Autoregressive 3D Shape Generation via Canonical Mapping

An-Chieh Cheng[1⋆] Xueting Li[2⋆] Sifei Liu[2⋆]
Min Sun[1,3] Ming-Hsuan Yang[4,5,6]

[1]National Tsing-Hua University [2]NVIDIA
[3]Joint Research Center for AI Technology and All Vista Healthcare
[4]Yonsei University [5]Google Research [6]UC Merced

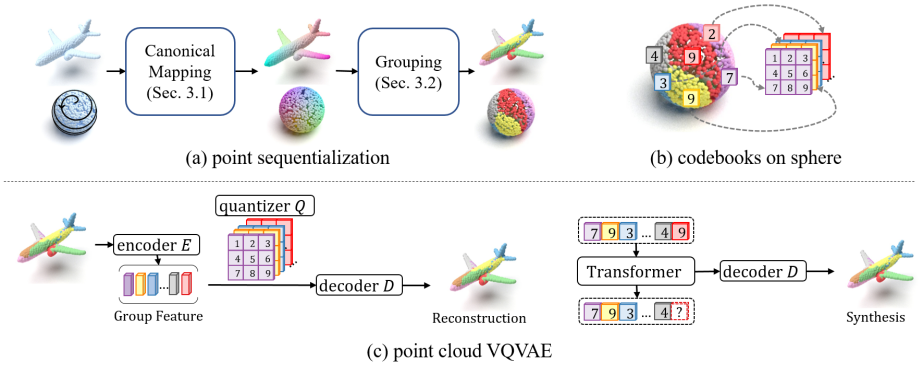

Fig. 1: Given a point cloud, we first decompose it into a sequence of perceptually meaningful shape compositions via a canonical auto-encoder in (a). A group of codebooks is then learned on the sequentialized shape compositions in (b). Finally, we introduce an autoregressive model for point cloud generation in (c).

Abstract. With the capacity of modeling long-range dependencies in sequential data, transformers have shown remarkable performances in a variety of generative tasks such as image, audio, and text generation. Yet, taming them in generating less structured and voluminous data formats such as high-resolution point clouds have seldom been explored due to ambiguous sequentialization processes and infeasible computation burden. In this paper, we aim to further exploit the power of transformers and employ them for the task of 3D point cloud generation. The key idea is to decompose point clouds of one category into semantically aligned sequences of shape compositions, via a learned canonical space. These shape compositions can then be quantized and used to learn a context-rich composition codebook for point cloud generation. Experimental results on point cloud

⋆ Equal contribution

reconstruction and unconditional generation show that our model performs favorably against state-of-the-art approaches. Furthermore, our model can be easily extended to multi-modal shape completion as an application for conditional shape generation. The source code and trained models can be found at https://github.com/AnjieCheng/CanonicalVAE.

Keywords: 3D Shape Generation; Autoregressive models

1 Introduction

In the past few years, transformers not only dominate the natural language processing area [27,9,2], but also consistently show remarkable performance in a variety of vision tasks such as image classification [10], semantic and instance segmentation [20,24] and image generation [12]. Compared to convolutional neural networks, transformers learn dependencies between visual elements from scratch without making any prior assumptions about data structure. As a result, they are more flexible and capable of capturing long-range dependencies in sequential data. Such property is especially desirable in the autoregressive generation of globally coherent long-range sequential data, such as high-resolution images. Indeed, promising performance of autoregressive generation via transformers has been demonstrated in [12] for image generation.

However, employing transformers for autoregression generation on less structured data, such as raw point clouds, has seldom been explored hitherto. The main challenge is that the sequentialization of such data is non-trivial. Naively arranging a point cloud as a sequence of points will break shape structural information and is computationally infeasible. To resolve the limitation, similar to the grid-like patches applied to 2D images [12,23], one can uniformly divide a point cloud into several groups and lay them out as a sequence. However, learning the sequential shape representation can be difficult since such shape compositions are entirely random.

In this paper, we resolve these issues and take the first step to employ transformers in 3D point cloud generation. The key idea is to decompose a point cloud into a sequence of semantically meaningful shape compositions, which are further encoded by an autoregressive model for point cloud generation. Specifically, we first learn a mapping function that maps each point cloud onto a shared canonical sphere primitive. Through a canonical auto-encoder with a few self-supervised objectives, the mapping function ensures that corresponding parts (e.g., tails of two airplanes) from different instances overlap when mapped onto the canonical sphere, i.e., dense correspondences of different instances are established and are explicitly represented via a canonical sphere (see Fig. 1 (a) middle). Grouping is carried out on the canonical sphere to obtain the shape compositions (see Fig. 1 (a) right). Thanks to the correspondence constraint, each group on the canonical sphere essentially corresponds to the same semantic part on all point cloud instances. As a result, each point cloud can be sequentialized into a set of shape compositions that are semantically aligned across different instances. Finally, we train a vector-quantized autoencoder (VQVAE) using these

sequentialized point cloud sequences, followed by learning a transformer that resolves the point cloud generation task.

The main contributions of this work include:

- We propose a novel transformer-based autoregressive model for point cloud generation.
- We introduce a canonical autoencoder and a self-supervised point grouping network to sequentialize point clouds into semantically aligned sequences of shape compositions.
- We train a VQVAE with group-specific codebooks, followed by learning a transformer model using the sequentialized point clouds to resolve the task of point cloud generation.
- Qualitative and quantitative comparisons demonstrate that our model can achieve state-of-the-art performance for point cloud auto-encoding and generation. We also extend our model to multi-modal shape completion as an application for conditional shape generation.

2 Related work

2.1 3D Shape Generation

3D shape generation targets at learning generative models on 3D shapes including but not limited to point clouds [33], voxels [29], implicit surfaces [5], etc. Some early works generate point clouds with a fixed-dimensional matrix [1,13]. Although these models can be easily applied with existing generative models (e.g., [13] a variational auto-encoder or [1] a generative adversarial network), they are restricted to generating a fixed number of points and are not permutation invariant. Several works [34,14,18] mitigate this issue by mapping a primitive to a point cloud. Specifically, they attach a global latent code to each sampled point on the primitive and then apply the transformation to the concatenation. In this work, we also generate point clouds from a shared primitive. Different from existing works, we decompose the primitive into different compositions and represent each group as a local latent code. Thanks to the local latent code representation, our model can generate point clouds with more fine-grained details.

Several recent works consider point clouds as samples from a distribution and propose different probabilistic models to capture the distribution. For example, PointFlow (PF) [33] applies normalizing flow to 3D point clouds. ShapeGF [3] models the gradient of the log-density field of shapes and generates point clouds using Langevin dynamics. DFM [21] and PVD [38] are both diffusion models that learn a probabilistic model over a denoising process on inputs.

Most related to our work, PointGrow [25], AutoSDF [22] and ShapeFormer [32] also use autoregressive models to generate 3D shapes. Specifically, the PointGrow discretizes point coordinates of a point cloud to fixed values and generates a shape in a point-wise manner following the spatial order. However, due to the large number of points in each point cloud, the size of generated point clouds is limited. Instead, our model decomposes a point cloud into compact shape

compositions that are both semantically meaningful and more efficient to process. The AutoSDF learns an autoregressive model on volumetric Truncated-Signed Distance Field (T-SDF), where a 3D shape is represented as a randomly permuted sequence of latent variables, while the proposed method takes raw point clouds as inputs and decomposes them into ordered sequences of shape compositions. The ShapeFormer represents 3D shapes as voxel grids that have limited capacity to encode details. Moreover, the ShapeFormer uses row-major order to turn voxels into sequences, which can be sensitive to object rotation and missing parts.

2.2 Transformers for Point Clouds

Recently, several works have started to apply transformers for point clouds due to their impressive representation ability. For example, Zhao et al. [37] design self-attention layers for point clouds and show improvement in point cloud classification and segmentation. Guo et al. [15] use a transformer to enhance local information within the point cloud. Nico et al. [11] propose to extract local and global features from point clouds and relate two features with the attention mechanism. Xiang et al. [31] leverage a transformer to extract local features with a focus on shape completion. Kim et al. [17] work on the shape generation task, in which they propose to encode a hierarchy of latent codes for flexible subset structures using a transformer. However, these works only employ transformer architectures on the encoding side. In contrast, we use transformer architectures as the decoder and focus on the autoregressive generation process.

3 Method

We propose a framework that employs transformers in the task of point cloud generation. The overview of our framework is illustrated in Fig. 1. Given a point cloud, our method first maps it onto a canonical sphere in Section 3.1. By adopting self-supervised training objectives (e.g., Chamfer distance loss, Earth Mover distance loss), we ensure that the semantically corresponding points from different instances overlap on the canonical sphere. Thus, by grouping points on the canonical sphere and serializing them as a sequence, we equivalently decompose each point cloud into an ordered sequence of shape compositions. This process is described in Section 3.2. We then learn a vector-quantized variational auto-encoder (VQVAE) using the sequentialized point clouds in Section 3.3, with codebooks as a library of the shape compositions in the point clouds. Finally, a transformer is trained for point cloud generation in Section 3.4.

3.1 Point Cloud Sequentialization

Different from convolutional neural networks, transformer models require sequential data as inputs. Taking the transformer in [12] as an example, an image is first sequentialized by starting from the top left patch and sequentially moving to the bottom right patch in a zigzag order. The underline key is that all images

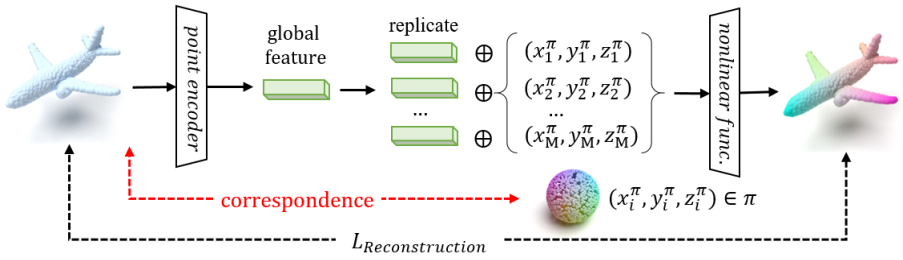

Fig. 2: Our Canonical Auto-encoder contains two parts: a point encoder that produces the shape feature of an input point cloud; a nonlinear function that decodes the canonical sphere, along with the shape feature, back to the input shape.

are sequentialized by the same "zigzag" order, allowing the transformer to learn to predict patches based on their surrounding context. However, when it comes to an orderless data structure such as point clouds, it remains challenging to sequentialize the unordered points similarly to images.

To resolve this issue, two key questions need to be answered: a) what forms a unit in a sequentialized point cloud? b) how to sort these units in a consistent order for different point clouds? In this section, for ease of understanding, we consider a single point as a unit in each point cloud and demonstrate how to sort these points in a consistent order for different point clouds. We then discuss how to learn more semantically meaningful and memory-friendly units (i.e., shape compositions) in the next section.

To sequentialize all point clouds in a consistent order, we first map them onto a shared canonical sphere. Each point on the sphere is from semantically corresponding points on different point clouds. Thus, by finding an order for points on the sphere, all the point clouds can be sequentialized accordingly. For instance, if a point on the canonical sphere is labeled as the kth point in a sequence, then all its corresponding points in different point clouds are also labeled as the kth point in the sequentialized point cloud. In the following, we first discuss how to map point clouds to a canonical sphere and then describe the order we choose to sort all points.

Mapping point clouds to a canonical sphere. We learn a nonlinear function to associate all point clouds with a canonical sphere π and thus obtain their correspondences, as inspired by [8]. As shown in Fig. 2, given an input point cloud $x \in \mathcal{R}^{M\times 3}$ including M points, we first encode it as a 256-dimensional global latent code by an encoder, e.g., DGCNN [28]. We then replicate the global code and concatenate it with points sampled from a canonical unit sphere as the input to the nonlinear function. At the output end of the function, we reconstruct the input point cloud via a Chamfer loss and an Earth Mover distance loss. The function thus performs a nonlinear transformation between the sphere and an individual instance, conditioned on its latent shape feature. We follow [8], in

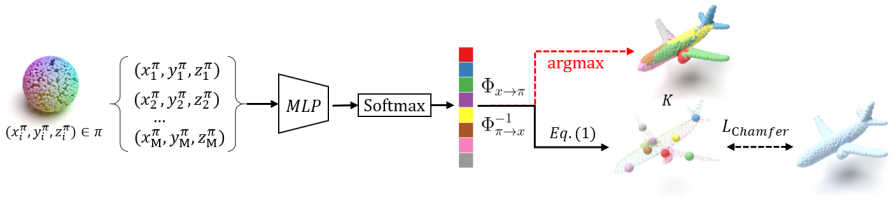

Fig. 3: We learn a self-supervised network to decompose the canonical sphere into non-overlapping groups. With canonical mapping, point clouds are simultaneously decomposed into semantically aligned shape compositions. See Sec. 3.2.

which the nonlinear function is a parameterized model implemented as a neural network. We name the combination of the shape encoder and the nonlinear function as *Canonical Auto-encoder*. For any point x_i from an input point cloud x, to locate its corresponding point on the sphere, we can (1) search its nearest neighbor $\hat{x}_i$ on the reconstructed point cloud $\hat{x}$, then (2) trace the point π_i on the sphere where $\hat{x}_i$ is mapped from.

As proved by Cheng et al. [8], points of all the reconstructed shapes are "re-ordered" according to the point indices of the canonical sphere (see Fig. 2). We note that as the key difference, we remove the "point cloud-to-sphere mapping network" designed in [8] to avoid processing the points that are inaccurately mapped to locations far away from the sphere surface. Our design also simplifies the following process in Sec. 3.2, i.e., the grouping and sequentialization can be conducted on a complete sphere instead of a subset of it [8]. For brevity, in the following, we denote the canonical mapping process as Φ, the corresponding point of $x_i \in x$ on the sphere π as $\Phi_{x\to\pi}(x_i)$, and the corresponding point of $\pi_i \in \pi$ on x as $\Phi_{\pi\to x}^{-1}(\pi_i)$.

Canonical sphere serialization. Since all point clouds are aligned with the canonical sphere by the Canonical Auto-encoder, any order defined on the canonical sphere can be easily transferred to any point cloud. In this paper, we traverse the canonical sphere from the pole with a Fibonacci spiral (see Fig. 1 (a)) and serialize the points in the spiral order along the way. As a result, the index of a point in a point cloud can be easily determined as the index of its corresponding point on the canonical sphere.

3.2 Shape Composition Learning

Though the re-ordered point clouds in Sec. 3.1 can be readily represented as sequences of points, such sequences usually include thousands of points and are intractable to be modeled by autoregressive models. In this section, we introduce a more semantically meaningful and memory-efficient unit for point cloud sequentialization.

Specifically, we decompose the points of each point cloud instance into G groups ($G = 128$ throughout all experiments). We call each group a shape

composition, which is analogous to an image patch in the 2D domain. As discussed above, since each point cloud is aligned to the canonical sphere, decomposing the point clouds is thus equivalent to decomposing the canonical sphere. A straightforward way is to uniformly divide the sphere by randomly sampling G points as center points and assigning each point to the nearest center point. However, this approach does not take the semantic prior into consideration and often produces discontinuous shape compositions.

Instead, we introduce a self-supervised grouping network as shown in Fig. 3. For each point on the sphere, we predict its group assignment by a multi-layer perceptron (MLP), followed by a SoftMax activation function. Both are shared by all the points on the sphere. This results in an assignment probability map P for all points $q \in \pi$, where P_i^j indicates the probability of assigning point π_i to the jth group.

To train this network and produce reasonable shape compositions, at the output end, we transfer the grouping assignment probability from each point $\pi_i \in \pi$ on the canonical sphere to its corresponding point $\Phi_{\pi \to x}^{-1}(\pi_i)$ on the input point cloud. As a result, we obtain an assignment probability map for each point cloud instance. To ensure the learned grouping captures the structure of a point cloud and formulates a decent abstraction of it, we compute G structure points $K \in \mathcal{R}^{G \times 3}$ [7], where each K_j is computed as:

$$K_j = \sum_{i=1}^{m} \Phi_{\pi \to x}^{-1}(\pi_i) P_i^j \quad \text{with} \quad \sum_{i=1}^{m} P_i^j = 1 \quad \text{for} \quad j = 1, 2, ..., G \tag{1}$$

Finally, a Chamfer distance is applied between the predicted structure points K and the input point cloud x, as $\mathcal{L}_{CD}(K, x)$.

After training, we assign each point on π to the group with the highest probability. To assign each point $x_i \in x$ to a group, we simply let it take the group label of its corresponding point $\Phi_{x \to \pi}(x_i)$ on the sphere. In different point clouds, points on corresponding semantic parts share the same grouping assignment through the canonical sphere. As a result, any point cloud instance is decomposed into a set of shape compositions, each of which includes the points assigned to this group.

These shape compositions form the basic shape units and are further sorted into a sequence following the Fibonacci spiral order described in Sec. 3.1. In the following sections, we still denote each point cloud as x for brevity, but we assume that all point clouds have been processed into sequences of shape compositions using the method described above.

3.3 Point Cloud Reconstruction through VQVAE

Now we introduce how to utilize the sequentialized point clouds to learn a VQVAE. Our VQVAE includes three components, an encoder E, a decoder D, and a vector quantizer Q, as shown in Fig. 1(c). We discuss each component in detail in the following.

Point cloud encoding. Given a sequentialized instance x, we first compute the point-wise feature by the encoder E. To compute the feature of each shape composition, we apply max-pooling to aggregate features of all points belonging to this shape composition. We denote the feature of the jth group as z^j.
Point cloud sequence quantization. Next, we quantize the group feature vectors z by a group of jointly learned codebooks. In conventional VQVAEs, a single codebook is learned and shared by all the compositions (e.g., image patches in 2D VQVAEs [26]). However, we found that this strategy often leads to low code utilization. The model struggles to capture the diverse feature of all groups via only a few codes while leaving all the others unused. Such design leads to the usage of an unnecessarily large codebook and inferior reconstruction results.

To resolve this issue, we learn an independent codebook for each group where at least one code from each codebook will be utilized. Since each codebook is only responsible for representing one particular shape composition, we can safely reduce the number of codes and the dimension of each code without degrading the performance. Specifically, given z^j for group j, we first reduce its dimension from 256 to 4 to obtain a low dimensional feature $\hat{z}^j$ by learning a linear projection. We then quantize $\hat{z}^j$ into z_q^{low} by finding its nearest neighbor token from the corresponding group codebook Z^j. Note that each group codebook Z^j contains 50 4-dimensional latent codes. Finally, we project the matched codebook token back to the high-dimension embedding space and denote the quantized group feature as z_q. We note that the recent work [35] also shows that this dimension reduction process improves the reconstruction quality. We show in Sec. 4.4 that our design choices for codebook significantly increase codebook usage.
Point cloud sequence decoding. To recover the input point cloud from z_q, we concatenate each point in the canonical sphere π with the corresponding quantized group feature and feed the concatenation to the decoder D.
VQVAE training. We use the same network architecture as in Sec. 3.1 for both E and D. We train them together with the codebooks by applying the Chamfer and Earth Mover Distance between the reconstructed point cloud $\hat{x}$ and the input point cloud x:

$$\mathcal{L}_{Quantization} = \mathcal{L}_{CD}(x, \hat{x}) + \mathcal{L}_{EMD}(x, \hat{x}) + \left\| sg[z_q^{low}] - \hat{z} \right\|_2^2 \tag{2}$$

where $sg[\cdot]$ is the stop-gradient operation. We use exponential moving average (EMA) [4] to maintain the embeddings in each of the group codebooks Z^j.

3.4 Point Cloud Generation through Transformers

Given the learned codebooks, we can represent a point cloud sequence as a sequence of codebook token indices in order to learn an auto-regressive model.

Specifically, we represent the codebook token indices as $s_1, s_2, ..., s_G$, where G is the total group number. Given indices $s_{<i}$, we train a transformer model to predict the distribution of possible next indices s_i based on its preceeding codebook tokens as:

$$\prod_{i=1}^{G} p(s_i | s_1, s_2, ..., s_{i-1}) \tag{3}$$

Table 1: Shape auto-encoding on the ShapeNet dataset. The best results are highlighted in bold. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

Dataset	Metric	AtlasNet		PF	ShapeGF	DPM	Ours	Oracle
		Sphere	Patches					
Airplane	CD	1.002	0.969	1.208	0.966	0.997	**0.889**	0.837
	EMD	2.672	2.612	2.757	2.562	2.227	**2.122**	2.062
Chair	CD	6.564	6.693	10.120	**5.599**	7.305	6.177	3.201
	EMD	5.790	5.509	6.434	4.917	4.509	**4.218**	3.297
Car	CD	5.392	5.441	6.531	5.328	5.749	**5.050**	3.904
	EMD	4.587	4.570	5.138	4.409	4.141	**3.614**	3.251

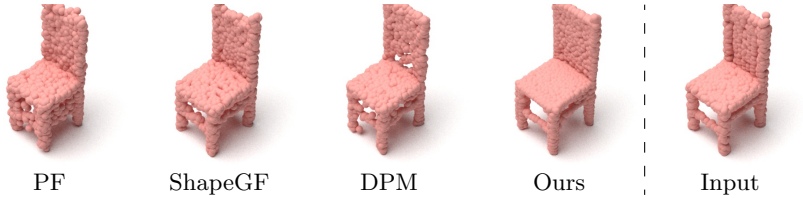

Fig. 4: Auto-encoding (reconstruction) results. We also shown results from PF (PointFlow) [33], ShapeGF [3], and DPM [21] on the left for comparison.

The training objective is to minimize the negative log-likelihood by

$$\mathcal{L}_{Transformer} = \mathbb{E}_{x\sim p(x))} \left[-\log p(s) \right] \tag{4}$$

The architecture of our transformer model is similar to as [12], where the indices are projected into the embedding space at each position together with an additive positional embedding. However, since each of our groups owns its own codebook, we do not use a shared embedding space for all codebook token indices. Instead, each index s_i is mapped to the embedding space using a separate linear layer.

Unconditional generation. With the learned transformer model, unconditional shape generation is carried out by sampling token-by-token from the output distribution. The sampled tokens are then fed into the decoder D in the VQVAE to decode output shapes.

Conditional generation. Going beyond unconditional generation, we further incorporate our transformer with a conditional input. Specifically, given a condition c (e.g., a depth image), we use our transformer to generate a shape that matches the semantic meaning of c. For instance, if c is a depth image, then the transformer is expected to generate a 3D shape that renders the depth image from the given viewpoint. To this end, we first encode the condition c into a feature vector in the same dimension of token embedding, then prepend the feature vector before the first token embedding.

4 Experiments

Datasets. Following previous works [33,3,17], we conduct our auto-encoding and generation experiments on the airplane, chair, and car category from the ShapeNet [6] dataset. For the multi-modal shape completion task, we follow [38] which uses the ShapeNet rendering data from Genre [36]. For baselines that take additional partial point clouds as inputs, we use the data provided by [38].

Evaluation metrics. For a fair comparison, we follow prior works [34,33,3] and use the symmetric Chamfer Distance (CD) as well as the Earth Mover's Distance (EMD) to evaluate the quality of the reconstructed point clouds. To evaluate the quality of the unconditionally generated point clouds, we use the Minimum Matching Distance (MMD) [1], the Coverage Score (COV) [1], and the 1-NN classifier accuracy (1-NNA) [33]. To evaluate the multi-modal shape completion performance for the conditional generation task, we follow Wu et al. [30] that uses a) the Total Mutual Difference (TMD) to measure the generation diversity and b) the Minimal Matching Distance (MMD) to measure the completion quality with Chamfer Distance (CD). We normalize each point cloud for all generation experiments to a unit sphere before measuring the metrics.

4.1 Shape Auto-encoding

We first evaluate how well our model can approximate a shape with quantized features. We quantitatively compare our results against the following state-of-the-art point cloud auto-encoders: AtlasNet [14] variants that deform from patches and from sphere, respectively, PointFlow (PF) [33], ShapeGF [3], and DPM [21]. Following [3], we also report the lower bound of the reconstruction errors in the "Oracle" column. As shown in Table 1, our method consistently outperforms other methods when measured by EMD. Note that EMD is usually considered a better metric to measure a shape's visual quality [38] as it requires the outputs to have the same density as the ground-truth shapes [19]. This suggests that our reconstructed point clouds have more uniformly distributed points on the surface. We also provide qualitative results compared to baselines in Figure 4 to validate the effectiveness of our model.

4.2 Unconditional Generation

We quantitatively compare our method with the following state-of-the-art generative models: PointGrow [25], ShapeGF [3], SP-GAN [18], PointFlow (PF) [33], SetVAEF [17], DPM [21], and PVD[38]. We summarize the quantitative results in Table 2. For most of the metrics, our model has comparable, if not better, performance than other baselines. This suggests that our model is capable of generating diverse and realistic samples. We provide qualitative results comparing to baselines in Figure 5.

Among these baselines, PointGrow is most relevant to our work that generates point clouds in an autoregressive manner. Our model significantly outperforms

Table 2: Shape generation results. ↑ 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 (%, ↓)	
Category	Model	CD	EMD	CD	EMD	CD	EMD
Airplane	PointGrow	3.07	11.64	10.62	10.62	99.38	99.38
	ShapeGF	**1.02**	6.53	**41.48**	32.84	80.62	88.02
	SP-GAN	1.49	8.03	30.12	23.21	96.79	98.40
	PF	1.15	6.31	36.30	38.02	85.80	83.09
	SetVAE	1.04	**6.16**	39.51	38.77	89.51	87.65
	DPM	1.10	7.11	36.79	25.19	86.67	90.49
	PVD	1.12	6.17	40.49	**45.68**	**80.25**	**77.65**
	Ours	**0.83**	**5.50**	**45.67**	**44.19**	**63.45**	**71.60**
Chair	PointGrow	16.23	18.83	12.08	13.75	98.05	99.10
	ShapeGF	**7.17**	**11.85**	**45.62**	44.71	61.78	64.27
	SP-GAN	8.51	13.09	34.74	26.28	77.87	84.29
	PF	7.26	12.12	42.60	**45.47**	65.56	65.79
	SetVAE	7.60	12.10	42.75	40.48	65.79	70.39
	DPM	**6.81**	11.91	43.35	42.75	64.65	69.26
	PVD	7.65	11.87	**45.77**	45.02	**60.05**	**59.52**
	Ours	7.37	**11.75**	**45.77**	**46.07**	**60.12**	**61.93**
Car	PointGrow	14.12	18.33	6.82	11.65	99.86	98.01
	ShapeGF	**3.63**	9.11	**48.30**	44.03	**60.09**	61.36
	PF	3.69	**9.03**	**44.32**	**45.17**	63.78	**57.67**
	SetVAE	**3.63**	9.05	39.77	37.22	65.91	67.61
	DPM	3.70	9.39	38.07	30.40	74.01	73.15
	PVD	3.74	9.31	43.47	39.49	65.62	63.35
	Ours	**3.31**	**8.89**	41.76	**47.72**	**55.68**	**57.81**

PointGrow in all metrics because PointGrow scales poorly [25] when generating large point sets. In contrast, our method can generate shapes in an arbitrary resolution ranging from low to high with sharp details within a single model. We show point clouds generated with different resolutions compared to PointGrow in Figure 6.

4.3 Conditional Generation

During inference, our transformer model generates a sequence by a probabilistic sampling of each token, which naturally allows multi-modal generation. On the other hand, the shape completion problem is multi-modal in nature since the incompleteness introduces significant ambiguity [30]. Motivated by this property, we extend our approach to shape completion as an application for conditional shape generation. Specifically, we use a depth map as the input condition to the transformer model. We employ a ResNet50 [16] encoder to extract global features from the depth map and prepend the feature vector to the transformer as discussed in follow Sec. 3.4.

We compare with two state-of-the-art approaches on multi-modal shape completion: MSC [30] and PVD [38]. Note that both MSC [30] and PVD [38] take aligned point clouds as inputs. Therefore, they require additional camera

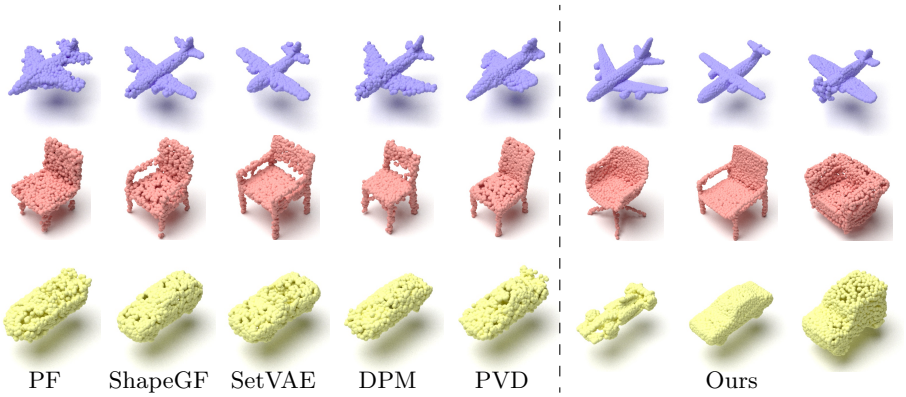

Fig. 5: Shape generation results. We shown results from PF (PointFlow) [33], ShapeGF [3], SetVAE [17], DPM [21], and PVD [38].

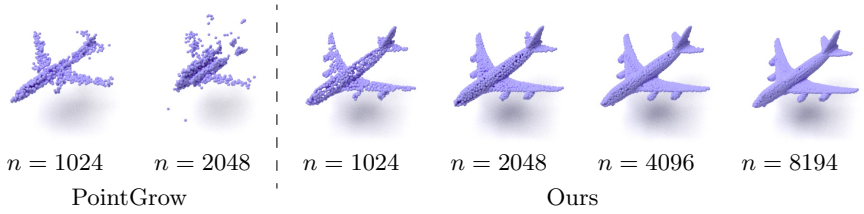

Fig. 6: High-resolution generation results comparing to PointGrow [25]. n refers to the number of output points.

parameters to obtain partial scans from the depth map. We present quantitative comparison results in Table 3 and show that our approach can achieve comparable or even better performance without requiring additional camera parameters. We also report results using different temperatures, which suggests that our model can provide controllable diversity by scaling the temperature parameter.

We visualize qualitative results in Figure 7. In general, our model produces shapes with better visual quality. However, due to the inherent ambiguity in single-view reconstruction, it is difficult to infer the real scale of objects without knowing the camera parameters, especially for depth maps rendered from the side views. For instance, given a depth map of a chair from side viewpoints, our model generates plausible but wider chairs than the ground truth 3D shapes, as shown in the first row of Fig. 7.

4.4 Ablation Study

Effectiveness of the canonical mapping function. Table 4 shows ablations on the effectiveness of our canonical mapping function, with empirical results

Table 3: Multi-modal completion on the Chair dataset. t denotes the temperature scaling factor. ↑ means the higher the better, ↓ means the lower the better. MMD and TMD are both multiplied by 10^3.

Category	Metric	Input	MMD (↓)	TMD (↑)
Airplane	MSC	Depth+Camera	1.475	0.925
	PVD	Depth+Camera	1.012	2.108
	Ours (t=1)	Depth	0.663	1.449
	Ours (t=2)	Depth	0.673	2.406
	Ours (t=3)	Depth	0.684	2.352
Chair	MSC	Depth+Camera	6.372	5.924
	PVD	Depth+Camera	5.042	7.524
	Ours (t=1)	Depth	5.142	6.553
	Ours (t=2)	Depth	5.261	8.174
	Ours (t=3)	Depth	6.427	13.341

Table 4: Ablation study on the effectiveness of primitive grouping on the Chair dataset. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

	Canonical Grouping			Uniform Grouping		
#groups	16	128	256	16	128	256
CD (↓)	7.542	6.177	7.933	8.646	7.676	7.278
EMD (↓)	4.466	4.218	4.645	4.772	4.541	4.627

on CD and EMD metrics using the ShapeNet Chair dataset. We report variants of our model using a different number of groups G and an alternative way to segment the canonical sphere (in the "Uniform Grouping" column). Specifically, we uniformly sample G points on the sphere as centers and use the nearest neighbor search to assign all points on the sphere to its nearest center point. Though straightforward, this approach does not provide any semantic correspondence across shape instances; thereby, our model consistently performs better than this baseline in different G settings. For the uniform setting, the auto-encoding results directly relate to the G because a larger G results in a finer segmentation. However, our model performs the best with a moderate $G = 128$. Since the size of each group is automatically determined in our model (i.e. rather than an equal size), therefore, some groups may include only a few points when G is large. This tends to hurt the encoding performance and results in over-fitting.

Effectiveness of group-wise codebooks. Table 5 demonstrates the effectiveness of the latent reduction and group-wise codebook (see Sec. 3.3) in our vector quantizer Q. In addition to CD and EMD, we also report the codebook usage for each model. The usage is computed as the percentage of codes that have been utilized at least once over the entire test set. For a fair comparison with our full model that uses 128 group codebooks in size 50, we use a global codebook in size 5000 ($\approx 128\times50$) for each variant that does not use group-wise codebooks. Our full model performs the best with dimension reduction from 256 to 4 together with a group-wise codebook. Reducing the lookup dimension in the codebook

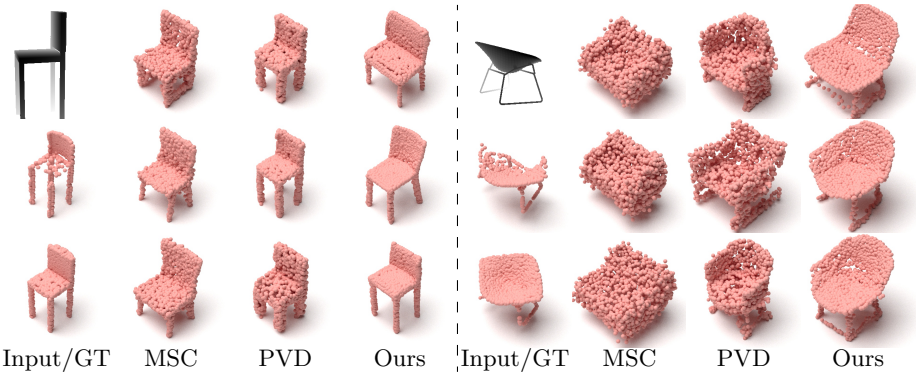

Fig. 7: Multi-modal shape completion results. We shown 4 samples comparing to MSC and PVD. The input depth-map, partial point cloud, and reference ground-truth shape for each sample is shown in the first column, respectively (from top to bottom).

Table 5: Ablation study on using different vector quantization on the Chair dataset. CD is multiplied by 10^4 and EMD is multiplied by 10^2.

Dimension Reduction	✗	256→64	256→4	256→64	256→4
Grouped Codebook	✗	✗	✗	✓	✓
CD (↓)	7.139	6.561	6.298	6.442	6.177
EMD (↓)	4.376	4.272	4.283	4.228	4.218
Codebook Usage (%, ↑)	11.72	21.04	35.72	70.22	79.28

and using a group-wise codebook significantly boost the codebook usage, thereby achieving better auto-encoding quality.

5 Conclusions

We propose a transformer-based autoregressive model for point cloud generation. The key idea is to decompose a point cloud into a sequence of semantically aligned shape compositions in a learned canonical space. We show that these compositions can be further used to learn a group of context-rich codebooks for point cloud generation. Experimental results demonstrate that the proposed method can achieve state-of-the-art performance for point cloud auto-encoding and generation. Finally, we show that our model can be easily extended to multi-modal shape completion as an application for conditional shape generation.

Acknowledgments. This work was supported in part by the MOST, Taiwan under Grants 110-2634-F-002-051, MOST Joint Research Center for AI Technology, All Vista Healthcare, and NSF CAREER grant 1149783. We thank National Center for High-performance Computing (NCHC) for providing computational and storage resources.

References

1. Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L.: Learning representations and generative models for 3d point clouds. In: ICML. pp. 40–49 (2018) 3, 10
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. NeurIPS (2020) 2
3. 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) 3, 9, 10, 12
4. Cai, Z., Ravichandran, A., Maji, S., Fowlkes, C., Tu, Z., Soatto, S.: Exponential moving average normalization for self-supervised and semi-supervised learning. In: CVPR (2021) 8
5. Chan, E.R., Lin, C.Z., Chan, M.A., Nagano, K., Pan, B., Mello, S.D., Gallo, O., Guibas, L., Tremblay, J., Khamis, S., Karras, T., Wetzstein, G.: Efficient geometry-aware 3D generative adversarial networks. In: CVPR (2022) 3
6. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., et al.: Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:1512.03012 (2015) 10
7. Chen, N., Liu, L., Cui, Z., Chen, R., Ceylan, D., Tu, C., Wang, W.: Unsupervised learning of intrinsic structural representation points. In: CVPR (2020) 7
8. Cheng, A.C., Li, X., Sun, M., Yang, M.H., Liu, S.: Learning 3d dense correspondence via canonical point autoencoder. In: NeurIPS (2021) 5, 6
9. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018) 2
10. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020) 2
11. Engel, N., Belagiannis, V., Dietmayer, K.: Point transformer. IEEE Access **9**, 134826–134840 (2021) 4
12. Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: CVPR (2021) 2, 4, 9
13. Gadelha, M., Wang, R., Maji, S.: Multiresolution tree networks for 3d point cloud processing. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 103–118 (2018) 3
14. Groueix, T., Fisher, M., Kim, V.G., Russell, B.C., Aubry, M.: A papier-mâché approach to learning 3d surface generation. In: CVPR. pp. 216–224 (2018) 3, 10
15. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: Pct: Point cloud transformer. Computational Visual Media p. 187–199 (Apr 2021) 4
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016) 11
17. 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, 10, 12
18. 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) 3, 10
19. Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S.M.: Morphing and sampling network for dense point cloud completion. In: AAAI. pp. 11596–11603 (2020) 10

20. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021) 2
21. 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) 3, 9, 10, 12
22. Mittal, P., Cheng, Y.C., Singh, M., Tulsiani, S.: Autosdf: Shape priors for 3d completion, reconstruction and generation. arXiv preprint arXiv:2203.09516 (2022) 3
23. Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. NeurIPS **32** (2019) 2
24. Strudel, R., Garcia, R., Laptev, I., Schmid, C.: Segmenter: Transformer for semantic segmentation. In: ICCV (2021) 2
25. 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) 3, 10, 11, 12
26. Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. NeurIPS **30** (2017) 8
27. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS (2017) 2
28. 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) 5
29. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. NeurIPS (2016) 3
30. 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) 10, 11
31. Xiang, P., Wen, X., Liu, Y.S., Cao, Y.P., Wan, P., Zheng, W., Han, Z.: SnowflakeNet: Point cloud completion by snowflake point deconvolution with skip-transformer. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV) (2021) 4
32. Yan, X., Lin, L., Mitra, N.J., Lischinski, D., Cohen-Or, D., Huang, H.: Shapeformer: Transformer-based shape completion via sparse representation. In: CVPR. pp. 6239–6249 (2022) 3
33. 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, 9, 10, 12
34. Yang, Y., Feng, C., Shen, Y., Tian, D.: Foldingnet: Point cloud auto-encoder via deep grid deformation. In: CVPR. pp. 206–215 (2018) 3, 10
35. Yu, J., Li, X., Koh, J.Y., Zhang, H., Pang, R., Qin, J., Ku, A., Xu, Y., Baldridge, J., Wu, Y.: Vector-quantized image modeling with improved vqgan. ICLR (202) 8
36. Zhang, X., Zhang, Z., Zhang, C., Tenenbaum, J.B., Freeman, W.T., Wu, J.: Learning to Reconstruct Shapes From Unseen Classes. In: NeurIPS (2018) 10
37. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16259–16268 (2021) 4
38. 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) 3, 10, 11, 12