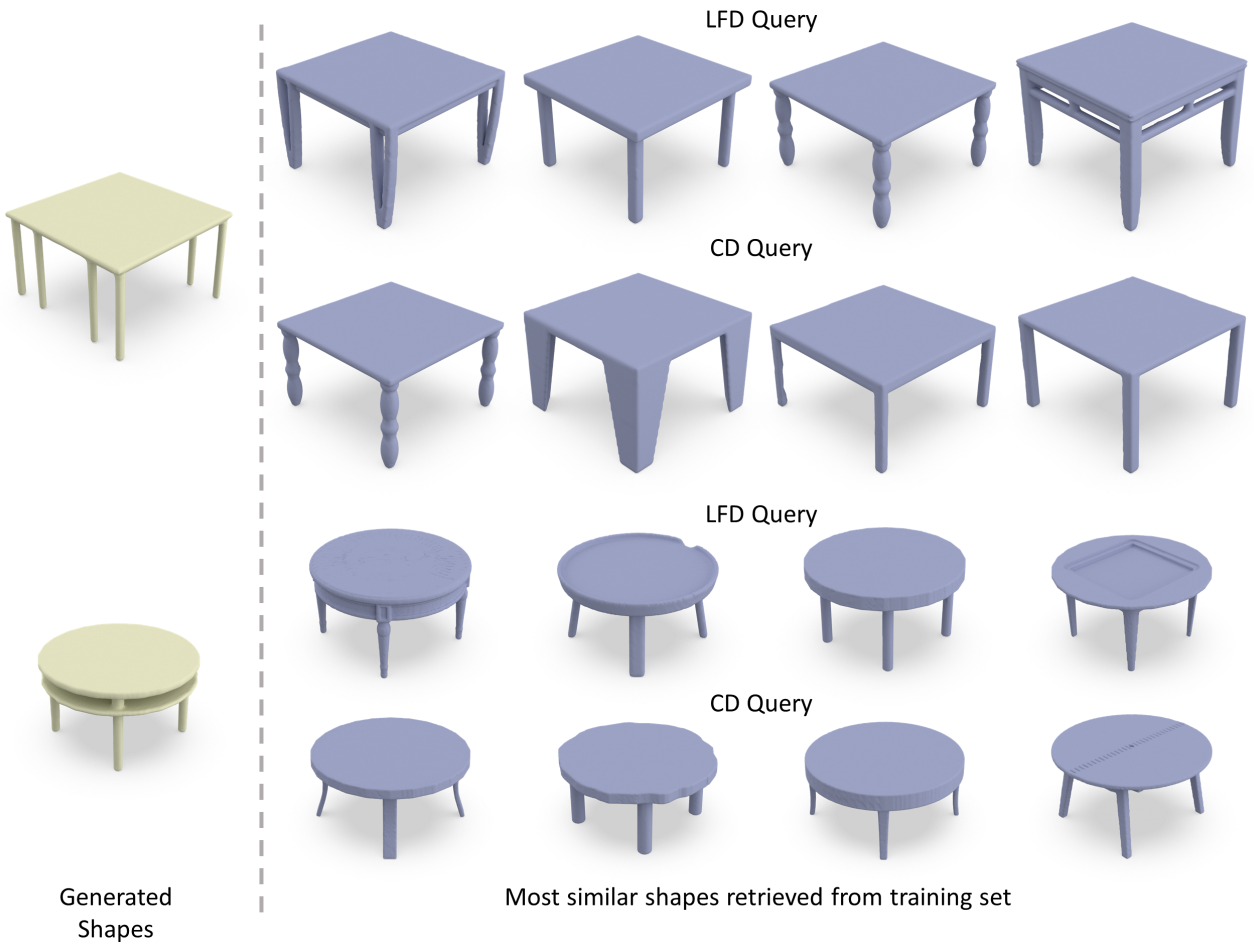

Figure 11. Shape novelty analysis. For each random shape (left) generated by our method, we retrieve top-four most similar shapes in the training set by CD and LFD (cont.).

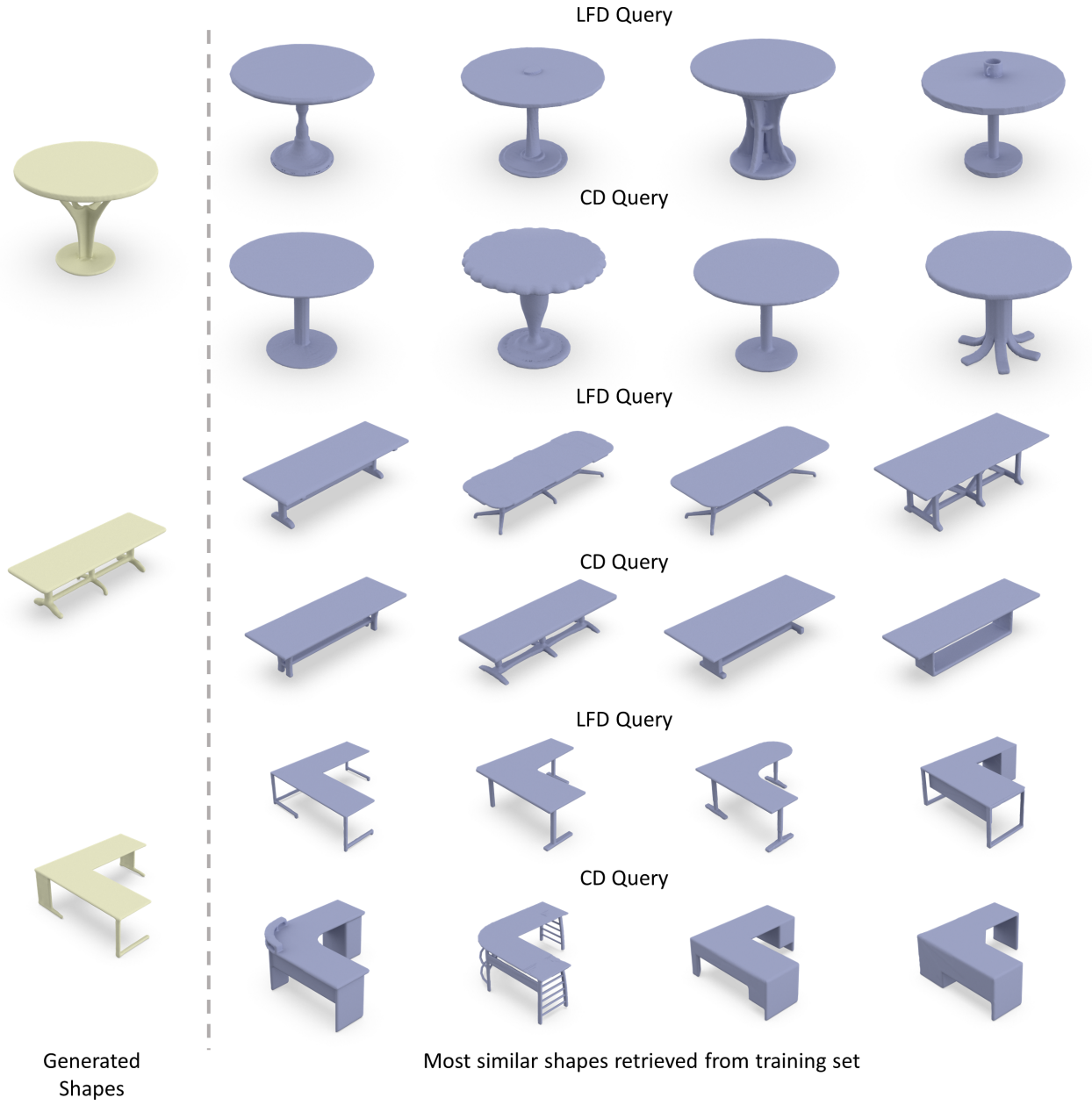

Figure 12. Shape novelty analysis. For each random shape (left) generated by our method, we retrieve top-four most similar shapes in the training set by CD and LFD (cont.).

F. Distance Metrics employed in Shape Novelty Analysis

In this section, we will give the details of the distance metrics employed in the shape novelty analysis, *i.e.*, Chamfer Distance (CD) and Light Field Distance (LFD).

- For CD, we first uniformly sample $2,048$ points on each generated shape and also on each training shape. Then, the CD metric can be used to find the top-four shapes in the training set that are most similar to each generated shape.
- For LFD, we uniformly sample 20 viewpoints to render images of each training shape and also each generated shape. For each image, we first compute the region shape descriptors and contour shape descriptors, and concatenate them to obtain a 45-dimensional feature vector per image. By then, the LFD between two shapes can be computed by summing up all pairwise L1 distances between the image feature vectors from the same viewpoint. As before, we take the training shape with the lowest LFD as the most similar one. Since LFD is defined in the image domain, it emphasizes visual similarity between shapes, making it more robust to shape retrieval.

G. Details on Wavelet Decomposition

In this section, we provide the details of the data preparation process via the wavelet decomposition. We will first introduce the theoretical framework of the wavelet decomposition, followed by the implementation details. For simplicity, we will focus on the one-dimensional function to develop the theorem, and the theory is mainly based on Velho et al. [1994] with some adaptions to our framework.

In general, wavelet decomposition employs a set of filter functions localized in space and scale, so that we can represent an arbitrary function as a linear combination of these filter functions. This family of filter functions can be constructed by scaling and translating a single function $\phi(x) \in L^2(R)$, which is called the scaling function. These functions can form a nested sequence of subspaces $\{V_j\}$, where j is the scale index of one subspace. The subspaces $\{V_j\}$ are nested if they satisfy the following property:

$$\cdots V_{-1} \supset V_0 \supset V_1 \cdots .$$

By varying the translation parameter k, we can define the basis functions for one subspace V_j as

$$\phi_{j,k}(x) = \frac{1}{\sqrt{2^j}} \phi(\frac{x}{2^j} - k). \tag{1}$$

Since subspaces $\{V_j\}$ are nested, *i.e.*, $V_{j+1} \subset V_j$, we can define another sequence of subspaces $\{W_j\}$ so that W_{j+1} is the orthogonal complement of V_{j+1} in V_j, *i.e.*, $V_{j+1} \oplus W_{j+1} = V_j$ and $V_{j+1} \perp W_{j+1}$. The basis functions of W_{j+1} are named the wavelet functions, denoted as $\psi_{j+1,k}$. Any function S in V_j can then be decomposed into a linear combination of two sets of basis functions:

$$S(x) = \sum_k C_{j+1,k} \phi_{j+1,k}(x) + \sum_k D_{j+1,k} \psi_{j+1,k}(x), \tag{2}$$

where the coefficients $C_{j+1,k}$ and $D_{j+1,k}$ can be computed by an orthogonal projection of S on the dual of the basis functions $\phi_{j+1,k}$ and $\psi_{j+1,k}$, respectively. By considering C_j as a function with a spatial input k, we can then apply the decomposition on C_j repeatedly for J times with $C_0 = S$. This will result in $J+1$ coefficients, *i.e.*, $\{C_J, D_J, ..., D_1\}$, where C_j and D_j are the coarse coefficient and the detail coefficient, respectively. Since W_j is orthogonal to V_l, with $l \geq j$, the detail coefficient D_j can extract the detail information of that scale. Lastly, since we know that $V_{j+1} \oplus W_{j+1} = V_j$, the inverse transform can be done without loss of information by projecting ϕ_{j+1} and ψ_{j+1} into ϕ_j.

In our framework, we can directly extend the above formulations to our 3D function (*i.e.*, TSDF) by applying Equation 2 independently to each dimension. Each decomposition step will result in one coarse coefficient volume and seven detail coefficient volumes. Instead of storing seven detail coefficient volumes, we adopt the efficient computation procedure suggested in Velho et al. [1994] to project detail coefficient volume D_{j+1} into V_j as a single coefficient volume. By applying the decomposition J times, we can obtain $J+1$ coefficient volumes to represent the original sampled TSDF.

Since the decomposition in Equation 2 involves a projection of a function onto the translated basis functions ϕ_{j+1} and ψ_{j+1}, we can implement the procedure as convolution operations with two filters. In particular, we adapt the 2D implementation provided in Cotter [2020] in our case using the PyTorch convolution operations. Further, the wavelet decomposition theory is proposed for the real-value functions, yet our sampled TSDF volume has a limited domain. Therefore, extra paddings on the domain's boundary are required to apply the wavelet decomposition in a finite domain. Therefore, we scale the mesh in the range of $[-0.45, +0.45]^3$ rather than the complete input domain ($[-0.5, +0.5]^3$) to avoid boundary artifacts during the reconstruction of TSDF.

H. Training Objectives Derivations

H.1. Training procedure of Diffusion model

As described in the main paper, we have a forward process (denoted as $q(C_{0:T})$) that progressively adds noises to corrupt C_0 into a random noise volume and a backward process (denoted as $p_\theta(C_{0:T})$) that iteratively removes noise from C_T by the generator network (with network parameter θ). Their associated transition probabilistic distribution functions can be parameterized in the following closed forms:

$$q(C_t|C_{t-1}) = \mathcal{N}(C_t; \sqrt{1-\beta_t}C_{t-1}, \beta_t\mathbf{I}) \tag{3}$$

$$p_\theta(C_{t-1}|C_t) = \mathcal{N}(C_{t-1}; \mu_\theta(C_t, t), \sigma_t\mathbf{I}), \tag{4}$$

where $\{\beta_t\}_{t=1}^T$ are hyperparameters; μ_θ is the mean of C_{t-1} estimated based on C_t and t; and σ_t is empirically set based on β_t.

Importantly, we model μ_θ using a neural network, *i.e.*, our generator network. Here, the generator network takes the C_t volume and t as the inputs to predict a 3D volume, in which each voxel value is the voxel's mean value of the sampling distribution for C_{t-1}. With this setting, we can produce the C_{t-1} volume at time step $t-1$ from C_t using a reparameterization of the Gaussian distribution:

$$C_{t-1} = \mu_\theta(C_t, t) + \sigma_t\epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{5}$$

where ϵ is a noise volume produced by sampling each voxel value from the unit Gaussian distribution independently.

Noise contamination. On the other hand, by integrating $C_1, ..., C_{t-1}$ over Equation (3) and making use of the fact that all distributions are Gaussian, we can obtain C_t at any time step t given target C_0:

$$C_t = \sqrt{\bar{\alpha}_t}C_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, \epsilon \sim \mathcal{N}(0, \mathbf{I}), \tag{6}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^t \alpha_s$. With Equation (6), we can directly produce C_t from C_0. Such C_t is a noise-contaminated volume for training the generator network.

Furthermore, by the Bayes' rule, the forward-process posterior distribution can be computed analytically as

$$q(C_{t-1}|C_t, C_0) = \mathcal{N}(C_{t-1}; \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}C_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}C_t, \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t\mathbf{I}). \tag{7}$$

Training the generator network. To train the generator network, we take C_i^3 from different input shapes in training set as C_0. Then, we randomize time step $t \in [1, ..., T]$ and apply Equation (6) to produce a noise-contaminated coefficient volume C_t from C_0.

Given C_t, we can apply the generator network to produce μ_θ at time step $t-1$ from C_t and produce probability distribution $p_\theta(C_{t-1}|C_t)$ using Equation (4). Also, given C_t and C_0, we can analytically evaluate the forward-process posterior distribution $q(C_{t-1}|C_t, C_0)$ using Equation (7). In the training process, we should align these two distributions $p_\theta(C_{t-1}|C_t)$ and $q(C_{t-1}|C_t, C_0)$. To do so, we adopt the KL divergence to regularize the statistical distance between them. Since both distributions are Gaussian, the KL divergence can be simplified into a mean-squares loss between the mean parameters of the two distributions.

Furthermore, Ho et al. [2020] suggests that we can further parameterize the mean estimator $\mu_\theta(C_t, t)$ as $\frac{1}{\sqrt{\alpha_t}}(C_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(C_t, t))$, so we can use our generator network to predict the ϵ_θ noise volume, instead of μ_θ. This can cancel out the terms in the mean-squares loss, resulting in a simpler objective adopted in our framework:

$$L_2 = E_{t,C_0,\epsilon}[\|\epsilon - \epsilon_\theta(C_t, t)\|^2], \epsilon \sim \mathcal{N}(0, \mathbf{I}). \tag{8}$$

H.2. Formulas derivations.

In the followings, we further present some derivations for the above formulas. We provide the details for completeness, and this part is mainly based on Ho et al. [2020].

Forward sampling. First, we introduce the derivation for the closed-form expression used in the forward process (Equation 6) based on the reparameterization trick of the Gaussian distribution.

$$\begin{aligned} C_t &= \sqrt{\alpha_t} C_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}; \epsilon_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= \sqrt{\alpha_t \alpha_{t-1}} C_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\epsilon_{t-2}; \text{where } \epsilon_{t-2} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ &= ... \\ &= \sqrt{\bar{\alpha}_t} C_0 + \sqrt{1-\bar{\alpha}_t}\epsilon; \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \end{aligned} \tag{9}$$

Forward-process posterior distribution. We can first denote the forward-process posterior $q(C_{t-1}|C_t, C_0)$ in Equation 7 as

$$q(C_{t-1}|C_t, C_0) = \mathcal{N}(C_{t-1}; \bar{\mu}_t(C_t, C_0), \bar{\beta}_t \mathbf{I}),$$

By using the Bayes' rule, it can be factorized by

$$\begin{aligned} q(C_{t-1}|C_t, C_0) &= q(C_t|C_{t-1}, C_0)\frac{q(C_{t-1}|C_0)}{q(C_t|C_0)} \\ &\propto exp(-\frac{1}{2}(\frac{(C_t - \sqrt{\alpha_t}C_{t-1})^2}{\beta_t} + \frac{(C_{t-1} - \sqrt{\bar{\alpha}_{t-1}}C_0)^2}{1-\bar{\alpha}_{t-1}} - \frac{(C_t - \sqrt{\bar{\alpha}_t}C_0)^2}{1-\bar{\alpha}_t})) \\ &= exp(-\frac{1}{2}((\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})C_{t-1}^2 - (2\frac{\sqrt{\alpha_t}}{\beta_t}C_t + \frac{2\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}C_0)C_{t-1} + F(C_t, C_0))), \end{aligned}$$

where $F(C_t, C_0)$ is a function not involving C_{t-1}, so we can get the followings by comparing the above with the standard Gaussian density function:

$$\bar{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

$$\begin{aligned} \bar{\mu}_t(C_t, C_0) &= \frac{(\frac{\sqrt{\alpha_t}}{\beta_t}C_t + \frac{\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_{t-1}}C_0)}{(\frac{\alpha_t}{\beta_t} + \frac{1}{1-\bar{\alpha}_{t-1}})} \\ &= \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}C_0 + \frac{\sqrt{\alpha_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}C_t. \end{aligned}$$

Derivation of the loss function. Lastly, we show that the optimization of the target generation probability $p_\theta(C_0)$ is related to aligning two distributions with the KL divergence, *i.e.*, $p_\theta(C_{t-1}|C_t)$ and $q(C_{t-1}|C_t, C_0)$, and the final objective can be further simplified into a mean-squares loss.

Direct optimization of $p_\theta(C_0)$ is generally intractable, so we optimize the variational lower bound as in the Variational Autoencoder (VAE) framework.

$$\begin{aligned}
-logp_\theta(C_0) &\leqslant -logp_\theta(C_0) + D_{KL}(q(C_{1:T}|C_0)||p_\theta(C_{1:T}|C_0)) \\
&= -logp_\theta(C_0) + E_{C_{1:T}\sim q(C_{1:T}|C_0)}[log\frac{q(C_{1:T}|C_0)}{p_\theta(C_{0:T})/p_\theta(C_0)}] \\
&= -logp_\theta(C_0) + E_{C_{1:T}\sim q(C_{1:T}|C_0)}[log\frac{q(C_{1:T}|C_0)}{p_\theta(C_{0:T})} + logp_\theta(C_0)] \\
&= E_{C_{1:T}\sim q(C_{1:T}|C_0)}[log\frac{q(C_{1:T}|C_0)}{p_\theta(C_{0:T})}]
\end{aligned}$$

Then, we can denote L_{VLB} as

$$L_{VLB} = E_{q(C_{0:T})}[log\frac{q(C_{1:T}|C_0)}{p_\theta(C_{0:T}))}] \geqslant -E_{q(C_0)}logp_\theta(C_0).$$

The loss can then be decomposed into different terms as follow:

$$\begin{aligned}
L_{VLB} &= E_{q(C_{0:T})}[log\frac{q(C_{1:T}|C_0)}{p_\theta(C_{0:T}))}] \\
&= E_{q(C_{0:T})}[log\frac{\prod_{t=1}^{T} q(C_t|C_{t-1})}{p_\theta(C_T)\prod_{t=1}^{T} p_\theta(C_{t-1}|C_t)}] \\
&= E_{q(C_{0:T})}[-logp_\theta(C_T) + \sum_{t=1}^{T} log\frac{q(C_t|C_{t-1})}{p_\theta(C_{t-1}|C_t)}] \\
&= E_{q(C_{0:T})}[-logp_\theta(C_T) + \sum_{t=2}^{T} log\frac{q(C_t|C_{t-1})}{p_\theta(C_{t-1}|C_t)} + log\frac{q(C_1|C_0)}{p_\theta(C_0|C_1)}] \\
&= E_{q(C_{0:T})}[-logp_\theta(C_T) + \sum_{t=2}^{T} log\frac{q(C_{t-1}|C_t, C_0)}{p_\theta(C_{t-1}|C_t)}\frac{q(C_t|C_0)}{q(C_{t-1}|C_0)} + log\frac{q(C_1|C_0)}{p_\theta(C_0|C_1)}] \\
&= E_{q(C_{0:T})}[-logp_\theta(C_T) + \sum_{t=2}^{T} log\frac{q(C_{t-1}|C_t, C_0)}{p_\theta(C_{t-1}|C_t)} + \sum_{t=2}^{T} log\frac{q(C_t|C_0)}{q(C_{t-1}|C_0)} + log\frac{q(C_1|C_0)}{p_\theta(C_0|C_1)}] \\
&= E_{q(C_{0:T})}[-logp_\theta(C_T) + \sum_{t=2}^{T} log\frac{q(C_{t-1}|C_t, C_0)}{p_\theta(C_{t-1}|C_t)} + log\frac{q(C_T|C_0)}{q(C_1|C_0)} + log\frac{q(C_1|C_0)}{p_\theta(C_0|C_1)}] \\
&= E_{q(C_{0:T})}[log\frac{q(C_T|C_0)}{p_\theta(C_T)} + \sum_{t=2}^{T} log\frac{q(C_{t-1}|C_t, C_0)}{p_\theta(C_{t-1}|C_t)} - logp_\theta(C_0|C_1)] \\
&= E_{q(C_{0:T})}[D_{KL}(q(C_T|C_0)||p_\theta(C_T)) + \sum_{t=2}^{T} D_{KL}(q(C_{t-1}|C_t, C_0)||p_\theta(C_{t-1}|C_t)) - logp_\theta(C_0|C_1)].
\end{aligned}$$

We can observe that most terms in L_{VLB} are for aligning two distributions, *i.e.*, $p_\theta(C_{t-1}|C_t)$ and $q(C_{t-1}|C_t, C_0)$, as described in the training procedure.

By further parameterizing the mean estimator $\mu_\theta(C_t, t)$ as $\frac{1}{\sqrt{\alpha_t}}(C_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(C_t, t))$, the KL divergence is reduced to minimize the difference of the mean parameters:

$$\begin{aligned} L_t &= E_{C_0,\epsilon_t}[\frac{1}{2||\Sigma_\theta(C_t,t)||^2}||\bar{\mu}_t(C_t, C_0) - \mu_\theta(C_t, t)||^2] \\ &= E_{C_0,\epsilon_t}[\frac{\beta_t^2}{2\alpha_t(1-\bar{\alpha}_t)||\Sigma_\theta||^2}||\epsilon_t - \epsilon_\theta(C_t, t)||^2]]. \end{aligned}$$

Empirically, Ho et al. [2020] find that the following simplified objective works better for training by dropping the weights:

$$L_t^{simple} = E_{C_0,\epsilon_t}[||\epsilon_t - \epsilon_\theta(C_t, t)||^2].$$

I. Details on Quantitative Evaluation Metrics (MMD, COV, and 1-NNA)

In this section, we present additional details on the quantitative evaluation. We will first describe details on each evaluation metric, followed by the implementation details.

I.1. Details on Evaluation Metrics

We first denote the generated shape set as S_g and the testing reference shape set as S_r. For each shape in the sets, we first convert it to a point cloud via sampling on the surface, except for the shapes by Point-Diff Luo and Hu [2021], which are already in the form of point clouds. Then, the computation of Minimum Matching Distance (MMD), Coverage (COV), and 1-NN classifier Accuracy (1-NNA) are defined as in Yang et al. [2019]:

- Minimum Matching Distance (MMD) computes the shortest distance from each referenced point cloud to its nearest neighbor in the generated set. In particular, the metric can be evaluated as

$$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X, Y). \tag{10}$$

- Coverage (COV) measures the fraction of shapes in the reference set that are matched to at least one of the shapes in the generated set. It can be computed as

$$\text{COV}(S_g, S_r) = \frac{|\{\text{argmin}_{Y \in S_r} D(X, Y) | X \in S_g\}|}{|S_r|}. \tag{11}$$

- 1-NN classifier Accuracy (1-NNA) classifies each shape whether its closest neighbor is from S_g or S_r. We first denote the closest neighbor of a shape X except itself as N_X, which is $\text{argmin}_{Y \in S}\{D(X, Y) | Y \in S\}$, and S is the union of S_g and S_r except for X. The metric can be computed as

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} 1[N_X \in S_g] + \sum_{Y \in S_r} 1[N_Y \in S_r]}{|S_g| + |S_r|}. \tag{12}$$

It is noted that the above metrics require a distance measure $D(\cdot, \cdot)$, where we adopt either the chamfer distance (CD) or earth mover's distance (EMD) in this work.

I.2. Other Implementation Details

In the evaluation, we first generate 2000 shapes from each evaluated method. Then, we normalize each of the generated shapes and reference shapes to fit a cube in the range of $[-1, 1]$, and sample 2048 points from the shape's surface. We use the whole testing set provided in Chen and Zhang [2019] as S_r, which is not used in our training procedure. It is noted that we use the whole S_g for MMD and COV, while using first $|S_r|$ of the generated shapes for 1-NNA, since $|S_r| = |S_g|$ is required for the computation of 1-NNA.

J. Ablation Study

J.1. Ablation Cases Setting

To evaluate the effectiveness of the major components in our method, we conduct an ablation study by successively changing our full pipeline in two steps. In the first step, we evaluate the performance of our generation results with/without the detail predictor. If the detail predictor is absent, we set the voxel values of the detail coefficient volume to be zero such that we can still perform inverse wavelet transform to produce a TSDF volume . In the next step, we focus on the generator network (without the detail predictor). We consider two independent ablation cases in the second step: (i) replace the diffusion model with a Variational Autodecoder (VAD) Park et al. [2019]; Hertz et al. [2022], which generates the coarse coefficient volume from a randomly sampled latent code; this substep 2.1 evaluates the capabilities of the diffusion model over a conventional generative model; and (ii) substep 2.2 uses a diffusion model to produce the TSDF volume directly in the same resolution as our coarse coefficient volume, without using our wavelet representation.

In each of the three ablation cases above, we re-trained a network model and tested its performance on the Chair category. It is noted that we conducted the last two ablation cases (substeps 2.1 & 2.2) without the detail predictor (step 1) because the replacement of the diffusion model in substep 2.1 has already shown a significant drop in performance, which is unlikely to be recovered by further predicting the detail coefficient volume. Also, there are no trivial ways to extract the detail information for TSDF without using our wavelet representation in substep 2.2.

J.2. Visual Ablation Comparisons

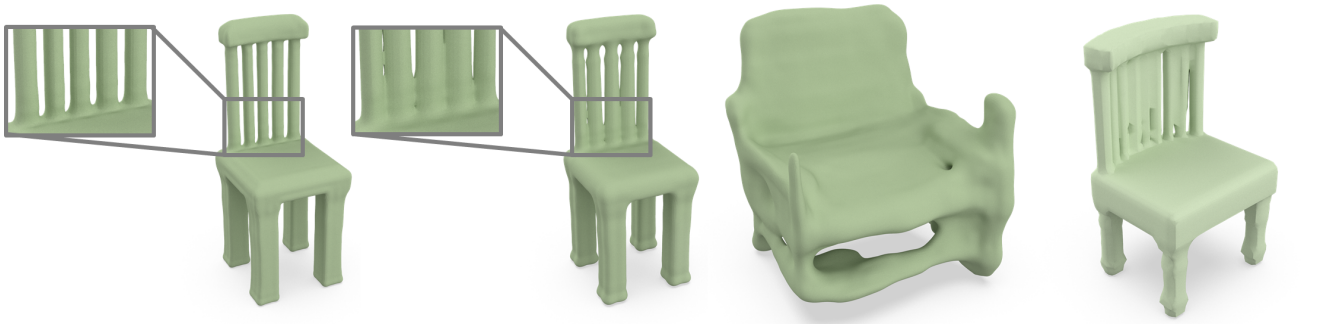

Figure 13. Visual ablation results generated by (a) our full pipeline; (b) removing the detail predictor; (c) replacing the diffusion model with the VAD; and (d) directly predicting the TSDF without the wavelet representation.

Figure 13 shows some visual ablation results. Comparing (a) & (b) reveals that fine details and thin structures are deprived after removing the detail predictor. Further, we provide additional visual ablation results in Figure 14 (a) & (b), showing that the detail predictor mainly accounts for the geometric surface details on the cabinets. Second, replacing the diffusion model with the VAD causes a substantial drop in performance, as shown in Figure 13 (c). Last, comparing (b) & (d) shows that our compact wavelet representation helps to promote more effective shape learning and generation than directly predicting the TSDF, as the grid resolution limits the explicit mesh resolution and the production of fine details. Note that the quantitative evaluation results for the ablation cases are shown in the main paper.

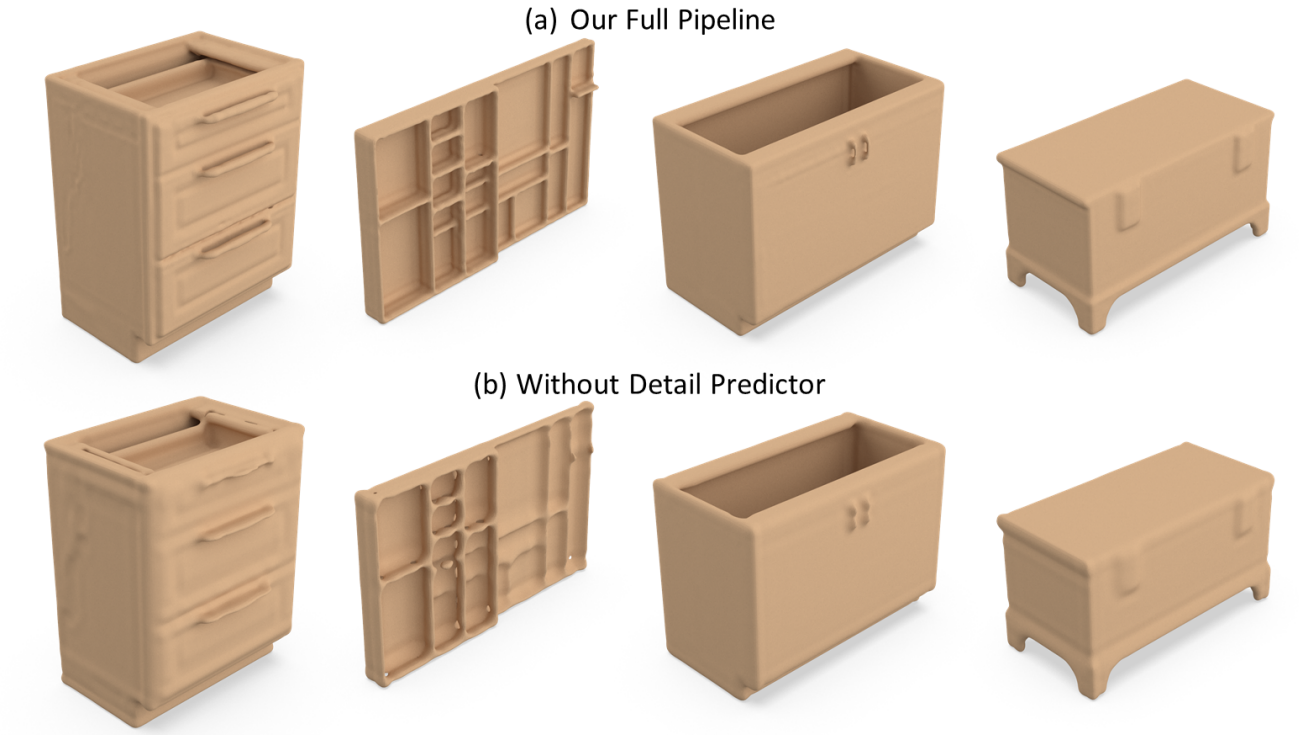

Figure 14. More visual ablation results on the cabinet category. (a) our full pipeline; (b) removing the detail predictor.

K. Discussion on Limitations

(a) Unmet Functionalities (b) Disconnected Complex Parts

Figure 15. Failure cases. Left: the chairs unlikely meet their basic functionality in real world. Right: artifacts may happen in complex and very thin structures.

First, while our method can generate diverse and realistic-looking shapes, the generated shapes may not meet the desired functionality as normal real objects. As Figure 15(a) shows, the generated chairs, despite of looking interesting and structurally reasonable, have an exceptionally tall seat back and low seat height for normal human bodies. In the future, we may incorporate functionality, *e.g.* Blinn et al. [2021], into the shape generation process. Second, although our method is learned efficiently via a wavelet-based representation, it still requires a large number of shapes for training. So, for categories with few training samples, especially those with complex and very thin structures, see, *e.g.*, Figure 15(b), the generated shape may exhibit artifacts like broken parts and structures. Lastly, although our method can better fit the data distribution than existing methods, the generated shapes still conform to the structures or appearances of the training set. Considering that most 3D models can be decomposed into component assemblies, exploring part-aware generation & composition has great potential to generate even more novel 3D shapes.

References

Bryce Blinn, Alexander Ding, Daniel Ritchie, R Kenny Jones, Srinath Sridhar, and Manolis Savva. 2021. Learning Body-Aware 3D Shape Generative Models. *arXiv preprint arXiv:2112.07022* (2021). 23

Zhiqin Chen and Hao Zhang. 2019. Learning implicit fields for generative shape modeling. In *CVPR*. 5939–5948. 8, 9, 20

Fergal Cotter. 2020. *Uses of Complex Wavelets in Deep Convolutional Neural Networks*. Ph.D. Dissertation. University of Cambridge. 15

Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2022. SPAGHETTI: Editing Implicit Shapes Through Part Aware Generation. *arXiv preprint arXiv:2201.13168* (2022). 8, 9, 21

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS*. 6840–6851. 17, 19

Marian Kleineberg, Matthias Fey, and Frank Weichert. 2020. Adversarial generation of continuous implicit shape representations. *arXiv preprint arXiv:2002.00349* (2020). 8, 9

Shitong Luo and Wei Hu. 2021. Diffusion probabilistic models for 3D point cloud generation. In *CVPR*. 2837–2845. 8, 9, 20

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. 2019. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*. 165–174. 21

Luiz Velho, Demetri Terzopoulos, and Jonas Gomes. 1994. Multiscale implicit models. In *Proceedings of SIBGRAPI*, Vol. 94. 93–100. 15

Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. 2019. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*. 4541–4550. 20

The End