ShapeFormer: Transformer-based Shape Completion via Sparse Representation

Supplementary Material

Xingguang Yan[1] Liqiang Lin[1] Niloy J. Mitra[2,3] Dani Lischinski[4] Daniel Cohen-Or[1,5] Hui Huang[1*]
[1]Shenzhen University [2]University College London [3]Adobe Research [4]Hebrew University of Jerusalem [5]Tel Aviv University

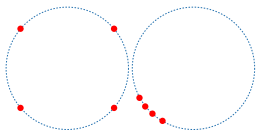
Abstract

*In this supplementary document, we first give a detailed description of the ambiguity measure, model architectures, and training/testing statistics in Appendix A. Then we show more visual comparisons between our method and previous methods for scans of both **high and low ambiguity** in Appendix B. Lastly, we will give more analysis on our method in Appendix C, such as a discussion of limitations. The code of our model is also included in the supplementary material.*

A. Implementation Details

A.1. Ambiguity measure for partial point cloud

The ambiguity for a partial point cloud measures the variety of its potential complete shapes. However, the direct measurement for ambiguity is difficult, if not impossible. In contrast, the incompleteness of a partial cloud toward its complete shape is relatively easy to compute. Although it can not fully reflect ambiguity (e.g., a top scan of a table as incomplete as a bottom scan could have a much greater ambiguity), the ambiguity is still strongly correlated. Hence, we seek to find a metric on the incompleteness of such a point cloud to indicate its ambiguity. Intuitively, we could use metrics like F-score [12] to measure the ratio of the approximate partial surface area toward the complete area. But as indicated in the inset figure, such measures will fail to differentiate the coverage difference of the partial cloud (red dots) to the complete one (in blue). Instead, we propose to use a metric based on Chamfer-L_2, which goes larger as the partial point cloud misses more global structure. Since the partial to complete distance is always negligible, we can only calculate the complete to partial distance. And to compare the ambiguity of scans on different shapes, we normalize the distance of a point according to its farthest distance in the complete shape. More specifically, we define the metric Amb evaluating the ambiguity of scan $\mathcal{C}$ given the complete point cloud as $\mathcal{B}$ as:



$$Amb(B, C) = \frac{1}{B}\Sigma_{x\in\mathcal{B}}\frac{\min_{y\in\mathcal{C}}||x-y||}{\max_{x'\in B}||x-x'||}, \quad (1)$$

Where B is the number of points in the complete cloud.

We sample 70 views for each shape, 64 of which are evenly sampled from the view sphere (via Fibonacci sampling), and the rest are the six orthogonal views. Then we sort these views according to the score. In Fig. 1, we use a teapot as an example to show the score distribution of these 70 scans. For scans with low ambiguity scores, the underlying shape's global structure is either captured or is clearly indicated by the captured shape salient features. For example, the scan covering the teapot's mouth, handle, and body can be completed easily. However, it would be more difficult to infer the complete shape when the score is high since it may have different global structures, and a single explanation is not satisfactory. As shown in the main paper, our method can better handle such scans than existing shape completion methods.

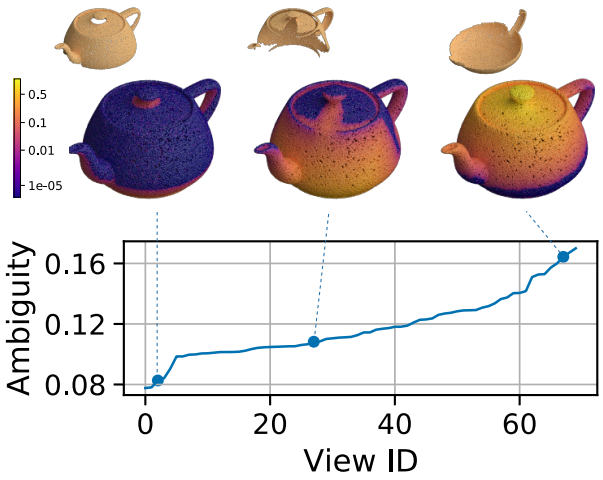


Figure 1. Viewing direction greatly influences the scan ambiguity. Our proposed scores for 70 scans of a teapot are shown in sorted order, with examples marked with their position on the curve. The example contains the scans (in gold insets) and complete shape color-coded scores for each point in it.

A.2. Architectures

We show the detailed architecture of VQDIF and ShapeFormer in Figs. 2 and 3, respectively. and the parameters of their sub-modules are listed in Tab. 1

VQDIF. As shown in Figure 2, VQDIF is an encoder-decoder architecture, where the encoder maps an input point

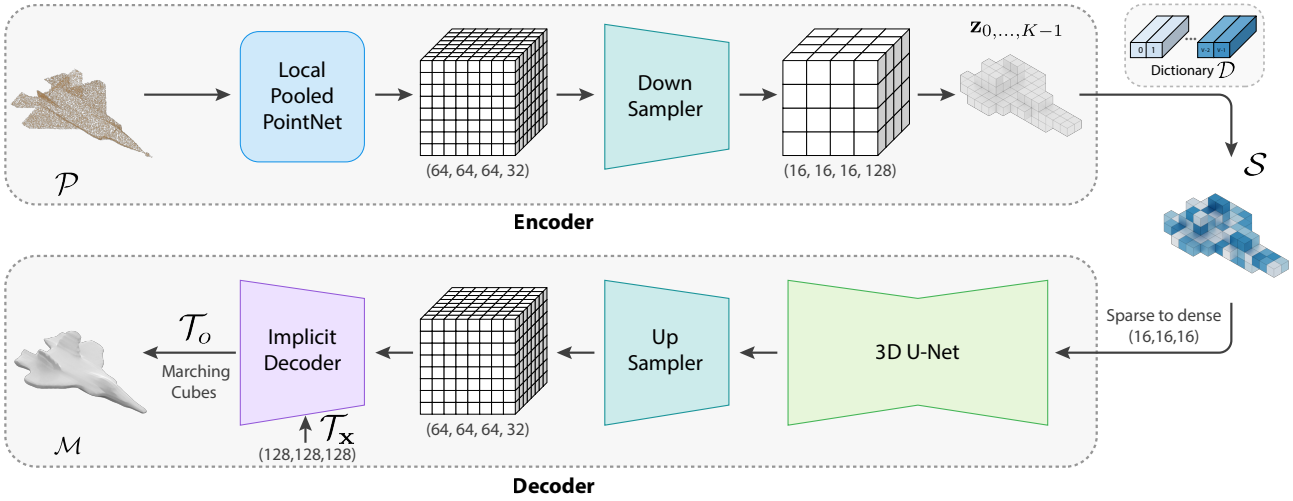


Figure 2. The architecture of VQDIF. The complete point cloud $\mathcal{P}$ is encoded to a feature grid and down-sampled into a lower resolution one. Its non-empty features are then flattened and quantized to form the VQDIF sequence which is then projected back to a feature grid, up-sampled and sent to an implicit decoder, from which the occupancy grid $\mathcal{T}_o$ of probes $\mathcal{T}_\mathbf{x}$ and the reconstruction $\mathcal{M}$ can be obtained.

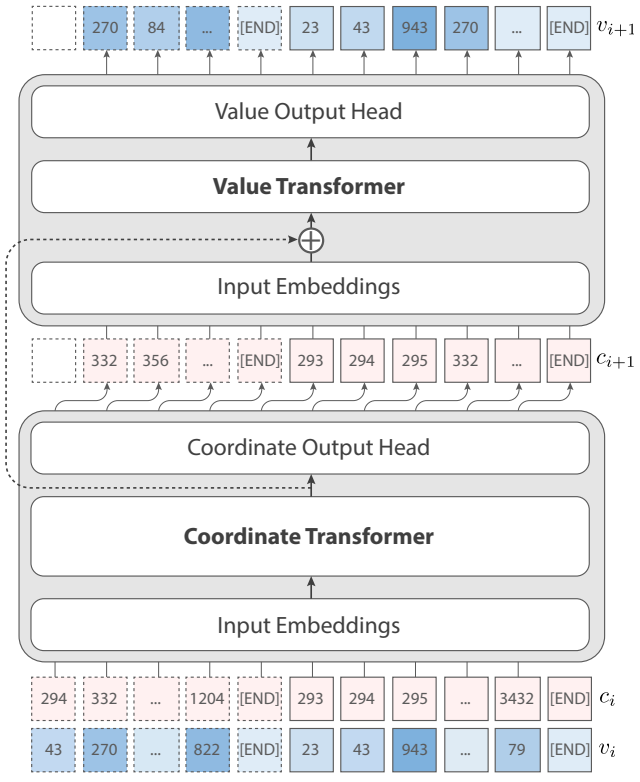


Figure 3. An extended view of ShapeFormer. Different from the figure in the main paper, we show the inside of each Transformer module. The input embeddings are obtained by additively mixing the location and value embeddings. And the output head converts the output embedding into categorical distributions.

Layer Name	Notes	Input Size
VQDIF		
Local Pooled Pointnet		$N \times 3$
Downsampler		
ConvLayer	k2s2p0	$64 \times 64 \times 64 \times 32$
ConvLayer	k1s1p0	$32 \times 32 \times 32 \times 64$
ConvLayer	k2s2p0	$64 \times 64 \times 64 \times 32$
ConvLayer	k1s1p0	$32 \times 32 \times 32 \times 64$
Quantizer		$16 \times 16 \times 16 \times 128$
UNet3D		$16 \times 16 \times 16 \times 128$
Upsampler		$16 \times 16 \times 16 \times 128$
Scaling	nearest mode	$16 \times 16 \times 16 \times 128$
ConvLayer	k3s1p1	$32 \times 32 \times 32 \times 128$
ConvLayer	k3s1p1	$32 \times 32 \times 32 \times 64$
Scaling	nearest mode	$32 \times 32 \times 32 \times 64$
ConvLayer	k3s1p1	$64 \times 64 \times 64 \times 64$
ConvLayer	k3s1p1	$64 \times 64 \times 64 \times 32$
Upsampler Output		$64 \times 64 \times 64 \times 32$
Implicit Decoder		$128^3 \times 3$
Implicit Decoder Output		$128^3 \times 1$
ShapeFormer		
Embedding Blocks	#4M	$K \times 2$
Coordinate Transformer Blocks $\times 20$	#251M	$K \times 1024$
Coordinate Output Heads	#4M	$K \times 4097$
Embedding Blocks	#4M	$K \times 2$
Value Transformer Blocks $\times 4$	#50M	$K \times 1024$
Value Output Heads	#4M	$K \times 4097$
Total params	#340M	
Trainable params	#323M	

Table 1. The detailed architecture information of our method. N is the point size. For both VQDIF and ShapeFormer, we list the input size of their components. For convolutional neural networks, the "k", "s", "p" stands for kernel size, stride, and padding, respectively. Also "ConvLayer" denotes the composition of CNN + ReLU + GroupNorm. We also list the number of parameters for each component and indicate them with #. The sequence length is denoted by K, with a maximum of 812.

cloud to a discrete sequence representation $\mathcal{S}$, while the decoder maps such a sequence to a deep implicit function $f(\mathbf{x})$. Unlike the main paper's completion pipeline, both the encoder and decoder only take complete input during training. The input to the encoder is a point cloud $\mathcal{P} \in \mathbb{R}^{N\times 3}$ representing the dense sampling of a shape or its partial observation. During the training phase, we use complete dense clouds with $N = 32768$ points to train VQDIF to capture local geometric details in the input. At test time, we use the trained encoder to directly encode partial point clouds, which may be sparse or dense.

The encoder first processes the input cloud with a local pooled PointNet [1] to obtain a feature grid. Similar to prior work [10], the local pooled PointNet aggregates features within a grid cell in contrast to the original PointNet, where all point features are pooled together to obtain a global feature. Specifically, we use a grid of resolution 64 with a feature size of 32.

Next, to reduce the number of local features, the high-resolution feature grid is down-sampled to lower resolution R, using several consecutive strided convolution blocks. As shown in Tab. 1, the parameters of these blocks are carefully set to have the least receptive field since a large receptive field lets each grid feature cover a larger region, reducing the sparsity of the representation. We can then extract the non-empty features by directly masking the encoded feature grid with the voxelized input point cloud (resolution R) thanks to the minimum receptive field. After flattening and quantizing the features (see the main paper), we get the 2-tuple sequence representation directly sent to the decoder. Note that we also save the "empty" feature to project the sequence back to the feature grid in the decoder.

The decoder consists of a 3D U-Net [4], an up-sampler, and an implicit decoder. It first projects the quantized sparse sequence back to a 3D feature grid, which serves as the input for the 3D U-Net. In contrast to the encoder, the decoder is designed to have a large receptive field. This is because, in order for the implicit decoder to infer whether a probe lies inside or outside of the shape, we need global knowledge. This is in alignment with prior works [6, 10]. More specifically, we use a 3-step U-Net to increase the receptive field, which integrates both local and global information. The up-sampler has the same number of scaling stages as the down-sampler, but it has a larger receptive field by design. Lastly, similarly to prior work [10], the implicit decoder consists of multiple ResNet blocks. It takes querying probe points $\mathcal{T}_\mathbf{x}$ and predicts their occupancy probability $\mathcal{T}_o$.

ShapeFormer. In Fig. 3, we show the detailed architecture of ShapeFormer. The input to the ShapeFormer consists of the concatenated sequence of $\mathcal{S}_\mathcal{P}$ and $\mathcal{S}_\mathcal{C}$. Since these sequences both have variable lengths, we append an end-token ([END]) to each sequence to indicate when the sequence terminates. Next, as in prior works [5, 9], all these indices are turned into learnable embeddings and are additively combined as the input embedding for ShapeFormer.

The main components of ShapeFormer are two causally-masked transformers, which consist of multiple decoder-only transformer blocks [11]. The first transformer learns to predict the coordinate of the next tuple, conditioned on previous tuples, while the second one learns to predict the value of the next element conditioned on previous tuples and the (predicted) coordinate index of the next element. Thus, the output feature of the first transformer is additively mixed with the input embedding of the second transformer delivering the encoded sequence information.

Each transformer is followed by an output head, which converts the feature produced by the transformer into a categorical distribution of the next sequence element. Both output heads consist of two fully connected layers, followed by a softmax layer to produce categorical conditional distributions for each of the sequence elements: $\{(p_{c_i}, p_{v_i})\}_{i=1}^{K}$. Note that this essentially shifts the complete sequence to the right by one element. For training, we also empirically find randomly masking out the partial sequence will improve generalization.

A.3. Details on training and sampling

We use Adam optimizer for training both VQDIF and ShapeFormer, and we set the learning rate as $1e-4$ for VQDIF and $1e-5$ for ShapeFormer. We use step decay for VQDIF with step size equal to 10 and $\beta = .9$ and do not apply learning rate scheduling for ShapeFormer. We train our network on a deep learning server with Intel Xeon CPU E5-2680 v4 CPU*56 and 256GB memory with 10 Nvidia Quadro P6000 graphics cards with a GPU memory size of 24GB. It takes 30 hours for our model to converge on our virtual scan dataset and 8 hours on the PartNet dataset. For D-Faust, the converging time is 16 hours. For sampling, we can obtain a single sample sequence in roughly 20 seconds, and we can also sample 24 sequences in parallel in 5 minutes.

B. More comparisons

We show more visual comparisons between our method and prior state-of-the-art methods in Figs. 4 to 6. Figs. 4 and 5 illustrates results on high-ambiguity scans, In these examples, we can see the averaging effect of the deterministic methods (See the scattering effect in ambiguous regions of the completions of PoinTr [14]). Our method produces significantly better results in terms of quality and diversity.

Also, we demonstrate our method can also achieve competitive accuracy for low-ambiguity scans in Fig. 6. Since there is limited ambiguity for such scans and the goal is to achieve accuracy toward ground truth, we put the ground truth in the first row and only sample 1 completion for each

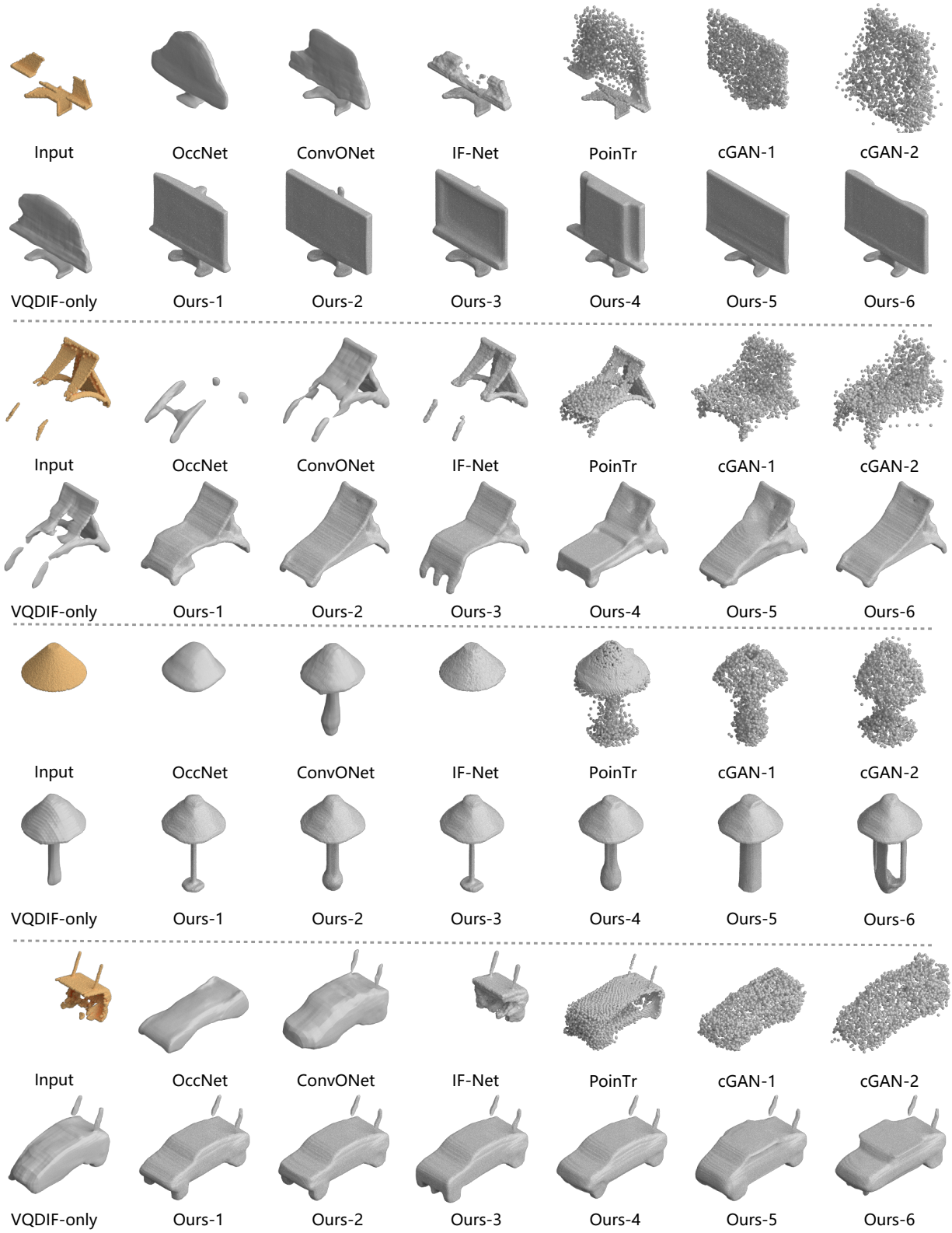


Figure 4. More comparisons on high ambiguity scans of ShapeNet objects.

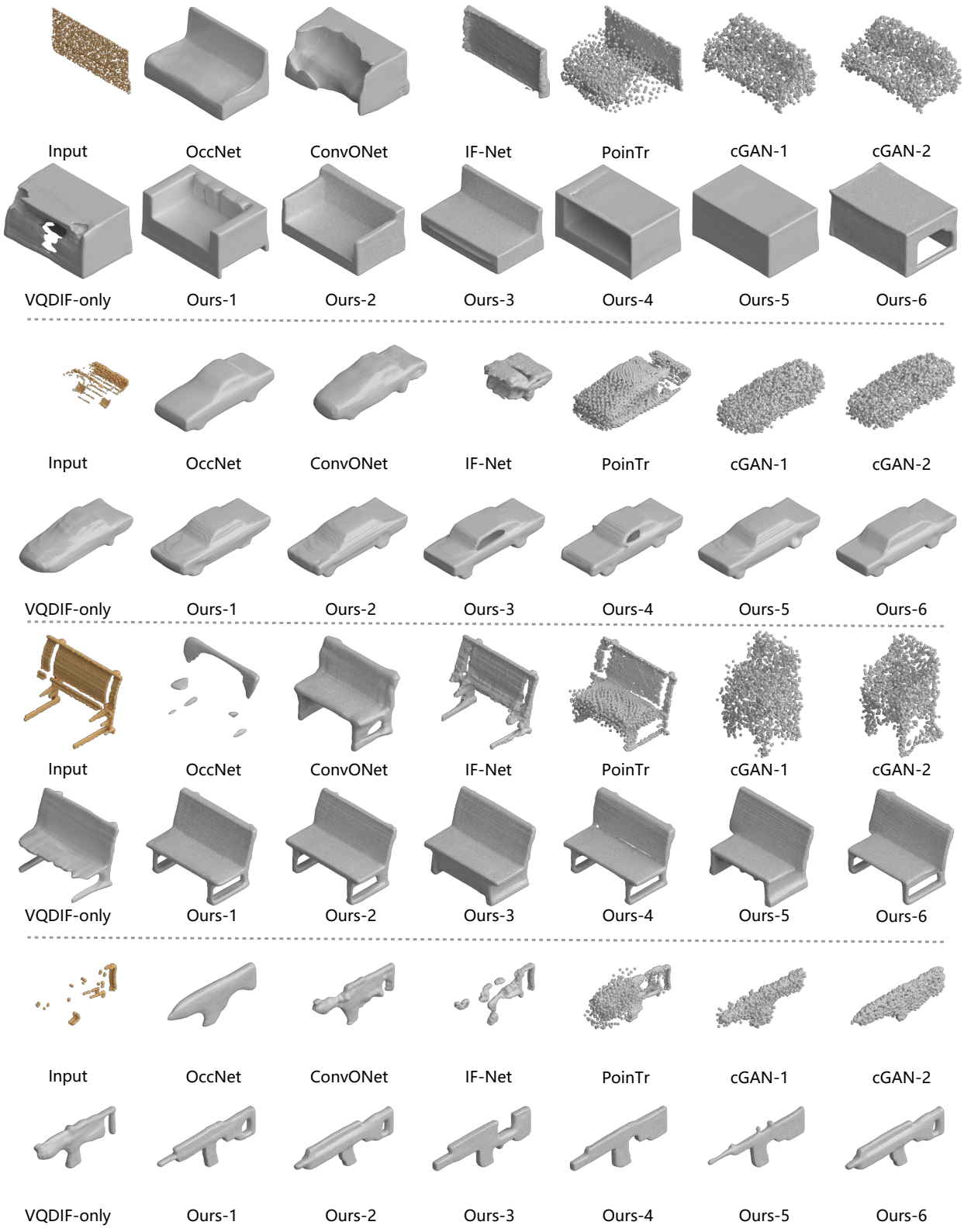


Figure 5. More comparisons on high ambiguity scans of ShapeNet objects.

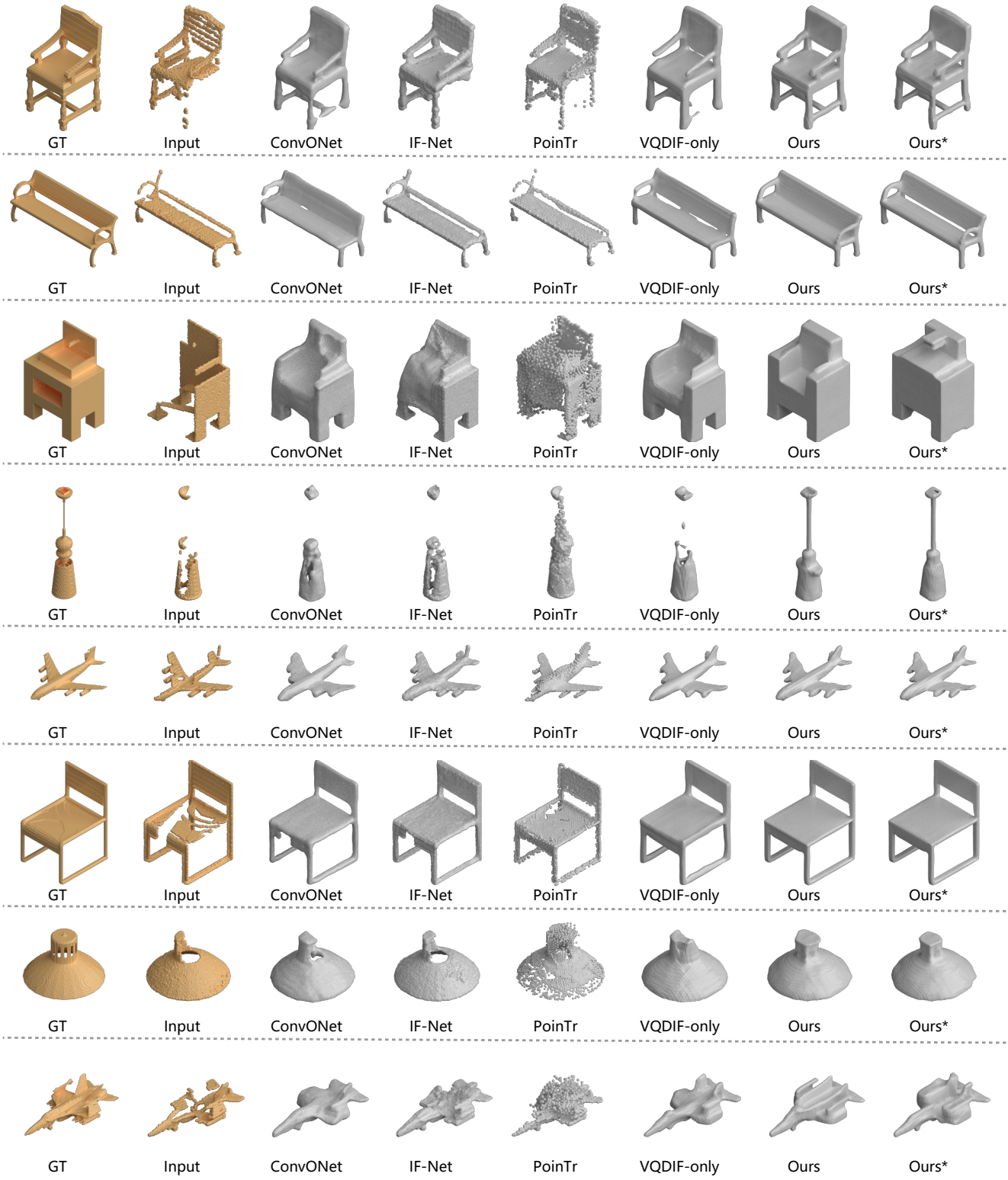


Figure 6. More comparisons on low ambiguity scans of ShapeNet objects. Ours=top-.4 sampling, Ours*=top-.0 sampling (best sampling).

of our sampling strategies (Ours: top-.4 sampling, Ours*: top-.0, e.g., best sampling). Also, we only compare state-of-the-art deterministic methods: ConvONet [10], IF-Net [2], and PoinTr [14] in these examples. As we can see, even the scans cover most areas of the ground truth shape; prior works can still produce unsatisfactory results for unseen regions. In contrast, our method can always produce more accurate, high-quality completions. Moreover, since Ours* always picks the coordinate and value indices with the highest probability, it often produces slightly more accurate shapes.

C. More analysis

Discussion of Limitation. ShapeFormer inherits the typical limitations of transformer-based autoregressive models. Mainly, the representation length cannot be too long, and thus the method currently can only use VQDIF with $R = 16$, which may fail to complete and reconstruct shapes with intricate structures; an example is shown in Figure 7. Another related limitation is the sampling speed, which prevents interactive applications.

Figure 7. An example of a shape completion failure case of ShapeFormer. The intricate details present in the input (second from left) are not preserved in the completions (gray shapes). The leftmost image shows the ground truth shape.

There are two research avenues to alleviate these problems: (i) Investigating more efficient attention mechanisms to reduce the transformer's quadratic complexity in the sequence length K to $O(K\sqrt{K})$ [7] or even $O(K)$ [3]. (ii) Designing an adaptive quantization scheme for the point clouds, which enables Transformers to focus dependencies on a lower local level while using higher-level features for faraway regions. (iii) Adopt advanced sampling techniques for autoregressive models such as parallel sampling [8].

Moreover, since we generate sequences of complete shapes from scratch, our results may slightly alter the input geometry to overcome the potential sparsity and noise. Besides using higher resolution quantized features to obtain more accurate generation, another possible improvement to this issue is to include high-resolution features of the input in the decoding procedure as in a recent image inpainting technique [13].

References

[1] R. Qi Charles, Hao Su, Mo Kaichun, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017. 3

[2] Julian Chibane, Thiemo Alldieck, and Gerard Pons-Moll. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*. IEEE, jun 2020. 7

[3] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers, 2021. 7

[4] Özgün Çiçek, Ahmed Abdulkadir, Soeren S Lienkamp, Thomas Brox, and Olaf Ronneberger. 3d u-net: learning dense volumetric segmentation from sparse annotation. In *International conference on medical image computing and computer-assisted intervention*, pages 424–432. Springer, 2016. 3

[5] Sander Dieleman, Charlie Nash, Jesse Engel, and Karen Simonyan. Variable-rate discrete representation learning. *arXiv preprint arXiv:2103.06089*, 2021. 3

[6] Philipp Erler, Paul Guerrero, Stefan Ohrhallinger, N. Mitra, and M. Wimmer. Points2surf learning implicit surfaces from point clouds. In *Proc. Euro. Conf. on Computer Vision*, 2020. 3

[7] Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers, 2019. 7

[8] Vivek Jayaram and John Thickstun. Parallel and flexible sampling from autoregressive models via langevin dynamics, 2021. 7

[9] Charlie Nash, Jacob Menick, Sander Dieleman, and Peter W Battaglia. Generating images with sparse representations. *arXiv preprint arXiv:2103.03841*, 2021. 3

[10] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *Proc. Euro. Conf. on Computer Vision*, 2020. 3, 7

[11] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. 3

[12] Maxim Tatarchenko, Stephan R Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3D reconstruction networks learn? In *Proc. IEEE Conf. on Computer Vision & Pattern Recognition*, pages 3405–3414, 2019. 1

[13] Ziyu Wan, Jingbo Zhang, Dongdong Chen, and Jing Liao. High-fidelity pluralistic image completion with transformers. *arXiv preprint arXiv:2103.14031*, 2021. 7

[14] Xumin Yu, Yongming Rao, Ziyi Wang, Zuyan Liu, Jiwen Lu, and Jie Zhou. Pointr: Diverse point cloud completion with geometry-aware transformers. In *ICCV*, 2021. 3, 7