Appendix: 3D Shape Generation and Completion through Point-Voxel Diffusion

A. Additional Generation Metrics

We present additional generation metrics in Table 1, following PointFlow [9]. We report coverage (COV), which measures the fraction of point clouds in the reference set that are matched to at least one point cloud in the generated set. We further report minimum matching distance (MMD), which measures for each point cloud in a reference set, the distance to its nearest neighbor in the generated set. Note that these generation metrics can vary depending on implementation and do not necessarily correlate to generation quality, as discussed in [9].

Table 2 includes generation results on Airplane, Chair, Car compared with the voxel-diffusion model, Vox-Diff, as described in the main paper, evaluated using the 1-NN metric. By generating less noisy point clouds, PVD significantly outperforms Vox-Diff.

B. Point Cloud Generation Visualization

We additionally visualize some generation results for Airplane, Car, and Chair in terms of the generation process and the final generated shapes from all angles in Figures 3, 4, 5. 6, 7, 8, 9, and 10.

C. Derivation of the Variational Lower Bound

$$\begin{aligned}
\mathrm{E}_{q(\mathbf{x}_0)}[\log p_\theta(\mathbf{x}_0)] &= \mathrm{E}_{q(\mathbf{x}_0)}\left[\log \int p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T)\, d\mathbf{x}_{1:T}\right] \\
&\geq \mathrm{E}_{q(\mathbf{x}_0)}\left[\int q(\mathbf{x}_1, ..., \mathbf{x}_T|\mathbf{x}_0)\right. \\
&\quad \left.\log \frac{p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T)}{q(\mathbf{x}_1, ..., \mathbf{x}_T|\mathbf{x}_0)}\, d\mathbf{x}_{1:T}\right] \\
&= \mathrm{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{p_\theta(\mathbf{x}_0, ..., \mathbf{x}_T)}{q(\mathbf{x}_1, ..., \mathbf{x}_T|\mathbf{x}_0)}\right],
\end{aligned}$$

where the inequality is by Jensen's inequality.

D. Properties of the Diffusion Model

$\{\beta_0, ..., \beta_T\}$ is a sequence of increasing parameters; $\alpha_t = 1 - \beta_t$ and $\tilde{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Two following properties are crucial to deriving the final $\mathcal{L}_2$ loss.

Property 1. Tractable marginal of the forward process:

$$\begin{aligned}
q(\mathbf{x}_t|\mathbf{x}_0) &= \int q(\mathbf{x}_{1:t}|\mathbf{x}_0)\, d\mathbf{x}_{1:(t-1)} \\
&= \mathcal{N}(\sqrt{\tilde{\alpha}_t}\mathbf{x}_0, (1-\tilde{\alpha}_t)\mathbf{I}).
\end{aligned}$$

This property is proved in the Appendix of [4] and provides convenient closed-form evaluation of $\mathbf{x}_t$ knowing $\mathbf{x}_0$:

$$\mathbf{x}_t = \sqrt{\tilde{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\tilde{\alpha}_t}\boldsymbol{\epsilon}, \tag{1}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Property 2. Tractable posterior of the forward process. We first note the Bayes' rule that connects the posterior with the forward process,

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \frac{q(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_0)q(\mathbf{x}_{t-1}|\mathbf{x}_0)}{q(\mathbf{x}_t|\mathbf{x}_0)}.$$

Since the three probabilities on the right are Gaussian, the posterior is also Gaussian, given by

$$\begin{aligned}
&q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) = \\
&\mathcal{N}\Big(\frac{\sqrt{\tilde{\alpha}_{t-1}}\beta_t}{1-\tilde{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}\mathbf{x}_t, \frac{(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}\beta_t\mathbf{I}\Big).
\end{aligned} \tag{2}$$

E. Derivation of $\mathcal{L}_2$ Loss

We need to match generative transition $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ with ground-truth posterior $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$, both of which are Gaussian with a pre-determined variance schedule $\beta_1, ..., \beta_T$. Therefore, maximum likelihood learning is reduced to simple $\mathcal{L}_2$ loss of the form with two cases:

$$L_t = \begin{cases} \left\| \frac{\sqrt{\tilde{\alpha}_{t-1}}\beta_t}{1-\tilde{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{\alpha_t}(1-\tilde{\alpha}_{t-1})}{1-\tilde{\alpha}_t}\mathbf{x}_t - \mu_\theta(\mathbf{x}_t, t)\right\|^2, & t > 1 \\ \left\|\mathbf{x}_0 - \mu_\theta(\mathbf{x}_t, t)\right\|^2, & t = 1 \end{cases}$$

where $\alpha_t = 1 - \beta_t$ and $\tilde{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The supervision target of case $t > 1$ comes from Eqn. 2. We can Further reduce the case when $t > 1$ by substituting $\mathbf{x}_0$ as an expression of $\mathbf{x}_t$ using Eqn. 1 and arrive at

$$\mathcal{L}_t = \begin{cases} \left\| \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}\right) - \mu_\theta(\mathbf{x}_t, t)\right\|^2, & t > 1 \\ \left\|\mathbf{x}_0 - \mu_\theta(\mathbf{x}_t, t)\right\|^2, & t = 1 \end{cases}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Note that when $t = 1$, $\tilde{\alpha}_1 = \alpha_1$ so that the supervision target of the first case above evaluated at $t = 1$ becomes:

$$\begin{aligned}
\frac{1}{\sqrt{\alpha_1}}\left(\mathbf{x}_1 - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_1}}\boldsymbol{\epsilon}\right) &= \frac{1}{\sqrt{\tilde{\alpha}_1}}\left(\mathbf{x}_1 - \sqrt{1-\tilde{\alpha}_1}\boldsymbol{\epsilon}\right) \\
&= \mathbf{x}_0,
\end{aligned} \tag{3}$$

where the last equality is by rewriting Eqn. 1. Therefore, in fact, the two cases are equivalent.

The final $\mathcal{L}_2$ loss is

$$\mathcal{L}_t = \left\| \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}\right) - \mu_\theta(\mathbf{x}_t, t)\right\|^2.$$

Model	Airplane				Chair				Car			
	MMD↓		COV↑ (%)		MMD↓		COV↑ (%)		MMD↓		COV↑ (%)	
	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD	CD	EMD
r-GAN [1]	0.4471	2.309	30.12	14.32	5.151	8.312	24.27	15.13	1.446	2.133	19.03	6.539
l-GAN (CD) [1]	0.3398	0.5832	38.52	21.23	2.589	2.007	41.99	29.31	1.532	1.226	38.92	23.58
l-GAN (EMD) [1]	0.3967	0.4165	38.27	38.52	2.811	1.619	38.07	44.86	1.408	0.8987	37.78	45.17
PointFlow [9]	0.2243	0.3901	47.90	46.41	**2.409**	1.595	42.90	50.00	**0.9010**	0.8071	46.88	50.00
SoftFlow [5]	0.2309	**0.3745**	46.91	47.90	2.528	1.682	41.39	47.43	1.187	0.8594	42.90	44.60
DPF-Net [6]	0.2642	0.4086	46.17	48.89	2.536	1.632	44.71	48.79	1.129	0.8529	45.74	49.43
Shape-GF [2]	2.703	0.6592	40.74	40.49	2.889	1.702	46.67	48.03	9.232	**0.7558**	**49.43**	50.28
Vox-Diff	1.322	0.5610	11.82	25.43	5.840	2.930	17.52	21.75	5.646	1.551	6.530	22.15
PVD (Ours)	**0.2243**	0.3803	**48.88**	**52.09**	2.622	**1.556**	**49.84**	**50.60**	1.077	0.7938	41.19	**50.56**

Table 1: Additional generation metrics, following PointFlow [9].

	Airplane		Chair		Car	
	CD	EMD	CD	EMD	CD	EMD
Vox-Diff	99.75	98.13	97.12	96.74	99.56	96.83
PVD (ours)	**73.82**	**64.81**	**56.26**	**53.32**	**54.55**	**53.83**

Table 2: Generation results on Airplane, Chair, Car compared with the voxel-diffusion model, Vox-Diff, as described in the main paper, evaluated using the 1-NN metric. By generating less noisy point clouds, PVD significantly outperforms Vox-Diff.

Since $\mathbf{x}_t$ is known when $\mathbf{x}_0$ is known, we can redefine the model output as $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$. Instead of directly predicting $\mu_\theta(\mathbf{x}_t, t)$, we instead predict a noise value $\epsilon_\theta(\mathbf{x}_t, t)$, where

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right). \quad (4)$$

Substituting in $\mu_\theta(\mathbf{x}_t, t)$ into the loss, we can arrive at the final loss

$$\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right\|^2, \ \ \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I}). \quad (5)$$

F. Point Cloud Generation Process

Since the transition mean $\mu_\theta(\mathbf{x}_t, t)$ of $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ is calculated by Eqn. 4, the generative process is performed by progressively sampling from $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as $t = T, ..., 1$:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\tilde{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right) + \sqrt{\beta_t}\mathbf{z}, \quad (6)$$

where $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. This approach is also similar to Langevin dynamics [3, 8] used in energy-based models, since it similarly adds scaled noise outputs from the model to current samples. Specifically, Langevin dynamics is in the following form:

$$\mathbf{x}_{t+1} = \mathbf{x}_t + s\frac{\partial}{\partial \mathbf{x}}\log p_\theta(\mathbf{x}) + \sqrt{2s}\mathbf{z}, \quad (7)$$

where s denotes step size, $p_\theta(\mathbf{x})$ denotes the model distribution, and $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$. Both processes are Markovian, shifting the previous output by a model-dependent term and a noise term. The scaled model output of our model can also be seen as an approximation of gradients of an energy function.

G. Controlled Completion

We show that our model can control the shape completion process in Figure 1. Given a pretrained completion model, our formulation also enables control over completion results through latent interpolation. The figure below shows one such case: we may have used the left depth map to obtain a completed shape. But it comes with an unwanted feature (cavity at the chair's back). We can refine the result by feeding another depth map (shown on the right), with a better view of the back. To retain features from both partial shapes, we can interpolate by $(1-\lambda)\hat{\mathbf{x}}_T + \lambda\hat{\mathbf{y}}_T$ in the latent space at time T, where the latent features are obtained by $\hat{\mathbf{x}}_T = \sqrt{\tilde{\alpha}_T}\hat{\mathbf{x}}_0 + \sqrt{1-\tilde{\alpha}_T}\epsilon$ (see Eqn. 1). Interpolation presents diverse choices and users can actively control how much features are shared by varying λ.

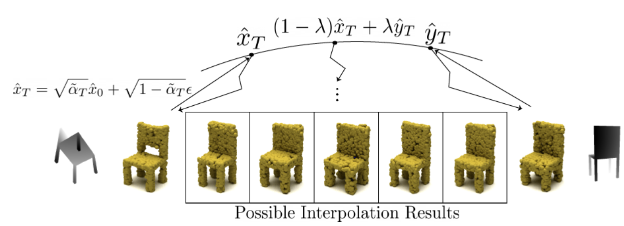

Figure 1: Controlled completion process.

H. Training Details

H.1. Model Architecture

Same as in [7], our point-voxel CNN architecture is modified from PointNet++, where we replace the PointNet substructure with point-voxel convolution, as shown in Figure 2. We specify our architecture in Table 3, Table 4, and and Table 5. Table 3 shows details of a single set abstraction (SA) module. Table 4 shows details of a single feature propagation (FP) module. Table 5 shows how these modules are combined together.

In particular, we concatenate the temporal embeddings with point features before sending input into the Set Abstraction or the Feature Propagation modules. To obtain temporal embeddings, we used a sinusoidal positional embedding, commonly used in Transformers. Given a time t and an embedding dimension d, the time embedding consists of pairs of $\sin$ and $\cos$ with varying frequencies, $(\sin(\omega_1 t), \cos(\omega_1 t), ..., \sin(\omega_{d/2} t), \cos(\omega_{d/2} t))$, where ω_k is $1/\left(10000^{2k/d}\right)$.

We use the same architecture for both generation and completion tasks. For shape completion specifically, the model takes as input a 200-point partial shape and 1,848 points sampled from noise, totaling 2048 points. At each step, the first 200 of the 2,048 points sampled by the model are replaced with the input partial shape. The updated point set is then used as input in the next time step.

H.2. Choices of β_t and T

For both hyper-parameters, we follow [4]. For Car and Chair, we set $\beta_0 = 10^{-4}, \beta_T = 0.01$ and linearly interpolate other β's. For Airplane, we interpolate between $\beta_0 = 10^{-5}$ and $\beta_T = 0.008$ for the first 90% steps and then fix $\beta_T = 0.008$. We also set $T = 1000$ for all experiments and we generally notice that lower timesteps (*e.g.*, 100) are not enough for the model to construct shapes.

H.3. Training Parameters

We use Adam optimizer with learning rate 2×10^{-4} for all experiments.

References

[1] Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3D point clouds. In *ICML*, 2018. 2

[2] Ruojin Cai, Guandao Yang, Hadar Averbuch-Elor, Zekun Hao, Serge Belongie, Noah Snavely, and Bharath Hariharan. Learning gradient fields for shape generation. In *ECCV*, 2020. 2

[3] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *NeurIPS*, 2019. 2

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1, 3

[5] Hyeongju Kim, Hyeonseung Lee, Woo Hyun Kang, Joun Yeop Lee, and Nam Soo Kim. SoftFlow: Probabilistic framework for normalizing flow on manifolds. In *NeurIPS*, 2020. 2

[6] Roman Klokov, Edmond Boyer, and Jakob Verbeek. Discrete point flow networks for efficient point cloud generation. In *ECCV*, 2020. 2

[7] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3D deep learning. In *NeurIPS*, 2019. 3

[8] Erik Nijkamp, Mitch Hill, Song-Chun Zhu, and Ying Nian Wu. Learning non-convergent non-persistent short-run MCMC toward energy-based model. In *NeurIPS*, 2019. 2

[9] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. PointFlow: 3D point cloud generation with continuous normalizing flows. In *ICCV*, 2019. 1, 2

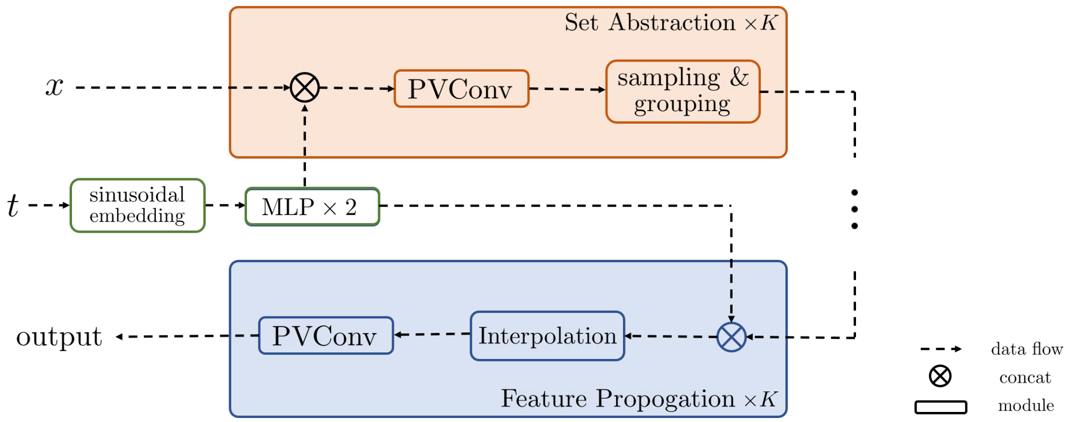

Figure 2: Model architecture diagram.

Set Abstraction		
Input Feature Size: $N_{\text{input}} \times C_{\text{input}}$		
Input Time Embedding Size: $N_{\text{input}} \times E_t$		
Output Feature Size: $N_{\text{output}} \times C_{\text{output}}$		
Voxelization Resolution: D		
Number of Point-Voxel Convolution (PVConv) blocks: L		
Whether to use attention mechanism: use_attn		
PVConv $\times L$		
Layers	In-Out Size	Stride
Input: (X, t)	$(D \times D \times D \times C_{\text{input}}, D \times D \times D \times E_t)$	
Concat(X, t)	-	-
3x3x3 conv(C_{output}), GroupNorm(8), Swish	$D \times D \times D \times C_{\text{output}}$	1
Dropout(0.1)		
3x3x3 conv(C_{output}), GroupNorm(8)	$D \times D \times D \times C_{\text{output}}$	1
Attention(use_attn)	$D \times D \times D \times C_{\text{output}}$	
MLP		
1x1x1 conv(C_{output}), GroupNorm(8), Swish	$D \times D \times D \times C_{\text{output}}$	1
Sampling & Grouping		
Number of Centers: N_{center}		
Grouping Radius: r		
Number of Neighbors: N_{neighbor}		

Table 3: Set Abstraction Layer. Input is first fed through L PVConv modules, then to an MLP module, and finally through the Sampling & Grouping module.

Feature Propagation		
Input Feature Size: $N_{\text{input}} \times C_{\text{input}}$		
Output Feature Size: $N_{\text{output}} \times C_{\text{output}}$		
Voxelization Resolution: D		
Number of Point-Voxel Convolution (PVConv) blocks: L		
Whether to use attention mechanism: use_attn		
Interpolation		
PVConv $\times L$		
Layers	In-Out Size	Stride
Input: X	$D \times D \times D \times C_{\text{input}}$	
3x3x3 conv(C_{output}), GroupNorm(8), Swish	$D \times D \times D \times C_{\text{output}}$	1
Dropout(0.1)		
3x3x3 conv(C_{output}), GroupNorm(8)	$D \times D \times D \times C_{\text{output}}$	1
Attention(use_attn)	$D \times D \times D \times C_{\text{output}}$	
MLP		
3x3x3 conv(C_{output}), GroupNorm(8), Swish	$D \times D \times D \times C_{\text{output}}$	1

Table 4: Feature Propagation Layer. Input is fed through Interpolation module, L PVConv modules, and an MLP module.

Input Feature Size: 2048×3				
Input Time Embedding Size: 64				
Output Feature Size: 2048×3				
Time Embedding				
Sinusoidal Embedding dim = 64 MLP(64, 64) LeakyRelU(0.1) MLP(64, 64)				
	SA 1	SA 2	SA 3	SA 4
L	2	3	3	0
C_{input}	3	32	64	-
E_t	64	64	64	-
C_{output}	32	64	128	-
D	32	16	8	-
use_attn	False	True	False	-
N_{center}	1024	256	64	16
r	0.1	0.2	0.4	0.8
N_{neighbor}	32	32	32	32
	FP 1	FP 2	FP 3	FP 4
L	3	3	2	2
C_{input}	128	256	256	128
C_{output}	256	256	128	64
D	8	8	16	32
use_attn	False	True	False	False
MLP(64,3)				

Table 5: Entire point-voxel CNN architecture. Input point clouds and time steps are sequentially passed through SA 1-4, FP 1-4, and an MLP to obtain output of the same dimension. At the start of each SA and FP module, time embedding and point features are first concatenated.

Figure 3: Airplane generation process.

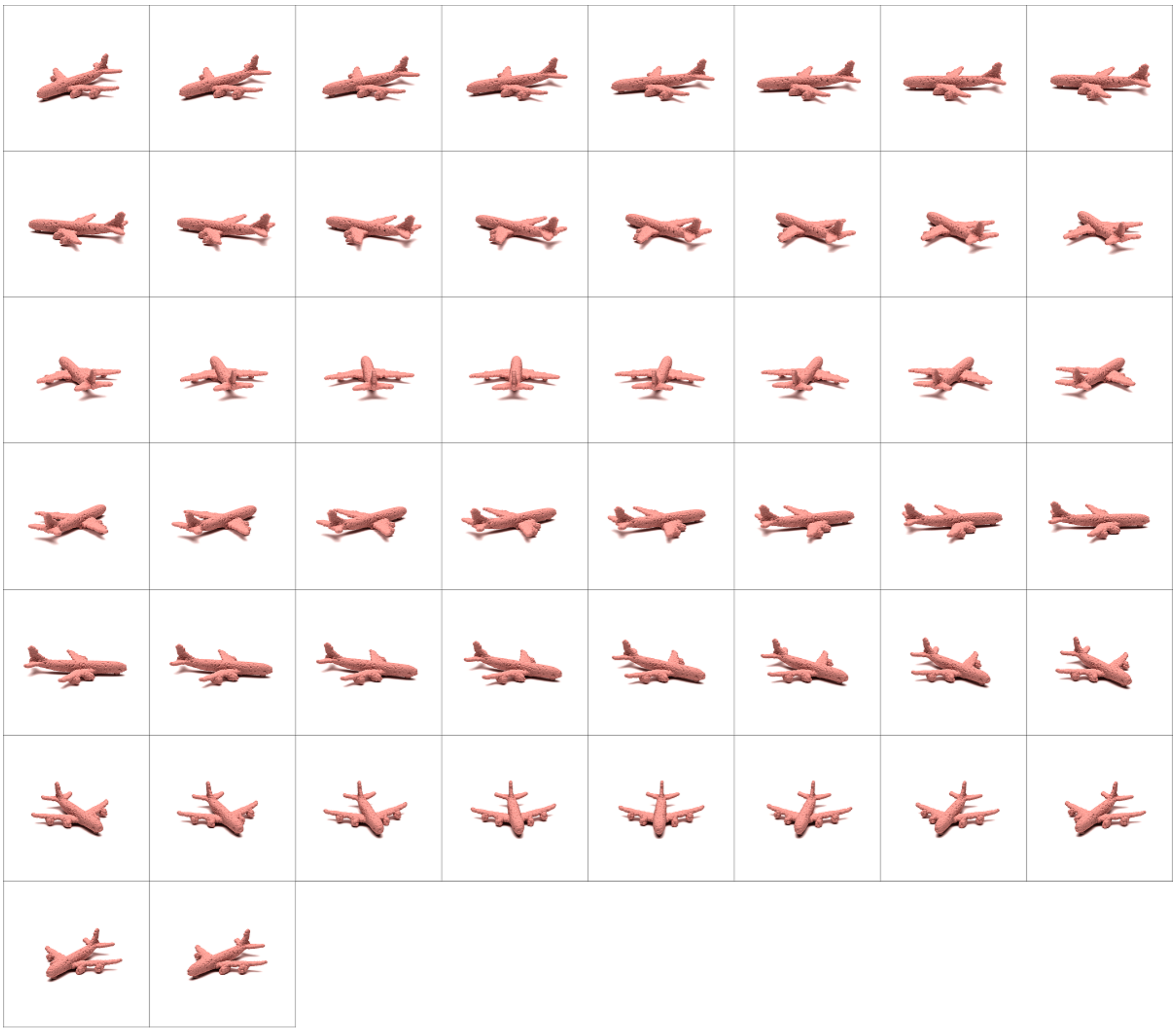

Figure 4: Airplane results from all angles.

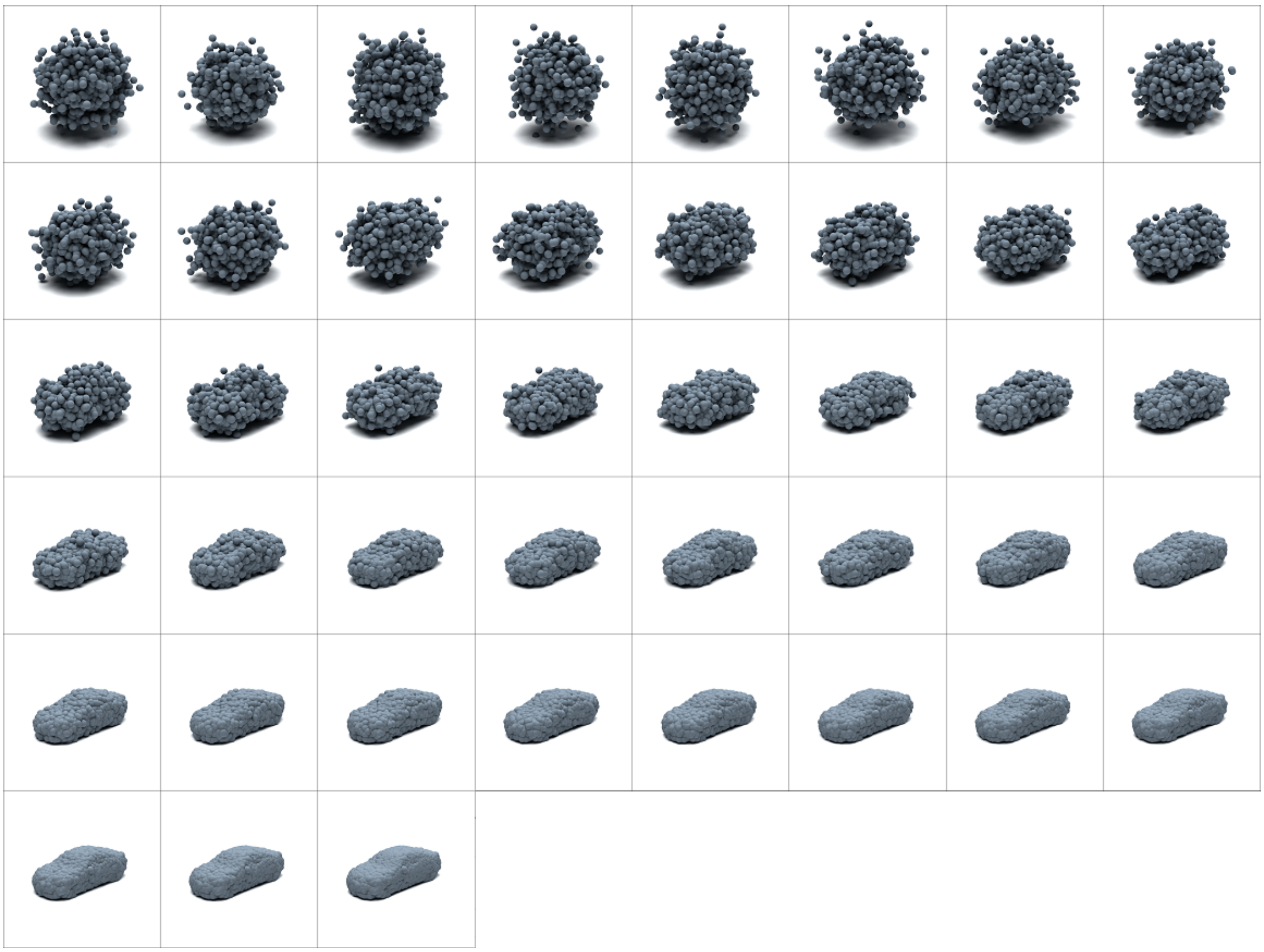

Figure 5: Car generation process.

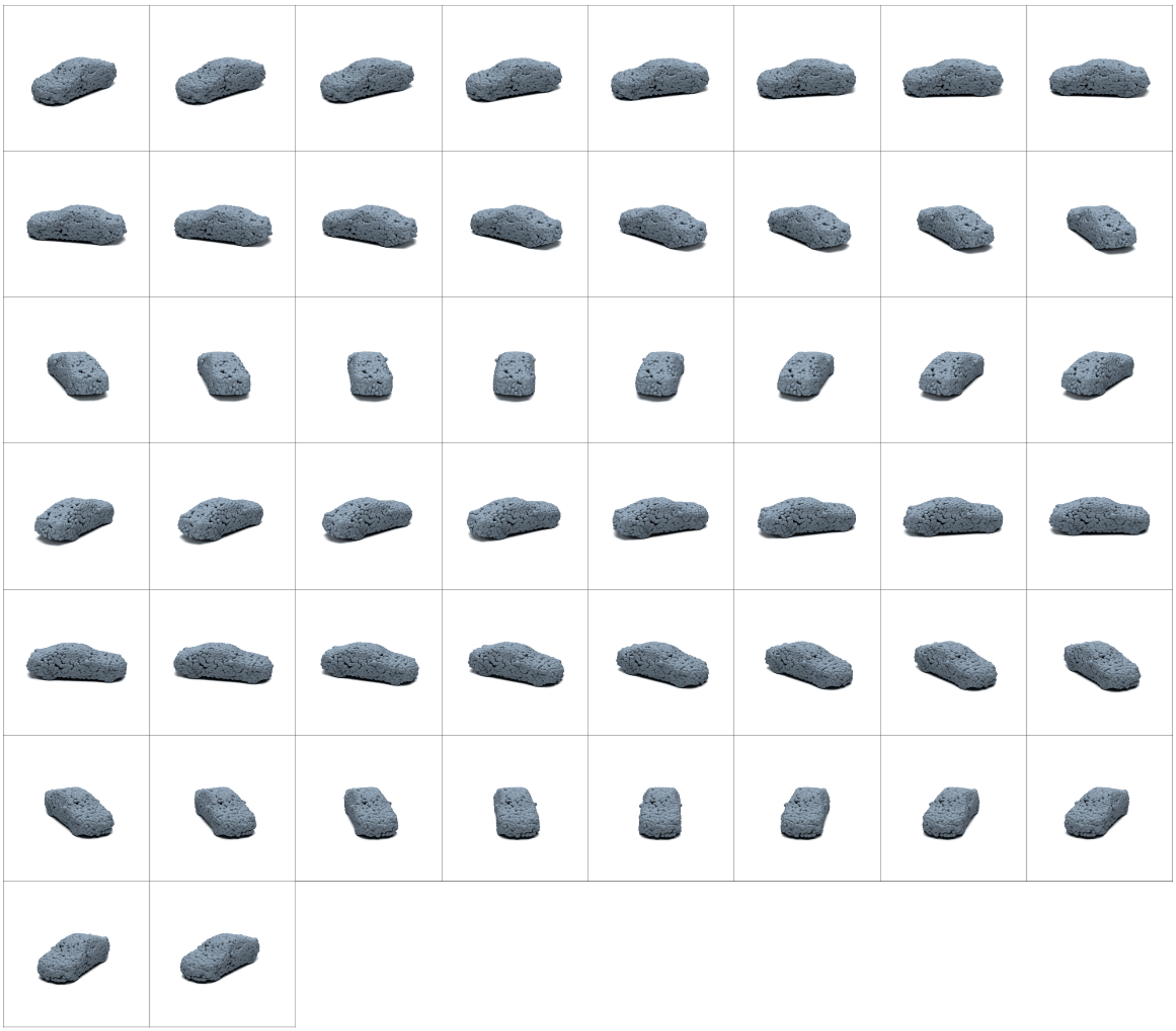

Figure 6: Car results from all angles.

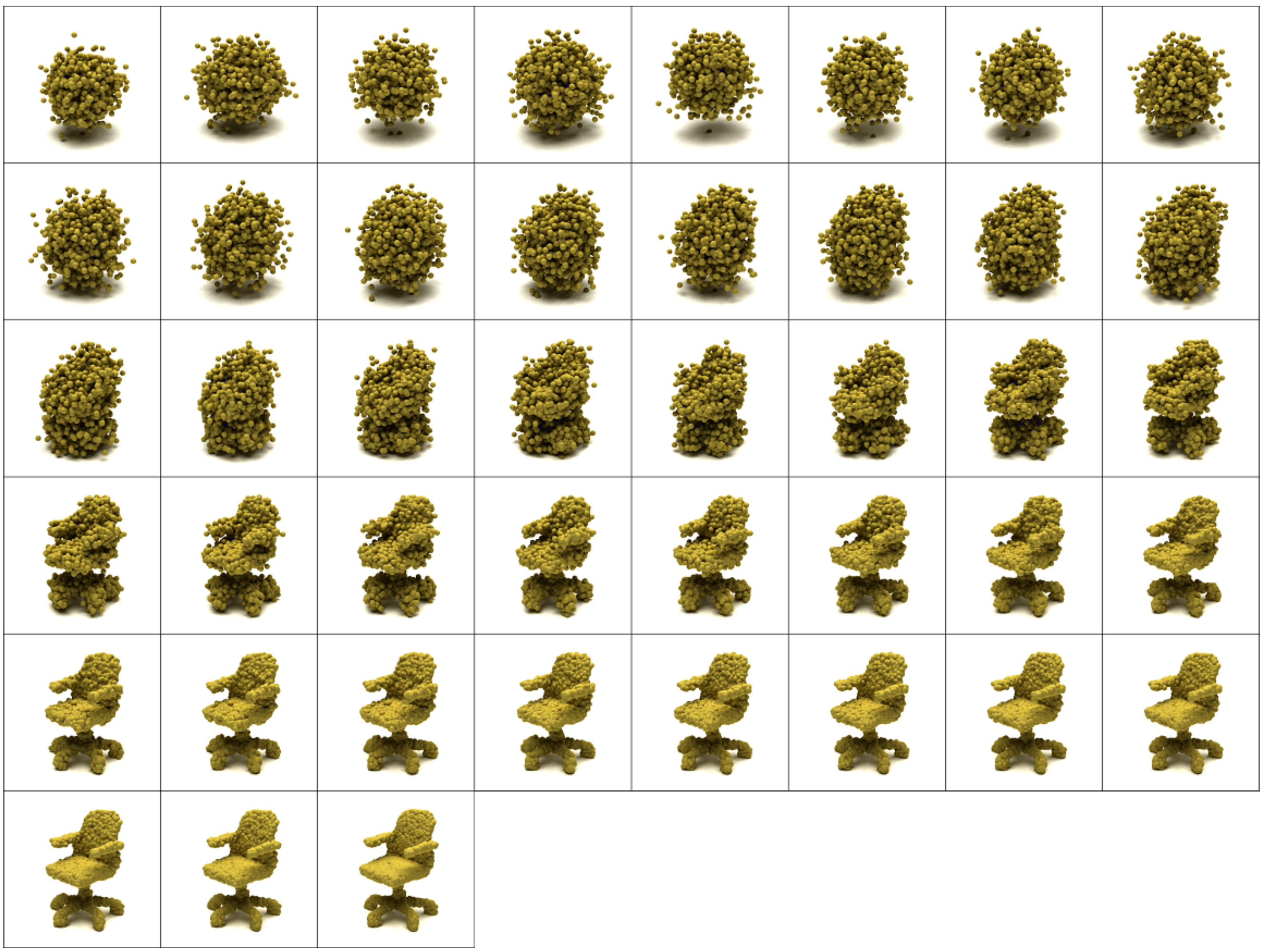

Figure 7: Chair generation process.

Figure 8: Chair results from all angles.

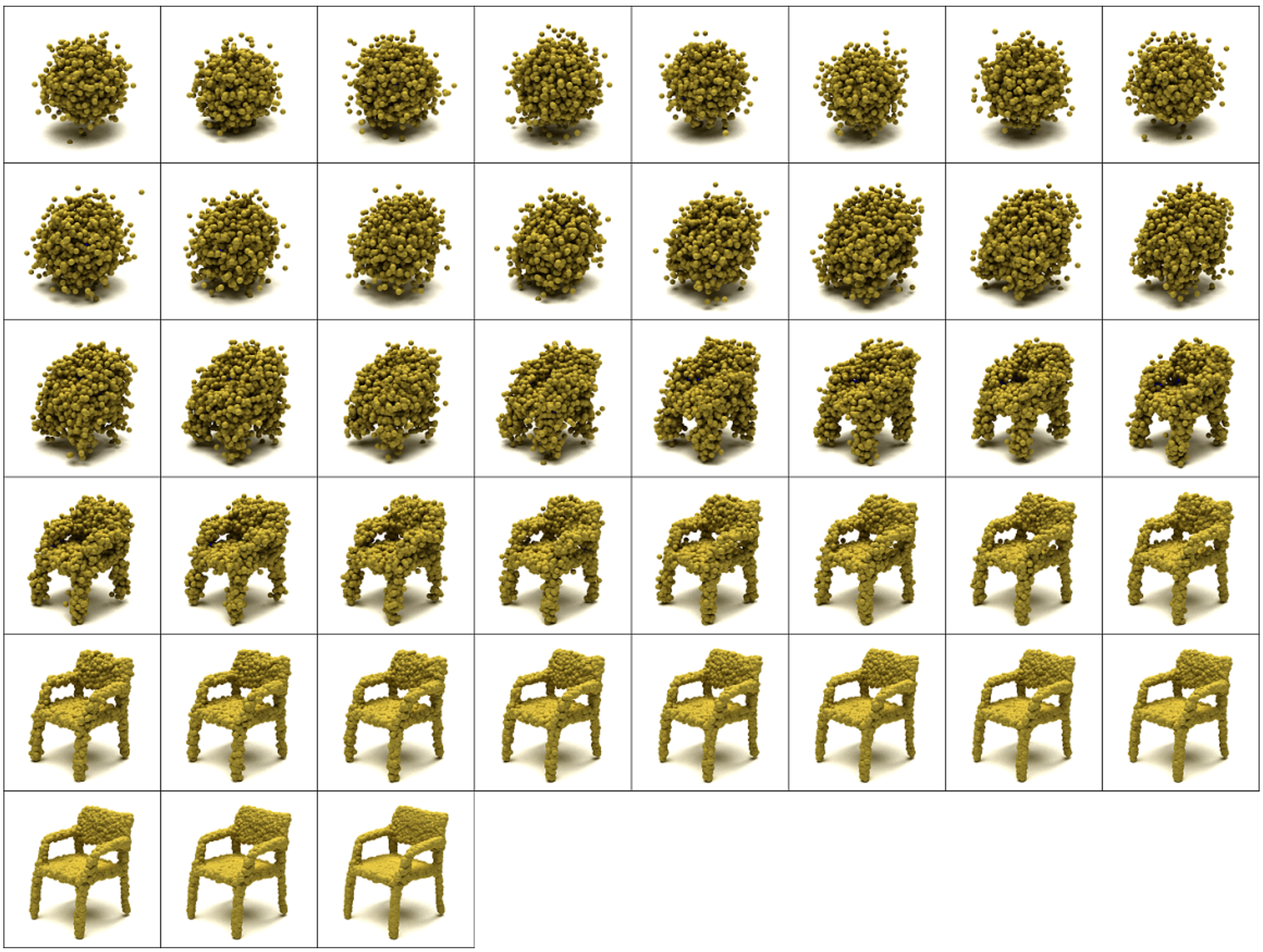

Figure 9: Chair generation process.

Figure 10: Chair results from all angles.