Scaling Mesh Generation via Compressive Tokenization

Haohan Weng[14], Zibo Zhao[24*], Biwen Lei[4], Xianghui Yang[4], Jian Liu[4], Zeqiang Lai[4], Zhuo Chen[4]
Yuhong Liu[4], Jie Jiang[4], Chunchao Guo[4], Tong Zhang[1], Shenghua Gao[3], C. L. Philip Chen[1]

[1]South China University of Technology, [2]ShanghaiTech University, [3]University of Hong Kong
[4]Tencent Hunyuan

https://whaohan.github.io/bpt

Figure 1. **Generated meshes conditioned on images or point cloud sampled from dense meshes.** Our model can generate meshes up to 8k faces based on the proposed compressive tokenization. The lower right dense meshes or images represent the conditions.
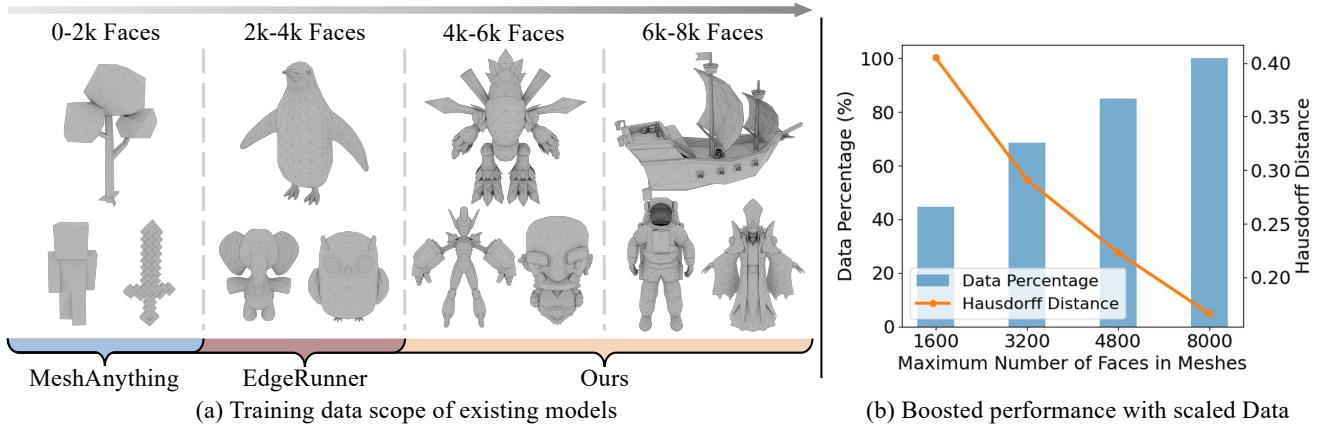
Abstract

*We propose a compressive yet effective mesh representation, **Blocked and Patchified Tokenization (BPT)**, facilitating the generation of meshes exceeding 8k faces. BPT compresses mesh sequences by employing block-wise indexing and patch aggregation, reducing their length by approximately 75% compared to the original sequences. This compression milestone unlocks the potential to utilize mesh data with significantly more faces, thereby enhancing detail richness and improving generation robustness. Empowered with the BPT, we have built a foundation mesh generative model training on scaled mesh data to support flexible control for point clouds and images. Our model demonstrates the capability to generate meshes with intricate details and accurate topology, achieving SoTA performance on mesh generation and reaching the level for direct product usage.*

1. Introduction

Meshes, the cornerstone of 3D geometric representation, are widely utilized in various applications, including video games, cinematic productions, and simulations. Despite their widespread adoption, the meticulous craft of meshes with functional topologies demands substantial design ef-

---

*Corresponding Author

Figure 2. **Scaling data for mesh generation with BPT.** (a) Existing models can only handle meshes with at most 4k faces, which still lack intricate details. Empowered by BPT, our model can leverage meshes exceeding 8k faces, effectively extending the training scope for mesh generation. (b) We train the same model on meshes with different maximum numbers of faces. As the number of mesh faces increases, the performance of mesh generation significantly improves, highlighting the value of high-poly training data. The generation performance is measured by the Hausdorff distance between the input point cloud and generated meshes (a lower distance indicates a better performance).

fort. This labor-intensive process acts as a bottleneck, impeding the evolution of 3D content creation and the progress of immersive human-computer interaction. Recent research [3–5, 36, 38, 45] has tried to automate the mesh sculpting process via auto-regressive Transformers. These methods directly generate vertices and faces as human-crafted to maintain the high-quality mesh topology, yielding promising results on low-poly mesh generation.

The foundation of modeling meshes with auto-regressive transformers is mesh tokenization, which converts the mesh into a one-dimensional sequence. PolyGen [26] and MeshXL [3] directly tokenize the mesh by converting the vertex coordinates to sorted triplets, each defining a tuple of quantized 3D coordinates. They learn the one-dimensional sequence with a joint or two separate auto-regressive transformers. MeshGPT [36] and its variants [4, 45] utilize an auto-encoder to convert meshes into latent sequences. MeshAnythingv2 [5] and Edgerunner [38] propose improved tokenization methods to compress vanilla mesh sequences further. However, these methods still convert meshes to relatively long sequences, limiting the ability of generative models to learn with high-poly meshes. *To scale up mesh generation, a more compressive representation is demanded to extend the scope of the training data.*

In this paper, we propose a compressive yet efficient mesh representation called Blocked and Patchified Tokenization (BPT). BPT converts Cartesian coordinates to block-wise indexes, which makes the initial attempt to compress mesh tokens at the vertex level. Then, we select the vertices connected with most faces (i.e., the highest vertex degree) as the patch center. The faces around the center vertices are aggregated as patches, compressing mesh tokens at the face level. Our approach significantly reduces the length

of the vanilla mesh sequence by around 75%, achieving the SoTA compression ratio across existing tokenization. Empowered with BPT, our mesh generative model can utilize millions of meshes with intricate details, significantly improving its performance and robustness.

BPT facilitates a wide range of 3D applications. We demonstrate its effectiveness via conditional mesh generation on point clouds and images. Our model empowers even unprofessional users to produce meshes at the product-ready level. Its applicability spans a spectrum of practical domains of 3D content creation, revealing the dawn of a new era in 3D generation.
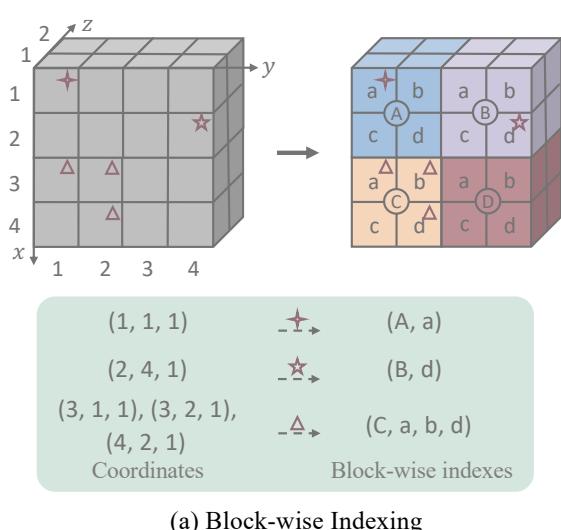
Our contributions can be summarized as follows:
- We introduce Blocked and Patchified Tokenization (BPT), a compressive yet effective tokenization with a state-of-the-art compression ratio of around 75%.
- Empowered by BPT, we investigated the scaling of mesh data across diverse face configurations, revealing that incorporating extended data improves generation performance and robustness.
- We build a mesh foundation model that supports conditional generation based on images and point clouds, enabling users to create product-ready 3D assets.
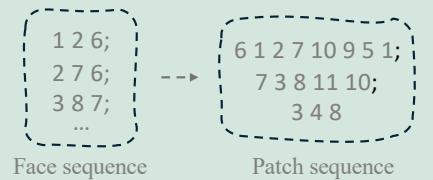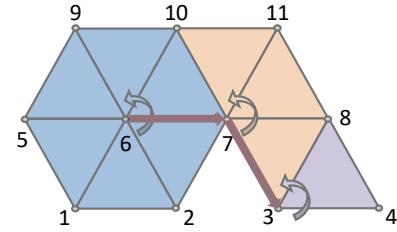
2. Scaling Data with Compressive Tokenization

In this section, we discuss the significant improvement brought by the scaled data, showing the necessity and importance of the proposed compressive tokenization.

Extend the training data scope. One of the keys to improving current mesh generative models is to extend its training scope (i.e., effectively utilizing the high-poly meshes). The training data of existing mesh generative

(a) Block-wise Indexing

(b) Patchified Aggregation

Figure 3. **The proposed Blocked and Patchified Tokenization (BPT).** (a) We convert the coordinates from the Cartesian system to block-wise indexes. The coordinates are first separated equally into several blocks. Then, vertices inside each block are located with 1-dim indexes. (b) The nearby faces are aggregated as patches to compress the mesh sequence. Each patch center is set as the vertex connected with the most unvisited faces. Subsequently, other vertices within the patch are included in the subsequence to create a complete patch.

models predominantly consists of low-poly meshes, significantly limiting their ability to express intricate details and handle complicated inputs. As shown in Figure 2(a), the training data from previous methods still lacks sufficient detail. In contrast, BPT accommodates meshes exceeding 8k faces, showing the potential to utilize vast amounts of high-quality mesh data with rich details.

Boosted performance and robustness. As shown in Figure 2(b), we train the same model on meshes with different maximum numbers of faces. To qualitatively demonstrate the performance improvement, we compute the Hausdorff distance between the input point cloud and generated meshes (the lower distance indicates better performance). The figure shows the surprising enhancement afforded by the scaled data. It illustrates the positive correlation between the number of faces and the performance of generative models. As the mesh data's complexity increases, the generative models' performance and robustness are significantly boosted.

3. Blocked and Patchified Tokenization

This section will explore Blocked and Partchified Tokenization (BPT) as illustrated in Figure 3 and Algorithm 1, which involves block-wise indexing and patch aggregation. We will begin by discussing the vanilla mesh tokenization in Section 3.1. Following that, Section 3.2 will detail the conversion of vertex representations from Cartesian coordinates to block-wise indexes. In Section 3.3, we will introduce how to directly aggregate nearby faces into patches and model these patches. With the implementation

of BPT, we can achieve a state-of-the-art compression ratio of approximately 75% while effectively preserving the mesh topology.

3.1. Preliminary

A triangle mesh $\mathcal{M} = (f_1, f_2, ..., f_n)$ with n faces can be formulated as the composition of faces f_i.

$$
\begin{aligned}
f_i &= (v_{i1}, v_{i2}, v_{i3}) \\
&= (x_{i1}, y_{i1}, z_{i1};\ x_{i2}, y_{i2}, z_{i2};\ x_{i3}, y_{i3}, z_{i3})
\end{aligned}
\tag{1}
$$

where each face f_i consists of 3 vertices and each vertex v_i contains 3D coordinates (x_i, y_i, z_i) discretized with a 7-bit uniform quantization. The vertices are sorted by z-y-x coordinates from lowest to highest, and the faces are ordered by their lowest vertices.

3.2. Block-wise Indexing

Naive indexing. The basic idea is to convert a 3D Cartesian coordinate (x_i, y_i, z_i) into a scaler index I_i. With the 7-bit quantization setting (resolution $r = 128$), the coordinates x_i, y_i, z_i are within the range $[0, r-1]$, thus the indexes I_i are ranged from $[0, r^3 - 1]$, which leads to an unaffordable vocabulary size r^3 for Transformer.

Indexing in each block. As shown in Figure 3(a), we equally partition the whole coordinate system into several blocks and index the coordinates as offsets in each block. Specifically, we equally separate the coordinates along each axis into B segments, where the length of each segment is O. Thus, for the vertex $v_i = (x_i, y_i, z_i)$, the block-wise

Table 1. **Compression ratio over different mesh tokenization methods.**

Method	MeshXL [3]	MeshAnything [4]	MeshGPT [36]	PivotMesh [45]	MeshAnythingv2 [5]	EdgeRunner [38]	BPT
Compression Ratio ↓	1.00	1.00	0.67	0.67	0.46	0.47	**0.26**

indexing (b_i, o_i) can be compute as follow:

$$b_i = (x_i \mid O) \cdot B^2 + (y_i \mid O) \cdot B + z_i \mid O$$
$$o_i = (x_i \% O) \cdot O^2 + (y_i \% O) \cdot O + z_i \% O \tag{2}$$

where $\mid$ denotes denotes division with no remainder and $\%$ denotes the modulo operation. The block index $b_i \in [0, B^3 - 1]$ and offset index $o_i \in [0, O^3 - 1]$ are all discrete integers, B^3 is the number of blocks and O^3 is the number of offsets in each block.

Block indexes compression. As the coordinates are sorted in the z-y-x order, the next predicted vertex frequently occurs in the same block. Thus, we can combine all the adjacent block indexes into one index to save more length. Specifically, for vertices $v_i, v_{i+1}, ..., v_n$ in the same block (i.e., their block index are the same), they can be represented as follows:

$$(v_i, v_{i+1}, ..., v_n) = (b_i, o_i, b_i, o_{i+1}, ..., b_i, o_n)$$
$$= (b_i, o_i, o_{i+1}, ..., o_n) \tag{3}$$

With the proposed block-wise indexing, a vertex can be represented with at most two tokens. The compression ratio so far is around 50%.

3.3. Patchified Aggregation

In vanilla mesh representation, each vertex appears as many times as the number of its connected faces, which results in considerable redundancy. Similar to image generation [28], we propose aggregating the faces connected to the same vertex as a patch without overlapping.

Next Patch Prediction. A crucial step in patchifying faces is finding each patch's center vertex. First, we initialize the whole sorted face sequence as unvisited. Then we select the first unvisited face and designate the vertex *connected with the most unvisited faces as the patch center*, as shown in Figure 3(b). Consequently, all faces connected to the center vertex are aggregated to form a patch, denoted as $P_c = (v_c, v_1, v_2), (v_c, v_2, v_3), ..., (v_c, v_{n-1}, v_n)$, we can convert it as follows:

$$P_c = (v_c, v_1, v_2, ..., v_n) \tag{4}$$

where v_c is the patch center and $v_{1:n}$ are other vertices in the patch. Then, we mark all the faces in the patch as visited and find the next unvisited face by sorting order until all the faces are visited. In this way, the occurrences of the

Algorithm 1 Blocked and Patchified Tokenization (BPT)

1: **procedure** BPT($\mathcal{M}$)
2:    $P \leftarrow \emptyset$ ▷ List of patches
3:    **while** $\mathcal{M} \neq \emptyset$ **do**
4:       $f \leftarrow$ select unvisited face in $\mathcal{M}$
5:       $v_c \leftarrow$ vertex in f with most unvisited faces
6:       $P_c \leftarrow$ Aggregate(v_c)   ▷ Eq. 4
7:       $P_c \leftarrow$ Blockwise-Index(P_c)   ▷ Eq. 2 & 3
8:       $P_c \leftarrow$ Dual-Block(P_c)   ▷ Eq. 5
9:       $P \leftarrow P \cup P_c, \quad \mathcal{M} \leftarrow \mathcal{M} - P_c$
10:   **return** P

patch center vertex v_c are reduced (average 6) to just 1. Additionally, for most cases, the occurrences of other vertices $v_{1:n}$ are decreased from the number of connected faces to the number of connected patches.

Seperate patches with dual-block indexes. An intuitive solution to denote the termination of each patch is appending a special token. However, this increases the length of the mesh sequence and the training cost. In contrast, we leverage the dual-block indexes to denote the starting of a patch. We create two vocabularies for block indexes, one for center vertex v_c and the other for the common vertices $v_{1:n}$. The offset vocabulary is shared by both the center vertex and other vertices.

$$P_c = (b'_c, o_c, b_1, o_1, b_2, o_2, ..., b_n, o_n) \tag{5}$$

The advantages of patch aggregation are twofold. First, a substantial decrease in vertex repetition leads to a condensed mesh sequence. Secondly, the inherent clustering of adjacent faces augments the sequence's spatial locality, thereby simplifying its generation. With Patchified Aggregation, the sequence length yields another 50% shorter.

3.4. Properties of BPT

Compactness: free lunch for training and inference efficiency. BPT compresses mesh from two perspectives. For the vertex level, one vertex is represented with at most two tokens (one for inner-block indexing and two for inter-block indexing). At the face level, faces are aggregated into patches to reduce the repeated vertices that are shared with adjacent faces. As shown in Table 1, the proposed BPT achieves the state-of-the-art compression ratio of the mesh sequence. Consequently, *BPT earns free training and inference efficiency for all mesh generative models.*
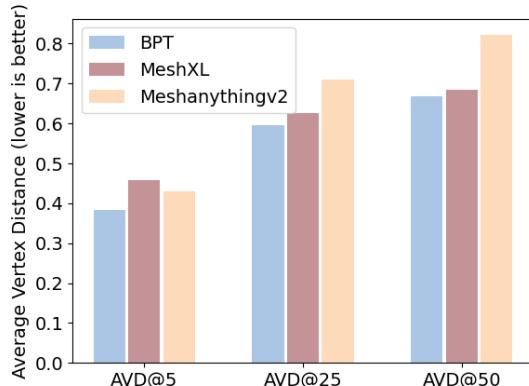
4

Figure 4. **The average vertex distance with previous t vertices** (denoted as AVD@t, lower is better). Among various context lengths t, BPT achieves the lowest AVD, showing its locality for effective mesh modeling.

Locality: effective mesh sequence modeling. The locality is essential for avoiding long-range dependency between tokens in the mesh sequence for transformer modeling. In BPT, faces are aggregated into patches for generation, guaranteeing each patch's locality. We quantize the locality with the Average Vertex Distance between vertices and their previous t vertices (denoted as AVD@t), as shown in Figure 4. For different values of t, BPT achieves the best locality. Furthermore, as the visual comparison shown in Section 5.2, models based on Adjacent Mesh Tokenization (AMT) [5] produce meshes with more incomplete meshes. While benefiting from the vital locality, models based on BPT avoid learning the long-range dependency in mesh sequence, thus fundamentally alleviating such problems.

4. Mesh Generation based on BPT

The proposed tokenization can be easily applied to mesh generation with high performance and robustness. This section details the architecture of conditional mesh generation on different modalities, including point clouds and images. We use a standard auto-regressive Transformer with parameters θ to model the sequence with our tokenization and leverage cross-attention for conditioning. The token sequences are modeled with a standard auto-regressive Transformer with parameter θ, maximizing the log probability. The *cross-attention* is leveraged for various conditions c.

$$L(\theta) = \prod_{i=1}^{|P|} p(p_i | p_{1:i-1}, c; \theta), \qquad (6)$$

Point-cloud to Mesh. We leverage a pre-trained point cloud encoder in Michelangelo-like [50] architecture with larger parameters. Furthermore, we add the linear projection and layer-norm layer before cross-attention for a more stable conditioning injection. To yield better generalization ability, we sample 50k points from the mesh and randomly select 4096 points as the condition for each iteration.

Image to Mesh. With the trained point-cloud conditional generative model, we further align the image latent to the point-cloud latent with an additional diffusion model as Michelangelo [50]. Specifically, we leverage DiT architecture [28] to generate the point-cloud condition produced by the point-cloud encoder. Our diffusion model is conditioned on the image features extracted from the pre-trained DINO encoder [27]. By feeding the generated point cloud feature into the trained point-cloud conditional generative model, our model can produce meshes faithfully following the given images.

5. Experiments

5.1. Experiment Settings

Datasets. Our model is trained on the mixture of ShapeNetV2 [2], 3D-FUTURE [13], Objaverse [11], Objaverse-XL [12] and internal data licensed from 3D content providers. We manually filter the objects with poor topology (e.g., scanning data) without decimation. We further use an effective filtering strategy to maximize data utilization, which filters meshes via their tokenized length based on the transformer's context window 9600. To trade off the generalizability and mesh quality, we first trained the model on large-scale data with around 1.5M meshes and then further fine-tuned it on 0.3M high-quality meshes.

Baselines. We benchmark our approach against leading mesh generation methods: **MeshAnything** [4], which introduces point-cloud conditioned mesh generation model with a noise-resistant decoder to boost mesh generation quality; **Meshanythingv2** [5], which proposes a new tokenization method called Adjacent Mesh Tokenization (AMT) to achieve half the token sequence length on average compared with vanilla representation; **EdgeRunner** [38], which features a novel mesh tokenization algorithm for efficient compression and a fixed-length latent space for better generalization. Since there is no publicly accessible code of Edgerunner, we can only conduct qualitative experiments with EdgeRunner on cases demonstrated on its webpage.

Metrics. For point cloud conditional generation, we apply the following metrics: 1) **Chamfer Distance(CD)**: It calculates the minimum cost required to transform one point set into another by summing the shortest distances from each point in one set to the nearest point in the other set. 2) **Hausdorff Distance(HD)**: It quantifies the maximum discrepancy between two point sets by measuring the greatest distance from any point in one set to its nearest neighbor
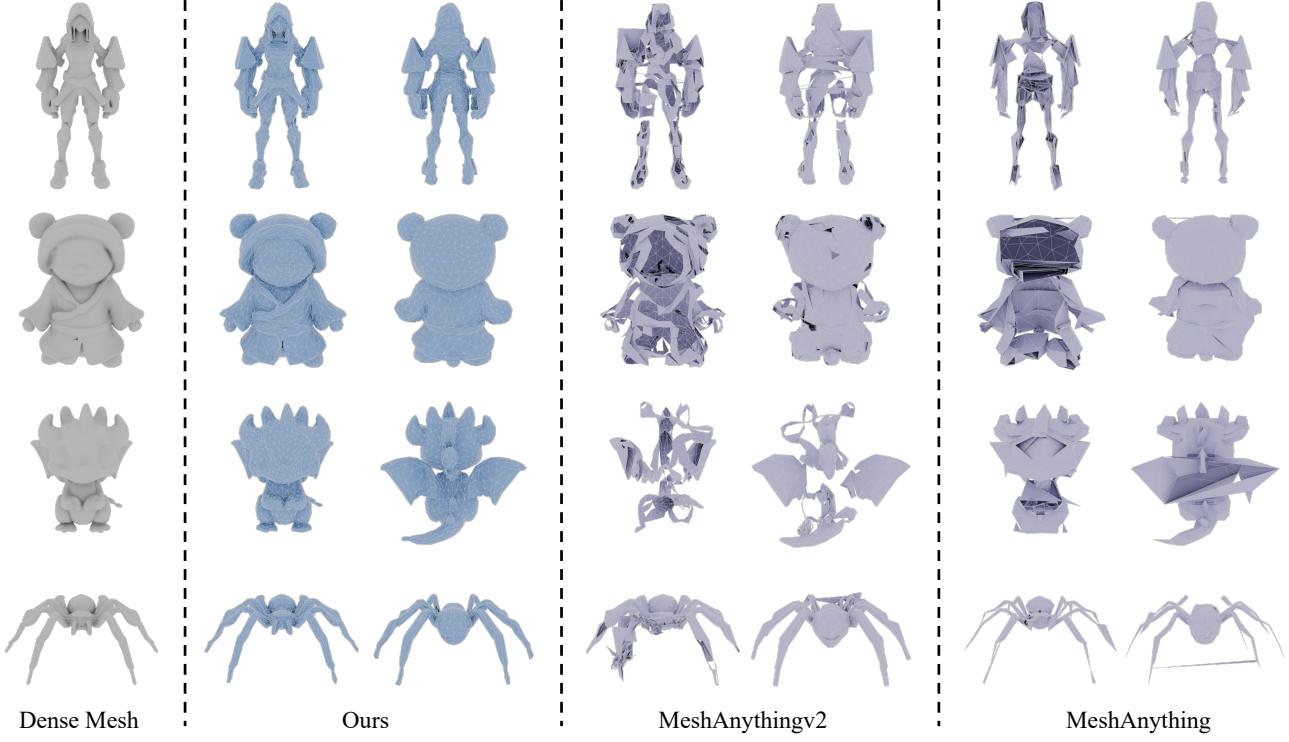
5

Figure 5. **Comparision on point-cloud conditional generation.** All the meshes are generated conditioned on the point cloud sampled from dense meshes. Our model can recover the details of dense meshes while maintaining high-quality topology.

Table 2. **Quantitative comparison with baselines.** With the proposed BPT, our model can utilize meshes with many more faces, thus greatly improving the generation performance and robustness.

Method	Hausdorff Distance ↓	Chamfer Distance ↓
MeshAnything [4]	0.301	0.136
MeshAnythingv2 [5]	0.265	0.114
Ours	**0.166**	**0.094**

in the other set. For both CD and HD, a lower distance indicates a better performance. We compute these metrics on 1024-dim point clouds uniformly sampled from meshes. For image conditional generation, we only conducted qualitative comparisons due to the lack of comparison samples for Edgerunner.

Implementation Details. Our auto-regressive Transformer has 24 layers with a hidden size of 1024. It is trained on 4 8×L40 machines for around 7 days with batch size 2 for each GPU. The temperature used for sampling is set to 0.7 to balance the quality and diversity. We use flash attention for all Transformer architecture and bf16 mixed precision to speed up the training process. We use AdamW [23] as the optimizer with $\beta_1 = 0.9$ and $\beta_2 = 0.99$ with a learning rate of 10^{-4} for all the experiments.

5.2. Point-cloud to Mesh Generation

To benchmark the capability of point-cloud conditional generation, we use around 500 dense meshes generated from other models for testing. We select MeshAnything [4], and MeshAnythingV2 [5] as the baselines. We calculate the Hausdorff distance and Chamfer distance between the point clouds sampled from the dense meshes and generated meshes. The quantitative comparisons are shown in Table 2, indicating our model achieves state-of-the-art performance for both Hausdorff distance and Chamfer distance with significant improvement. We also qualitatively compare Meshanything and MeshanythingV2 as shown in Figure 5. Limited with inefficient representations, MeshAnything and MeshAnythingV2 are only trained on meshes with maximum faces of 800 and 1600 separately. Thus, these models struggle to generate meshes with intricate details and produce more incomplete meshes, as shown in the qualitative comparison. In contrast, with the proposed BPT, our model can utilize meshes with many more faces and learn the mesh sequence with vital locality, thus significantly improving the generation performance.

5.3. Image to Mesh Generation

There are minimal works [3, 38] to generate meshes conditioned on images directly, and none of them are publicly accessible. To demonstrate the proposed BPT's effective-
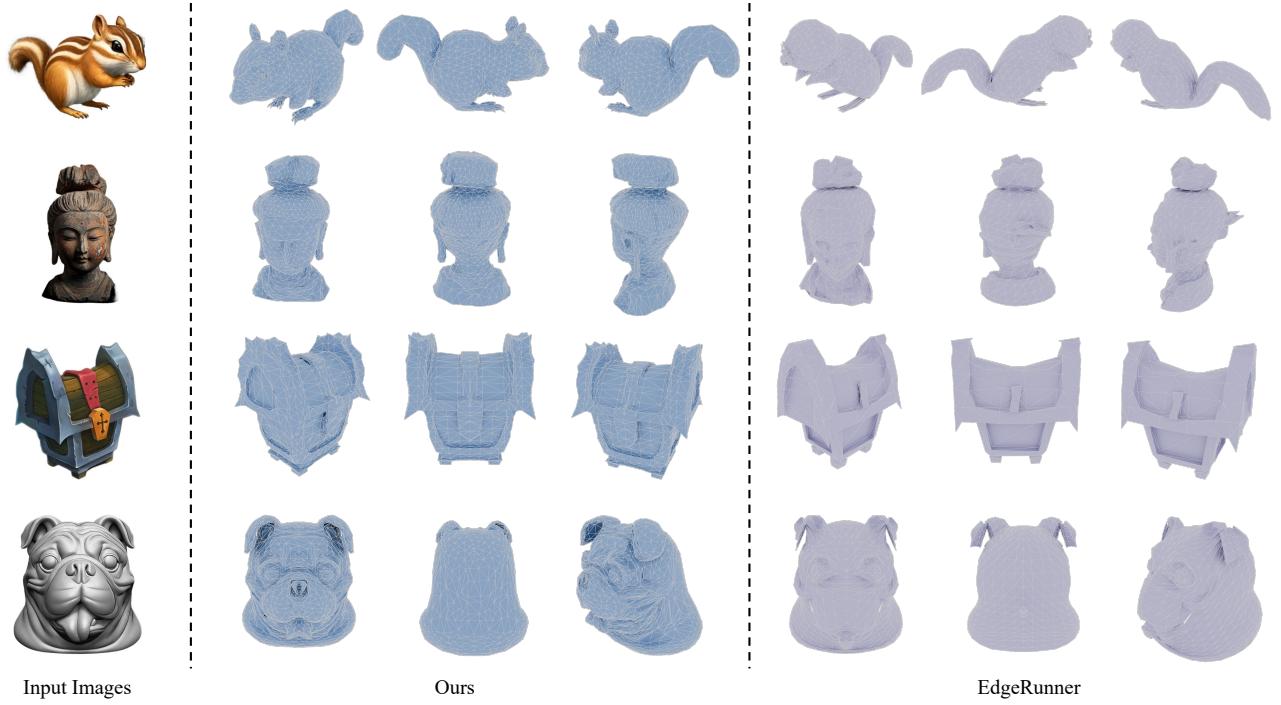
Figure 6. **Comparision on image conditional generation.** The results of Edgerunner are downloaded from its gallery for faithful and comprehensive evaluations. The meshes generated from our model have more details and faithfully follow the input image.

Table 3. **Abaltion study on block size and offset size.** The block size 8 and offset size 16 achieve the best generation performance.

$(|B|, |O|)$	Hausdorff Distance $\downarrow$	Chamfer Distance $\downarrow$
(4, 32)	0.209	0.111
(8, 16)	**0.166**	**0.094**
(16, 8)	0.256	0.126

ness, we download Edgerunner's gallery from its webpage and generate the meshes with our model conditioned on the same images. With BPT, our model is trained on meshes with more faces, and thus, it can produce meshes with more details. As shown in Figure 6, the meshes generated from our model faithfully follow the input images in more detail compared with Edgerunner's.

5.4. Ablation Studies

Select block size and offset size for BPT. The block size $|B|$ and offset size $|O|$ are crucial hyper-parameters for BPT. We conduct the ablation experiments under the fixed mesh quantization resolution (i.e., $|B| \cdot |O| = 128$). With smaller block sizes, BPT's compression ratio is higher, thus making it better for mesh modeling. However, with too small block sizes (i.e., too large offset sizes), the vocabulary becomes unaffordable for prediction, thus reducing the generation performance. As shown in Table 3, the block size 8 and offset size 16 achieve the best generation performance.

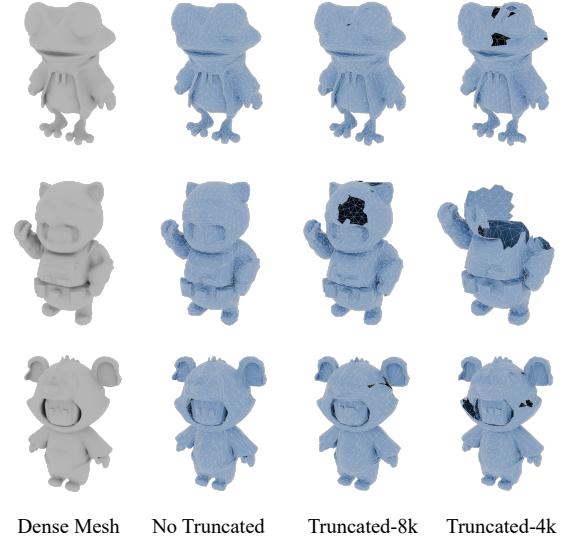Dense Mesh   No Truncated   Truncated-8k   Truncated-4k

Figure 7. **Trained on truncated meshes with the different context windows, while inference with sliding window attention.** Our results show that such truncation would reduce the model's robustness and make it easy to produce incomplete meshes.

Comparison with truncated training. There are also some engineering tricks to improve the utilization of training meshes. A typical solution is to train on truncated mesh sequences and inference with sliding window attention. We
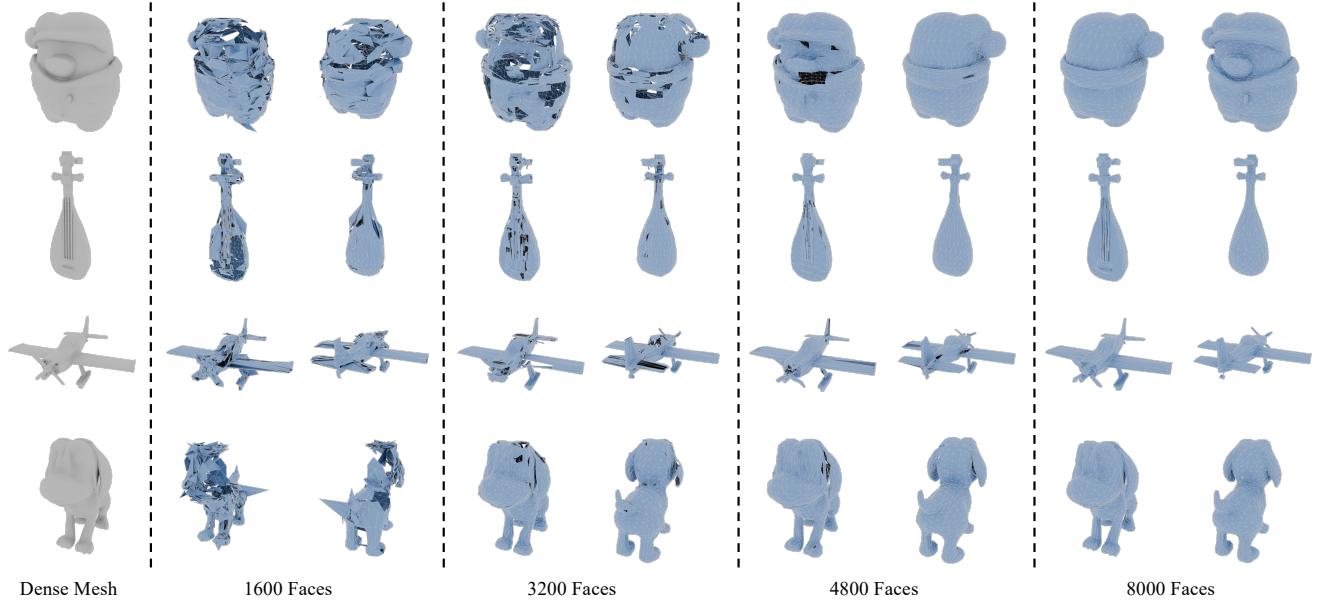
Figure 8. **As the maximum face of the training meshes increases, the performance of mesh generation is significantly boosted.** It shows that scaling the mesh data with more faces is crucial to ensure the mesh generation performance.

conduct the ablation study on truncated meshes with the Transformer context window of 4k and 8k, as shown in Figure 7. However, our experiments show that such truncation would reduce the robustness of mesh generation and make it easier to produce incomplete meshes.

Face matters for mesh generation. As the number of faces increases, the training meshes contain more details, improving the performance and robustness of mesh generative models. As shown in Figure 2(b) and Figure 8, we conduct the ablation study with different training data, with the maximum faces of 1600, 3200, 4800, and 8000 separately. The experiment results show that scaling the mesh data with more faces is crucial to ensure the generation performance.

6. Related Works

3D generation with neural representation. Most previous attempts learn 3D shapes with various representations, e.g., SDF grids [8, 9, 35, 52] and neural fields [15, 17, 21, 24, 25, 41, 49]. To improve the generalization ability, researchers start to leverage pre-trained 2D diffusion models [20, 30, 31] with score distillation loss [19, 29, 42] in a per-shape optimization manner. Multi-view diffusion models [7, 33, 34, 39, 44, 51] are used to enhance the quality further and alleviate the Janus problem. Recently, Large Reconstruction Models (LRM) [16, 18, 37, 43, 46–48] train the Transformer backbone on large scale dataset [11] to effectively generates generic neural 3D representation and shows the great performance of scaling. However, these neural 3D shape generation methods require post-conversion [22, 32] for downstream applications, which is non-trivial and easy

to produce dense and over-smooth meshes.

Native mesh generation. Compared with the well-developed generative models of neural shape representations, mesh generative models remain under-explored. Some pioneering works try to tackle this problem by formulating the mesh representation as surface patches [14], deformed ellipsoids [40], mesh graphs [10], and binary space partitioning [6]. PolyGen [26], Polydiff [1], and MeshGPT [36] construct the promising generative models, but their performance is limited by the single-category datasets (e.g., ShapeNet). Recent researches [3, 4, 45] build more generic generative models for native mesh generation within large-scale datasets. Our research, along with several concurrent works [5, 38], is designed to optimize the mesh representation, thus improving the model's scalability and robustness.

7. Conclusion

In this paper, we introduce Blocked and Patchified Tokenization (BPT), a fundamental improvement for mesh tokenization that reduces sequence length by approximately 75%. Such compression efficiency utilizes meshes with over 8k faces, allowing us to scale the training data and enhance generation performance. By leveraging BPT, we have built a foundational mesh generative model conditioned on point clouds and images.

Limitations and future work. The current number of parameters (500M) is still insufficient, and we will conduct additional scaling experiments with BPT. Furthermore, we can explore other promising architectures for sequence modeling to better utilize the inductive bias of meshes.

8

References

[1] Antonio Alliegro, Yawar Siddiqui, Tatiana Tommasi, and Matthias Nießner. Polydiff: Generating 3d polygonal meshes with diffusion models. *arXiv preprint arXiv:2312.11417*, 2023. 8

[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. 5

[3] Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Yanru Wang, Zhibin Wang, Chi Zhang, et al. Meshxl: Neural coordinate field for generative 3d foundation models. *arXiv preprint arXiv:2405.20853*, 2024. 2, 4, 6, 8

[4] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. *arXiv preprint arXiv:2406.10163*, 2024. 2, 4, 5, 6, 8

[5] Yiwen Chen, Yikai Wang, Yihao Luo, Zhengyi Wang, Zilong Chen, Jun Zhu, Chi Zhang, and Guosheng Lin. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. *arXiv preprint arXiv:2408.02555*, 2024. 2, 4, 5, 6, 8

[6] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 45–54, 2020. 8

[7] Zilong Chen, Yikai Wang, Feng Wang, Zhengyi Wang, and Huaping Liu. V3d: Video diffusion models are effective 3d generators. *arXiv preprint arXiv:2403.06738*, 2024. 8

[8] Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4456–4465, 2023. 8

[9] Gene Chou, Yuval Bahat, and Felix Heide. Diffusion-sdf: Conditional generative modeling of signed distance functions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2262–2272, 2023. 8

[10] Angela Dai and Matthias Nießner. Scan2mesh: From unstructured range scans to 3d meshes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5574–5583, 2019. 8

[11] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 5, 8

[12] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 5

[13] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *International Journal of Computer Vision*, 129:3313–3337, 2021. 5

[14] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 216–224, 2018. 8

[15] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3DGen: Triplane Latent Diffusion for Textured Mesh Generation, 2023. 8

[16] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *arXiv preprint arXiv:2311.04400*, 2023. 8

[17] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 8

[18] Jiahao Li, Hao Tan, Kai Zhang, Zexiang Xu, Fujun Luan, Yinghao Xu, Yicong Hong, Kalyan Sunkavalli, Greg Shakhnarovich, and Sai Bi. Instant3d: Fast text-to-3d with sparse-view generation and large reconstruction model. *arXiv preprint arXiv:2311.06214*, 2023. 8

[19] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3D: High-Resolution Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 8

[20] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot One Image to 3D Object, 2023. 8

[21] Zhen Liu, Yao Feng, Michael J. Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. MeshDiffusion: Score-based Generative 3D Mesh Modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 8

[22] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. arXiv, 1998. 8

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6

[24] Zhaoyang Lyu, Jinyi Wang, Yuwei An, Ya Zhang, Dahua Lin, and Bo Dai. Controllable mesh generation through sparse latent point diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 271–280, 2023. 8

[25] Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4328–4338, 2023. 8

[26] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of

3d meshes. In *International conference on machine learning*, pages 7220–7229. PMLR, 2020. 2, 8

[27] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 5

[28] William Peebles and Saining Xie. Scalable Diffusion Models with Transformers, 2023. 4, 5

[29] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *ICLR*. arXiv, 2023. 8

[30] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022. 8

[31] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S. Sara Mahdavi, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022. 8

[32] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021. 8

[33] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023. 8

[34] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 8

[35] Jaehyeok Shim, Changwoo Kang, and Kyungdon Joo. Diffusion-based signed distance fields for 3d shape generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20887–20897, 2023. 8

[36] Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. *arXiv preprint arXiv:2311.15475*, 2023. 2, 4, 8

[37] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. *arXiv preprint arXiv:2402.05054*, 2024. 8

[38] Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024. 2, 4, 5, 6, 8

[39] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. *arXiv preprint arXiv:2403.12008*, 2024. 8

[40] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018. 8

[41] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, et al. Rodin: A generative model for sculpting 3d digital avatars using diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4563–4573, 2023. 8

[42] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-Fidelity and Diverse Text-to-3D Generation with Variational Score Distillation, 2023. 8

[43] Zhengyi Wang, Yikai Wang, Yifei Chen, Chendong Xiang, Shuo Chen, Dajiang Yu, Chongxuan Li, Hang Su, and Jun Zhu. Crm: Single image to 3d textured mesh with convolutional reconstruction model. *arXiv preprint arXiv:2403.05034*, 2024. 8

[44] Haohan Weng, Tianyu Yang, Jianan Wang, Yu Li, Tong Zhang, CL Chen, and Lei Zhang. Consistent123: Improve consistency for one image to 3d object synthesis. *arXiv preprint arXiv:2310.08092*, 2023. 8

[45] Haohan Weng, Yikai Wang, Tong Zhang, CL Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. *arXiv preprint arXiv:2405.16890*, 2024. 2, 4, 8

[46] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024. 8

[47] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, et al. Dmv3d: Denoising multi-view diffusion using 3d large reconstruction model. *arXiv preprint arXiv:2311.09217*, 2023.

[48] Yinghao Xu, Zifan Shi, Wang Yifan, Hansheng Chen, Ceyuan Yang, Sida Peng, Yujun Shen, and Gordon Wetzstein. Grm: Large gaussian reconstruction model for efficient 3d reconstruction and generation. *arXiv preprint arXiv:2403.14621*, 2024. 8

[49] Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):1–16, 2023. 8

[50] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in Neural Information Processing Systems*, 36, 2024. 5, 1

[51] Chuanxia Zheng and Andrea Vedaldi. Free3d: Consistent novel view synthesis without 3d representation. *arXiv preprint arXiv:2312.04551*, 2023. 8

[52] Xin-Yang Zheng, Hao Pan, Peng-Shuai Wang, Xin Tong, Yang Liu, and Heung-Yeung Shum. Locally attentional sdf diffusion for controllable 3d shape generation. *ACM Transactions on Graphics (TOG)*, 42(4):1–13, 2023. 8

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 9 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.
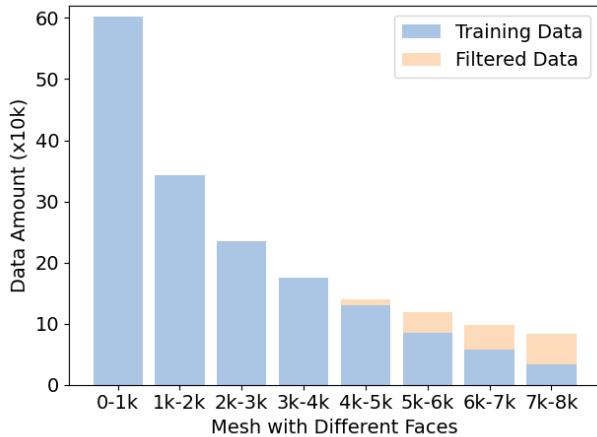

Figure 9. **Effective utilization of training data.** Almost all meshes under 5k faces are used, and around 58% of meshes with more than 5k faces are utilized further.

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.

B. Model Architecture

As shown in Figure 10, the overall architecture of our model follows Michelangelo [50]. As shown in Figure 10, 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

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 11, the meshes generated by our model are at the product-ready level.
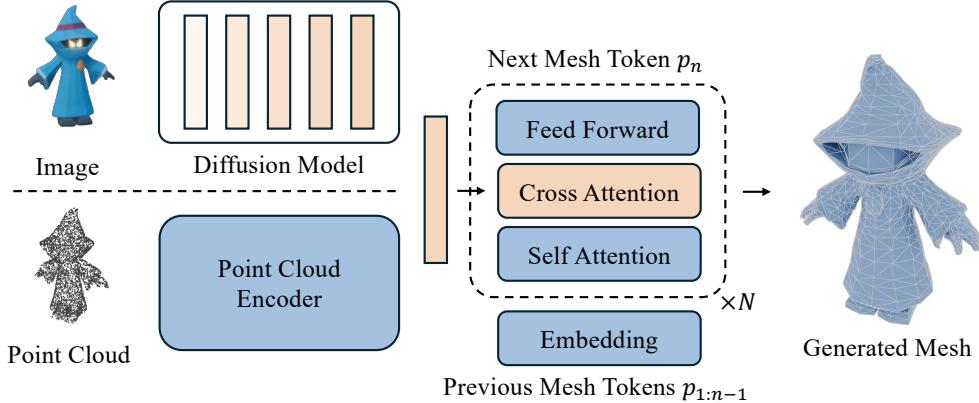
Figure 10. **Model architecture for conditional mesh generation.** First, we leverage an auto-regressive transformer to generate meshes conditioned on point-cloud features via cross-attention layers. Next, we train an additional diffusion model to generate point-cloud features based on images, enabling image-to-mesh generation.
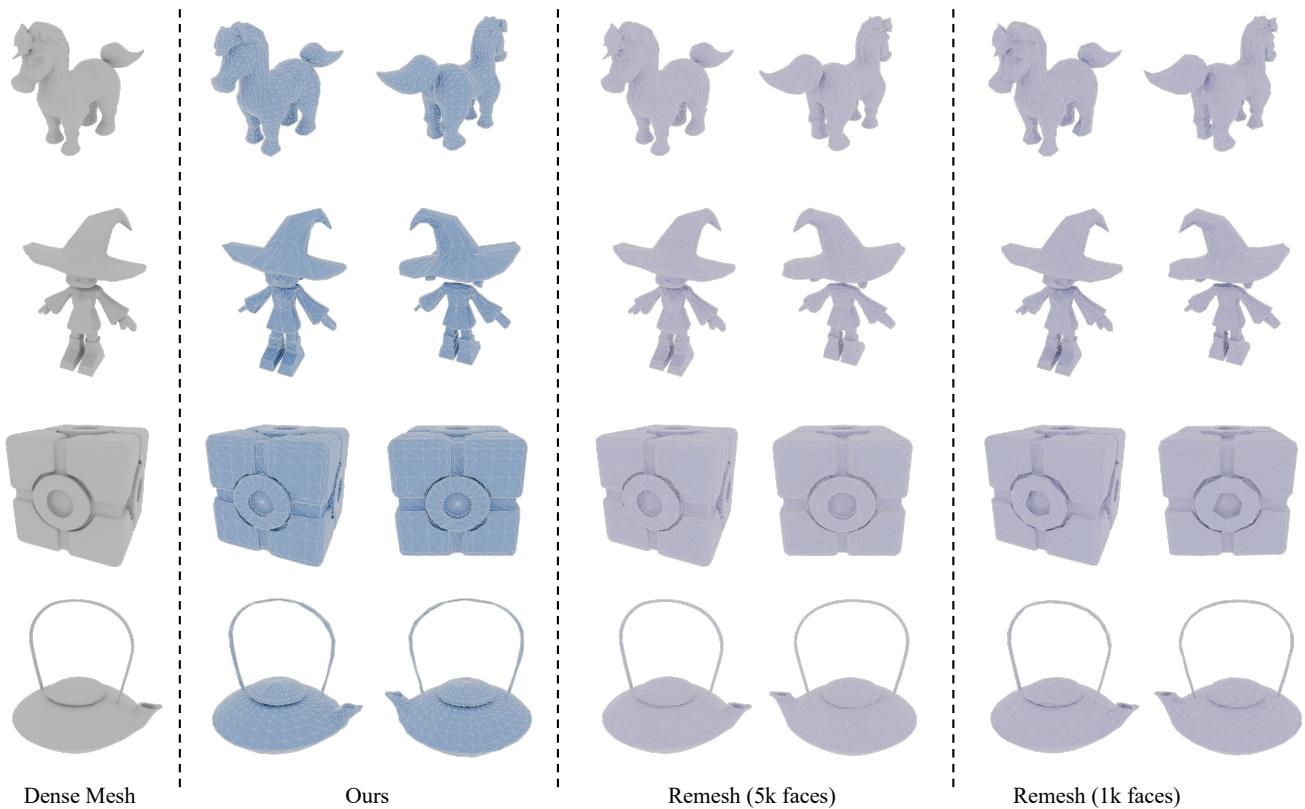

Figure 11. **Comparison with remeshing.** Our method can generate appropriate topology from dense meshes, while remesh algorithms fundamentally fail to capture models' geometry and produce poor topology.

2