SALAD: Part-Level Latent Diffusion for 3D Shape Generation and Manipulation — Supplementary Material

Juil Koo* Seungwoo Yoo* Minh Hieu Nguyen* Minhyuk Sung
KAIST
{63days,dreamy1534,hieuristics,mhsung}@kaist.ac.kr

Contents

S.1. Overview . . . 3
S.2. SALAD Implementation Details . . . 3
S.3. Experiment Details . . . 3
S.3.1 Details on Part Completion Experiment Setup — Section 5.2 . . . 3
S.3.2 Details on Text-Guided Shape Generation — Section 5.4 . . . 4
S.3.3 Details on GaussGlot — Section 5.5 . . . 5
S.4. Multi-Class Generation . . . 5
S.5. Shape Generation with More Classes . . . 5
S.6. Shape Generation with Different Number of Parts . . . 6
S.7. More Qualitative Comparisons on Shape Generation . . . 7
S.8. More Qualitative Comparisons on Part Completion . . . 9
S.9. More Qualitative Results on Part Mixing and Refinement . . . 11
S.10More Qualitative Comparisons on Text-Guided Shape Generation . . . 13
S.11More Qualitative Results on Text-Guided Part Completion . . . 14

*Equal contribution.

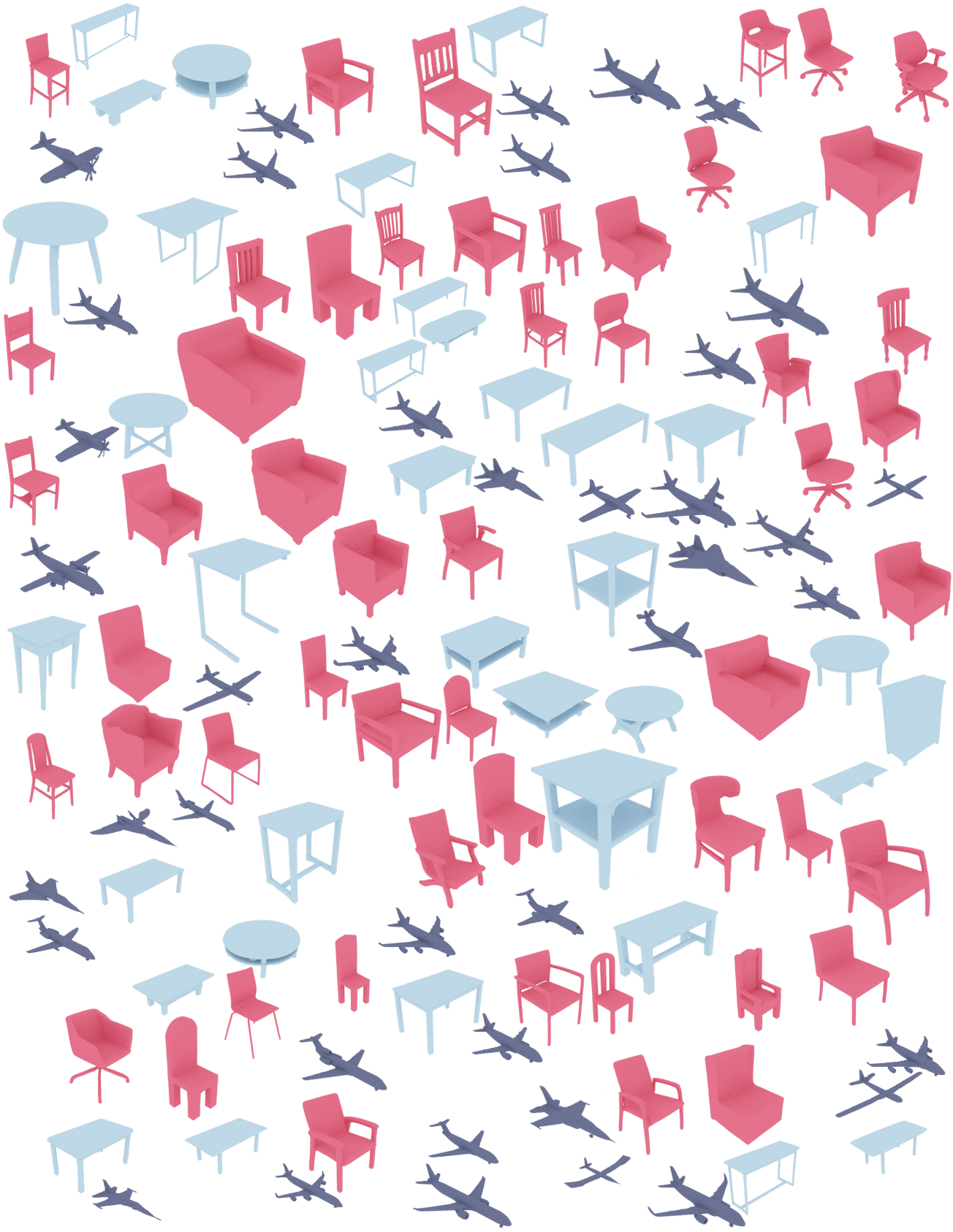


Figure S1: **A visual gallery of *airplanes*, *chairs*, and *tables* generated by SALAD.**

S.1. Overview

In this supplementary material, we first illustrate additional details in the implementation of SALAD (Section S.2) and details of the experiments discussed in the main paper (Section S.3). Then, we report additional experimental results: multi-class generation (Section S.4), shape generation with more classes (Section S.5) and shape generation with different number of parts (Section S.6). Lastly, we report more *qualitative* results of the experiments reported in the main paper: shape generation (Section S.7), part completion (Section S.8), part mixing and refinement (Section S.9), text-guided shape generation (Section S.10), and text-guided part completion (Section S.11).

S.2. SALAD Implementation Details

As discussed in Section 3.2 of the main paper, an extrinsic vector $\mathbf{e}_i$ is represented by $\{\mathbf{c}_i, \lambda_i^1, \lambda_i^2, \lambda_i^3, \mathbf{u}_i^1, \mathbf{u}_i^2, \mathbf{u}_i^3, \pi_i\}$, where the eigenvectors $\{\mathbf{u}_i^j\}_{j=1}^3$ must be orthogonal to each other. Therefore, the diffusion processes for $\{\mathbf{e}_i\}_{i=1}^N$ need to model distributions in a product space of an orthogonal group $\mathrm{O}(3)$ and Euclidean group, not in the Euclidean space. Recent work [11, 1] introduce diffusion models on Lie group or its product space, however, we empirically find that learning diffusion without considering the orthogonality also performs well. It is ensured only at the test time by taking the projection of the generated eigenvectors $\mathbf{U}_i = [\mathbf{u}_i^1, \mathbf{u}_i^2, \mathbf{u}_i^3]$ to $\mathrm{O}(3)$ space. We follow Schönemann [16] and project $\mathbf{U}_i$ as

$$\tilde{\mathbf{U}}_i = [\tilde{\mathbf{u}}_i^1, \tilde{\mathbf{u}}_i^2, \tilde{\mathbf{u}}_i^3] = \mathbf{A}\mathbf{B}^l, \tag{1}$$

where $\mathbf{U}_i = \mathbf{A}\mathbf{\Sigma}\mathbf{B}^T$ is a singular value decomposition of $\mathbf{U}_i$. We also clip negative eigenvalues in $\{\lambda_i^j\}_{j=1}^3$ to 1×10^{-4} since the covariance matrix is positive-definite.

We normalize elements of $\mathbf{e}_i$ to avoid arbitrary high-variance latent space. Specifically, during the training of "Diffusion of $\{\mathbf{e}_i\}_{i=1}^N$", we normalize π_i and $\{\lambda_i^j\}_{j=1}^3$ using element-wise means and standard deviations pre-computed from all training data. At test time, we re-scale these elements by the means and the standard deviations. We do not apply normalization to the others.

The Transformer-based network of SALAD introduced in Section 4 of the main paper consists of an embedding layer, which maps an input to 512-dimensional embeddings, and 6 Transformer blocks. Each Transformer block is a stack of a self-attention block and an MLP, each of which is followed by an AdaLN layer. We set the dimension of the output of the positional encoding $\gamma(\cdot)$ to 128.

As SALAD consists of two diffusion models, each trained for 5000 epochs, we train the baselines for $10{,}000$ epochs for a fair comparison. We use a batch size of 64 and an initial learning rate 10^{-4} with a polynomial decaying scheduler (power=0.999). The diffusion process is configured with $T = 1000$, $\beta^{(1)} = 10^{-4}$, and $\beta^{(T)} = 0.05$.

S.3. Experiment Details

In this section, we provide details of the experiments whose results are reported in the main paper.

S.3.1 Details on Part Completion Experiment Setup — Section 5.2

As mentioned in Section 5.2 of the main paper, part completion via a *guided* reverse process [14] requires binary masks indicating the parts to be ablated. We describe how such masks are constructed for SALAD and Neural Wavelet [8] in this section.

SALAD. We define a binary mask $m \in \{0, 1\}^N$ for pairs $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$ to have value 0 at completed parts, 1 otherwise. To this end, we first *transfer* the part labels of the annotated point clouds from ShapeNet [2] dataset to each $(\mathbf{e}_i, \mathbf{s}_i)$. Assume a point cloud $\{(\mathbf{x}_j, l_j)\}_{j=1}^K$ of K points where $\mathbf{x}_j \in \mathbb{R}^3$ and $l_j \in \{1, 2, \ldots, L\}$, denote 3D coordinate and part label of j-th point, respectively. Each $(\mathbf{e}_i, \mathbf{s}_i)$ is assigned a part label $l_i \in \{1, 2, \ldots, L\}$ based on the proximity of $\mathbf{e}_i$ to the points $\{\mathbf{x}_j\}_{j=1}^K$. Since $\mathbf{e}_i$ parameterizes a Gaussian distribution in 3D space, we employ Mahalanobis distance [13] as a distance measure. For each Gaussian represented by $\mathbf{e}_i$, we compute the distance to every point $\mathbf{x}_j$ and select the closest 100 points. We then count the number of part label occurrences over the points and assign the most frequently occurred label to the pair.

Having assigned the part labels to each of $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$, we define a mask m selecting a part whose label is l as

$$m_i = \begin{cases} 0 & \text{if } l_i = l \\ 1 & \text{otherwise} \end{cases}, \tag{2}$$

where m_i denotes the i-th element of m.

Neural Wavelet [8]. Note that there is neither a publicly available official code nor detailed instructions for shape manipulation using Neural Wavelet [8]. Although a concurrent work of ours, Hu *et al.* [7], demonstrates shape manipulation using Neural Wavelet, it does not provide a detailed implementation.

Following Hui *et al.* [8], we derive the wavelet coefficients of the shapes in our training set. We compute signed distance functions (SDFs) of the shapes and truncate their values into $[-0.1, 0.1]$. We denote S the resulting truncated signed distance function (TSDF) of a shape. We leverage Biorthogonal wavelet-6-8 filter [3] to decompose S into a coarse wavelet coefficient volume at a scale 3 (C^3) and a detail wavelet coefficient volume at a scale 2 (D^2). Refer to Hui *et al.* [8] for details on preprocessing.

We then aim to derive binary masks for C^3, necessary for leveraging pre-trained Neural Wavelet [8] for part completion. Note that selecting a part to complete is a *nontrivial* task for a voxel-based representation adapted by Neural Wavelet, as opposed to SALAD where we can define binary masks for $\{(\mathbf{e}_i, \mathbf{s}_i)\}_{i=1}^N$ to select parts directly. As one solution, we compute bounding boxes enclosing semantic parts of 3D shapes, and use them to designate the *regions* to complete. Such bounding boxes are used to compute binary masks for C^3 via a heuristic based on the property of wavelet transforms extracting local spectral information. Through experiments, we empirically find a set of wavelet coefficients that vary when the TSDF values in a 3D volume are set to 0.1 (i.e.,outside of a shape). For instance, we set the TSDF values in the bounding box enclosing the back of a chair to 0.1 to discover a set of wavelet coefficients corresponding to the part. We assign 0 to the coefficients whose amount of change is above a threshold δ and 1 to the others.

Rigorously, let $M \in \{0, 1\}^{256^3}$ denote a binary voxel grid of the same resolution as S with 0 indicating the semantic part of interest and 1 otherwise. Such M is derived from a bounding box enclosing a semantic part of a 3D shape, and is used to derive a *masked* TSDF S^* defined as

$$S_v^* = \begin{cases} 0.1 & \text{if } M_v = 0 \\ S_v & \text{otherwise} \end{cases}, \tag{3}$$

for all $v \in \{(0,0,0), (0,0,1), ..., (255,255,255)\}$. After marking all values inside a bounding box as *outside*, we obtain the wavelet coefficients C^{3*} via forward wavelet transform. A mask m for C^3 is then defined as

$$m_{v'} = \begin{cases} 0 & \text{if } |C_{v'}^{3*} - C_{v'}^3| > \delta \\ 1 & \text{otherwise} \end{cases}. \tag{4}$$

for all $v' \in \{(0,0,0), (0,0,1), ..., (47,47,47)\}$. Here, we use $\delta = 0.001$.

ShapeFormer [17]. As discussed in Section 5.2 of the main paper, after constructing the axis-aligned bounding box of a part, we make a partial point cloud by masking out the points inside the bounding box, and pass it to ShapeFormer [17] as an input.

S.3.2 Details on Text-Guided Shape Generation — Section 5.4

Implementation Details of Text-Conditioned SALAD. We impose text conditions on both the first and the second phase models by feeding text features from our text encoder. We use LSTM [6] for the text encoder and train it jointly with the first and the second phase models. We also apply the classifier-free guidance [5]. More precisely, we jointly train a conditional diffusion model $\boldsymbol{\epsilon}_\theta(\mathbf{x}^{(t)}, t, \mathbf{c})$ and an unconditional diffusion model $\boldsymbol{\epsilon}_\theta(\mathbf{x}^{(t)}, t, \emptyset)$, where $\mathbf{c}$ denotes a condition feature vector and $\emptyset$ is a null condition vector. We randomly set $\mathbf{c}$ to $\emptyset$ with a 20% dropout probability during training. To make $\emptyset$, we feed an empty sequence as an input text and zero vectors for $\{\mathcal{E}(\mathbf{e}_i)\}_{i=1}^N$. $\mathbf{c}$ is solely a text feature for the first phase model. For the second phase model conditioned on the features from extrinsic vectors $\{\mathbf{e}_i\}_{i=1}^N$, we use the concatenation of the features and a text feature as a condition.

At sampling time, the noise prediction is adjusted by an extrapolation between the noise prediction of the conditional diffusion model and the unconditional diffusion model as follows:

$$\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{x}^{(t)}, t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{x}^{(t)}, t, \emptyset), \tag{5}$$

where $\tilde{\boldsymbol{\epsilon}}_t$ is the noise prediction with the classifier-free guidance applied, and w is a hyperparameter controlling guidance strength. We use $w = 2$ for sampling.

Experiment Setup. To measure Neural-Evaluator-Preference (NEP) discussed in Section 5.4 of the main paper, we leverage a modified PartGlot [10] for a neural evaluator. The modified architecture takes point clouds as inputs instead of super-segments. Refer to the PartGlot [10] paper for more details. We adapt the training and test set of PartGlot [10] to create binary classification examples. The modified PartGlot achieves 73.98% test accuracy on the binary classification. Following Mittal *et al.* [15], we consider an example to be confused if the absolute difference between the neural evaluator's confidence is ≤ 0.2.

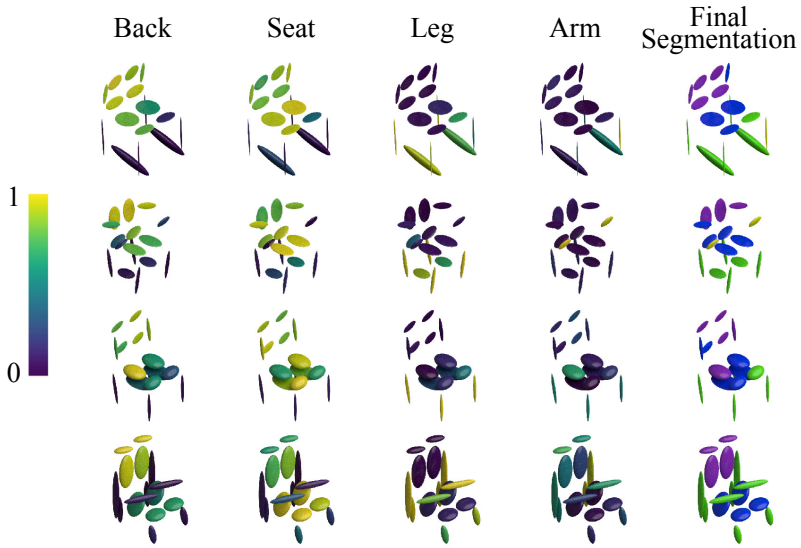


Figure S2: **GAUSSGLOT qualitative results.** The attention maps for each semantic part achieved by GAUSSGLOT are shown in the left columns of the figure. The colors of the attention maps change from dark blue to yellow as the attention weights increase from 0 to 1. The final part segmentation results are depicted in the rightmost column of the figure, where purple, blue, green, and yellow indicate *back*, *seat*, *leg*, and *arm*, respectively.

S.3.3 Details on GaussGlot — Section 5.5

Inspired by Koo *et al.* [10], we design a text-driven self-supervised semantic part segmentation network, GAUSSGLOT, where a set of Gaussian primitives is employed as super-segments. As discussed in Section 5.4 of the main paper, PartGlot is a neural evaluator that classifies shapes from a query text. While solving this text-conditioned shape classification, PartGlot learns semantic part segmentation in an unsupervised manner by learning the attention maps between the input text and the super-segments. Refer to the PartGlot [10] paper for more details. Specifically, we train GAUSSGLOT with $\{\mathbf{e}_i\}_{i=1}^{N}$ excluding π_i elements which is inessential to define 3D Gaussian primitives. Based on the architecture of PartGlot, 15-dimensional Gaussian parameters are mapped to 256-dimensional features through MLPs. We embed text tokens into 128 dimensions and use LSTM as a text encoder with 256-dimensional hidden states. Our trained GAUSSGLOT achieves 76.03% test accuracy and 56.85% mIoU. Qualitative part segmentation examples and the attention maps of each semantic part from GAUSSGLOT can be found in Figure S2.

S.4. Multi-Class Generation

We further demonstrate that SALAD is capable of multi-class generation. We construct the multi-class latent space by pre-training SPAGHETTI [4] with a training data set consisting of 200 *airplanes* and 200 *cars*. Next, we train class-label-conditioned SALAD with the latents extracted from the pre-trained SPAGHETTI. Figure S3 shows the same initial latents are decoded into different class shapes, airplanes and cars, through the class-label-guided reverse process.

S.5. Shape Generation with More Classes

In the main paper, we used *chairs* and *airplanes* for the quantitative comparison as done in the previous work [8]. Figure S3 and Figure S4 show qualitative results of SALAD trained with more other classes, *cars*, *lamps* and *cabinets*.

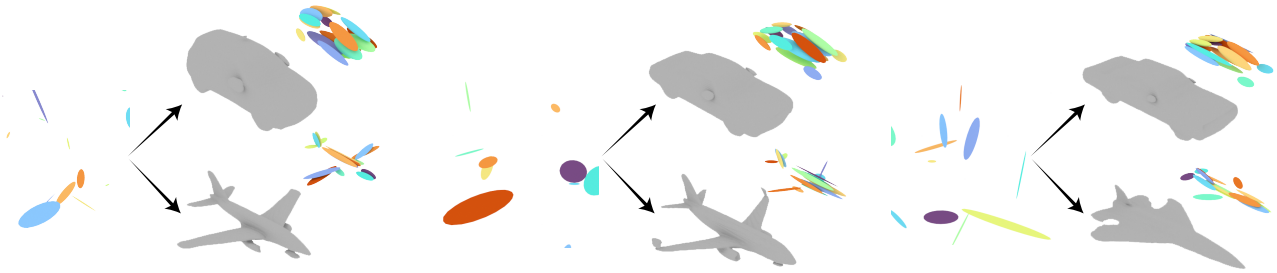


Figure S3: **Class-label-guided generation of SALAD trained with *airplanes* and *cars*.**

Figure S4: **Generation of *lamps* and *cabinets*.**

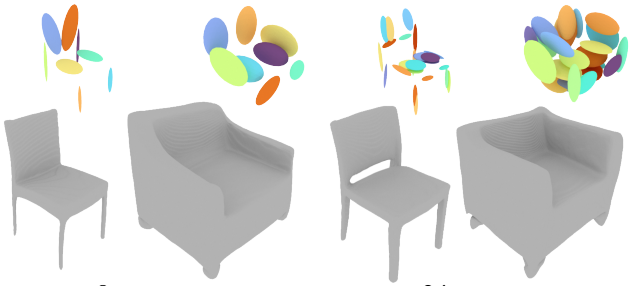


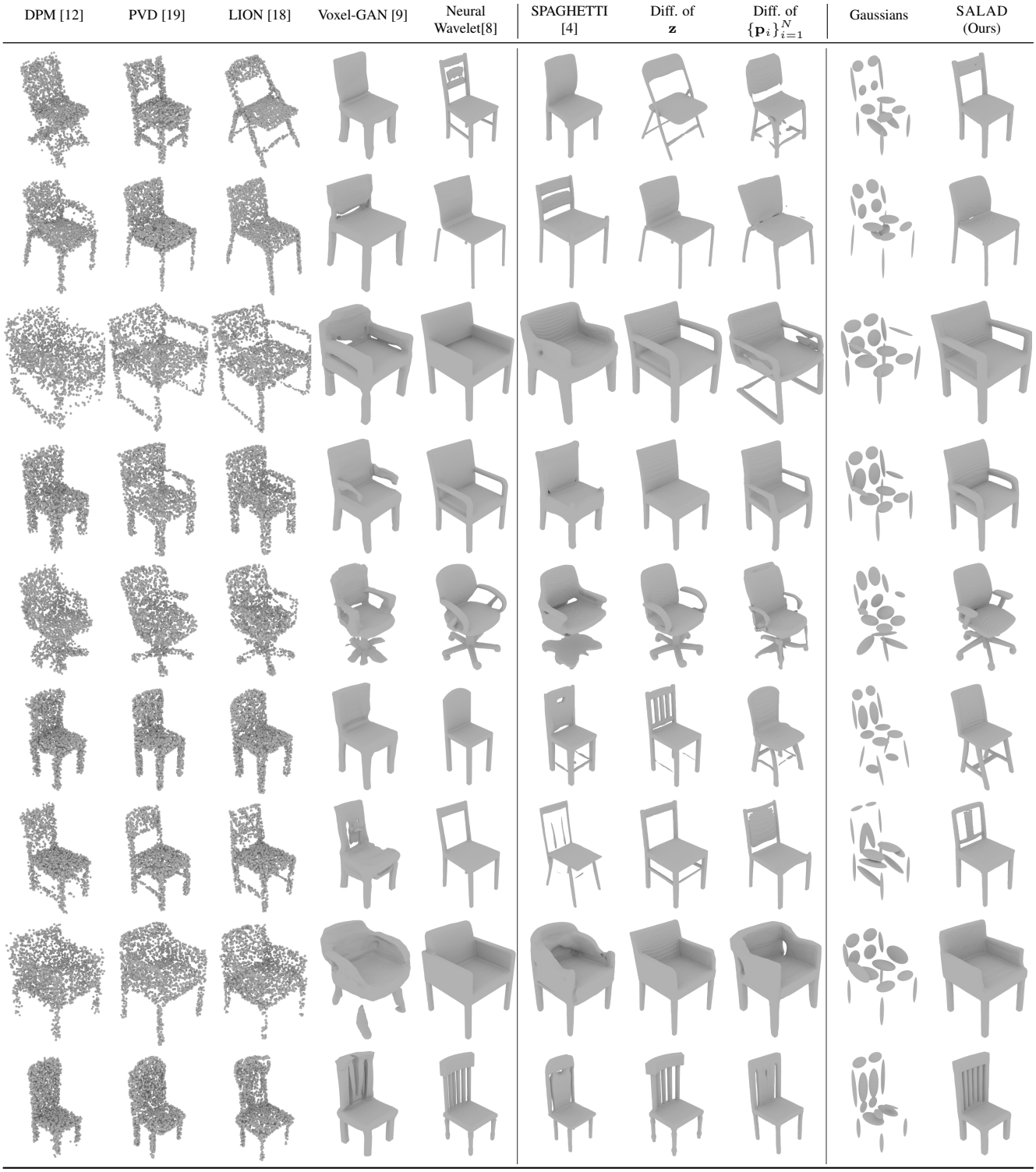
Figure S5: **Generation with varying number of parts.**

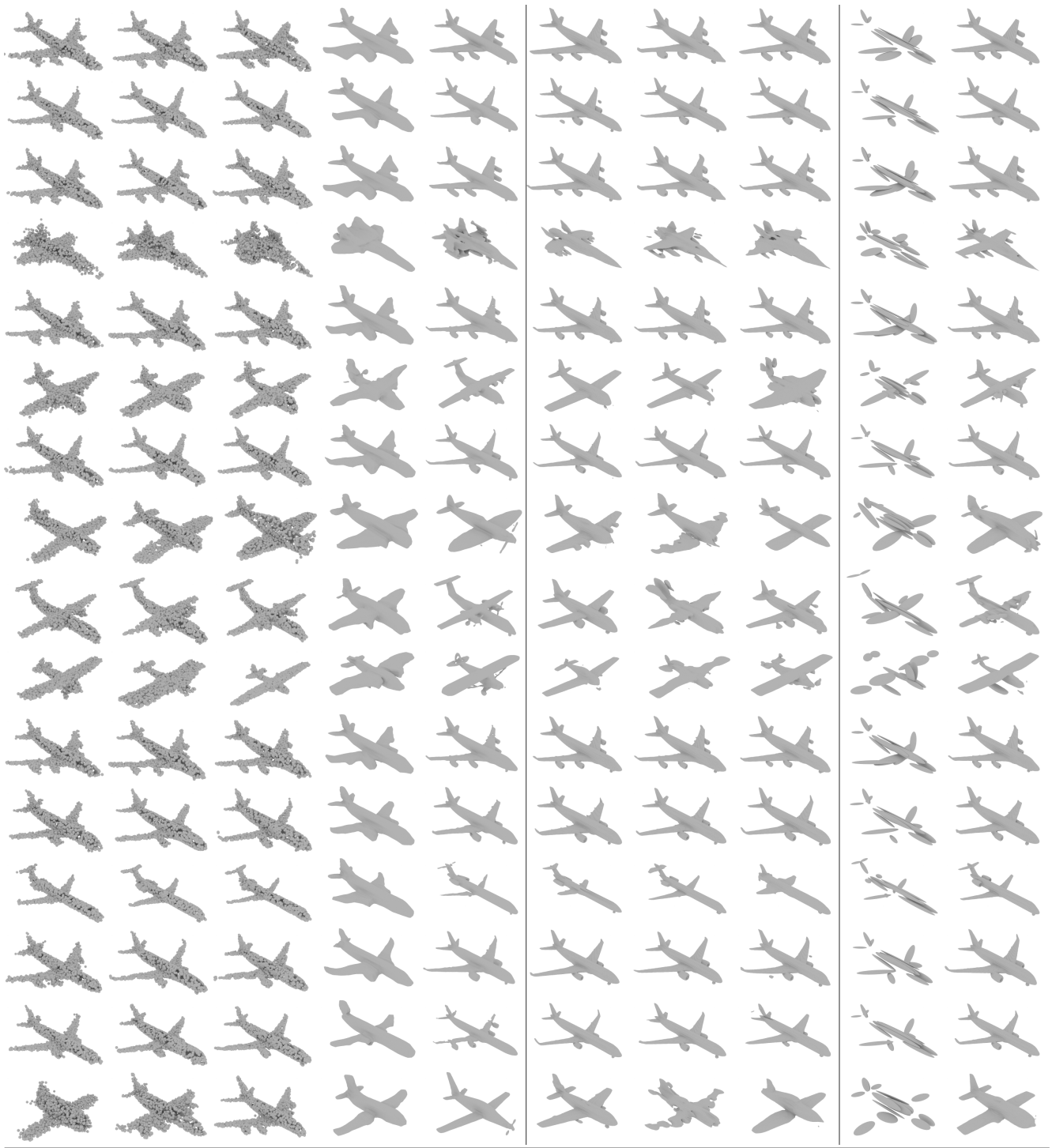
S.6. Shape Generation with Different Number of Parts

Although we used 16 parts in the main paper, Figure S4 shows qualitative results of with varying number of parts, 8 and 24 parts, respectively. It demonstrates that SALAD is agnostic to the number of parts. Furthermore, the experiments of Figure S3, Figure S4 and Figure S5 use 400 training shapes, a significantly smaller number than the train set of the main paper. It demonstrates that SALAD can generate high-quality shapes with a small number of training data.

S.7. More Qualitative Comparisons on Shape Generation

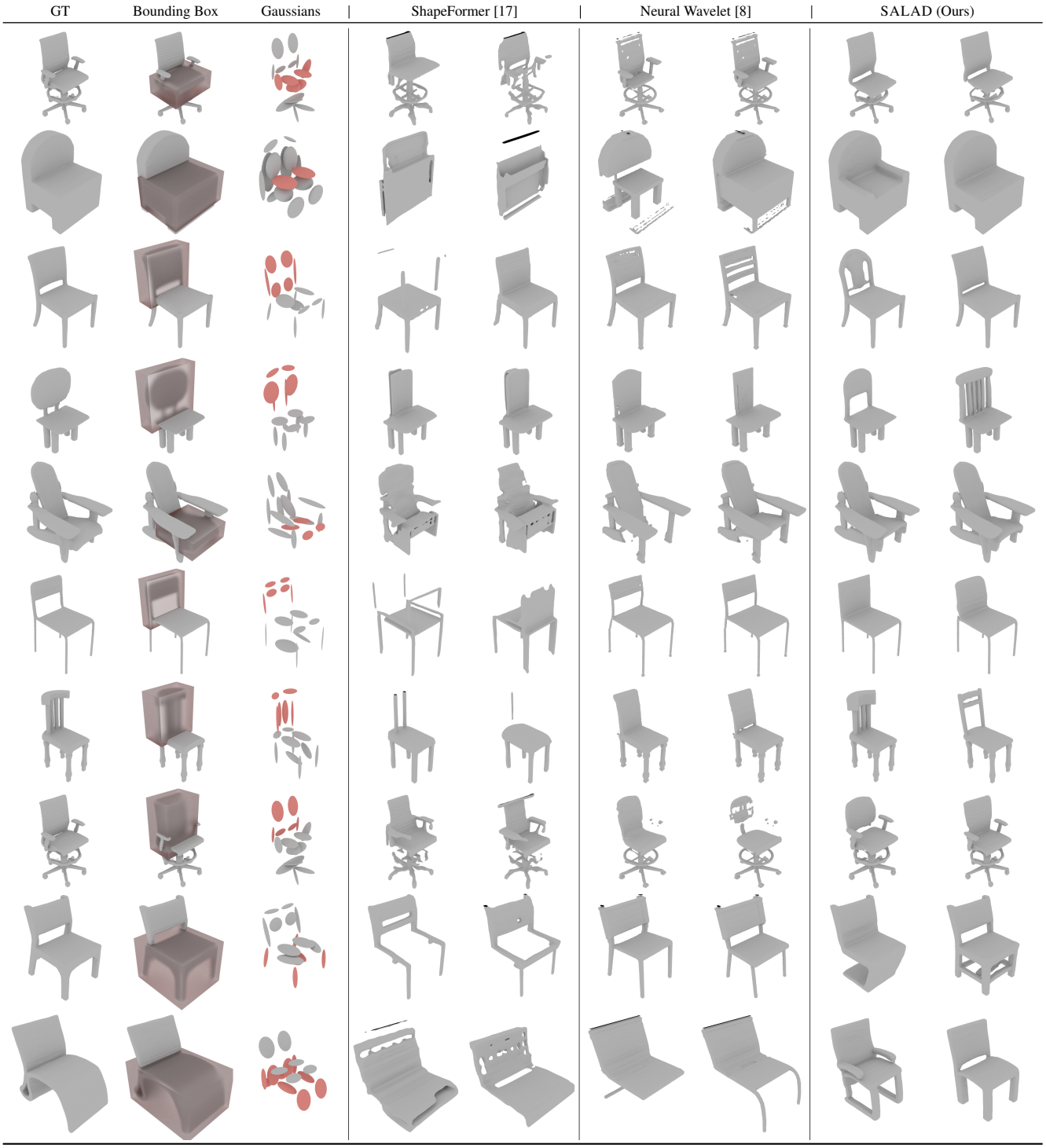
In the following, we provide more qualitative comparisons on shape generation with *chair* and *airplane* classes, as shown in Figure 4 of the main paper.

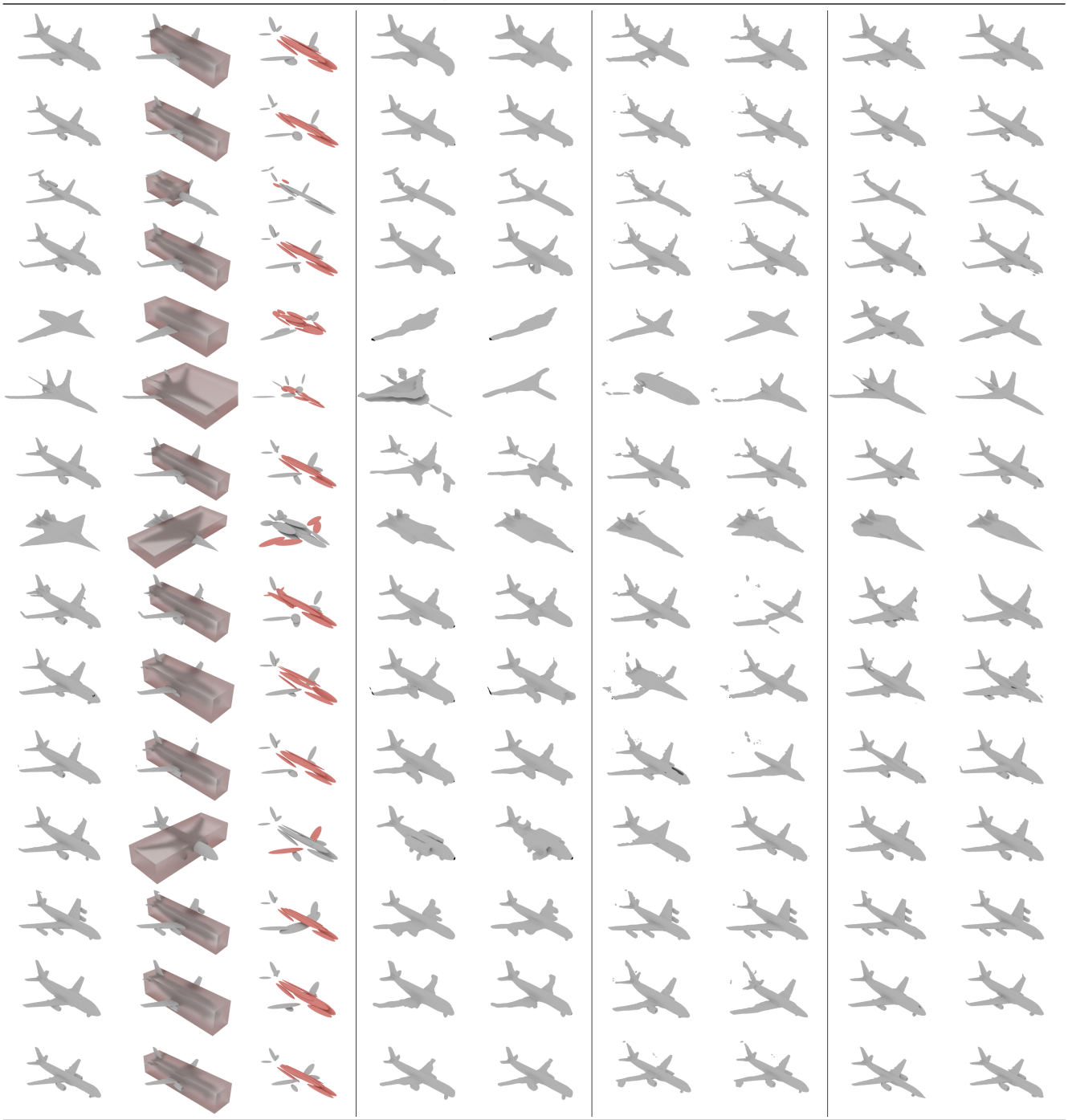



DPM [12]
PVD [19]
LION [18]
Voxel-GAN [9]
Neural Wavelet[8]
SPAGHETTI [4]
Diff. of $\mathbf{z}$
Diff. of $\{\mathbf{p}_i\}_{i=1}^{N}$
Gaussians
SALAD (Ours)

S.8. More Qualitative Comparisons on Part Completion

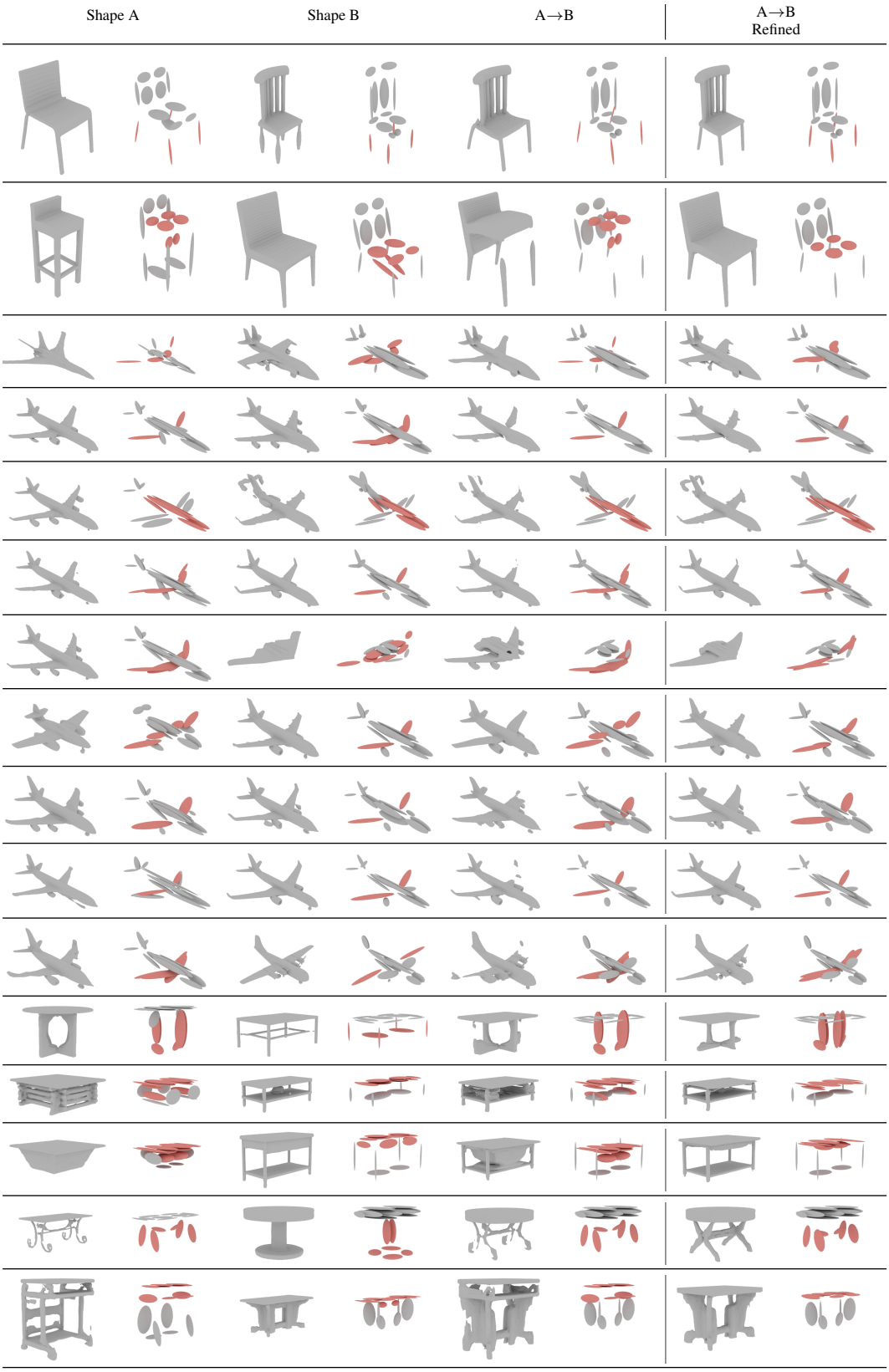
We report more qualitative comparisons on part completion with *chair* and *airplane* classes, as shown in Figure 5 in the main paper.




GT
Bounding Box
Gaussians
ShapeFormer [17]
Neural Wavelet [8]
SALAD (Ours)

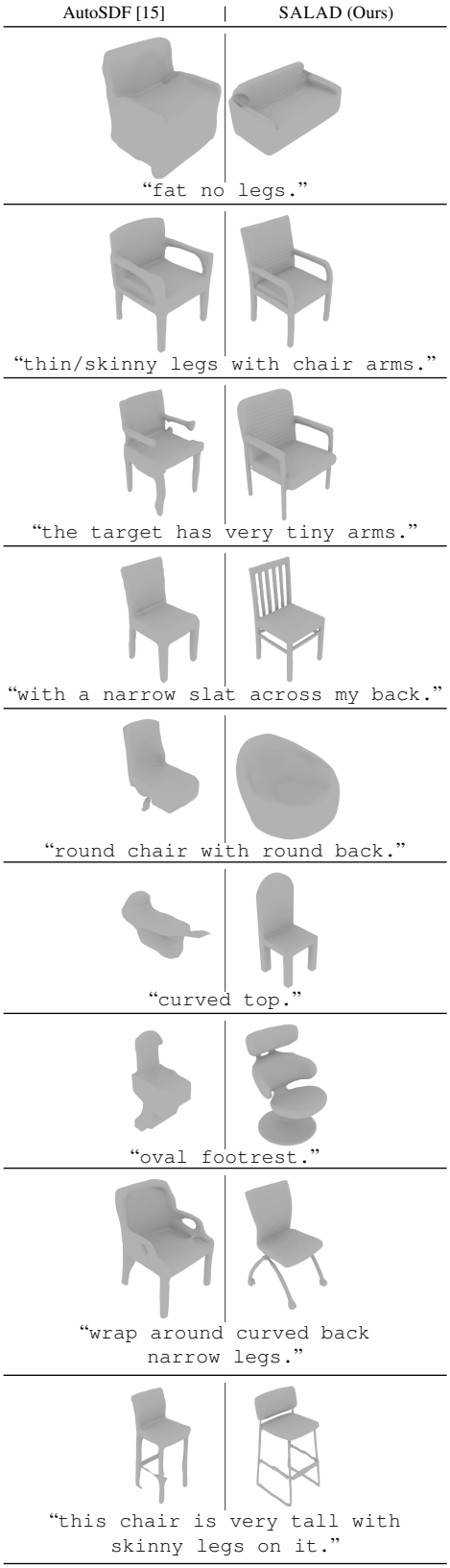
S.9. More Qualitative Results on Part Mixing and Refinement

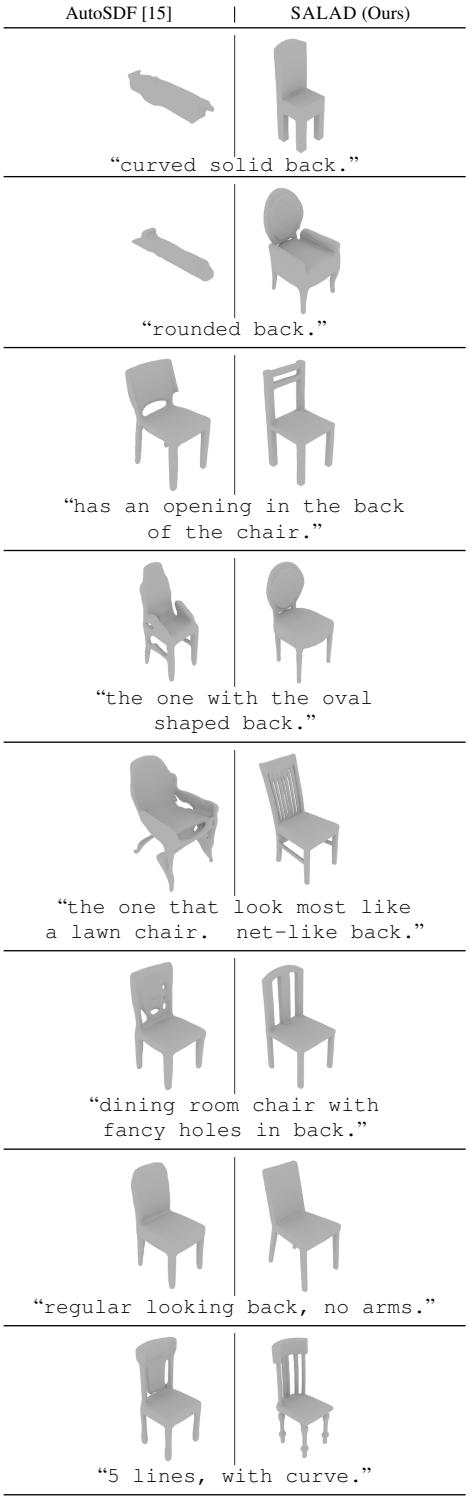
We report more qualitative results on part mixing and refinement with *chair*, *airplane* and *table* classes, as shown in Figure 6 in the main paper.


Shape A
Shape B
A→B
A→B
Refined

S.10. More Qualitative Comparisons on Text-Guided Shape Generation

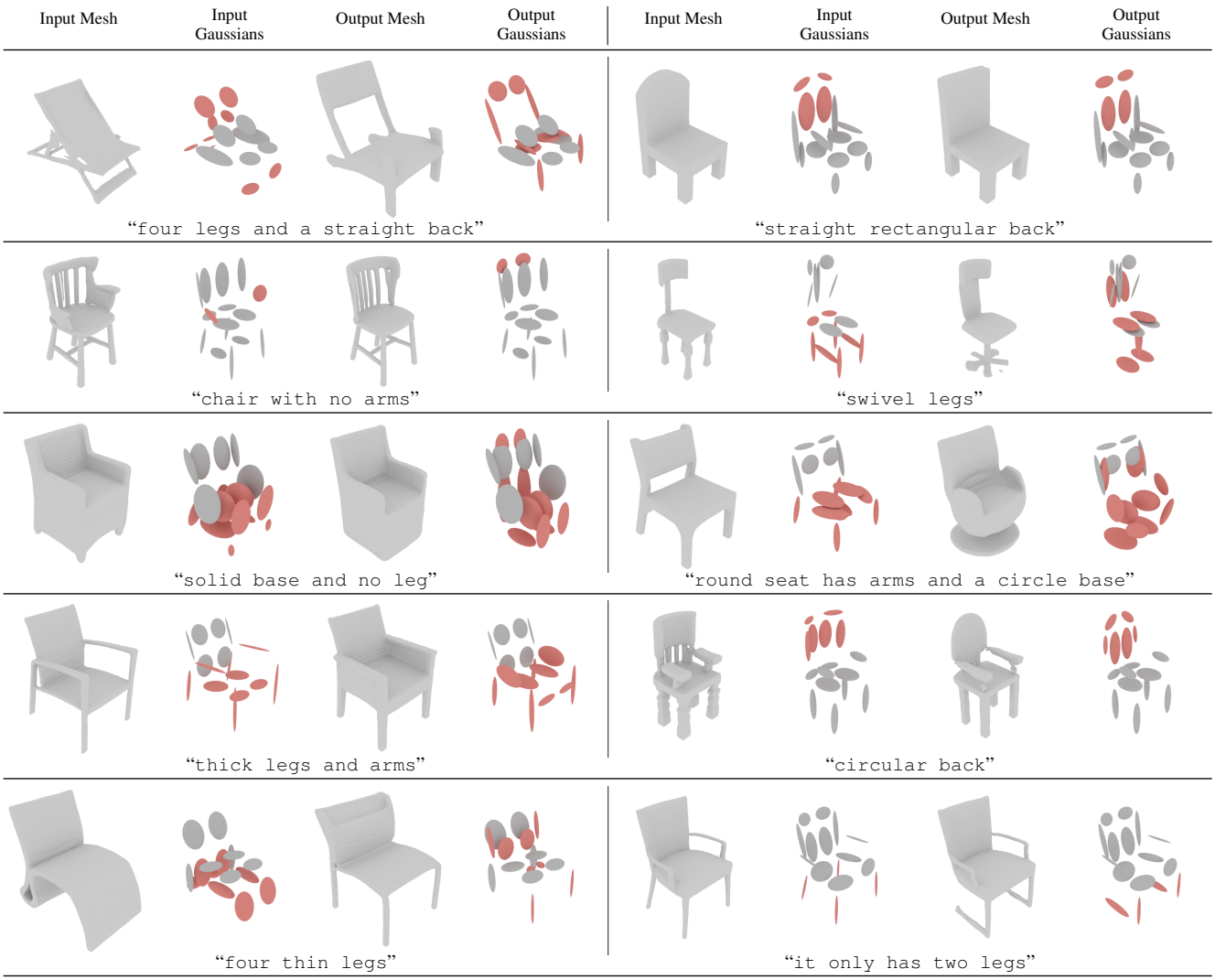
We report more qualitative comparisons on text-guided shape generation between AutoSDF [15] and SALAD.





S.11. More Qualitative Results on Text-Guided Part Completion

We report more qualitative results on text-guided part completion leveraging SALAD and GAUSSGLOT. In the figure below, the parts selected by GAUSSGLOT from the text are highlighted by red. Text-conditioned SALAD completes the selected parts to match the text via the guided reverse process.



References

[1] Valentin De Bortoli, Emile Mathieu, Michael John Hutchinson, James Thornton, Yee Whye Teh, and Arnaud Doucet. Riemannian score-based generative modelling. In *NeurIPS*, 2022.

[2] 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.

[3] Albert Cohen. *Biorthogonal Wavelets*. 1993.

[4] Amir Hertz, Or Perel, Raja Giryes, Olga Sorkine-Hornung, and Daniel Cohen-Or. SPAGHETTI: Editing implicit shapes through part aware generation. *ACM TOG*, 2022.

[5] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021.

[6] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[7] Jingyu Hu, Ka-Hei Hui, Zhengzhe Liu, Ruihui Li, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation, inversion, and manipulation. *arXiv preprint arXiv:2302.00190*, 2023.

[8] Ka-Hei Hui, Ruihui Li, Jingyu Hu, and Chi-Wing Fu. Neural wavelet-domain diffusion for 3d shape generation. In *SIGGRAPH ASIA*, 2022.

[9] Marian Kleineberg, Matthias Fey, and Frank Weichert. Adversarial generation of continuous implicit shape representations. *Eurographics - Short Papers*, 2020.

[10] Juil Koo, Ian Huang, Panos Achlioptas, Leonidas J Guibas, and Minhyuk Sung. PartGlot: Learning shape part segmentation from language reference games. In *CVPR*, 2022.

[11] Adam Leach, Sebastian M Schmon, Matteo T. Degiacomi, and Chris G. Willcocks. Denoising diffusion probabilistic models on SO(3) for rotational alignment. In *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[12] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*, 2021.

[13] Prasanta Chandra Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Sciences of India*, 1936.

[14] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *ICLR*, 2022.

[15] Paritosh Mittal, Yen-Chi Cheng, Maneesh Singh, and Shubham Tulsiani. Autosdf: Shape priors for 3d completion, reconstruction and generation. In *CVPR*, 2022.

[16] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 1966.

[17] Xingguang Yan, Liqiang Lin, Niloy J Mitra, Dani Lischinski, Daniel Cohen-Or, and Hui Huang. Shapeformer: Transformer-based shape completion via sparse representation. In *CVPR*, 2022.

[18] 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 *NeurIPS*, 2022.

[19] Linqi Zhou, Yilun Du, and Jiajun Wu. 3d shape generation and completion through point-voxel diffusion. In *ICCV*, 2021.