FastMesh: Efficient Artistic Mesh Generation via Component Decoupling

Jeonghwan Kim Yushi Lan Armando Fortes Yongwei Chen Xingang Pan
S-Lab, Nanyang Technological University
https://jhkim0759.github.io/projects/FastMesh

Figure 1. Example of meshes generated by FastMesh. Our approach efficiently produces 3D objects by substantially reducing the number of tokens required for generation. Note that all meshes are directly generated from point clouds.

Abstract

Recent mesh generation approaches typically tokenize triangle meshes into sequences of tokens and train autoregressive models to generate these tokens sequentially. Despite substantial progress, such token sequences inevitably reuse vertices multiple times to fully represent manifold meshes, as each vertex is shared by multiple faces. This redundancy leads to excessively long token sequences and inefficient generation processes. In this paper, we propose an efficient framework that generates artistic meshes by treating vertices and faces separately, significantly reducing redundancy. We employ an autoregressive model solely for vertex generation, decreasing the token count to approximately 23% of that required by the most compact existing tokenizer. Next, we leverage a bidirectional transformer to complete the mesh in a single step by capturing inter-vertex relationships and constructing the adjacency matrix that defines the mesh faces. To further improve the generation quality, we introduce a fidelity enhancer to refine vertex positioning into more natural arrangements and propose a post-processing framework to remove undesirable edge connections. Experimental results show that our method achieves more than 8× faster speed on mesh generation compared to state-of-the-art approaches, while producing higher mesh quality.

1. Introduction

Industries such as gaming, visual effects, and virtual reality rely heavily on 3D meshes as the core representation, due to their compactness and compatibility with mature rendering pipelines. As the demand for 3D assets continues to grow, creating high-quality meshes through traditional manual modeling remains time-consuming and labor-intensive. This has driven a wave of research into data-driven generative models that aim to automate mesh creation, either from scratch or conditioned on point clouds [5, 8, 12, 14, 36, 45, 47]. These approaches typically represent a mesh as

a sequence of tokens and train an LLM–style architecture to generate the tokens autoregressively.

Despite promising advances in mesh generation, the next-token prediction paradigm faces a key limitation: it is inefficient in both time and memory due to the long token sequences. In a mesh, multiple faces often share common vertices. As a result, tokenization requires repeatedly recording the same vertices across different tokens, leading to significant redundancy. Consequently, these methods typically take 30 seconds to one minute to generate a mesh with 500 vertices on an A6000 GPU, and they struggle to scale to complex meshes with a large number of vertices. While some works have proposed more efficient tokenization strategies to improve compression ratios [7, 22, 43, 48], repeated references to the same vertex tokens remain unavoidable when encoding geometric information into a single sequence.

In this paper, we introduce FastMesh, an efficient framework for generating high-quality 3D meshes within seconds. The key idea is to decouple the generation of mesh components—vertices and faces—and process them sequentially to avoid duplication issues commonly encountered in mesh tokenization. For vertex generation, we employ an autoregressive model, as it inherently accommodates varying vertex counts. Once vertices are generated, we recognize that their connectivity primarily depends on local information, allowing for parallel processing. To facilitate this, we leverage a bidirectional transformer [42] to model the relationships between vertices, from which edge connections can be directly derived in one step. These edges form an adjacency matrix that is then used to extract faces by identifying closed triangles. Thanks to this design, FastMesh reduces the token count to approximately 23% of that required by the previous most compact tokenizer [48], significantly mitigating issues associated with long token sequences, such as high inference latency and quality degradation.

Beyond this, we introduce a fidelity enhancer module and a prediction filtering process to further improve mesh quality. The fidelity enhancer maps discretized vertex positions, which are constrained by indexing [48], back to continuous coordinates. This results in smoother surfaces and more natural vertex distributions. In addition, the prediction filtering, a post-processing technique for face generation, refines the adjacency matrix by progressively masking irrelevant connections. This reduces redundant or spurious faces while preserving the intended geometric structure, yielding meshes that are cleaner, more compact, and better aligned with downstream requirements.

Extensive experiments on the Toys4K dataset [37] demonstrate that FastMesh can generate higher-quality 3D meshes with significantly less time than previous methods. When generating meshes with 1,000 and 4,000 vertices, our method only takes about 7 seconds and 30 seconds, respectively, achieving an $8\times$ speedup over BPT [48]. In particular, since our mesh is generated in continuous space with a higher vertex count, it achieves not only greater geometric accuracy but also higher visual quality. Furthermore, we show that our method can be effectively integrated with other 3D generation pipelines that produce non-mesh 3D assets, enabling the creation of artistic meshes aligned with diverse input modalities such as images or text. As illustrated in Fig. 1, FastMesh accurately represents a diverse range of complex structures conditioned on shape inputs. Our contributions can be summarized as follows:

- We propose an efficient framework for high-quality mesh generation, which treats vertices and faces separately, each with a dedicated model suitable for the task. In particular, once the vertices are ready, all the mesh edges are generated in parallel via a bidirectional transformer. Our design reduces the token count to about 23% of that required by the previous most compact tokenizer [48].
- We integrate a fidelity enhancer to improve the precision of vertex representations by restoring information lost during quantization. Additionally, we propose prediction filtering to reduce connectivity errors while preserving geometric structure and mesh quality.
- FastMesh shows a clear advantage through the Toys4K dataset by generating more detailed and accurate meshes at a significantly faster speed compared to prior methods.

2. Related Work

2.1. 3D Mesh Generation

Early studies [13, 16, 26, 29, 41, 50] on mesh generation, constrained by the scarcity of large-scale 3D datasets, trained on limited object categories, and often failed to generalize to out-of-distribution classes. DreamFusion [30] pioneered the use of score distillation loss to train 3D models guided by diffusion priors [32, 33], inspiring a series of follow-up works [4, 6, 21, 39, 44] aimed at mitigating the lack of 3D data. With the release of large-scale 3D datasets [10, 11], feed-forward methods [15, 20, 23, 46, 51] demonstrated high-speed 3D generation, while subsequent diffusion-based approaches [9, 17, 18, 24, 49] produced more detailed and higher-quality outputs. Most of these methods utilize intermediate representations (*e.g.*, point clouds, implicit functions, and triplane) rather than meshes to facilitate the training of 3D geometry. These are typically converted into meshes through post-processing techniques [2, 25, 34], enabling use in downstream tasks. However, such pipelines often yield dense meshes with overly smooth surfaces or cause distortions through interpolation, failing to faithfully represent the intended 3D structure. Motivated by these limitations, a line of research has emerged on generating complete meshes end-to-end with

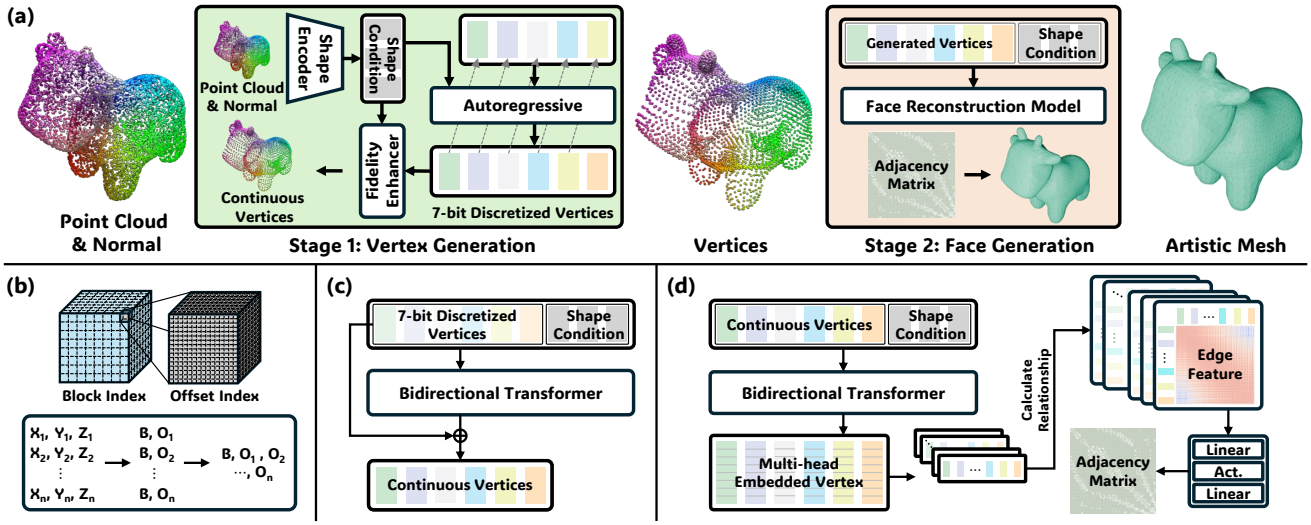

Figure 2. **(a)** Overall architecture of FastMesh. Note that our pipeline consists of two stages, where we first generate the vertices from the shape condition and then construct the faces to complete the mesh. **(b)** Visualization of the block-wise indexing scheme introduced by BPT [48], which we adopt for vertex tokenization. **(c)** Structure of the fidelity enhancer in the first stage. The 7-bit discretized vertices and shape condition are fed into the network to estimate the offset that can make the coordinate a continuous value. **(d)** Details of face reconstruction. The generated vertices are embedded to capture inter-vertex relationships in a multi-head manner. Each head computes a matrix, where the output represents one feature dimension used in edge prediction.

neural networks. Notably, PolyGen [28], MeshGPT [36], and PolyDiff [1] were among the first to leverage generative models for creating mesh triangles, producing results that closely resemble those crafted by human artists.

2.2. Shape-conditioned Artistic Mesh Generation

Since the initial works on artistic mesh generation were published, MeshXL [5] and MeshAnything [8] proposed the autoregressive model to create meshes by using shape information in the form of point clouds as input. It is worth noting that these designs enhance the scalability of artistic mesh generation, since all 3D representations can be easily converted into point clouds. However, since each triangle requires three vertex coordinates—nine tokens in total—these approaches struggle as the sequence length grows with the number of faces. This makes it difficult for the network to handle complex meshes. Meshtron [14] designed a hierarchical structure to efficiently process large amounts of tokens. More recent efforts [7, 22, 40, 43, 48] have proposed compact tokenization mechanisms that encode mesh geometry into shorter sequences while preserving topological consistency. Among them, BPT [48] introduces two promising schemes—block-wise indexing and patchified aggregation—achieving a compression rate of about 75%. This substantial reduction in computational burden enables the network to process more complex and large-scale meshes with up to 8,000 faces. However, token compression alone does not fully resolve scalability—the sequence length remains lower-bounded by the number of faces, and discretization artifacts introduced by block-wise indexing persist. On the other hand, SpaceMesh [35] adopts a diffusion model for vertex generation, followed by embeddings to capture inter-vertex relationships, and constructs faces by traversing edges in a half-edge representation. Although this approach can generate meshes within a few seconds, it suffers from low geometric accuracy and is restricted to low-poly outputs, primarily due to instability in predicting both vertices and edges. In contrast, FastMesh is capable of generating high-complexity meshes with detailed structures while maintaining faster inference. This design not only overcomes the limitations of existing methods but also underscores the value of revisiting decoupled design principles.

3. Proposed Method

The proposed method addresses the inherent redundancy in vertex usage when tokenizing entire meshes as a single sequence, thereby facilitating faster inference and more effective handling of complex mesh structures. Similar to prior works [7, 40, 48], we extract shape condition features from input point clouds and normals using a pretrained encoder [52]. Subsequently, the network generates vertices and faces to complete the mesh. Accordingly, the overall architecture consists of two main stages, as shown in Fig. 2(a).

3.1. Stage I: Vertex Generation

Since vertex generation significantly influences the overall mesh quality, we designed our approach to ensure high-

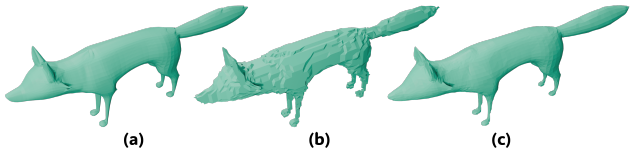

Figure 3. **(a)** Example of high-resolution mesh containing 7,694 vertices. **(b)** Mesh obtained by discretizing (a) in 7-bit coordinate space, resulting in 3,636 vertices. **(c)** Mesh reconstructed using the same 3,636 vertices as in (b), with continuous coordinates refined by our fidelity enhancer.

quality vertex prediction. We first generate vertex sequences using an autoregressive model and then refine the vertex positions with the fidelity enhancer to achieve more precise and naturally arranged vertex distributions. To support efficient generation in the autoregressive model, we employ the block-wise indexing method to compress the vertex representation sequence effectively. This technique, originally introduced in BPT [48], is illustrated in Fig. 2(b). Specifically, block-wise indexing maps XYZ coordinates into two distinct values (*i.e.*, the block index and the offset index), which represent the discretized 3D space separated by the global block and the local offset. Furthermore, the vertices are integrated within each block, assigning the block index once per group in the sequence. This compresses the final sequence length to roughly match the number of vertices. Where the number of faces (F) is about twice the number of vertices (V), and each face in the vanilla mesh representation (S) requires nine tokens, we obtain $S = 9F \approx 18V$. This reduction in sequence length significantly improves inference speed and stabilizes generation by reducing the accumulation of prediction error.

Even though block-wise indexing efficiently compresses the sequential representation, its reliance on 3D space discretizations leads to the loss of fine geometric details. As the complexity of 3D objects increases and the number of vertices grows, higher spatial resolution becomes necessary, which further amplifies this degradation. This issue primarily stems from two factors: the shifting of vertex positions to quantized points and the reduction of points through the merging of neighbors. As shown in Fig. 3, discretizing an object with 7-bit resolution noticeably reduces geometric detail, while adjusting only the vertex positions to continuous values restores most of the original geometry. Motivated by this observation, we design the fidelity enhancer, a small transformer structure, that receives 7-bit discretized vertex positions and shape information to convert each vertex into a continuous coordinate by predicting the residual, as illustrated in Fig. 2(c).

3.2. Stage II: Face Generation

In this stage, we embed the vertices by using the bidirectional transformer to capture inter-vertex relationships and predict the edges to finalize the mesh structure. Since SpaceMesh [35] demonstrates the effectiveness of the spacetime distance function [19] for modeling relational structures, we adopt this function to compute interactions between vertices, defined as follows:

$$\begin{aligned} \text{dst}(u, v) &= \text{dst}([u_1, u_2], [v_1, v_2]) \\ &= \|u_1 - v_1\|_2^2 - \|u_2 - v_2\|_2^2 \end{aligned} \tag{1}$$

where $\text{dst}(\cdot)$ denotes the spacetime distance function, and u and v are input vectors, corresponding to embedded vertex features in our setting. Each vector is split in half, and the Euclidean distance is computed separately for the two parts. The final value is computed by subtracting the second distance from the first, enabling the representation of both positive and negative values. Instead of using the function as the final activation, as done in SpaceMesh, we employ a multi-head approach in which the feature vector is split and the function is applied independently across the heads, with each result forming the distinct dimension of the edge feature. This edge feature is then passed through a prediction network, which outputs the edge logits for determining vertex connectivity(see Fig. 2). This strategy significantly improves the network's capacity to represent diverse and complex connectivity. Lastly, we threshold the logits at zero to determine edge connectivity, then construct mesh faces by identifying triplets of vertices that are mutually connected.

In the training phase, we adopt the asymmetric loss [31] to guide edge predictions, emphasizing the scarce positives whose proportion declines as the vertex count grows. Additionally, we prioritize reducing false-positive edges that create holes, rather than pursuing class balance, to preserve geometric fidelity. The loss is defined as follows:

$$\text{AsyLoss}(p) = \begin{cases} L_+ = (1-p)^{\gamma_+} \log(p), \\ L_- = p^{\gamma_-} \log(1-p), \end{cases} \tag{2}$$

where L_+ and L_- represent the loss values for positive and negative samples, respectively, and γ_+ and γ_- are scaling factors controlling how steeply the loss decreases as predictions become more confident, set to 0 and 3 in our experiments. It is noteworthy that the face generation process is executed in a single feedforward, which can achieve fast inference while producing sophisticated results.

3.3. Post-processing: Prediction Filtering

Our face generation process occasionally predicts incorrect links, most of which do not significantly affect the mesh structure due to overlaps, but they increase computational cost. To address this, we introduce prediction filtering, a post-processing strategy applied after the initial adjacency matrix prediction. The bandwidth of an adjacency matrix is defined as the maximum $|i - j|$ for which $A[i][j] \neq 0$,

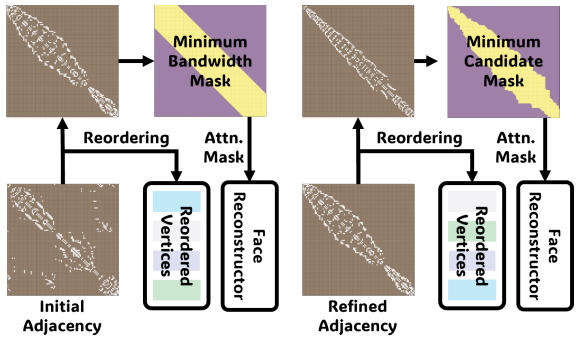

Figure 4. The detailed structure of the prediction filtering. We use the initial adjacency matrix from the first face generation to perform BFS reordering. Based on this reordering, we apply the minimum bandwidth mask and the minimum candidate mask as attention masks.

representing the farthest distance from the diagonal where a connection occurs. We first reorder nodes via breadth-first search to reduce bandwidth and construct a minimal bandwidth mask that retains edges with $|i - j| \leq B$. Using this mask, we re-predict edges over two iterations, narrowing the candidate set at each step. Subsequently, we apply a minimum candidate mask, which assigns each node i a maximum valid connection distance r_i (retaining neighbors j with $|i-j| \leq r_i$), and repeat the two-iteration refinement. The entire process is illustrated in Fig 4. This two-stage, iterative filtering reduces unnecessary faces while preserving geometric fidelity.

4. Experiments

4.1. Implementation details

Model Variants. We introduce two variants of our method, i.e., FastMesh-V4K and FastMesh-V1K. The two variants share an identical model structure, but differ in the filtering applied to the training data: FastMesh-V4K is trained on meshes with up to 4,000 vertices, whereas FastMesh-V1K is trained on a subset filtered to include only meshes with no more than 1,000 vertices.

Datasets. To train the full network, we combine ShapeNet [3], Objaverse [10], and a portion of Objaverse-XL [11], selecting samples with fewer than 4,000 vertices. We further filter out undesirable meshes with our algorithm and finally obtain a dataset of 100K high-quality meshes. For inference, we use the Toys4K dataset [37], which contains 4,000 meshes with a diverse range of complexity.

Evaluation Metrics. To quantitatively evaluate the quality of mesh generation, we adopt Chamfer Distance (**CD**) and Hausdorff Distance (**HD**) as evaluation metrics. Both the generated and ground-truth meshes are normalized in scale, and 5,000 points are uniformly sampled from each mesh surface. CD measures the average closest-point distance between two point sets, capturing the overall structural similarity. HD captures the largest deviation by measuring the point that is farthest from any point in the other set, making it sensitive to local errors such as holes or incorrectly reconstructed regions. Additionally, we calculate the average inference time (**Inf. Time**) and number of vertices (**#V**) per mesh to demonstrate the efficiency of our framework. In the experiment for ablation studies, we also use the number of faces (**#F**), F1-score, and recall.

Table 1. Quantitative comparison on the Toys4K dataset. The best and second best results in each category are **bold** and underlined respectively.

Method	CD (%) ↓	HD(%) ↓	Inf. Time(s) ↓	#V
MeshAnything	12.02	26.87	26.06	218.6
MeshAnythingV2	10.23	24.98	31.94	533.3
TreeMeshGPT	5.46	13.96	27.32	706.3
BPT	5.71	12.02	49.23	525.5
FastMesh-V1K	4.09	10.32	**3.41**	467.2
FastMesh-V4K	**4.05**	**10.22**	6.60	1040.6

4.2. Performance Evaluation

To demonstrate the effectiveness of our method, we conduct both quantitative and qualitative evaluations by comparing it with state-of-the-art approaches for shape-conditioned artistic mesh generation (*i.e.*, MeshAnything [7, 8], TreeMeshGPT [22] and BPT [48]). For quantitative evaluation, we use the Toys4K dataset [37], which none of the compared models have used for training. In qualitative evaluation, we additionally utilize the samples from the ObjaverseXL dataset [11]. The results for comparisons are conducted without using post-processing.

Quantitative. For evaluation, we used the Toys4K dataset by selecting meshes with fewer than 5,000 vertices, resulting in a total of 2,063 samples. The comparison results are presented in Table 1. As shown in the results, the FastMesh-V4K achieves the best performance, with scores of 4.05 and 10.22 on Chamfer distance and Hausdorff distance, respectively. In terms of generation time per vertex, FastMesh-V4K is eight times faster compared with BPT while having better geometry representation. Moreover, FastMesh-V1K demonstrates the fastest inference time, with an average of approximately 3.41 seconds per mesh, which is nearly twice as fast as FastMesh-V4K, owing to its use of fewer vertices. It also shows that the stability of mesh generation by using such long sequences often fails due to the accumulation of prediction errors during generation, which further deteriorates overall performance. It is noteworthy that our proposed approach

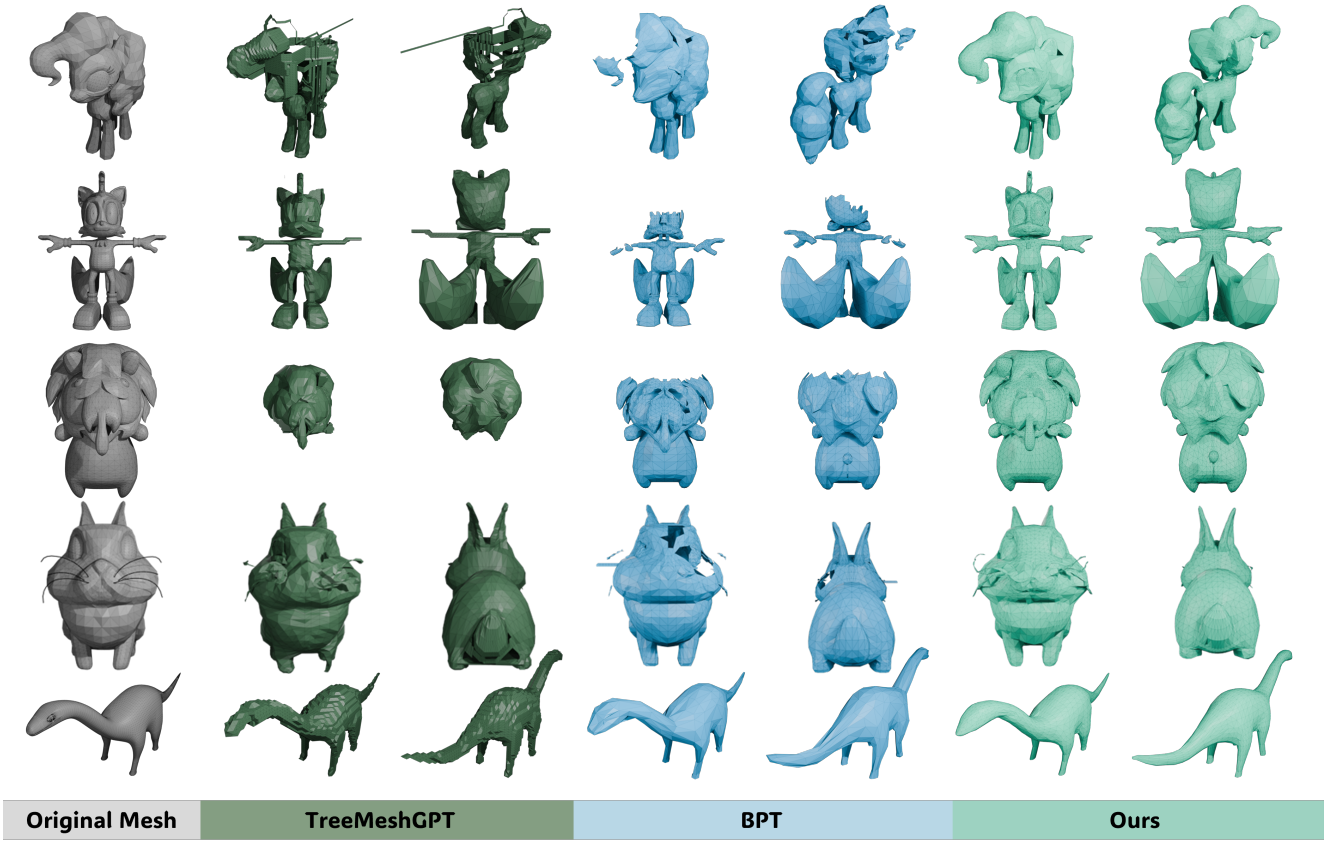

Figure 5. Qualitative comparison of shape-conditioned mesh generation on the Toys4K dataset [37]. All meshes were generated from the same input point clouds that were sampled from the original meshes.

deals with shorter sequences that mitigate not only the inference speed but also the overall performance.

Qualitative. We conducted a qualitative evaluation using high-resolution meshes from the Toys4K dataset, as shown in Fig 5. As illustrated, the proposed method consistently produces more accurate and refined meshes compared to other approaches. Notably, other methods often fail to finalize the mesh structure, resulting in incomplete or distorted shapes. In contrast, our method generates complete and coherent geometry, effectively preserving fine-grained features throughout the generation process. This is due not only to token-length constraints but also to the growing instability at longer sequences, as shown in the first example. There are also cases where the overall shape is preserved, but detailed features are not well represented. For instance, in the fourth row of the comparison with TreeMeshGPT, the rabbit exhibits simplified facial structures, particularly poorly defined eyes and unnaturally thin arms, leading to lower visual quality. We also demonstrate the effectiveness of our fidelity enhancer through the example in the last row by showing smoother surfaces. Additionally, we present multi-angle views of our generated meshes across samples of varying complexity from the Objaverse dataset in Fig. 6, demonstrating that our results are consistently well-structured.

4.3. Ablation Studies

In this subsection, we demonstrate the validity structure and loss function for face generation by the comparative experimental results according to the change of components. For the experiment, we train the model on the ShapeNet dataset and evaluate it on the Toys4K dataset by using meshes with fewer than 500 vertices in both. Moreover, we demonstrate the effectiveness of the fidelity enhancer, which locates vertex positions in a natural arrangement.

Edge Prediction. To demonstrate the effectiveness of the structure for the final step in the face generation, i.e., predicting edge connectivity using embedded vertex features, we compare our method against four different designs: using spacetime distance function as final activation as used in SpaceMesh [35]; computing the function multiple times across split feature groups and averaging the results as logits; replacing the spacetime distance function to cosine similarity; and following the multi-head manner which use the

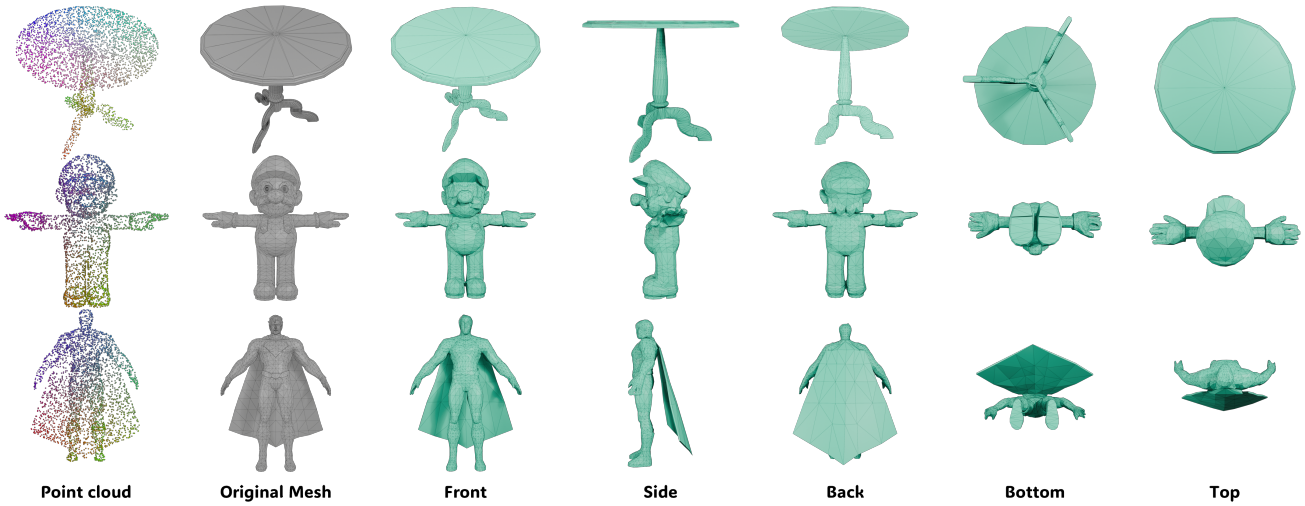

Figure 6. Diverse view of mesh results from the proposed method in the Objaverse dataset [10].

Table 2. Performance analysis of the face generation according to the change of the edge prediction structure based on the Toys4K dataset.

Function	Multi-head	MLP	CD (%) ↓	HD(%) ↓
Spacetime	✗	✗	7.27	25.32
Spacetime	✓	✗	5.72	21.41
Cosine	✓	✓	5.78	22.05
Spacetime	✓	✓	**5.06**	**18.55**

Table 3. Performance analysis of the face generation according to the change of the loss function based on the Toys4K dataset.

Loss	CD(%) ↓	HD(%) ↓	F1(%) ↑	Recall(%) ↑
BCE	11.62	36.27	69.45	64.07
Dice	10.88	35.68	67.60	71.20
Asymmetric	**5.06**	**18.55**	**70.58**	**81.32**

MLP block to predict logits. The following results are shown in Table 2. Based on the comparison between the first and second settings, the multi-head configuration provides a significant improvement over the case that uses a single vector. This indicates that a single vector is insufficient to capture all complex relationships between vertices. Furthermore, the design using the spacetime distance function outperforms the one using cosine similarity, reaffirming the effectiveness of the spacetime distance function in graph representation, as demonstrated in previous work. [19, 35].

Loss Function. We conducted a comparative analysis of loss functions to evaluate their importance in training a face generation model. We compared asymmetric loss [31] with binary cross-entropy, dice loss [27], which are commonly used in binary classification tasks. In addition to standard geometry-based metrics, we also used F1 score and recall as evaluation metrics to better understand the relationship between the accuracy of matrices and the actual quality of the mesh. The experimental results show that the asymmetric loss consistently outperformed the other loss functions across all metrics. Since the asymmetric loss forces the model to focus more heavily on learning from positive samples, it leads to more accurate predictions, despite the low proportion of positive samples in the matrix. Dice loss, which also emphasizes learning from positive values, showed improved performance in Chamfer Distance (CD) and Hausdorff Distance (HD) compared to BCE loss. However, it does not enforce this emphasis as strongly as the asymmetric loss, resulting in comparatively lower overall performance. Notably, although the F1 score remains similar across all loss functions, the model trained with asymmetric loss achieves a significantly higher recall with improved shape metric scores. This suggests that positive predictions contribute more critically to the representation of mesh structures, as improved recall is strongly associated with better shape metric scores.

Table 4. Performance analysis of face generation with and without prediction filtering (**PF**) on the Toys4K dataset. #F and #V denote the number of faces and vertices, respectively.

PF	CD(%) ↓	HD(%) ↓	Inf. Time(s) ↓	#F	#V
✗	4.05	10.22	6.77	6799.2	1040.6
✓	4.03	11.02	12.59	2811.1	1040.1

Prediction Filtering. We conducted an experiment to evaluate how effectively unnecessary faces can be removed through prediction filtering. We compare with generation metrics along with the average number of faces and vertices in each result, as shown in Table 4. As observed, the results without post-processing show that the number of faces is excessively higher than the number of vertices. Most of these extra faces are overlapping surfaces lying on the same plane. Although they do not affect the mesh geom-

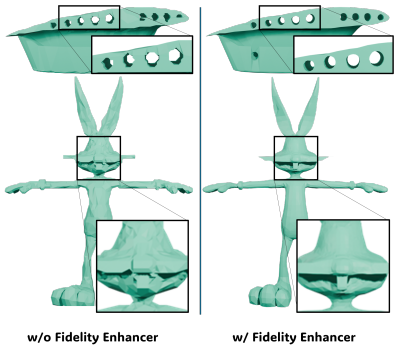

Figure 7. Comparison of the mesh results from FastMesh-V4K according to the usage of the fidelity enhancer.

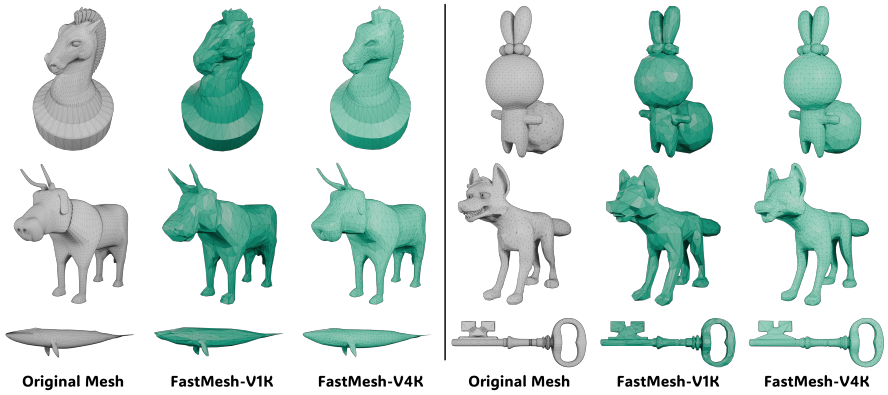

Figure 8. Comparison results of the two variants of the proposed method. The meshes are generated from input point clouds extracted from the original meshes for each model.

etry, they result in unnecessary computational overhead in downstream applications. Although post-processing leads to a slight decrease in Hausdorff Distance due to removing and making holes in the failure cases, it improves Chamfer Distance while substantially reducing the number of faces. This indicates that the method effectively removes unnecessary faces without compromising geometric quality.

Fidelity Enhancer We evaluated meshes generated with and without the fidelity enhancer under identical conditions and found clear improvements in facial details and smoother surfaces (Fig. 7), demonstrating that our simple refinement network can significantly improve mesh quality by adjusting vertex arrangements.

4.4. Discussion

Variants We introduced two variants that exhibit different generation behaviors by training with different data filtering. FastMesh-V1K is aimed at accelerating the process, while FastMesh-V4K prioritizes mesh quality. As shown in Table 1, although FastMesh-V1K handles significantly fewer vertices than FastMesh-V4K, the performance in terms of Chamfer Distance (CD) and Hausdorff Distance (HD) remains comparable. This is because these geometry-based metrics are more sensitive to overall structural accuracy than to fine-grained details, and FastMesh-V1K is still able to capture the mesh structure effectively with fewer vertices. As illustrated in Fig. 8, even when given a complex structure, FastMesh-V1K successfully reconstructs the overall geometry without structural collapse. On the other hand, FastMesh-V4K processes a larger number of tokens, trading off some generation speed for higher detail. This allows it to generate meshes with greater vertex density, resulting in smoother and more elaborate surfaces.

Limitations and Future Work While our method shows strong performance, several limitations remain. In vertex generation, the model can occasionally produce overly fine-grained sequences that exceed the maximum vertex limit. In face generation, it may remove valid faces, retain invalid ones, or produce overlapping faces, meaning manifoldness is not guaranteed. Details and examples are provided in the supplementary material. Future work will explore relative positional encoding [38] to allow longer sequence generation, refine the face generation architecture, and introduce constraints or additional pipelines to ensure manifoldness.

5. Conclusion

In this paper, we propose a simple yet powerful framework for artistic mesh generation. We address the challenge of representing the mesh in a single sequence by treating the vertices and surfaces of the mesh separately. We enable the autoregressive model to process only a few tokens for the process, which results in faster inference speed and more stable results. To mitigate the discretization issue introduced by the vertex quantization process, we apply the fidelity enhancer, which refines vertex positions. Furthermore, we reduce the number of unnecessary faces through the prediction reduction process, making the results from our method to be in well-designed. Our experimental results demonstrate that FastMesh delivers significantly more robustness across diverse shapes compared to prior methods, while generating highly detailed meshes up to $8\times$ faster. We have shown that the decoupled approach offers an efficient solution for mesh generation, and we believe it will inspire further research in the field.

Acknowledgements

This research is supported by the National Research Foundation, Singapore, under its NRF Fellowship Award NRF-NRFF16-2024-0003. This research is also supported by NTU SUG-NAP, as well as cash and in-kind funding from NTU S-Lab and industry partner(s).

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. 3

[2] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Cláudio Silva, and Gabriel Taubin. The ball-pivoting algorithm for surface reconstruction. *IEEE transactions on visualization and computer graphics*, 5(4):349–359, 2002. 2

[3] 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

[4] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. In *CVPR*, pages 22246–22256, 2023. 2

[5] Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Zhibin Wang, Jingyi Yu, Gang Yu, BIN FU, and Tao Chen. MeshXL: Neural coordinate field for generative 3d foundation models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 1, 3

[6] Yongwei Chen, Tengfei Wang, Tong Wu, Xingang Pan, Kui Jia, and Ziwei Liu. Comboverse: Compositional 3d assets creation using spatially-aware diffusion guidance. In *ECCV*, pages 128–146. Springer, 2024. 2

[7] 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, 2024. 2, 3, 5

[8] Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. Meshanything: Artist-created mesh generation with autoregressive transformers, 2025. 1, 3, 5

[9] Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia-xl: High-quality 3d pbr asset generation via primitive diffusion. In *CVPR*, 2025. 2

[10] 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. *arXiv preprint arXiv:2212.08051*, 2022. 2, 5, 7

[11] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-xl: A universe of 10m+ 3d objects. *arXiv preprint arXiv:2307.05663*, 2023. 2, 5

[12] Daoyi Gao, Yawar Siddiqui, Lei Li, and Angela Dai. Meshart: Generating articulated meshes with structure-guided transformers. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025. 1

[13] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. Get3d: A generative model of high quality 3d textured shapes learned from images. *Advances In Neural Information Processing Systems*, 35:31841–31854, 2022. 2

[14] Zekun Hao, David W. Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3d mesh generation at scale, 2024. 1, 3

[15] 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. In *The Twelfth International Conference on Learning Representations*, 2024. 2

[16] Heewoo Jun and Alex Nichol. Shap-e: Generating conditional 3d implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. 2

[17] Yushi Lan, Fangzhou Hong, Shuai Yang, Shangchen Zhou, Xuyi Meng, Bo Dai, Xingang Pan, and Chen Change Loy. Ln3diff: Scalable latent neural fields diffusion for speedy 3d generation. In *European Conference on Computer Vision*, pages 112–130. Springer, 2024. 2

[18] Yushi Lan, Shangchen Zhou, Zhaoyang Lyu, Fangzhou Hong, Shuai Yang, Bo Dai, Xingang Pan, and Chen Change Loy. Gaussiananything: Interactive point cloud latent diffusion for 3d generation. In *International Conference on Learning Representations*, 2025. 2

[19] Marc T Law and James Lucas. Spacetime representation learning. In *The Eleventh International Conference on Learning Representations*, 2022. 4, 7

[20] 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. 2

[21] 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 *CVPR*, pages 300–309, 2023. 2

[22] Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. *CVPR*, 2025. 2, 3, 5

[23] Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Mukund Varma T, Zexiang Xu, and Hao Su. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[24] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10072–10083, 2024. 2

[25] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 2

[26] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4460–4470, 2019. 2

[27] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. Ieee, 2016. 7

[28] Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter W. Battaglia. PolyGen: An autoregressive generative model of 3d meshes. 2020. 3

[29] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022. 2

[30] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[31] Tal Ridnik, Emanuel Ben-Baruch, Nadav Zamir, Asaf Noy, Itamar Friedman, Matan Protter, and Lihi Zelnik-Manor. Asymmetric loss for multi-label classification. In *CVPR*, pages 82–91, 2021. 4, 7

[32] 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. 2

[33] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022. 2

[34] Tianchang Shen, Jacob Munkberg, Jon Hasselgren, Kangxue Yin, Zian Wang, Wenzheng Chen, Zan Gojcic, Sanja Fidler, Nicholas Sharp, and Jun Gao. Flexible isosurface extraction for gradient-based mesh optimization. *ACM TOG*, 42 (4), 2023. 2

[35] Tianchang Shen, Zhaoshuo Li, Marc Law, Matan Atzmon, Sanja Fidler, James Lucas, Jun Gao, and Nicholas Sharp. SpaceMesh: A continuous representation for learning manifold surface meshes. In *SIGGRAPH Asia 2024 Conference Papers (SA Conference Papers '24)*, page 11, New York, NY, USA, 2024. ACM. 3, 4, 6, 7

[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. In *CVPR*, pages 19615–19625, 2024. 1, 3

[37] Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *CVPR*, pages 1798–1808, 2021. 2, 5, 6

[38] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2024. 8

[39] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22819–22829, 2023. 2

[40] 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. In *The Thirteenth International Conference on Learning Representations*, 2025. 3

[41] Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, Karsten Kreis, et al. Lion: Latent point diffusion models for 3d shape generation. *Advances in Neural Information Processing Systems*, 35:10021–10039, 2022. 2

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2

[43] Yuxuan Wang, Xuanyu Yi, Haohan Weng, Qingshan Xu, Xiaokang Wei, Xianghui Yang, Chunchao Guo, Long Chen, and Hanwang Zhang. Nautilus: Locality-aware autoencoder for scalable mesh generation, 2025. 2, 3

[44] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 2

[45] Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. LLaMA-Mesh: Unifying 3d mesh generation with language models, 2024. 1

[46] 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. In *European Conference on Computer Vision*, pages 57–74. Springer, 2024. 2

[47] Haohan Weng, Yikai Wang, Tong Zhang, C. L. Philip Chen, and Jun Zhu. Pivotmesh: Generic 3d mesh generation via pivot vertices guidance. In *The Thirteenth International Conference on Learning Representations*, 2025. 1

[48] Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, Tong Zhang, Shenghua Gao, and C. L. Philip Chen. Scaling mesh generation via compressive tokenization. *CVPR*, 2025. 2, 3, 4, 5

[49] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3d latents for scalable and versatile 3d generation. *arXiv preprint arXiv:2412.01506*, 2024. 2

[50] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2690–2698, 2019. 2

[51] 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. 2

[52] 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. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 3