Scaling Mesh Generation via Compressive Tokenization

Supplementary Material

A. Data Curation

Effective data filtering. For meshes with the same faces, their tokenized sequence length may differ due to the patch aggregation and block compression of BPT. We design an effective data-filtering strategy to maximize the utilization of our training data. Specifically, we filter meshes with their sequence length lower than the context window of the Transformer (i.e., 9600). Figure 1 shows that almost all meshes under 5k faces are used, and around 58% of meshes with more than 5k faces are further utilized. This strategy allows the utilization of some complicated meshes and improves the model's robustness and performance.

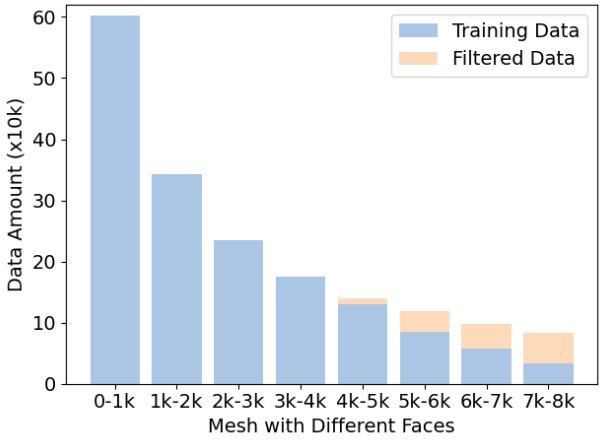

Figure 1. **Effective utilization of training data.** Almost all meshes under 5k faces are used, and around 58% of meshes with more than 5k faces are utilized further.

Two-stage training. Objaverse-XL contains many low-poly data with simple geometries, such as CAD meshes. In the initial stage of training, the model may benefit from these meshes to learn the geometry prior. However, their topology is typically different from human-crafted meshes and may prevent the model from learning the delicate topology. Therefore, we leverage a two-stage training strategy to trade off the generalizability and topology quality. The model is first pretreated on the large-scale data with around 1.5M meshes and then further fine-tuned on 0.3M high-quality meshes without simple geometry.

Testset Distribution. We build testset with 651 shapes in Table 1, generated by 3D diffusion models and remeshed to around 100k faces and 50k vertices.

Category	Amount	Category	Amount
Human	119	Structure	119
Product	99	Animal	81
Style	62	Geometry	51
Weapons	45	Transportation	44
Others	31	Total	651

Table 1. **The distribution of the test set.**

B. Model Architecture

As shown in Figure 2, the overall architecture of our model follows Michelangelo. As shown in Figure 2, we first train an auto-regressive transformer to generate meshes conditioned on point-cloud features extracted from the point-cloud encoder. Then, we train an additional diffusion model to generate point-cloud features conditioned on images.

C. Additional Results

Finer Discretization of BPT. For fair comparison, BPT follows the 7-bit discretization of previous works. However, BPT can be easily extended to finer discretization by increasing vocabulary, similar to modern LLMs (e.g., 128k of LLAMA3). We show the 9-bit discretization ($|B| = 16$, $|O| = 32$) in Figure 3 and found that the finer discretization contributes to better details and the smoother surface.

Compression Ratio of Different BPT Settings. The compression ratio is empirically calculated in our training dataset. With smaller $|B|$, vertices are easier to fall into the same block and be merged, thus the compression ratio is lower as shown in Table 2.

$(\|B\|, \|O\|)$	(4, 32)	(8, 16)	(16, 8)
Compression Ratio↓	**0.2456**	0.2594	0.2802

Table 2. **The compression ratios with different BPT settings**

Contribution of each component. We combine block-wise indexing (BI) with AMT to further show the effectiveness of patch aggregation in Table 4. Due to the long-range dependency on AMT in Figure 4, the results with BI are even worse than those with vanilla AMT.

Tokenization Speed. All tokenization methods can serialize meshes in near real-time in Table 5.

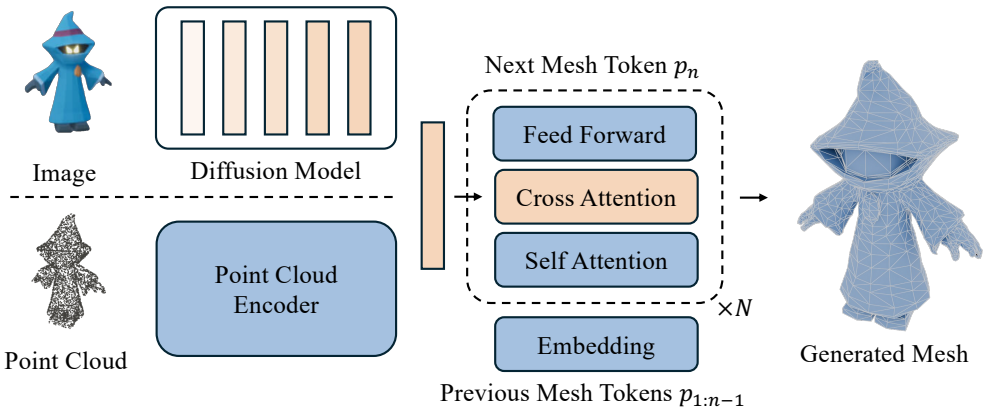

Figure 2. **Model architecture for conditional mesh generation.** First, we leverage an auto-regressive transformer to generate meshes conditioned on point-cloud features via cross-attention layers. Next, we train an additional diffusion model to generate point-cloud features based on images, enabling image-to-mesh generation.

Metrics	ECD(10^{-3})↓	NC↑	#V	#F	V Ratio(10^{-2})	F Ratio(10^{-2})
MeshAnythingv1	6.09	0.63	416	713	0.83	0.71
MeshAnythingv2	4.49	0.64	1100	1943	2.20	1.94
BPT	**3.84**	**0.85**	1207	2369	4.73	2.36

Table 3. **Additional metrics for quantitative evaluation.**

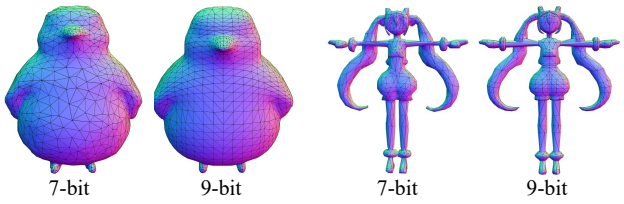

Figure 3. **Results of 7-bit and 9-bit discretization of BPT**

Method	BPT	AMT	AMT w/ BI
Hausdorff Distance↓	**0.166**	0.265	0.605
Chamfer Distance↓	**0.094**	0.114	0.291

Table 4. **Ablation on block-wise indexing**

Method	AMT	Edgerunner	BPT
Speed (s/mesh)	0.028	**0.014**	0.026

Table 5. **Average tokenization time for a mesh**

Additional Metrics. We extend Table 2 in the main paper with more metrics in Table 3. Note that meshes generated by Meshanything v1/v2 are mainly incomplete, leading to a smaller number of faces and vertices.

Serilization Order. For generic datasets, the sequence order only slightly affects the generation performance. We tried several orders in Table 6 and found that the z-y-x order (bottom-up order) performs best.

Sequence Order	z-y-x	x-y-z	y-z-x
Hausdorff Distance↓	**0.166**	0.199	0.210
Chamfer Distance↓	**0.094**	0.104	0.109

Table 6. **Ablation on sequence face order**

Point-cloud Number Ablation. We try different points in Table 7 and found 4096 points achieve the best performance. With fewer points, details are hard to preserve. With more points, the model tends to produce smaller faces for more details, making it easier to generate incomplete meshes.

Point-cloud Amount	2048	4096	8192
Hausdorff Distance↓	0.184	**0.166**	0.202
Chamfer Distance↓	0.102	**0.094**	0.107

Table 7. **Ablation on the number of points**

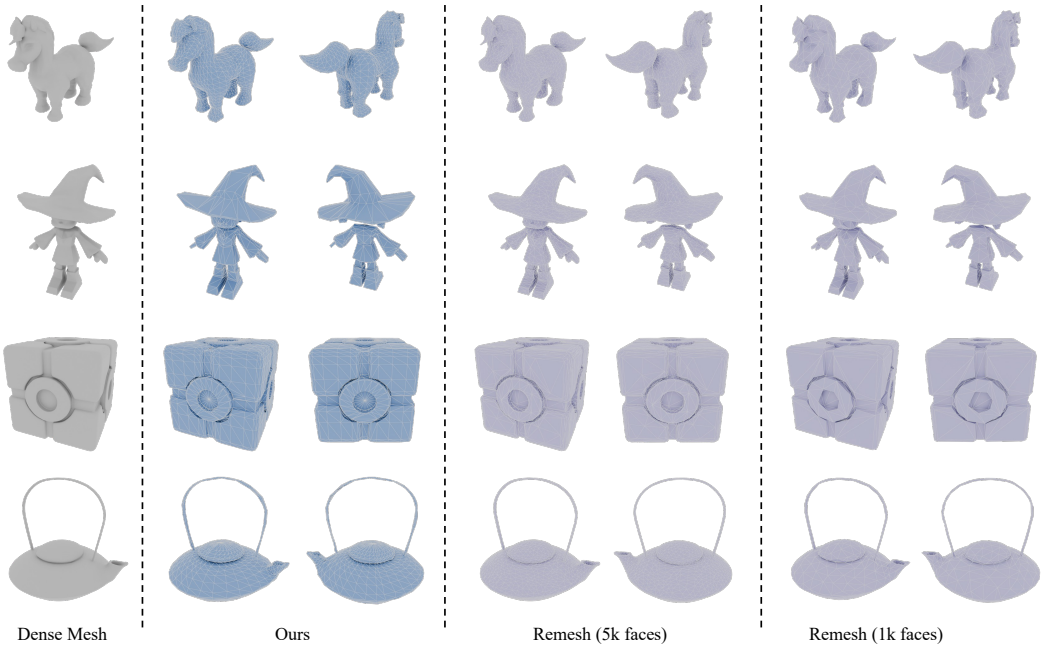

Figure 4. **Comparison with remeshing.** Our method can generate appropriate topology from dense meshes, while remesh algorithms fundamentally fail to capture models' geometry and produce poor topology.

Textured Mesh Generation. We synthesize texture with an off-the-shelf texture model in Figure 5.

Figure 5. **Textured Mesh Generation.**

Comparison with remesh. Compared with remeshing algorithms, our method can generate appropriate topology from dense meshes, while remesh algorithms fundamentally fail to capture models' geometry and produce poor topology. As shown in Figure 4, the meshes generated by our model are at the product-ready level.