Supplementary Material of Autoregressive 3D Shape Generation via Canonical Mapping

1 Overview

This supplementary material provides more details about the implementation, model efficiency, and additional results of our work. We start by introducing implementation details in Section 2, followed by model size and computational time discussion in Section 3. We then show ablation studies on different shape composition serialization methods in Section 4. In Section 5, Section 6 and Section 7, we demonstrate more qualitative results on auto-encoding, unconditional generation and conditional generation. We further show the visualization of latent space in Section 8 and provide additional experiments on the full ShapeNet collection in Section 9. Finally, we discuss limitations of our method in Section 10.

2 Implementation Details

2.1 Model Architecture

We use the same model architecture for all encoders and decoders used in our method. Specifically, we adopt the encoder structure from DGCNN [11], which contains 3 EdgeConv layers using neighborhood size 20. Our decoder follows the two branch structure as in SP-GAN [7]. Given a matrix that consists of sphere points and features, the decoder first feeds sphere points to a graph attention module to extract point-wise spatial features. On the other branch, we use a nonlinear feature embedding to extract style features from the latent. Then, we use adaptive instance normalization [4] to fuse the local styles with the spatial features. We repeat the process with another round of style embedding and fusion, then predict the final output from the fused feature. For our grouping network, we follow [2], using a two-layer 128-neuron MLP with ReLU activations and BatchNorm layers. Our transformer model is modified from [5], where we reduce their number of layers and heads to 24 and 16, respectively.

2.2 Training Details

We use the airplane category in ShapeNet as an example to illustrate the training pipeline. We perform all the experiments on a workstation with Intel Xeon Gold 6154 CPU (3.00GHz) and 4 NVIDIA Tesla V100 (32GB) GPUs. We implement our framework with Pytorch 1.10. Please see Algorithm 1 for more details.

Algorithm 1 : The training phase of our approach consists of learning four components: (1) Canonical Auto-encoder (2) Canonical Grouping Network (3) VQVAE (4) Transformer

(A) Canonical Auto-encoder ▷ *8 hours on Airplane category*

Sub-sample M points from the input point cloud x and canonical sphere π;
Initialize weight of the encoder $E_c(\cdot)$ and decoder $D_c(\cdot)$;
while not converged **do**
  foreach iteration **do**
    $\boldsymbol{z_x} \leftarrow E_c(x)$;
    $\hat{\boldsymbol{x}} \leftarrow D_c([\pi_i, z_A])$, where $\pi_i \in \pi$;
    Obtain reconstruction loss $L_{CD}(\hat{x}, x)$ and $L_{EMD}(\hat{x}, x)$;
    Update weight;

(B) Canonical Grouping Network ▷ *2 hours on Airplane category*

Generate randomly M points from the canonical sphere π;
Initialize weight of the canonical grouping network $MLP(\cdot)$;
while not converged **do**
  foreach iteration **do**
    Obtain the corresponding point of $\pi_i \in \pi$ on x as $\Phi_{\pi \to x}(\pi_i)$;
    $\boldsymbol{P_i} \leftarrow MLP([x_i, \pi_i)$, where $x_i \in x, \pi_i \in \pi$;
    $\boldsymbol{K_j} \leftarrow \sum_{i=1}^{m} \Phi_{\pi \to x}(\pi_i) P_i^j$, where $j = 1, 2, ..., G$;
    Obtain loss $L_{CD}(K, x)$;
    Update weight;

(C) VQVAE ▷ *24 hours on Airplane category*

Sub-sample M points from the input point cloud x and canonical sphere π;
Sequentialize x;
Initialize weight of $E(\cdot)$, $D(\cdot)$, and $VQ(\cdot)$;
while not converged **do**
  foreach iteration **do**
    Obtain group feature $\boldsymbol{z} \leftarrow E(x)$
    $\boldsymbol{z_q} \leftarrow VQ(z)$
    $\hat{\boldsymbol{x}} \leftarrow D([\pi_i, z_q])$, where $\pi_i \in \pi$;
    Obtain loss $L_{Quantization}$;
    Update weight;

(D) Transformer ▷ *16 hours on Airplane category*

Sub-sample M points from the input point cloud x;
Initialize weight of the transformer;
Vector Quantize x to a sequence s;
while not converged **do**
  foreach iteration **do**
    Obtain the probability distribution for each $s_i \in s$ autoregressively.
    Obtain loss $L_{Transformer}$;
    Update weight;

2.3 Inference Details

Esser et al. [5] introduce several test-time hyper-parameters (e.g., top-k and top-p heuristics, temperature scaling factor t) for transformer to obtain best results. Following [5], we use the top-p sampling heuristic for the transformer model which we empirically set $p = 0.92$ throughout all experiments. We do not use the top-k sampling heuristic and the temperature scaling factor t is set to 1 unless otherwise specified. We provide unconditional generation results on ShapeNet Chair using different p in Table 1.

Table 1: Shape generation results on ShapeNet Chair. ↑ means the higher the better, ↓ means the lower the better. MMD-CD is multiplied by 10^3 and MMD-EMD is multiplied by 10^2.

	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
Model	CD	EMD	CD	EMD	CD	EMD
$p = 0.85$	7.70	11.87	41.84	44.25	61.40	64.72
$p = 0.92$	7.37	11.75	**45.77**	**46.07**	**60.12**	**61.93**
$p = 0.99$	**7.22**	**11.73**	44.86	45.46	60.19	62.38

2.4 Evaluation Metrics

- **Minimum matching distance (MMD)** [13] measures the visual quality of the generated set. For each sample in the reference set, we compute the distance to its nearest neighbor in the generated set. The final MMD is the average of the computed distances. Note that the nearest neighbor can be searched with different distance measurements such as Chamfer distance or Earth Mover distance.
- **Coverage (COV)** [13] is able to detect mode-collapse in the generated set. Specifically, COV measures the fraction of samples in the reference set that are matched to at least one sample in the generated set. Specifically, for each sample in the generated set, we mark its nearest neighbor in the reference set as a matched sample. Similar to the MMD metric, the nearest neighbor can be searched with different distance measurements such as Chamfer distance or Earth Mover distance.
- **1-nearest neighbor accuracy (1-NNA)** [13] is a metric that performs two-sample tests [8] on the generated set and the reference set. Therefore, if the generated set is drawn from the reference set, the classifier will result in a random model (i.e., close to 50% accuracy).
- **Total Mutual Difference (TMD)** [12] is a metric that measures the diversity given a conditional input (e.g., depth-map, partial point cloud). Specifically, for each shape i in the k generated set, we calculate its average Chamfer distance d_i^{CD} to the other $k-1$ shapes. The total TMD is calculated as $\sum_{i=1}^{k} d_i^{CD}$.

3 Computational Time and Model Size

Table 2: The parameter size and inference time for different models.

Model	# Parameters	Inference Time
PointGrow [10]	0.31M	5303.70
ShapeGF [1]	0.13M	0.2659
SP-GAN [7]	0.58M	0.2407
PointFlow [13]	1.61M	0.3506
SetVAE [6]	0.55M	0.0158
DPM [9]	3.87M	0.0943
PVD [14]	27.6M	38.35
Ours (Transformer)	20.6M	1.5391
Ours (VQ)	0.91M	0.0981
Ours (Total)	21.5M	1.6372

We report the inference time and model size for different models in Table 2. To be precise, each model's inference time and model size is measured as the time and number of parameters needed for generating a shape instance. All results are measured with their official implementation on a workstation with Intel Xeon Gold 6154 CPU (3.00GHz) and a single NVIDIA Tesla V100 (32GB). Note that PointGrow requires forwarding the model with the same times as the desired number of points (e.g., 2048). Therefore, PointGrow is slow to compute. PVD is a diffusion-based approach that involves multi-step refinement from random noise, therefore, is computationally intensive, too.

4 Ablation on Shape Composition Serialization

To analyze the effect of different shape composition serialization, we train our transformer model with (1) random order (2) Fibonacci spiral order (3) inverse Fibonacci spiral order (Spiral$\star$). As shown in Table 3, using Fibonacci spiral order in either direction is generally better than using a random order.

Table 3: Shape generation results on ShapeNet Chair. ↑ means the higher the better, ↓ means the lower the better. MMD-CD is multiplied by 10^3 and MMD-EMD is multiplied by 10^2.

	MMD (↓)		COV (%, ↑)		1-NNA (%, ↓)	
Model	CD	EMD	CD	EMD	CD	EMD
Random	7.44	11.85	43.20	42.14	61.02	65.18
Spiral	7.37	11.75	**45.77**	**46.07**	60.12	**61.93**
Spiral$\star$	**7.17**	**11.61**	44.56	44.71	**59.36**	62.23

5 Qualitative Results of Auto-encoding

In Figure 1, we show more auto-encoding results. Thanks to the context-rich codebook, our model is able to reconstruct shapes with better local details. Moreover, the points are more uniformly distributed on the surface.

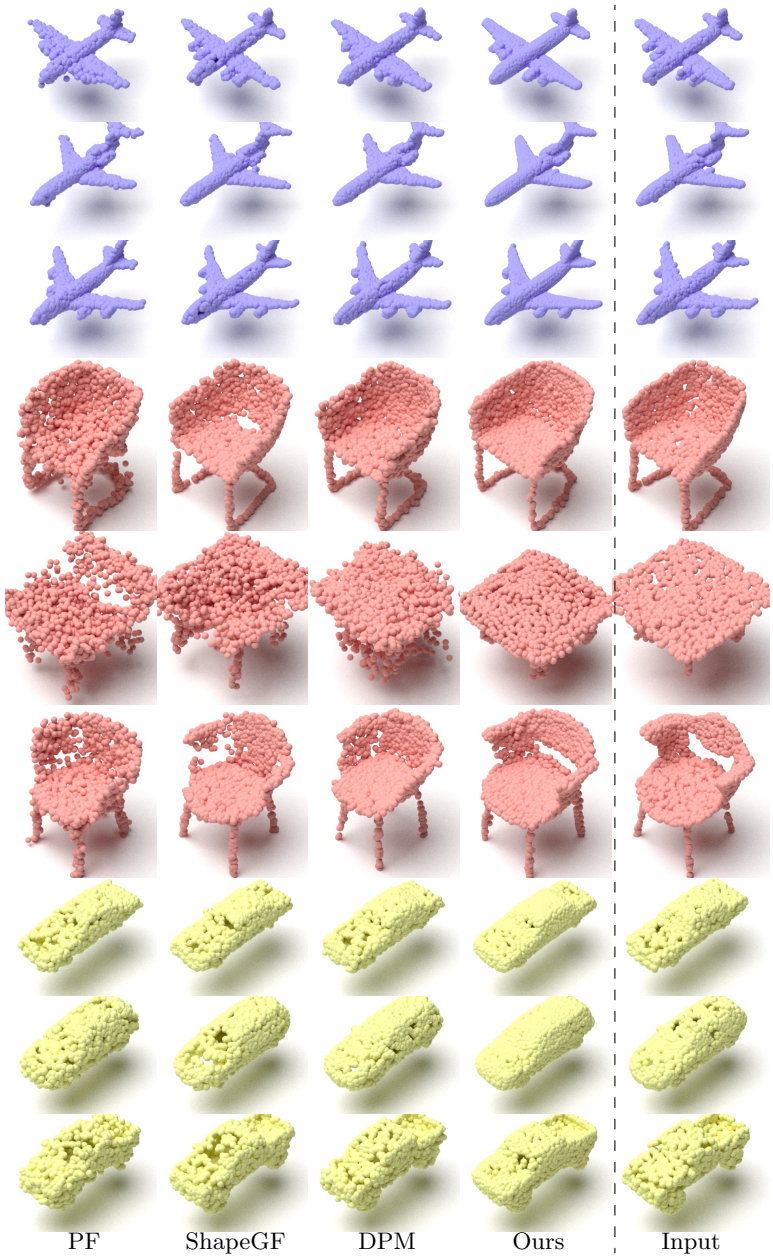


Fig. 1: Auto-encoding (reconstruction) results. We also shown results from PF (PointFlow) [13], ShapeGF [1], and DPM [9] on the left for comparison.

6 Qualitative Results of Unconditional Generation

In Figure 2, Figure 3, and Figure 4, we show more unconditional generation results. The results suggest that our model can generate diverse shape in high fidelity.

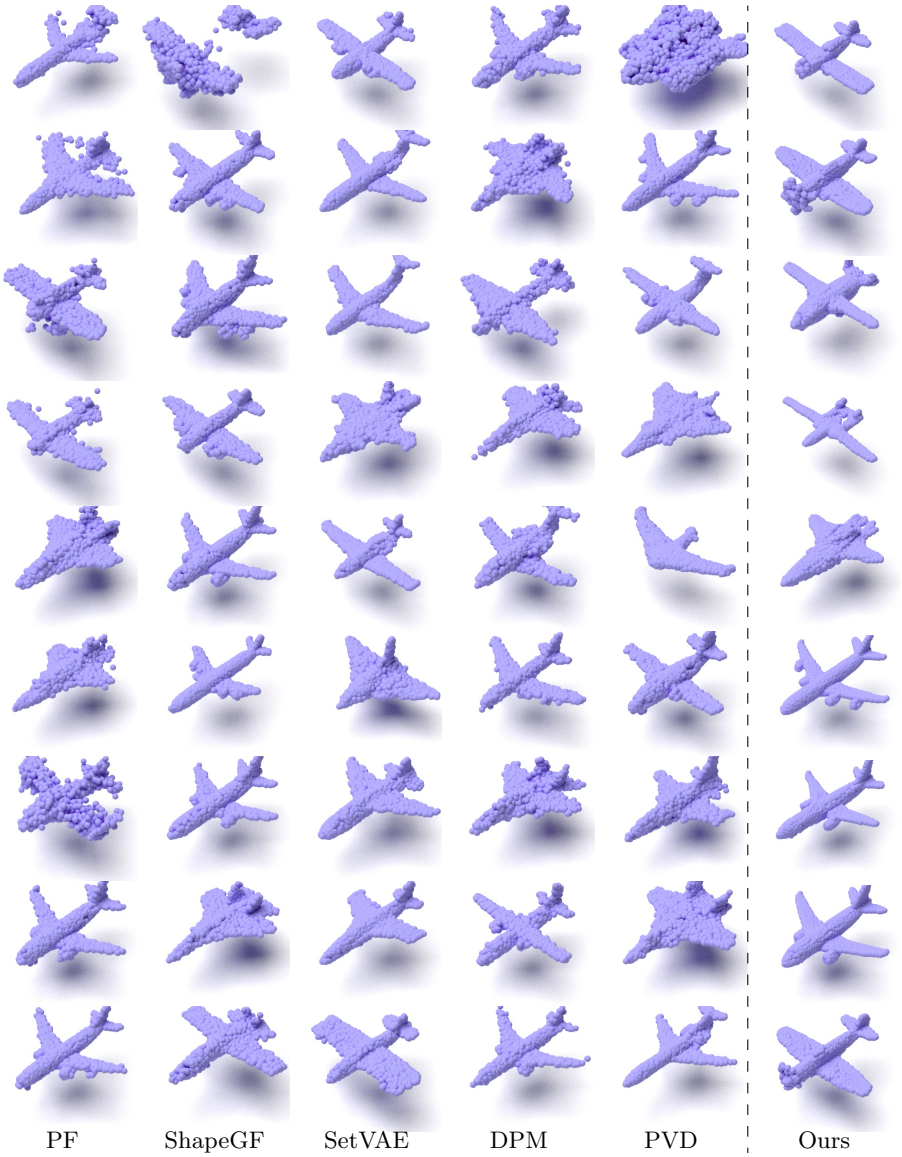


Fig. 2: Shape generation results on ShapeNet Airplane. We shown results from PF (PointFlow) [13], ShapeGF [1], SetVAE [6], DPM [9], and PVD [14].

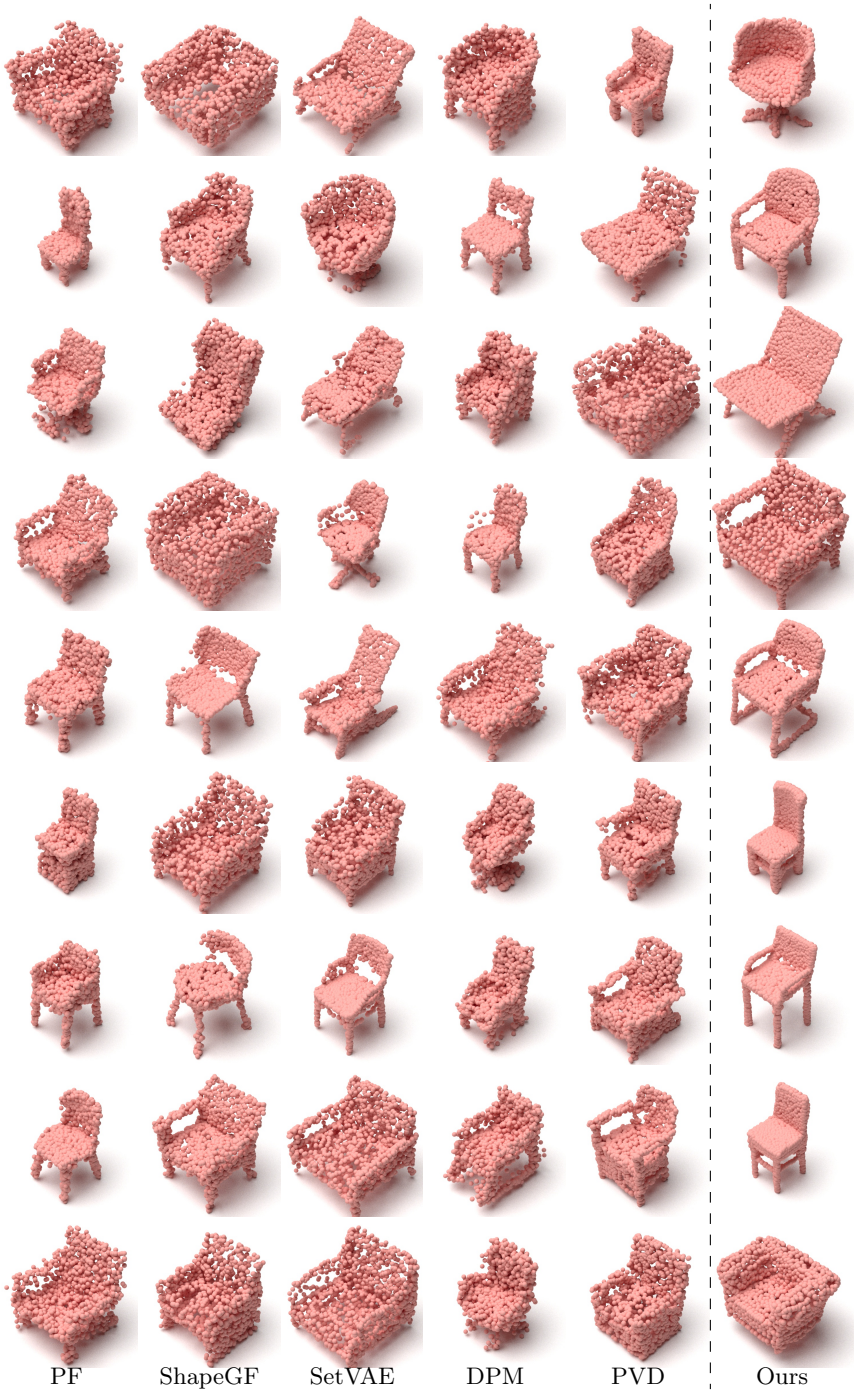


Fig. 3: Shape generation results on ShapeNet Chair. We shown results from PF (PointFlow) [13], ShapeGF [1], SetVAE [6], DPM [9], and PVD [14].

Fig. 4: Shape generation results on ShapeNet Car. We shown results from PF (PointFlow) [13], ShapeGF [1], SetVAE [6], DPM [9], and PVD [14].

7 Qualitative Results of Conditional Generation

In Figure 5, we show 4 more samples of the conditional generation results. Our shape completion results tend to show more variation and have better visual quality comparing to MSC [12] and PVD [14].

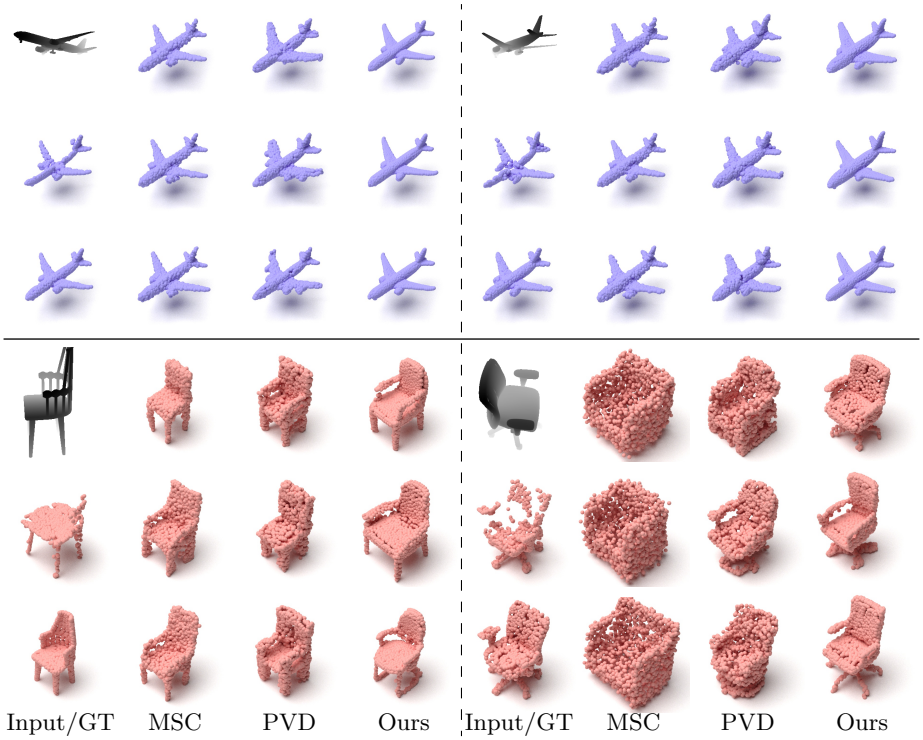


Fig. 5: Multi-modal shape completion results. The input depth-map, partial point cloud, and reference ground-truth shape for each sample is shown in the first column, respectively (from top to bottom).

8 Visualization of Latent Space

To further show that the proposed model can learn codebooks as a library of local shapes, we visualize the learned VQ codes in different groups in Figure 6, where each codebook clearly captures one meaningful part of the chair category.

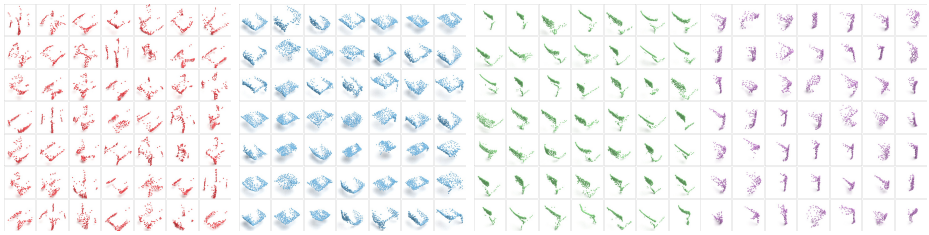


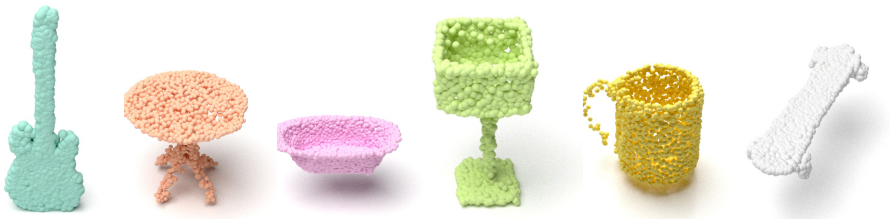
Fig. 6: Visulization of codebooks. Each of the 7×7 grid corresponds to a learned group codebook and each shape in the inner square represents a decoded code.

9 Generalizing to Different Object Categories.

To show that our model is capable of generalizing to different object categories, we report the auto-encoding performance of our model on the full ShapeNet collections, which consist of shapes from 55 categories. We use the same model configuration and training hyper-parameters as mentioned in the paper. As shown in Table 4, our method achieves lower EMD and comparable CD scores. We demonstrate reconstructions of other categories (e.g., guitar, table, bathtub, lamp, mug, skateboard) by our model in the figure below.

Table 4: Shape auto-encoding on the full ShapeNet dataset. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

	AtlasNet		PF	ShapeGF	Ours	Oracle
Metric	Sphere	Patches				
CD	5.301	**5.121**	7.551	5.154	5.164	3.031
EMD	5.553	5.493	5.176	4.603	**3.799**	3.103



10 Limitations

Our model relies on the learned correspondence from the canonical mapping function, therefore, inherits similar limitations from Cheng et al. [3]. Our model fails to reconstruct certain samples with holes or with complex topology. We show some failure cases of our model in Figure 7.

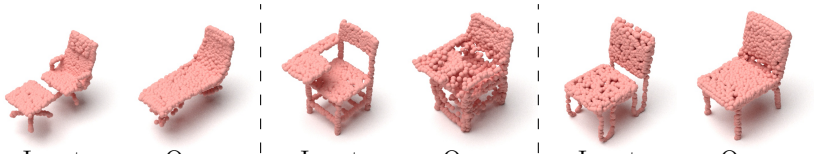


Fig. 7: Failure cases.

References

1. Cai, R., Yang, G., Averbuch-Elor, H., Hao, Z., Belongie, S., Snavely, N., Hariharan, B.: Learning gradient fields for shape generation. In: ECCV. pp. 364–381. Springer (2020) 4, 5, 6, 7, 8
2. Chen, N., Liu, L., Cui, Z., Chen, R., Ceylan, D., Tu, C., Wang, W.: Unsupervised learning of intrinsic structural representation points. In: CVPR (2020) 1
3. Cheng, A.C., Li, X., Sun, M., Yang, M.H., Liu, S.: Learning 3d dense correspondence via canonical point autoencoder. In: NeurIPS (2021) 10
4. Dumoulin, V., Shlens, J., Kudlur, M.: A learned representation for artistic style. ICLR (2017) 1
5. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021) 1, 3
6. Kim, J., Yoo, J., Lee, J., Hong, S.: Setvae: Learning hierarchical composition for generative modeling of set-structured data. In: CVPR. pp. 15059–15068 (2021) 4, 6, 7, 8
7. Li, R., Li, X., Hui, K.H., Fu, C.W.: Sp-gan: Sphere-guided 3d shape generation and manipulation. TOG **40**(4), 1–12 (2021) 1, 4
8. Lopez-Paz, D., Oquab, M.: Revisiting classifier two-sample tests. ICLR (2017) 3
9. Luo, S., Hu, W.: Diffusion probabilistic models for 3d point cloud generation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2837–2845 (2021) 4, 5, 6, 7, 8
10. Sun, Y., Wang, Y., Liu, Z., Siegel, J., Sarma, S.: Pointgrow: Autoregressively learned point cloud generation with self-attention. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 61–70 (2020) 4
11. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph cnn for learning on point clouds. TOG **38**(5), 1–12 (2019) 1
12. Wu, R., Chen, X., Zhuang, Y., Chen, B.: Multimodal shape completion via conditional generative adversarial networks. In: European Conference on Computer Vision. pp. 281–296. Springer (2020) 3, 9
13. Yang, G., Huang, X., Hao, Z., Liu, M.Y., Belongie, S., Hariharan, B.: Pointflow: 3d point cloud generation with continuous normalizing flows. In: CVPR. pp. 4541–4550 (2019) 3, 4, 5, 6, 7, 8
14. Zhou, L., Du, Y., Wu, J.: 3d shape generation and completion through point-voxel diffusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 5826–5835 (2021) 4, 6, 7, 8, 9