EdgeRunner: Auto-regressive Auto-encoder for Artistic Mesh Generation

Jiaxiang Tang[1*] **Zhaoshuo Li**[2] **Zekun Hao**[2] **Xian Liu**[2] **Gang Zeng**[1]
Ming-Yu Liu[2] **Qinsheng Zhang**[2]
[1]State Key Laboratory of General Artificial Intelligence, Peking University. [2]NVIDIA Research.

https://research.nvidia.com/labs/dir/edgerunner/

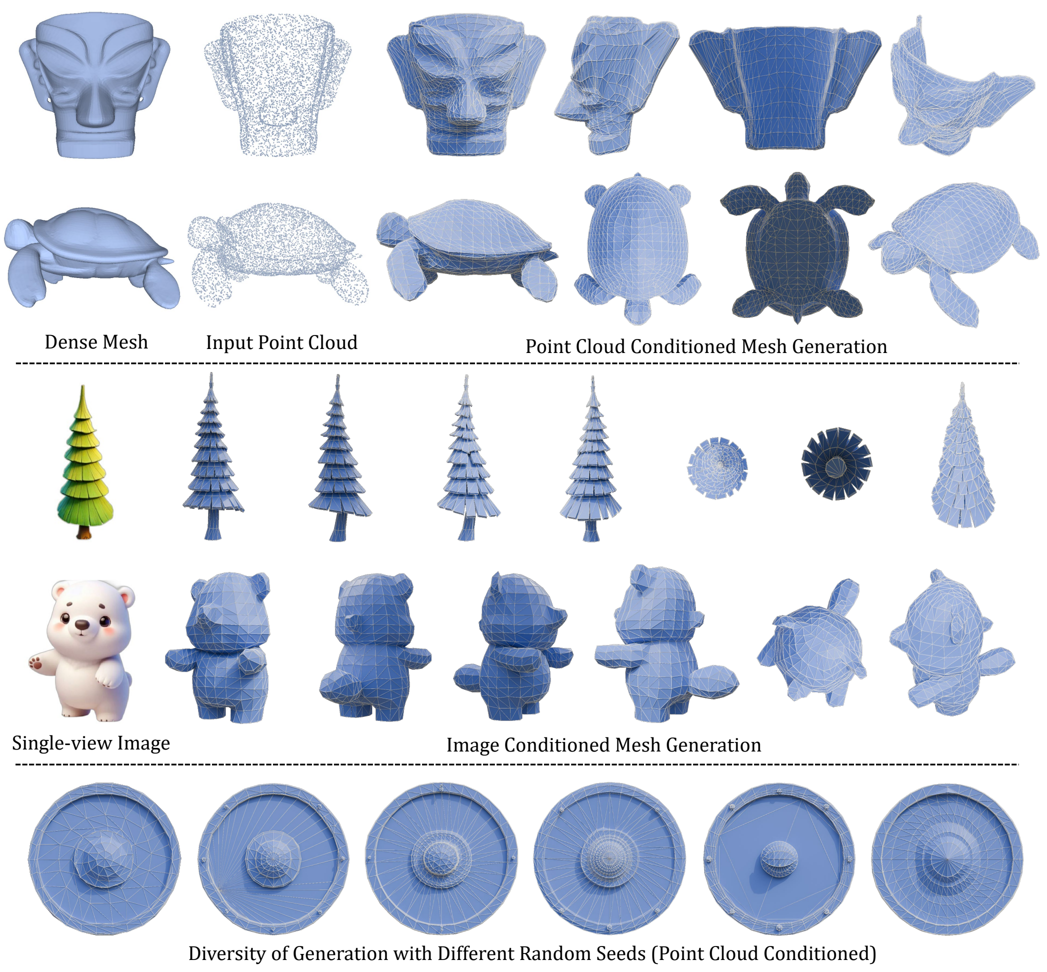

Dense Mesh
Input Point Cloud
Point Cloud Conditioned Mesh Generation
Single-view Image
Image Conditioned Mesh Generation
Diversity of Generation with Different Random Seeds (Point Cloud Conditioned)

Figure 1: **EdgeRunner** efficiently generates diverse, high-quality artistic meshes conditioned on point clouds or single-view images.

Abstract

Current auto-regressive mesh generation methods suffer from issues such as incompleteness, insufficient detail, and poor generalization. In this paper, we propose an Auto-regressive Auto-encoder (ArAE) model capable of generating high-quality 3D meshes with up to 4,000 faces at a spatial resolution of 512^3. We introduce a novel mesh tokenization algorithm that efficiently compresses triangular meshes into 1D token sequences, significantly enhancing training efficiency. Furthermore, our model compresses variable-length triangular meshes into a fixed-length latent space, enabling training latent diffusion models for better generaliza-

[*]This work is done while interning with NVIDIA.

tion. Extensive experiments demonstrate the superior quality, diversity, and generalization capabilities of our model in both point cloud and image-conditioned mesh generation tasks.

1 Introduction

Automatic 3D content generation, particularly the generation of widely used polygonal meshes, holds the potential to revolutionize industries such as digital gaming, virtual reality, and filmmaking. Generative models can make 3D asset creation more accessible to non-experts by drastically reducing the time and effort involved. This democratization opens up opportunities for a wider range of individuals to contribute to and innovate within the 3D content space, fostering greater creativity and efficiency across these sectors.

Previous research on 3D generation has explored a variety of approaches. For example, optimization-based methods, such as using score distillation sampling (SDS) (Poole et al., 2022; Lin et al., 2023; Liu et al., 2023b; Tang et al., 2023a), lift 2D diffusion priors into 3D without requiring any 3D data. In contrast, large reconstruction models (LRM) (Hong et al., 2023; Wang et al., 2023b; Xu et al., 2023b; Li et al., 2023; Weng et al., 2024b) directly train feed-forward models to predict neural radiance fields (NeRF) or Gaussian Splatting from single or multi-view image inputs. Lastly, 3D-native latent diffusion models (Zhang et al., 2024c; Wu et al., 2024b; Li et al., 2024c) encode 3D assets into latent spaces and generate diverse contents by performing diffusion steps in the latent space. However, all these approaches rely on continuous 3D representations, such as NeRF or SDF grids, which lose the discrete face indices in triangular meshes during conversion. Consequently, they require post-processing, such as marching cubes (Lorensen & Cline, 1998) and re-meshing algorithms, to extract triangular meshes. These meshes differ significantly from artist-created ones, which are more concise, symmetric, and aesthetically structured. Additionally, these methods are limited to generating watertight meshes and cannot produce single-layered surfaces.

Recently, several approaches (Siddiqui et al., 2024a; Chen et al., 2024d;b; Weng et al., 2024a; Chen et al., 2024e) have attempted to tokenize meshes into 1D sequences and leverage auto-regressive models for direct mesh generation. Specifically, MeshGPT (Siddiqui et al., 2024a) proposes to empirically sort the triangular faces and apply a vector-quantization variational auto-encoder (VQ-VAE) to tokenzie the mesh. MeshXL (Chen et al., 2024b) directly flattens the vertex coordinates and does not use any compression other than vertex discretization. Since these methods directly learn from mesh vertices and faces, they can preserve the topology information and generate artistic meshes. However, these auto-regressive mesh generation approaches still face several challenges. (1) Generation of a large number of faces: due to the inefficient face tokenization algorithms, most prior methods can only generate meshes with fewer than 1,600 faces, which is insufficient for representing complex objects. (2) Generation of high-resolution surface: previous works quantize mesh vertices to a discrete grid of only 128^3 resolution, which results in significant accuracy loss and unsmooth surfaces. (3) Model generalization: training auto-regressive models with difficult input modalities is challenging. Previous approaches often struggle to generalize beyond the training domain when conditioning on single-view images.

In this paper, we present a novel approach named **EdgeRunner** to address the aforementioned challenges. Firstly, we introduce a mesh tokenization method based on EdgeBreaker (Rossignac, 1999) that compresses sequence length by 50% and reduces long-range dependency between tokens, significantly improving the training efficiency. Secondly, we propose an Auto-regressive Auto-encoder (ArAE) that compresses variable-length triangular meshes into fixed-length latent codes. This latent space can be used to train latent diffusion models conditioned on other modalities, offering better generalization capabilities. We also enhance the training pipeline to support higher quantization resolution. These improvements enable EdgeRunner to generate diverse, high-quality artistic meshes with up to 4,000 faces and vertices discretized at a resolution of 512^3 — resulting in sequences that are twice as long and four times higher in resolution compared to previous methods.

In summary, our contributions are as follows:

1. We introduce a novel mesh tokenization algorithm, adapted from EdgeBreaker, which supports lossless face compression, prevents flipped faces, and reduces long-range dependencies to facilitate learning.

2. We propose an Auto-regressive Auto-encoder (ArAE), comprising a lightweight encoder and an auto-regressive decoder, capable of compressing variable-length triangular meshes into fixed-length latent codes.
3. We demonstrate that the latent space of ArAE can be leveraged to train latent diffusion models for better generalization, enabling conditioning of different input modalities such as single-view images.
4. Extensive experiments show that our method generates high-quality and diverse artistic meshes from point clouds or single-view images, exhibiting improved generalization and robustness compared to previous methods.

2 Related Work

2.1 Optimization-based 3D Generation

Early 3D generation methods relied on SDS-based optimization techniques (Jain et al., 2022; Poole et al., 2022; Wang et al., 2023a; Mohammad Khalid et al., 2022; Michel et al., 2022) due to limited 3D data. Subsequent works advanced in generation quality (Lin et al., 2023; Wang et al., 2023d; Chen et al., 2023c;e; Sun et al., 2023; Qiu et al., 2024), reducing generation time (Tang et al., 2023a; Yi et al., 2023; Lorraine et al., 2023; Xu et al., 2024a), enabling 3D editing (Zhuang et al., 2023; Singer et al., 2023; Raj et al., 2023; Chen et al., 2024c), and conditioning on images (Xu et al., 2023a; Tang et al., 2023b; Melas-Kyriazi et al., 2023; Liu et al., 2023b; Qian et al., 2023; Shi et al., 2023). Other approaches first predict multi-view images, then apply reconstruction algorithms to generate the final 3D models (Long et al., 2023; Li et al., 2024b; Pang et al., 2024; Tang et al., 2024b). Recently, Unique3D (Wu et al., 2024a) introduced a method that combines high-resolution multi-view diffusion models with an efficient mesh reconstruction algorithm, achieving both high quality and fast image-to-3D generation.

2.2 Feed-forward 3D Generation

With the introduction of large-scale datasets (Deitke et al., 2023b;a), more recent works propose to use feed-forward 3D models. The Large Reconstruction Model (LRM)(Hong et al., 2023) demonstrated that end-to-end training of a triplane-NeRF regression model scales effectively to large datasets and generates 3D assets within seconds. While LRM significantly accelerates generation speed, the resulting meshes often exhibit lower quality and a lack of diversity. Subsequent research has sought to improve generation quality by incorporating multi-view images as inputs(Xu et al., 2024b; Li et al., 2023; Wang et al., 2023b; He & Wang, 2023; Siddiqui et al., 2024b; Xie et al., 2024; Wang et al., 2024) and by adopting more efficient 3D representations (Zhang et al., 2024a; Li et al., 2024a; Wei et al., 2024; Zou et al., 2023; Tang et al., 2024a; Xu et al., 2024d; Zhang et al., 2024b; Chen et al., 2024a; Yi et al., 2024).

2.3 Diffusion-based 3D Generation

Analogous to 2D diffusion models for image generation, significant efforts have been made to develop 3D-native diffusion models capable of conditional 3D generation. Early approaches typically rely on uncompressed 3D representations, such as point clouds, NeRFs, tetrahedral grids, and volumes (Nichol et al., 2022; Jun & Nichol, 2023; Gupta et al., 2023; Cheng et al., 2023; Ntavelis et al., 2023; Zheng et al., 2023; Zhang et al., 2023; Liu et al., 2023c; Müller et al., 2023; Chen et al., 2023d; Cao et al., 2023; Chen et al., 2023a; Wang et al., 2023c; Yariv et al., 2023; Liu et al., 2023a; Xu et al., 2024c; Yan et al., 2024) to train diffusion models. However, these methods are often limited by small-scale datasets and struggle with generalization or producing high-quality assets. More recent approaches have focused on adapting latent diffusion models to 3D (Zhao et al., 2023; Zhang et al., 2024c; Wu et al., 2024b; Li et al., 2024c; Lan et al., 2024; Hong et al., 2024; Tang et al., 2023c; Chen et al., 2024f). These methods first train a VAE to compress 3D representations into a more compact form, which enables more efficient diffusion model training. Unlike the straightforward image representations in 2D, 3D latent diffusion models involve numerous design choices, leading to varied performance outcomes. For example, CLAY (Zhang et al., 2024c) has demonstrated that a transformer-based 3D latent diffusion model can scale to large datasets and generalize well across diverse input conditions.

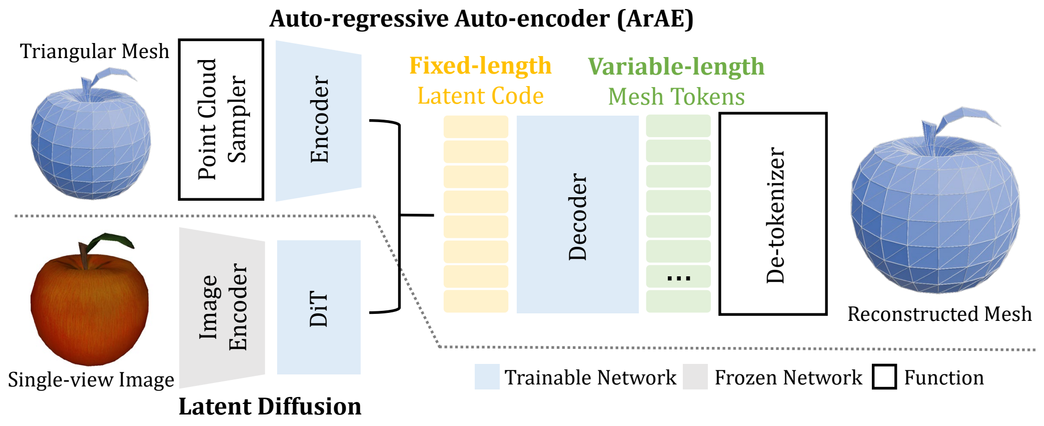


Figure 2: **Pipeline of our method**. Our ArAE model compresses variable-length mesh into fixed-length latent code, which can be further used to train latent diffusion models conditioned on other input modalities, such as single-view images.

2.4 Auto-regressive Mesh Generation

The above works require additional post-processing steps to extract triangular meshes and fail to model the mesh topology. Recenlty, approaches using auto-regressive models to directly generate meshes have emerged. MeshGPT (Siddiqui et al., 2024a) pioneered this approach by tokenizing a mesh through face sorting and compression with a VQ-VAE, followed by using an auto-regressive transformer to predict the token sequence. This method allows for the generation of meshes with varying face counts and incorporates direct supervision from topology information, which is often overlooked in other approaches. Subsequent works (Chen et al., 2024b; Weng et al., 2024a; Chen et al., 2024d) have explored different model architectures and extended this approach to conditional generation tasks, such as point cloud generation. However, these methods are limited to meshes with fewer than 1,000 faces due to the computational cost of mesh tokenization and exhibit limited generalization capabilities. A concurrent work, MeshAnythingV2 (Chen et al., 2024e), introduces an improved mesh tokenization technique, increasing the maximum number of faces to 1,600. Our approach also falls under the category of auto-regressive mesh generation but aims to further extend the maximum face count and provide control over the target face number during inference.

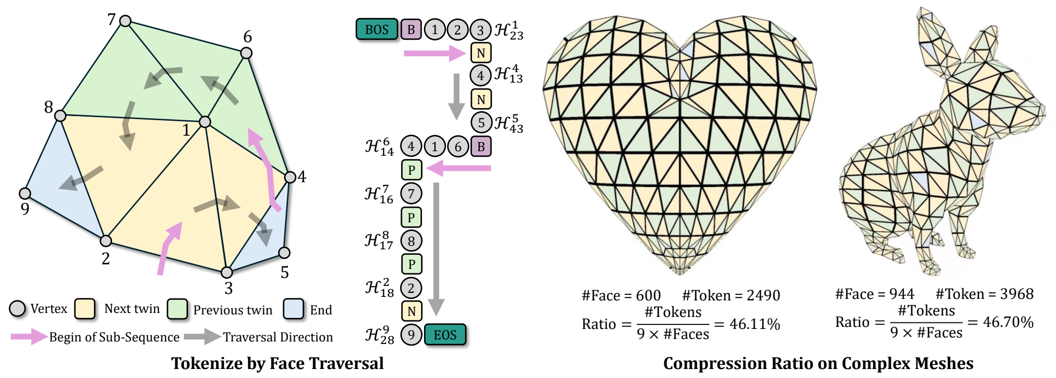
3 EdgeRunner

3.1 Compact Mesh Tokenization

Auto-regressive models process information in the form of discrete token sequences. Thus, compact tokenization is crucial as it allows information to be represented with fewer tokens accurately. For example, text tokenizers have been a central research direction for large language models (LLMs). The GPT and Llama series (Touvron et al., 2023; Brown et al., 2020) utilize the byte-pair encoding (BPE) tokenizer, which combines sub-word units into single tokens for highly compact and lossless compression.

In contrast, tokenization techniques used in prior auto-regressive mesh generation works mainly suffer from two issues. Some prior works use *lossy* VQ-VAEs (Siddiqui et al., 2024a; Chen et al., 2024d; Weng et al., 2024a), which sacrifices the mesh generation quality. Others opt for *zero-compression* by not using face tokenizers (Chen et al., 2024b), which poses training challenges due to the inefficiency.

In this paper, we introduce a tokenization scheme that allows us to represent a mesh compactly and efficiently, which is based on the well-established triangular mesh compression algorithm EdgeBreaker (Rossignac, 1999). **The key insight for mesh compression is to maximize edge sharing between adjacent triangles**. By sharing an edge with the previous triangle, the next triangle re-



7
6
8
1
9
4
2
5
3
Vertex
Next twin
Previous twin
End
Begin of Sub-Sequence
Traversal Direction
Tokenize by Face Traversal
BOS B 1 2 3 $\mathcal{H}^1_{23}$
N
4 $\mathcal{H}^4_{13}$
N
5 $\mathcal{H}^5_{43}$
$\mathcal{H}^6_{14}$ 4 1 6 B
P
$\mathcal{H}^7_{16}$ 7
P
$\mathcal{H}^8_{17}$ 8
P
$\mathcal{H}^2_{18}$ 2
N
$\mathcal{H}^9_{28}$ 9 EOS
#Face = 600 #Token = 2490
Ratio = #Tokens / 9 × #Faces = 46.11%
#Face = 944 #Token = 3968
Ratio = #Tokens / 9 × #Faces = 46.70%
Compression Ratio on Complex Meshes

Figure 3: **Illustration of our mesh tokenizer**. Our tokenizer traverses the 3D mesh triangle-by-triangle and converts it into a 1D token sequence. Through edge sharing, we reach a compression rate of 50% (4 or 5 tokens per face on average) compared to naïve tokenization of 9 tokens per face.

quires only one additional vertex instead of three. We illustrate our mesh tokenization process with an example below, and provide more details in the appendix.

Half-edge. EdgeBreaker (Rossignac, 1999) uses half-edge data structures (Weiler, 1986) for triangular face traversal. An illustration is provided in Figure 4. We use $\mathcal{H}^{\cdot}_{\cdot\cdot}$ to denote a half-edge. For example, $\mathcal{H}^3_{41}$ is the half-edge pointing from vertex 4 to 1, with vertex 3 across the face. Starting from $\mathcal{H}^3_{41}$, we can traverse to the *next* half-edge $\mathcal{H}^4_{13}$ and the *next twin* half-edge $\mathcal{H}^2_{31}$. Reversely, the *previous* half-edge is $\mathcal{H}^1_{34}$ and the *previous twin* half-edge is $\mathcal{H}^5_{43}$. The half-edge data structure has also been used in recent learning-based mesh generation work (Shen et al., 2024).

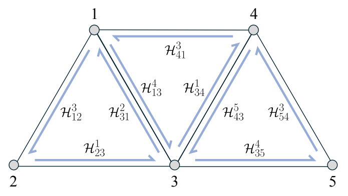


Figure 4: Half-edge representation for triangular faces.

Vertex Tokenization. To tokenize a mesh into a discrete sequence, vertex coordinates require discretization. Following previous works (Siddiqui et al., 2024a), we normalize the mesh to a unit cube and quantize the continuous vertex coordinates into integers according to a quantization resolution, which is 512 in this work. Each vertex is therefore represented by three integer coordinates, which are then flattened in XYZ order as tokens. With some abuse of notion, we denote the XYZ tokens as a single vertex token using ●.

Face Tokenization. We traverse through all faces following the half-edges. To illustrate the process, we use the mesh example in Figure 3. The process starts with one half-edge, where $\mathcal{H}^1_{23}$ is picked as the beginning of the current traversal. We signify the start of a traversal as B. We then append the vertex across the half-edge 1 as the first vertex token. Within the same triangular face, the two vertices 2 3 are also appended following the direction of $\mathcal{H}^1_{23}$.

During traversal, we visit the *next twin* half-edge whenever possible, and only reverse the half-edge direction to the *previous twin* half-edge when we exhaust all triangles in the current traversal. Returning to the example in Figure 3, we follow $\mathcal{H}^1_{23}$ and reach $\mathcal{H}^4_{13}$. Thus, we append N to signify the *next twin* traversal direction and we only need to append 4 as 1 3 are shared. The same process is repeated for $\mathcal{H}^5_{43}$ with N 5 added to the current sub-sequence.

We have completed the current traversal as no adjacent faces can be found for $\mathcal{H}^5_{43}$. The sub-sequence for the current traversal is thus B 1 2 3 N 4 N 5.

To begin a new sub-sequence, we reversely retrieve the last-added half-edges to traverse in the opposite directions. As the last-added half-edge $\mathcal{H}^5_{43}$ doesn't have any adjacent faces, we skip it and instead consider $\mathcal{H}^4_{13}$. We go opposite to its *previous twin* half-edge $\mathcal{H}^6_{14}$. As this is a new sub-sequence, B 6 1 4 are added.

We continue finding the un-visited faces in the neighborhood of $\mathcal{H}^6_{14}$ and arriving at its *previous twin* half-edge $\mathcal{H}^7_{16}$. Thus, we add P 7 to the current sub-mesh sequence as 6 1 are shared. The process is repeated and P 8 P 2 N 9 are added. As all triangular faces have been visited, the face tokenization process of the mesh is complete.

Auxiliary Tokens. Similar to prior work in LLMs, we prepend a BOS at the beginning of a mesh sequence and append a EOS at the end of a mesh sequence.

Detokenization. It is straightforward to reconstruct the original mesh from a mesh token sequence. We iterate over the tokens while maintaining a state machine. Each B of a sub-sequence is always followed by three vertex tokens. Each N or P is followed by a single vertex token, and we retrieve two previous vertex tokens based on the traversal direction to reconstruct the triangle. Finally, we merge duplicate vertices, as they may appear multiple times from different sub-sequences, and output the reconstructed mesh.

Advantages. Our tokenizer benefits model training in several ways: (1) Each face requires an average of 4 to 5 tokens, achieving approximately 50% compression compared to the 9 tokens used in previous works (Chen et al., 2024b;d). This increased efficiency enables the model to generate more faces with the same number of tokens and facilitates training on datasets containing a higher number of faces. (2) Our traversal is designed to avoid long-range dependency between tokens. Each token only relies on a short context of previous tokens, which further mitigate the difficulty of learning. (3) The traversal ensures that each face's orientation remains consistent within each sub-mesh. Consequently, the generated mesh can be accurately rendered using back face culling, a feature not consistently achieved in prior methods. We will further illustrate these benefits in Section 4.

3.2 AUTO-REGRESSIVE AUTO-ENCODER

Although our decoder is auto-regressive and generates variable-length token sequences, we observe limitations in both generation diversity and its ability to follow conditioning. In contrast, diffusion models, which have been extensively studied (Rombach et al., 2022; Zhang et al., 2024c), demonstrate promising results in addressing these challenges. To apply diffusion models, a key challenge is the requirement of fixed-length data, as mesh generation has a variable-length data structure and the number of triangle faces can vary significantly due to different complexities. Therefore, **we propose an Auto-regressive Auto-encoder (ArAE) to encode the variable-length mesh into a fixed-length latent space**, similar to the role of variational auto-encoders (VAEs) in latent diffusion models. To train our ArAE model, we choose point clouds as the encoder input for geometric information and our tokenized mesh sequences as output. Thus the ArAE model itself also works as a point cloud conditioned mesh generator.

Architecture. The architecture of our ArAE model is illustrated in Figure 2. ArAE consists of a lightweight encoder and an auto-regressive decoder. To extract geometric information from the mesh surface, we follow prior works to sample a point cloud and apply a transformer encoder (Chen et al., 2024d;e; Zhang et al., 2023; 2024c). Specifically, we sample N random points, $\mathbf{X} \in \mathbb{R}^{N\times 3}$, from the surface of the input mesh and use a cross-attention layer to extract the latent code:

$$\mathbf{Z} = \text{CrossAtt}\big(\mathbf{Q}, \text{PosEmbed}(\mathbf{X})\big) \tag{1}$$

where $\mathbf{Q} \in \mathbb{R}^{M\times C}$ represents the trainable query embedding with a hidden dimension of C, $\text{PosEmbed}(\cdot)$ is a frequency embedding function for 3D points (Zhang et al., 2023), and $\mathbf{Z} \in \mathbb{R}^{M\times L}$ represents the latent code, $M < N$ and $L < C$ denote the latent size and dimension, respectively. The decoder is an auto-regressive transformer, designed to generate a variable-length mesh token sequence. For simplicity, we adopt the OPT architecture (Zhang et al., 2022), as used in prior works (Siddiqui et al., 2024a; Chen et al., 2024d;b). A learnable embedding converts discrete tokens into continuous features, and a linear head maps predicted features back to classification logits. Stacked causal self-attention layers are employed to predict the next token based on previous tokens. The latent code $\mathbf{Z}$ is prepended to the input before the BOS token, allowing the decoder to learn how to generate a mesh token sequence conditioned on the latent code.

Face Count Condition. Another issue is that given the same input point cloud, multiple plausible meshes with varying numbers of faces and topologies can be generated. The number of faces is particularly crucial as it directly affects the mesh's complexity (low-poly *vs.* high-poly) and the generation speed. To manage meshes with a broad range of face counts, we aim to provide some level of explicit control over the targeted number of faces. This control facilitates the estimation of generation time and the complexity of the generated mesh during inference. We propose a simple face count conditioning method for coarse-grained control. Specifically, we append a learnable face count token after the latent code condition tokens. We bucket face count into different ranges and assign different tokens to each range. For instance, we use four distinct tokens to represent face counts in the following ranges: less than or equal to 1000, between 1000 and 2000, between 2000 and 4000, and greater than 4000. Additionally, during training, we randomly replace these tokens with a fifth unconditional token. This approach ensures that the model still learns to generate meshes without specifying a targeted face count.

Loss Function. The ArAE is trained using the standard cross-entropy loss on the predicted next tokens:

$$\mathcal{L}_{\text{ce}} = \text{CrossEntropy}(\hat{\mathbf{S}}[:-1], \mathbf{S}[1:]) \tag{2}$$

where $\mathbf{S}$ denotes the one-hot ground truth token sequence, and $\hat{\mathbf{S}}$ represents the predicted classification logits sequence. Additionally, to constrain the range of the latent space for easier training of subsequent diffusion models, we apply an L2 norm penalty to the latent code:

$$\mathcal{L}_{\text{reg}} = ||\mathbf{Z}||_2^2 \tag{3}$$

The final loss is a weighted combination of the cross-entropy loss and the regularization term.

3.3 Image-conditioned Latent Diffusion

With the fixed-length latent space provided by our ArAE architecture, it is now feasible to train mesh generation models conditioned on different inputs, akin to how 2D image generation models are trained. Among various input modalities, we showcase the capabilities of our model using single-view images, which are among the most commonly used conditions for mesh generation.

We follow the approach of previous methods (Chen et al., 2023b) and utilize a diffusion transformer (DiT) as the backbone. Specifically, we employ the image encoder from CLIP (Ilharco et al., 2021; Mohammad Khalid et al., 2022) to extract image features for conditioning. Cross-attention layers are used to integrate the image condition into the denoising features, while AdaLN layers incorporate timestep information. We use the DDPM framework (Ho et al., 2020) and mean square error (MSE) loss to train the DiT model. At each training step, we randomly sample a timestep t and a Gaussian noise $\epsilon \in \mathbb{R}^{M \times L}$. The loss is calculated between the predicted noise and the random noise.

4 Experiments

4.1 Qualitative Results

Point Cloud Conditioned Generation. We first compare the generated meshes conditioned on point clouds in Figure 5. For the test samples, we use other image-to-3D methods (Zhang et al., 2024c; Li et al., 2024c) to generate dense meshes, ensuring that these samples have never been seen in the training dataset. Notably, MeshAnything (Chen et al., 2024d;e) uses both point coordinates and normal vectors as input to a pretrained encoder (Zhao et al., 2023). In contrast, our encoder takes only point coordinates and is trained from scratch, while still achieving better performance. Our model demonstrates greater stability on challenging test cases, producing more aesthetically pleasing meshes. Additionally, the proposed tokenization algorithm enables the generation of longer sequences, facilitating the modeling of complex shapes and improving the closure of loops for watertight surfaces.

Image Conditioned Generation. In Figure 6, we compare the meshes generated from single-view images. Since other auto-regressive methods cannot directly condition on images, we compare our results with a recent optimization-based approach (Wu et al., 2024a). Our end-to-end image-to-mesh generation produces artistic meshes that better capture the semantics of the input images. Additionally, our model exhibits strong generalization on challenging 2D-style or realistic-lighting images, despite being trained exclusively on images rendered from 3D meshes with simple shading.

Face Count Control. In practice, users may have specific requirements for the face count or level of detail in the generated mesh. Our design enables coarse-grained control over the face count by allowing users to specify the face count condition. As demonstrated in Figure 7, our model is capable of generating meshes with different ranges of face count from the same input point cloud.

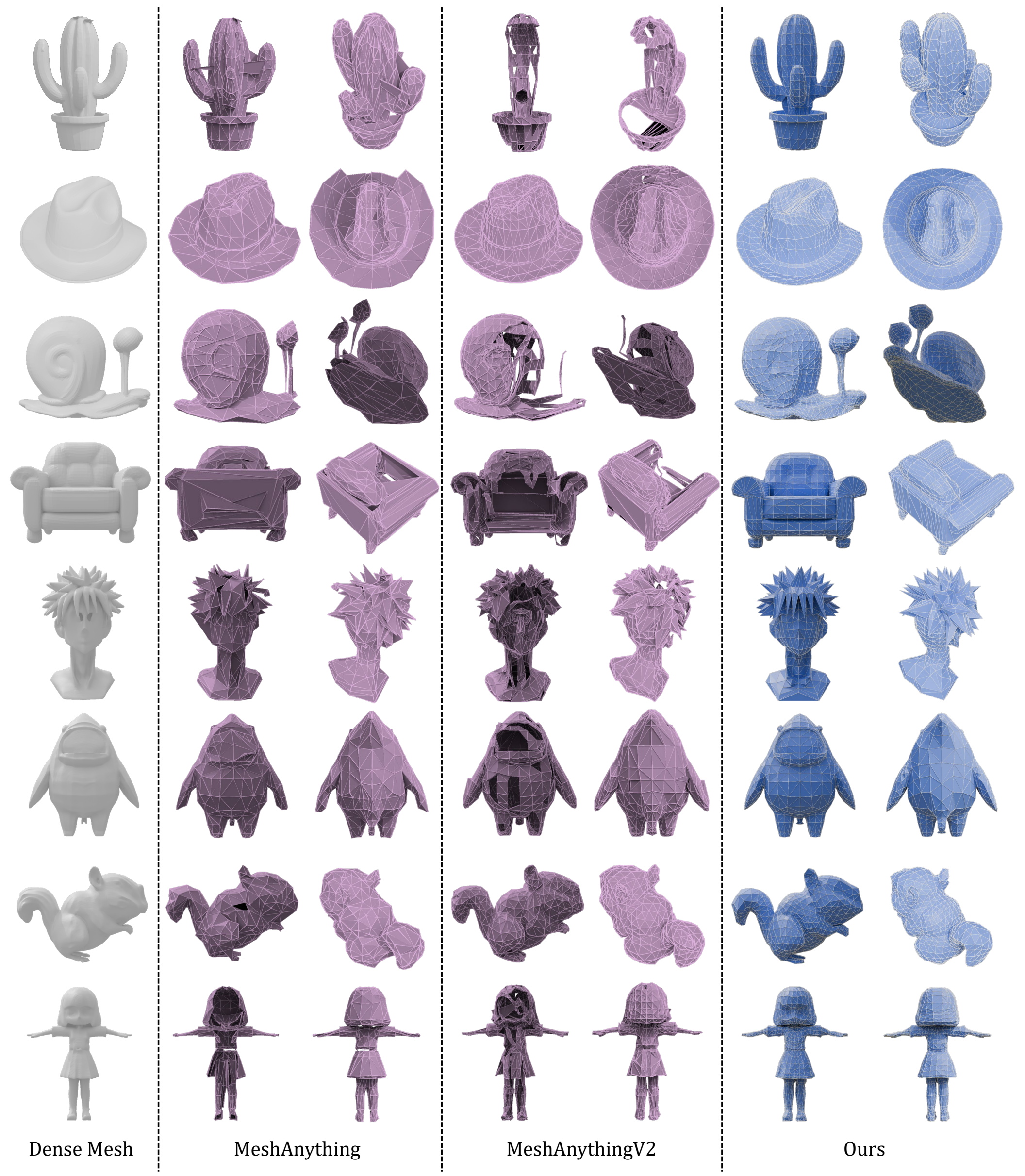


Figure 5: **Comparison on point cloud conditioned generation**. We show the reference dense mesh and generated meshes conditioned on randomly sampled point cloud.

4.2 Quantitative Results

User Study. Since evaluation of the mesh aesthetic quality is challenging, we majorly rely on a user study for quantitative comparisons. For a collection of 8 test cases, we render the mesh geometry and wireframe generated from MeshAnything (Chen et al., 2024d), MeshAnythingV2 (Chen et al., 2024e), and our method. Each volunteer is shown 8 samples from mixed random methods, and asked to rate in three aspects: input consistency with the point cloud, triangle aesthetic, and overall mesh quality. We collect results from 20 volunteers with 480 valid scores. As shown in Table 1, our method is preferred across all evaluated aspects.

Inference Speed. Since we are using next-token prediction, the inference time for each generation is dependent on the length of the sequence. Our model is basically a large transformer, and the major compute is used for attention calculation. On a A100 GPU with flash-attention (Dao, 2024) and

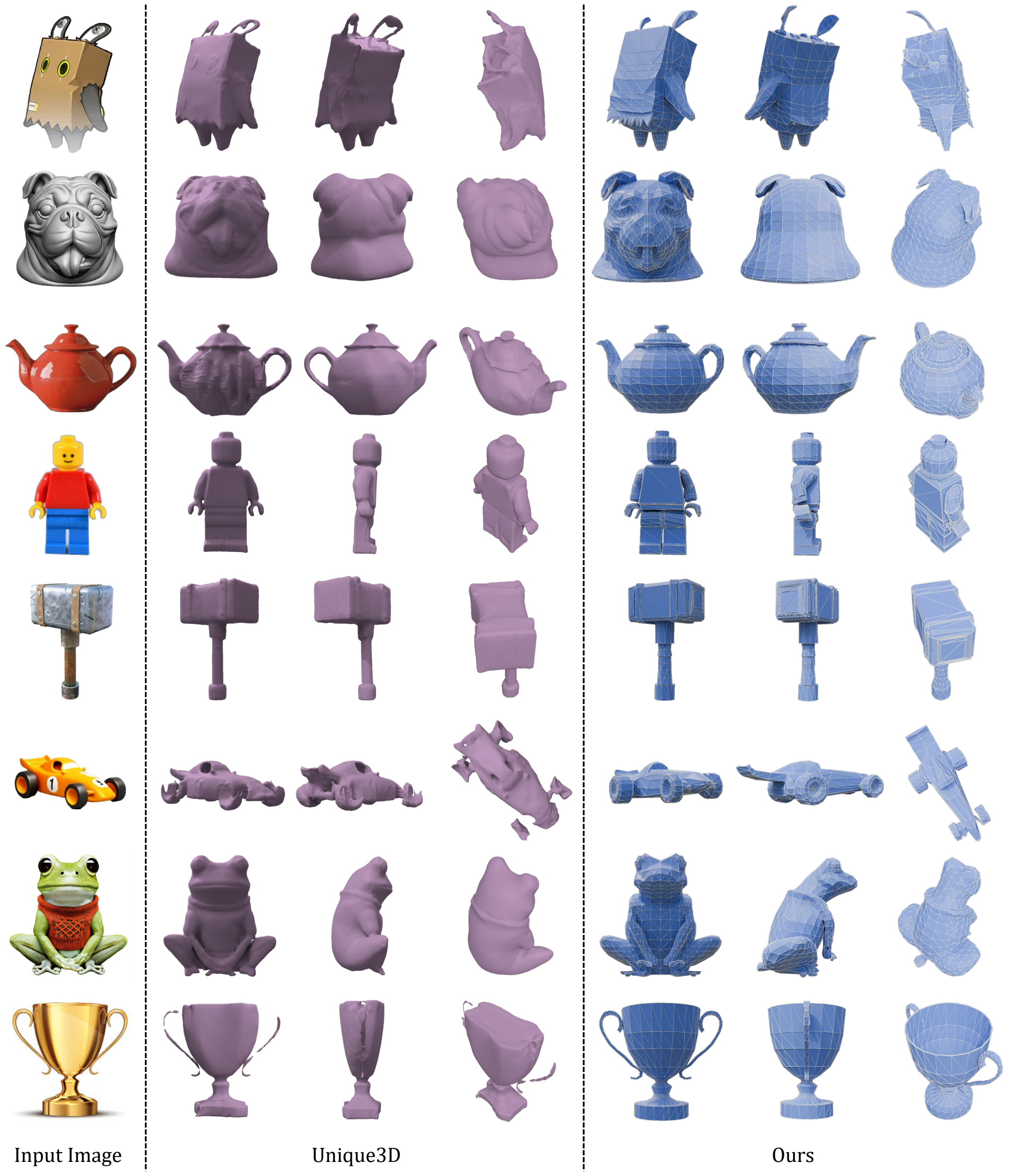


Figure 6: **Comparison on image conditioned generation**. We show the input image and generated meshes. Since Unique3D outputs dense meshes, we only visualize the surface without wireframe.

	Input Consistency	Triangle Aesthetic	Overall Quality
MeshAnything	2.93	2.43	2.43
MeshAnythingV2	2.64	2.36	2.14
Ours	**4.83**	**4.54**	**4.58**

Table 1: **User Study** on point-conditioned mesh generation. The rating is of scale 1-5, the higher the better.

KV cache enabled, our model runs at about 100 tokens per second. Specifically, it takes about 45 seconds to generate a mesh with 1,000 faces, about 90 seconds for 2,000 faces, and about 3 minutes for 4,000 faces.

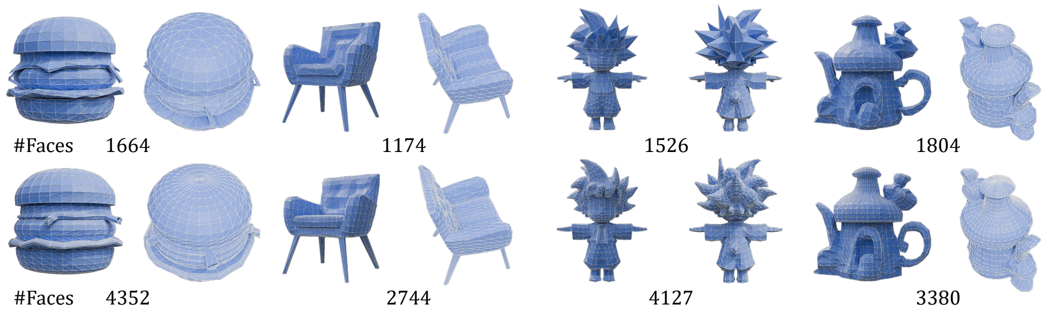


Figure 7: **Coarse-grained face count control**. Our model supports controlling the targeted number of face, which allows to generate meshes with different levels of detail.

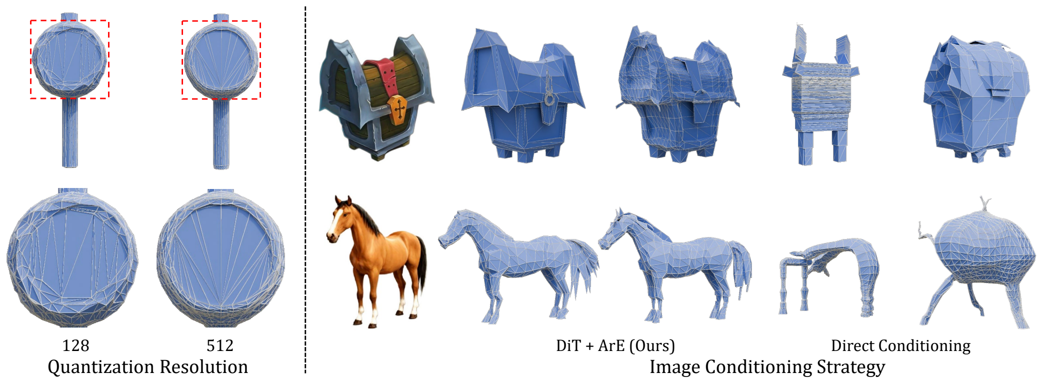


Figure 8: **Ablation Study** on different quantization resolutions and image conditioning strategies.

4.3 Ablation Studies

Quantization Resolution. We initially experimented with a quantization resolution of 128^3 as used in previous works. While this lower resolution made it easier to train the classification head, we observed noticeable quality degradation, as shown in the left part of Figure 8. To address this problem, we increased the quantization resolution to 512^3 for our final model, which resulted in more accurate vertex positions and smoother surfaces.

Direct Image Conditioning. We also attempted to directly train an image-conditioned auto-regressive mesh decoder. The image features from the CLIP vision encoder were prepended as condition tokens, and the model directly predicted the mesh token sequence. However, we found that this approach was more difficult to converge and generally exhibited poorer generalization on unseen test cases, as shown in the right part of Figure 8. Consequently, we designed the ArAE model to learn a fixed-length latent space using a simpler point-cloud condition, and we trained latent diffusion models to map complex condition modalities to this latent space. This approach led to better generalization and more efficient training.

5 Conclusion

In this work, we propose a novel Auto-regressive Auto-encoder that compresses variable-length triangular meshes into fixed-length latent codes, and we further demonstrate the potential of this latent space for image-to-mesh generation. Additionally, we introduce an efficient mesh tokenization algorithm that enables scaling up the model, resulting in longer context lengths and higher resolution. Our approach highlights the potential for improving the scalability of auto-regressive models for practical 3D mesh generation, offering a promising direction for future research and applications.

Acknowledgments

This work is supported by the Sichuan Science and Technology Program (2023YFSY0008), National Natural Science Foundation of China (61632003, 61375022, 61403005), Grant SCITLAB-20017 of Intelligent Terminal Key Laboratory of SiChuan Province, Beijing Advanced Innovation Center for Intelligent Robots and Systems (2018IRS11), and PEK-SenseTime Joint Laboratory of Machine Vision. We thank Yongxin Chen for introducing the EdgeBreaker paper. We thank Nicholas Sharp and Frank Shen for sharing the mesh visualization code.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 4

Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Large-vocabulary 3d diffusion model with transformer. *arXiv preprint arXiv:2309.07920*, 2023. 3

Anpei Chen, Haofei Xu, Stefano Esposito, Siyu Tang, and Andreas Geiger. Lara: Efficient large-baseline radiance fields. *arXiv preprint arXiv:2407.04699*, 2024a. 3

Hansheng Chen, Jiatao Gu, Anpei Chen, Wei Tian, Zhuowen Tu, Lingjie Liu, and Hao Su. Single-stage diffusion nerf: A unified approach to 3d generation and reconstruction. *arXiv preprint arXiv:2304.06714*, 2023a. 3

Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023b. 7

Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023c. 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*, 2024b. 2, 4, 6

Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21476–21485, 2024c. 3

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*, 2024d. 2, 4, 6, 7, 8, 21

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, 2024e. URL https://arxiv.org/abs/2408.02555. 2, 4, 6, 7, 8, 21

Zhaoxi Chen, Fangzhou Hong, Haiyi Mei, Guangcong Wang, Lei Yang, and Ziwei Liu. Primdiffusion: Volumetric primitives diffusion for 3d human generation. *arXiv preprint arXiv:2312.04559*, 2023d. 3

Zhaoxi Chen, Jiaxiang Tang, Yuhao Dong, Ziang Cao, Fangzhou Hong, Yushi Lan, Tengfei Wang, Haozhe Xie, Tong Wu, Shunsuke Saito, et al. 3dtopia-xl: Scaling high-quality 3d asset generation via primitive diffusion. *arXiv preprint arXiv:2409.12957*, 2024f. 3

Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3d using gaussian splatting. *arXiv preprint arXiv:2309.16585*, pp. 21401–21412, 2023e. 3

Yen-Chi Cheng, Hsin-Ying Lee, Sergey Tulyakov, Alexander G Schwing, and Liang-Yan Gui. Sdfusion: Multimodal 3d shape completion, reconstruction, and generation. In *CVPR*, pp. 4456–4465, 2023. 3

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. 8

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. *arXiv preprint arXiv:2307.05663*, 2023a. 3, 18

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 *CVPR*, pp. 13142–13153, 2023b. 3, 18

Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 209–216, 1997. 19

Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3dgen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023. 3

Tiankai Hang, Shuyang Gu, Chen Li, Jianmin Bao, Dong Chen, Han Hu, Xin Geng, and Baining Guo. Efficient diffusion training via min-snr weighting strategy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7441–7451, 2023. 20

Zexin He and Tengfei Wang. Openlrm: Open-source large reconstruction models. `https://github.com/3DTopia/OpenLRM`, 2023. 3

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33: 6840–6851, 2020. 7

Fangzhou Hong, Jiaxiang Tang, Ziang Cao, Min Shi, Tong Wu, Zhaoxi Chen, Tengfei Wang, Liang Pan, Dahua Lin, and Ziwei Liu. 3dtopia: Large text-to-3d generation model with hybrid diffusion priors. *arXiv preprint arXiv:2403.02234*, 2024. 3

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. 2, 3

Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL `https://doi.org/10.5281/zenodo.5143773`. If you use this software, please cite it as below. 7, 19

Ajay Jain, Ben Mildenhall, Jonathan T Barron, Pieter Abbeel, and Ben Poole. Zero-shot text-guided object generation with dream fields. In *CVPR*, pp. 867–876, 2022. 3

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

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. *arXiv preprint arXiv:2403.12019*, 2024. 3

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, 3

Mengfei Li, Xiaoxiao Long, Yixun Liang, Weiyu Li, Yuan Liu, Peng Li, Xiaowei Chi, Xingqun Qi, Wei Xue, Wenhan Luo, et al. M-lrm: Multi-view large reconstruction model. *arXiv preprint arXiv:2406.07648*, 2024a. 3

Peng Li, Yuan Liu, Xiaoxiao Long, Feihu Zhang, Cheng Lin, Mengfei Li, Xingqun Qi, Shanghang Zhang, Wenhan Luo, Ping Tan, et al. Era3d: High-resolution multiview diffusion using efficient row-wise attention. *arXiv preprint arXiv:2405.11616*, 2024b. 3

Weiyu Li, Jiarui Liu, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. Craftsman: High-fidelity mesh generation with 3d native generation and interactive geometry refiner. *arXiv preprint arXiv:2405.14979*, 2024c. 2, 3, 7

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*, pp. 300–309, 2023. 2, 3

Minghua Liu, Chao Xu, Haian Jin, Linghao Chen, Zexiang Xu, Hao Su, et al. One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization. *arXiv preprint arXiv:2306.16928*, 2023a. 3

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. *arXiv preprint arXiv:2303.11328*, 2023b. 2, 3

Zhen Liu, Yao Feng, Michael J Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. Meshdiffusion: Score-based generative 3d mesh modeling. *arXiv preprint arXiv:2303.08133*, 2023c. 3

Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, et al. Wonder3d: Single image to 3d using cross-domain diffusion. *arXiv preprint arXiv:2310.15008*, 2023. 3

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*, pp. 347–353, 1998. 2

Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. Att3d: Amortized text-to-3d object synthesis. *arXiv preprint arXiv:2306.07349*, 2023. 3

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 18

Luke Melas-Kyriazi, Iro Laina, Christian Rupprecht, and Andrea Vedaldi. Realfusion: 360deg reconstruction of any object from a single image. In *CVPR*, pp. 8446–8455, 2023. 3

Oscar Michel, Roi Bar-On, Richard Liu, Sagie Benaim, and Rana Hanocka. Text2mesh: Text-driven neural stylization for meshes. In *CVPR*, pp. 13492–13502, 2022. 3

Nasir Mohammad Khalid, Tianhao Xie, Eugene Belilovsky, and Tiberiu Popa. Clip-mesh: Generating textured meshes from text using pretrained image-text models. In *SIGGRAPH Asia*, pp. 1–8, 2022. 3, 7

Norman Müller, Yawar Siddiqui, Lorenzo Porzi, Samuel Rota Bulo, Peter Kontschieder, and Matthias Nießner. Diffrf: Rendering-guided 3d radiance field diffusion. In *CVPR*, pp. 4328–4338, 2023. 3

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. 3

Evangelos Ntavelis, Aliaksandr Siarohin, Kyle Olszewski, Chaoyang Wang, Luc Van Gool, and Sergey Tulyakov. Autodecoding latent 3d diffusion models. *arXiv preprint arXiv:2307.05445*, 2023. 3

Yatian Pang, Tanghui Jia, Yujun Shi, Zhenyu Tang, Junwu Zhang, Xinhua Cheng, Xing Zhou, Francis EH Tay, and Li Yuan. Envision3d: One image to 3d with anchor views interpolation. *arXiv preprint arXiv:2403.08902*, 2024. 3

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988*, 2022. 2, 3

Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023. 3

Lingteng Qiu, Guanying Chen, Xiaodong Gu, Qi Zuo, Mutian Xu, Yushuang Wu, Weihao Yuan, Zilong Dong, Liefeng Bo, and Xiaoguang Han. Richdreamer: A generalizable normal-depth diffusion model for detail richness in text-to-3d. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9914–9925, 2024. 3, 19

Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Nataniel Ruiz, Ben Mildenhall, Shiran Zada, Kfir Aberman, Michael Rubinstein, Jonathan Barron, et al. Dreambooth3d: Subject-driven text-to-3d generation. *arXiv preprint arXiv:2303.13508*, 2023. 3

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pp. 10684–10695, 2022. 6

Jarek Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE transactions on visualization and computer graphics*, 5(1):47–61, 1999. 2, 4, 5, 18

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)*, pp. 11, New York, NY, USA, 2024. ACM. doi: 10.1145/3680528.3687634. URL https://doi.org/10.1145/3680528.3687634. 5

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. 3

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19615–19625, 2024a. 2, 4, 5, 6

Yawar Siddiqui, Tom Monnier, Filippos Kokkinos, Mahendra Kariya, Yanir Kleiman, Emilien Garreau, Oran Gafni, Natalia Neverova, Andrea Vedaldi, Roman Shapovalov, et al. Meta 3d assetgen: Text-to-mesh generation with high-quality geometry, texture, and pbr materials. *arXiv preprint arXiv:2407.02445*, 2024b. 3

Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, et al. Text-to-4d dynamic scene generation. *arXiv preprint arXiv:2301.11280*, 2023. 3

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020. 20

Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. *arXiv preprint arXiv:2310.16818*, 2023. 3

Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023a. 2, 3

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*, 2024a. 3, 18

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. *arXiv preprint arXiv:2303.14184*, 2023b. 3

Shitao Tang, Jiacheng Chen, Dilin Wang, Chengzhou Tang, Fuyang Zhang, Yuchen Fan, Vikas Chandra, Yasutaka Furukawa, and Rakesh Ranjan. Mvdiffusion++: A dense high-resolution multi-view diffusion model for single or sparse-view 3d object reconstruction. *arXiv preprint arXiv:2402.12712*, 2024b. 3

Zhicong Tang, Shuyang Gu, Chunyu Wang, Ting Zhang, Jianmin Bao, Dong Chen, and Baining Guo. Volumediffusion: Flexible text-to-3d generation with efficient volumetric encoder. *arXiv preprint arXiv:2312.11459*, 2023c. 3

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 4

Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation. In *CVPR*, pp. 12619–12629, 2023a. 3

Peng Wang, Hao Tan, Sai Bi, Yinghao Xu, Fujun Luan, Kalyan Sunkavalli, Wenping Wang, Zexiang Xu, and Kai Zhang. Pf-lrm: Pose-free large reconstruction model for joint pose and shape prediction. *arXiv preprint arXiv:2311.12024*, 2023b. 2, 3

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 *CVPR*, pp. 4563–4573, 2023c. 3

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. *arXiv preprint arXiv:2305.16213*, 2023d. 3

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. 3

Xinyue Wei, Kai Zhang, Sai Bi, Hao Tan, Fujun Luan, Valentin Deschaintre, Kalyan Sunkavalli, Hao Su, and Zexiang Xu. Meshlrm: Large reconstruction model for high-quality mesh. *arXiv preprint arXiv:2404.12385*, 2024. 3

Kevin J Weiler. *Topological structures for geometric modeling (Boundary representation, manifold, radial edge structure)*. Rensselaer Polytechnic Institute, 1986. 5

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*, 2024a. 2, 4

Zhenzhen Weng, Jingyuan Liu, Hao Tan, Zhan Xu, Yang Zhou, Serena Yeung-Levy, and Jimei Yang. Single-view 3d human digitalization with large reconstruction models. *arXiv preprint arXiv:2401.12175*, 2024b. 2

Kailu Wu, Fangfu Liu, Zhihan Cai, Runjie Yan, Hanyang Wang, Yating Hu, Yueqi Duan, and Kaisheng Ma. Unique3d: High-quality and efficient 3d mesh generation from a single image, 2024a. 3, 7

Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Jingxi Xu, Philip Torr, Xun Cao, and Yao Yao. Direct3d: Scalable image-to-3d generation via 3d latent diffusion transformer. *arXiv preprint arXiv:2405.14832*, 2024b. 2, 3

Desai Xie, Sai Bi, Zhixin Shu, Kai Zhang, Zexiang Xu, Yi Zhou, Sören Pirk, Arie Kaufman, Xin Sun, and Hao Tan. Lrm-zero: Training large reconstruction models with synthesized data. *arXiv preprint arXiv:2406.09371*, 2024. 3

Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Yi Wang, and Zhangyang Wang. Neurallift-360: Lifting an in-the-wild 2d photo to a 3d object with 360deg views. In *CVPR*, pp. 4479–4489, 2023a. 3

Dejia Xu, Ye Yuan, Morteza Mardani, Sifei Liu, Jiaming Song, Zhangyang Wang, and Arash Vahdat. Agg: Amortized generative 3d gaussians for single image to 3d. *arXiv preprint arXiv:2401.04099*, 2024a. 3

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*, 2024b. 3

Xiang Xu, Joseph Lambourne, Pradeep Jayaraman, Zhengqing Wang, Karl Willis, and Yasutaka Furukawa. Brepgen: A b-rep generative diffusion model with structured latent geometry. *ACM Transactions on Graphics (TOG)*, 43(4):1–14, 2024c. 3

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*, 2023b. 2

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*, 2024d. 3

Xingguang Yan, Han-Hung Lee, Ziyu Wan, and Angel X Chang. An object is worth 64x64 pixels: Generating 3d object via image diffusion. *arXiv preprint arXiv:2408.03178*, 2024. 3

Lior Yariv, Omri Puny, Natalia Neverova, Oran Gafni, and Yaron Lipman. Mosaic-sdf for 3d generative models. *arXiv preprint arXiv:2312.09222*, 2023. 3

Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023. 3

Xuanyu Yi, Zike Wu, Qiuhong Shen, Qingshan Xu, Pan Zhou, Joo-Hwee Lim, Shuicheng Yan, Xinchao Wang, and Hanwang Zhang. Mvgamba: Unify 3d content generation as state space sequence modeling. *arXiv preprint arXiv:2406.06367*, 2024. 3

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *arXiv preprint arXiv:2301.11445*, 2023. 3, 6, 18

Chubin Zhang, Hongliang Song, Yi Wei, Yu Chen, Jiwen Lu, and Yansong Tang. Geolrm: Geometry-aware large reconstruction model for high-quality 3d gaussian generation. *arXiv preprint arXiv:2406.15333*, 2024a. 3

Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. *arXiv preprint arXiv:2404.19702*, 2024b. 3

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. Clay: A controllable large-scale generative model for creating high-quality 3d assets. *arXiv preprint arXiv:2406.13897*, 2024c. 2, 3, 6, 7

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 6

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. *arXiv preprint arXiv:2306.17115*, 2023. 3, 7

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. 3

Jingyu Zhuang, Chen Wang, Lingjie Liu, Liang Lin, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. *arXiv preprint arXiv:2306.13455*, 2023. 3

Zi-Xin Zou, Zhipeng Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Yan-Pei Cao, and Song-Hai Zhang. Triplane meets gaussian splatting: Fast and generalizable single-view 3d reconstruction with transformers. *arXiv preprint arXiv:2312.09147*, 2023. 3

A APPENDIX

A.1 IMPLEMENTATION DETAILS

A.1.1 MESH TOKENIZER

Algorithm 1: Tokenization

```
Data: Discretized Vertices $\mathbf{V} = \{\mathbf{v}_i\}_N$, Triangle Faces $\mathbf{T} = \{\mathbf{t}_i\}_M$.
Result: Array $\mathbf{O}$ to hold the token sequence.
/* Write a vertex to output.                                         */
def WriteVertex(Vertex $\mathbf{v}$):
    $\mathbf{O}$.append($\mathbf{v}$.x, $\mathbf{v}$.y, $\mathbf{v}$.z);
/* Recursive function to compress one face.                          */
def TokenizeFace(HalfEdge $\mathbf{c}$, bool first):
    $\mathbf{c}.\mathbf{t}.vis$ = true;
    if not first:
        WriteVertex($\mathbf{c}.\mathbf{v}$);
    if not $\mathbf{c}.\mathbf{v}.vis$:
        $\mathbf{c}.\mathbf{v}.vis$ = true;
        $\mathbf{O}$.append(N);
        TokenizeFace($\mathbf{c}.\mathbf{n}.\mathbf{o}$, false);
    elif $\mathbf{c}.\mathbf{n}.\mathbf{o}.\mathbf{t}.vis$ and $\mathbf{c}.\mathbf{p}.\mathbf{o}.\mathbf{t}.vis$:
        return; /* End of recursion.                                 */
    elif $\mathbf{c}.\mathbf{n}.\mathbf{o}.\mathbf{t}.vis$:
        $\mathbf{O}$.append(P);
        TokenizeFace($\mathbf{c}.\mathbf{p}.\mathbf{o}$, false);
    elif $\mathbf{c}.\mathbf{p}.\mathbf{o}.\mathbf{t}.vis$:
        $\mathbf{O}$.append(N);
        TokenizeFace($\mathbf{c}.\mathbf{n}.\mathbf{o}$, false);
    else: /* both left and right triangles are not visited.          */
        $\mathbf{O}$.append(N);
        TokenizeFace($\mathbf{c}.\mathbf{n}.\mathbf{o}$, false);
        Traverse($\mathbf{c}.\mathbf{p}.\mathbf{o}$); /* Begin a new sub-sequence for $\mathbf{c}.\mathbf{p}.\mathbf{o}.\mathbf{t}$.  */
/* Begin a new sub-sequence                                          */
def Traverse(HalfEdge $\mathbf{c}$):
    if not $\mathbf{c}.\mathbf{t}.vis$:
        $\mathbf{O}$.append(B);
        WriteVertex($\mathbf{c}.\mathbf{v}$);
        WriteVertex($\mathbf{c}.\mathbf{s}$);
        WriteVertex($\mathbf{c}.\mathbf{e}$);
        $\mathbf{c}.\mathbf{s}.vis = 1$; $\mathbf{c}.\mathbf{e}.vis = 1$;
        TokenizeFace($\mathbf{c}$, true);
/* Initialization                                                    */
for Vertex $\mathbf{v}$ in $\mathbf{V}$:
    if $\mathbf{v}$ is a boundary vertex:
        $\mathbf{v}.vis$ = true;
$\mathbf{O}$.append(BOS);
/* Loop all unvisited faces and traverse.                            */
for Face $\mathbf{t}$ in $\mathbf{T}$:
    if not $\mathbf{t}.vis$:
        Traverse($\mathbf{t}$.halfedges[0]);
$\mathbf{O}$.append(EOS);
```

Tokenization Algorithm. To formally define the algorithm, we first describe how to use the half-edge notation to refer the mesh data in Figure 9. For a vertex $\mathbf{v}$, its three coordinates are denoted as

$(\mathbf{v}.x, \mathbf{v}.y, \mathbf{v}.z)$. For a face $\mathbf{t}$, we store its three half-edges in an array $\mathbf{t}$.halfedges. Additionally, we use $\mathbf{v}.vis$ and $\mathbf{t}.vis$ to indicate whether this vertex or face has already been visited in the algorithm.

Comparison. We compare the original EdgeBreaker algorithm with our modified version and highlight the key differences. For clarity, we use N and P tokens in the main paper to represent moving to the right or left triangles (the next twin or previous twin half-edges). To remain consistent with the original algorithm, we now revert to using `L` and `R`.

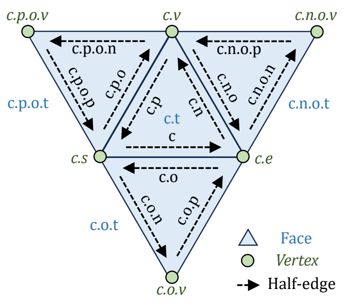


Figure 9: Mesh notation starting from half-edge $\mathbf{c}$.

(1) In the original EdgeBreaker algorithm, newly added vertices are stored in a residual form relative to the previous triangle, following the parallelogram rule (Rossignac, 1999). This method can potentially reduce the number of bits when compressed further in binary format. However, since our objective is to convert the mesh into a discrete token sequence rather than binary bits, relative vertex encoding is unnecessary. Moreover, our discretization reduces the accuracy of relative coordinates, so we simply use the absolute coordinates for each vertex.

(2) As illustrated in Figure 10, the original algorithm employs five face type tokens: `C`, `L`, `E`, `R`, and `S`. The special `S` token is used to indicate a bifurcation in the traversal tree. However, since the binary tree must be flattened into a 1D sequence, both branches dependent on the `S`-type triangle can introduce long-range dependencies, particularly if the first-visited branch is long, as shown in Figure 11. To avoid such dependencies, we explicitly duplicate the necessary vertices before visiting the second branch, which can be interpreted as restarting a sub-mesh sequence. While this approach slightly worsens the compression rate, it effectively eliminates long-range dependencies, which can be difficult for the transformer model to learn.

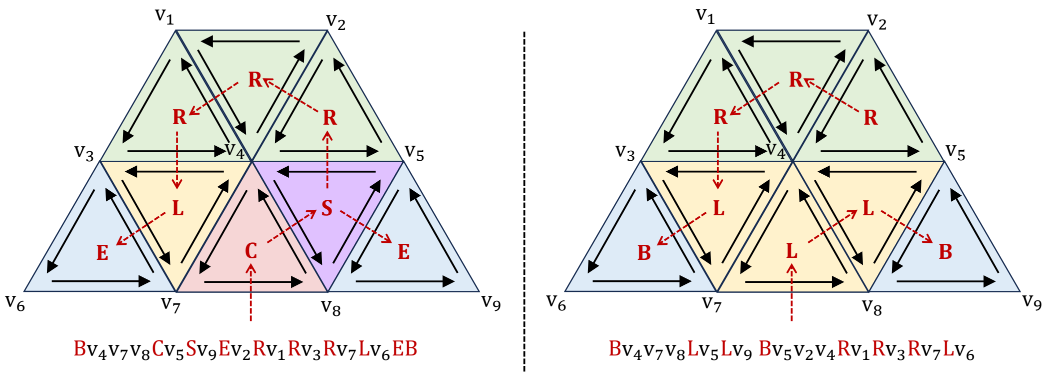
(3) During tokenization, the `C` token must be distinguished from the others, but it behaves similarly to the `L` token during de-tokenization. Since tokenization is unnecessary during inference, we replace all `C` tokens with `L` for training. Additionally, we merge the `E` token into `B`, as `E` always follows `B` after removing the `S` token. For simplicity, we don't consider special cases such as holes or handles, since they can still be represented at the cost of several degenerated faces, which can be easily cleaned during post-processing. As a result, our modified algorithm uses only three face types: `L`, `R`, and `B`.

A.1.2 Auto-regressive Auto-encoder

Datasets. For the ArAE model, we use meshes from the Objaverse and Objaverse-XL datasets (Deitke et al., 2023b;a). Given that the original datasets include many low-quality 3D assets, we filter the meshes empirically based on metadata captions or file names (Tang et al., 2024a). Our training set comprises approximately 112K meshes with fewer than 4,000 faces from both Objaverse and Objaverse-XL. For finetuning, we select only the higher-quality meshes from Objaverse, totaling around 44K. Each mesh undergoes a preprocessing pipeline where close vertices are merged, collapsed faces are removed, and the mesh is normalized to fit within a unit cube. The vertices are quantized at a resolution of 512^3. Our tokenizer's vocabulary thus includes 512 coordinate tokens, 3 face type tokens (N, P, B) and 3 special tokens (BOS, EOS, PAD), resulting in a total vocabulary size of 518.

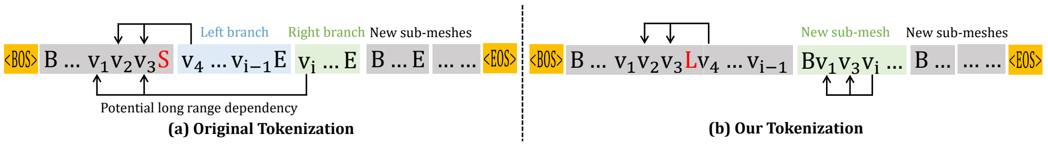
Architecture. We illustrate the detailed architecture of our network in Figure 12. The encoder of our ArAE model employs a cross-attention layer as outlined by (Zhang et al., 2023). We sample 8,192 points uniformly from the mesh surface and use a $4\times$ down-sampled embedding to query the latent code. Consequently, the latent code has a shape of $\mathbf{Z} \in \mathbb{R}^{2048\times 64}$. The decoder transformer is comprised of 24 self-attention layers, each with a hidden dimension of 1,536 and 16 attention heads. Overall, our ArAE model contains approximately 0.7 billion trainable parameters.

Training and Inference. We train the ArAE model on 64 A100 (80GB) GPUs for approximately one week. The batch size is 4 per GPU, leading to an effective batch size of 256. We use the AdamW optimizer (Loshchilov & Hutter, 2017) with a cosine-decayed learning rate that ranges from 5×10^{-5} to 5×10^{-6}, a weight decay of 0.1, and betas of $(0.9, 0.95)$. Gradient clipping is



v1
v2
v3
v4
v5
v6
v7
v8
v9
R
L
E
C
S
B
Bv4v7v8Cv5Sv9Ev2Rv1Rv3Rv7Lv6EB
Bv4v7v8Lv5Lv9 Bv5v2v4Rv1Rv3Rv7Lv6

Figure 10: Comparison of the original EdgeBreaker traversal order with our modified version.



Left branch
Right branch
New sub-meshes
<BOS> B ... v1v2v3S v4 ... vi−1E vi ... E B ... E ... ... <EOS>
Potential long range dependency
(a) Original Tokenization
New sub-mesh
New sub-meshes
<BOS> B ... v1v2v3Lv4 ... vi−1 Bv1v3vi ... B ... ... <EOS>
(b) Our Tokenization

Figure 11: Our tokenization reduces potential long range dependency in the token sequence.

applied with a maximum norm of 1.0. Given that each point cloud input can correspond to multiple plausible meshes but only one ground truth mesh is available, we apply robust data augmentation techniques to address data insufficiency during training: (1) The input mesh is randomly scaled by a factor between $[0.75, 0.95]$. (2) The mesh is randomly rotated along the vertical axis by up to 30 degrees. (3) Random quadric edge collapse decimation (Garland & Heckbert, 1997) is applied to perturb the number of faces. There is a 50% chance at each iteration that the mesh will be decimated to no more than one-quarter of its original number of faces. During inference, we employ a multinomial sampling strategy with a top-k of 10. To ensure that the generated sequence is valid for decoding by the tokenizer, we apply the following rules to post-process the logits of the next token based on the prefix token sequence: (1) The token following BOS should be B. (2) After B, there should be 9 coordinate tokens. (3) After N or P, there should be 3 coordinate tokens. (4) Otherwise, the next token should be one of N, P, B, or EOS.

A.1.3 Latent Diffusion

Datasets. For the image-conditioned diffusion model, we use meshes from the Objaverse dataset along with rendered images from the G-Objaverse dataset (Qiu et al., 2024). The final training dataset comprises approximately 75K 3D meshes, each accompanied by rendered RGB images from 38 different camera poses. We select only 21 camera poses that align with the front face of most objects in Objaverse empirically.

Architecture. Our DiT model consists of 24 layers of self-attention and cross-attention, as detailed in Section 3. Each layer can be expressed as follows:

$$\gamma_1, \beta_1, \alpha_1, \gamma_2, \beta_2, \alpha_2 = \text{Chunk}(\mathbf{T} + \mathbf{t}_e) \tag{4}$$

$$\mathbf{x} = \mathbf{x} + \alpha_1 \times \text{SelfAttention}(\text{LayerNorm}_1(\mathbf{x}) \times (1 + \beta_1) + \gamma_1) \tag{5}$$

$$\mathbf{x} = \mathbf{x} + \text{CrossAttention}(\mathbf{x}, \mathbf{c}) \tag{6}$$

$$\mathbf{x} = \mathbf{x} + \alpha_2 \times \text{FeedForward}(\text{LayerNorm}_2(\mathbf{x}) \times (1 + \beta_2) + \gamma_2) \tag{7}$$

where $\mathbf{T}$ is a layer-wise learnable scale shift table, $\mathbf{t}_e$ denotes the timestep features, and $\mathbf{c}$ denotes the condition features. All attention layers have a hidden dimension of 1,024 and use 16 attention heads. We utilize the pretrained CLIP-H model from OpenCLIP (Ilharco et al., 2021) as the image feature encoder. Features from the last layer of the vision transformer are employed as conditions for the cross-attention layers. The DiT model has approximately 0.5 billion trainable parameters in total.

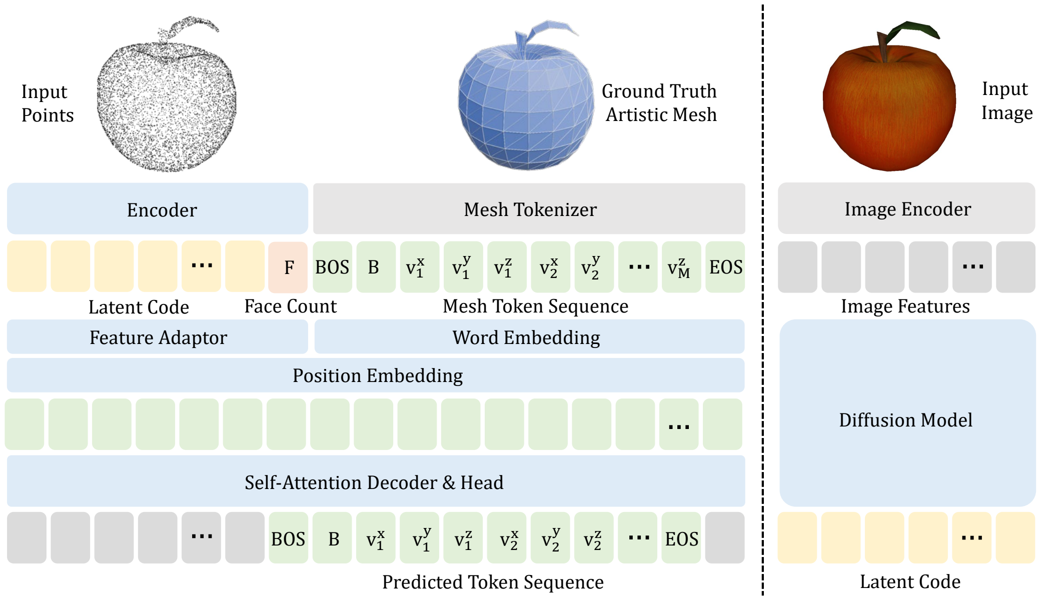


Figure 12: **Network Architecture** of our ArAE and DiT models. Our ArAE contains an encoder that takes a point cloud as input and encodes it to a fixed-length latent code, and an auto-regressive decoder that predicts variable-length mesh token sequence. The latent space can be used to train diffusion models conditioned on other more difficult conditions, such as single-view images.

	AI-Mesh		Artist-Mesh	
	CD↓	HD↓	CD↓	HD↓
MeshAnything	0.072	0.170	0.066	0.195
MeshAnythingV2	0.081	0.241	0.050	0.164
Ours	**0.034**	**0.079**	**0.036**	**0.112**

Table 2: **Reconstruction Quality** on point-cloud conditioned mesh generation.

Training and Inference. We train the DiT model on 16 A100 (40GB) GPUs for approximately one week. The lightweight ArAE encoder allows for on-the-fly calculation of the latent code during training, so we can also apply random down-sampling and rotation as data augmentation techniques for the point cloud input. The batch size is set to 32 per GPU, resulting in an effective batch size of 512. We use the same optimizer as for the ArAE model. Training employs the min-SNR strategy (Hang et al., 2023) with a weight of 5.0. The DDPM is configured with 1,000 timesteps and utilizes a scaled linear beta schedule. For classifier-free guidance during inference, 10% of the image conditions are randomly set to zero for unconditional training. During inference, we use the DDIM scheduler (Song et al., 2020) with 100 denoising steps and a Classifier-Free Guidance (CFG) scale of 7.5.

A.2 More Results

A.2.1 Reconstruction Quality

We conducted a qualitative comparison experiment to evaluate the Chamfer Distance (CD) and Hausdorff Distance (HD) between the generated meshes and the input point clouds. Due to the lack of widely recognized evaluation datasets, we tried to construct two test datasets from different sources: (1) AI-Mesh: This dataset comprises 30 meshes generated by other image-to-3D models, extracted using the Marching Cubes algorithm. (2) Artist-Mesh: This dataset consists of 100 artistic meshes obtained from basemesh[1], which were not seen during the training process. All meshes are

[1]https://www.thebasemesh.com/

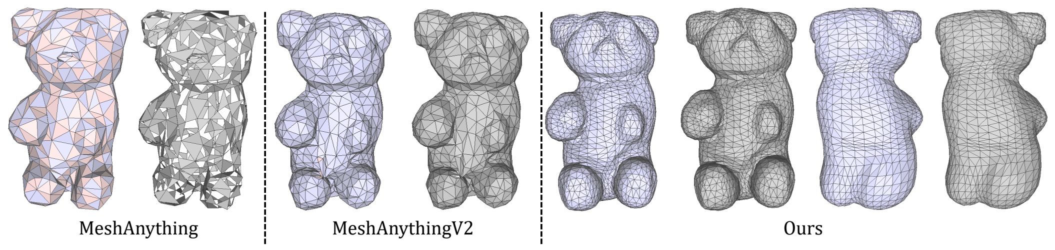


Figure 13: **Face orientation**. Our model ensures correct face orientation of the generated mesh.

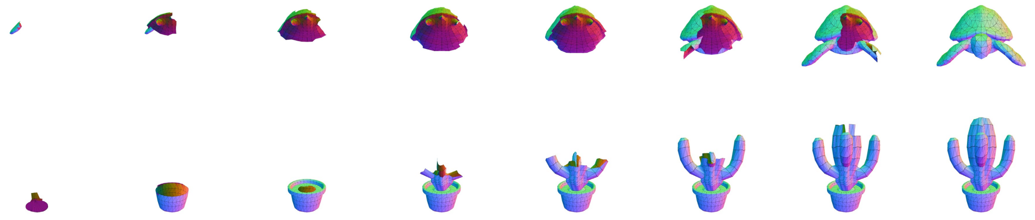


Figure 14: Visualization of the generation progress. Please visit our project page for video results.

	Compression Ratio ↓	Sub-sequence Count ↓	Tokenization Speed ↑
AMT	**46.2%**	199.5	8.6
Ours	47.4%	**54.7**	**25.2**

Table 3: **Comparison of the tokenizer algorithm.** Our tokenizer has a comparable compression ratio as AMT (Chen et al., 2024e) but uses much fewer sub-sequences per mesh, which makes model training easier. Moreover, our implementation is three times faster (mesh per second).

normalized into $[-1, 1]^3$. The summarized evaluation scores are presented in Table 2. We observed the following: (1) Our model achieves the best performance on surface metrics, even when using only point clouds as input. In contrast, the MeshAnything (Chen et al., 2024d) series further leverage surface normals. (2) MeshAnythingV2 (Chen et al., 2024e) performs significantly worse on the more challenging AI-Mesh dataset, often failing to generate complete surfaces in these cases (consistent with the results shown in Figure 5).

A.2.2 Face Orientation

In Figure 13, we show the face orientation and rendering with back-face culling. The tokenization algorithm of our method and MeshAnythingV2 (Chen et al., 2024e) ensures that the face orientation in each mesh sequence is correct, while MeshAnything (Chen et al., 2024d) fails to do this.

A.2.3 Generation Progress

In Figure 14, we show the generation progress by simultaneously de-tokenizing the sequence. Our model successfully learns how to generate a valid sequence for the modified EdgeBreaker algorithm to de-tokenize.

A.2.4 Tokenization Algorithm

In Table 3, we compare our mesh tokenization algorithm with AMT from MeshAnythingV2 (Chen et al., 2024e). Both algorithms aim to traverse all triangular faces. While AMT can only move to a fixed side that shares the last two vertices in the sequence, our algorithm allows movement to both the left and right sides. This flexibility reduces the number of sub-sequences, making training easier and improving model robustness, as demonstrated in Figure 5. Although AMT requires only

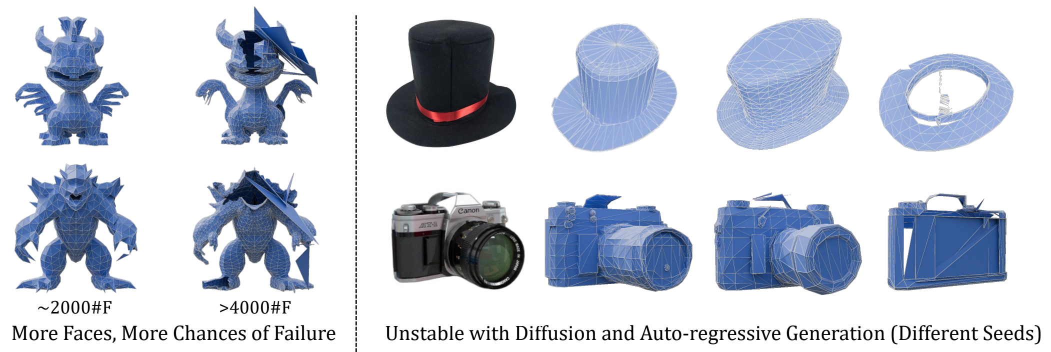

~2000#F
>4000#F
More Faces, More Chances of Failure
Unstable with Diffusion and Auto-regressive Generation (Different Seeds)

Figure 15: **Limitations**. We show some failure cases of our method to illustrate our limitations.

3 coordinate tokens for a subsequent face and our method requires 4 tokens (with an additional face type token), the amortized compression ratio remains similar.

A.2.5 LIMITATIONS

Despite the promising results, our approach has several limitations: (1) The compression ratio of our tokenization algorithm remains insufficient. Complex meshes, such as those representing game characters, typically require a much higher number of faces, exceeding the current capacity of our model. (2) Although our model can generate meshes with up to 4,000 faces, the success rate decreases as the sequence length increases, and the generation time becomes progressively longer. (3) While the auto-regressive next-token prediction allows for diverse outputs from the same input, it also leads to potentially unsatisfactory results. Since there are infinite plausible face topologies for the same surface, precise control over the generation is difficult to achieve. We illustrate some failure cases in Figure 15.