# FACE: A Face-based Autoregressive Representation for High-Fidelity and Efficient Mesh Generation

Supplementary Material

## 1. Architecture Details

This section details the architectures of the Face Embedding and the Face Decoding Head. These modules serve as the interface between the discrete geometric codebook and the continuous latent embedding space of the Transformer backbone.

### 1.1. Face Embedding

The Face Encoder is responsible for projecting a raw face (represented as a sequence of 9 discrete coordinate tokens) into a compact, continuous face embedding $h \in \mathbb{R}^{D_{\text{latent}}}$.

Let a triangular face $f$ be defined by a sequence of discrete tokens $\mathbf{t} = \{t_1, t_2, \ldots, t_9\}$, where each $t_j \in \{1, \ldots, V\}$ represents a quantized coordinate index from a vocabulary of size $V$.

**Token Projection.** First, each token $t_j$ is mapped to a dense vector via a shared token embedding layer $E_{\text{tok}} : \{1, \ldots, V\} \to \mathbb{R}^{D_{\text{embed}}}$.

**Feature Fusion.** To capture the holistic geometry of the triangle, we concatenate these individual coordinate embeddings into a unified feature vector:

$$h_{\text{flat}} = \text{Concat}\left(E_{\text{tok}}(t_1), E_{\text{tok}}(t_2), \ldots, E_{\text{tok}}(t_9)\right) \tag{1}$$

**Dimensionality Reduction.** This high-dimensional vector is then processed by a Multi-Layer Perceptron (MLP), denoted as $\mathcal{F}_{\text{enc}}$, to fuse intra-face geometric information and project it to the transformer's latent dimension:

$$h = \mathcal{F}_{\text{enc}}(h_{\text{flat}}) \tag{2}$$

Structurally, $\mathcal{F}_{\text{enc}}$ comprises a projection layer mapping $9 \times D_{\text{embed}} \to 2 \times D_{\text{latent}}$, followed by a GELU activation and a final projection to $d_{\text{latent}}$. This process ensures the embedding $h$ captures the holistic geometry of the triangular face.

### 1.2. Face Decoding Head

The Face Decoding Head performs the inverse operation: it reconstructs the 9 discrete coordinate tokens from the continuous face embedding $h$. To ensure geometric validity, we model this process autoregressively.

**Causal Formulation.** We employ a CausalMLP [2] approach where the probability of the $j$-th coordinate token is conditioned on both the global face context $h$ and all preceding coordinates within the face. The joint probability is factorized as:

$$P(\mathbf{t}|h) = \prod_{j=1}^{9} P(t_j|h, t_{<j}) \tag{3}$$

**Implementation.** We realize this using 9 distinct, position-specific prediction heads $\mathcal{H}_1, \ldots, \mathcal{H}_9$. Each head $\mathcal{H}_j$ consists of a 3-layer MLP (Linear-ReLU-Linear-ReLU-Linear) that maps the input features to logits over the vocabulary $V$.

**Input Formulation.** To preserve strict causal dependency and positional information, the input feature $u_j$ for the $j$-th head to each head is constructed by concatenating the face embedding $h$ with the embeddings of all previously generated tokens. To preserve positional information, we use distinct embedding layers $\{E^{(k)}_{\text{coord}}\}_{k=1}^{8}$ to encode the conditional history. The input feature $u_j$ for the $j$-th head is defined as:

$$u_j = \begin{cases} h, & \text{if } j = 1 \\ \text{Concat}\left(h, E^{(1)}_{\text{coord}}(t_1), \ldots, E^{(j-1)}_{\text{coord}}(t_{j-1})\right), & \text{if } j > 1 \end{cases} \tag{4}$$

Finally, the probability distribution for the $j$-th token is computed via a Softmax layer:

$$P(t_j|h, t_{<j}) = \text{Softmax}(\mathcal{H}_j(u_j)) \tag{5}$$

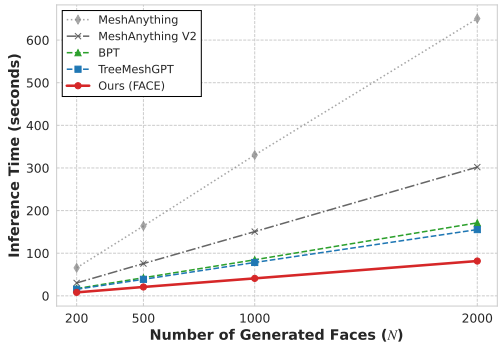


Figure 1. **Inference speed comparison.** We benchmark generation time across varying face number. By reducing the effective sequence length, **FACE** achieves significantly faster inference compared to state-of-the-art autoregressive baselines, exhibiting a nearly $2\times$ speedup over TreeMeshGPT and an $8\times$ speedup over MeshAnything at 2000 faces.

## 2. Inference Speed Analysis

Computational efficiency is a critical factor for mesh generation. To conduct a rigorous evaluation of inference speed,

we re-implemented the core token compression logic of all baseline methods within our unified codebase. This ensures that all models are tested under identical environmental conditions, hardware configurations (a single NVIDIA A100 GPU), and optimization levels (e.g., KV-caching).

We measured the time required to generate meshes with face counts ranging from 200 to 2000. As illustrated in Figure 1, our method consistently outperforms the baselines.

The performance gap widens as mesh complexity increases, highlighting the scalability of our approach. Specifically, at a resolution of 2000 faces, our approach achieves an inference speed approximately $2\times$ faster than TreeMeshGPT [2] and $8\times$ faster than MeshAnything [1]. This confirms that our "one-face-one-token" strategy effectively mitigates the quadratic bottleneck of standard autoregressive transformers.

## 3. Qualitative Comparison

We present extensive qualitative comparisons on the Objaverse and Famous datasets in Figure 2 and Figure 3, respectively. These visual results demonstrate the ***geometric quality*** (captures global shapes and proportions) and ***detail preservation*** (maintains high-frequency details and clean topology) of our generated meshes compared to baseline methods.

## References


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

[2] Stefan Lionar, Jiabin Liang, and Gim Hee Lee. Treemeshgpt: Artistic mesh generation with autoregressive tree sequencing. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 26608–26617, 2025. 1, 2


Figure 2. Qualitative results on the Famous dataset.

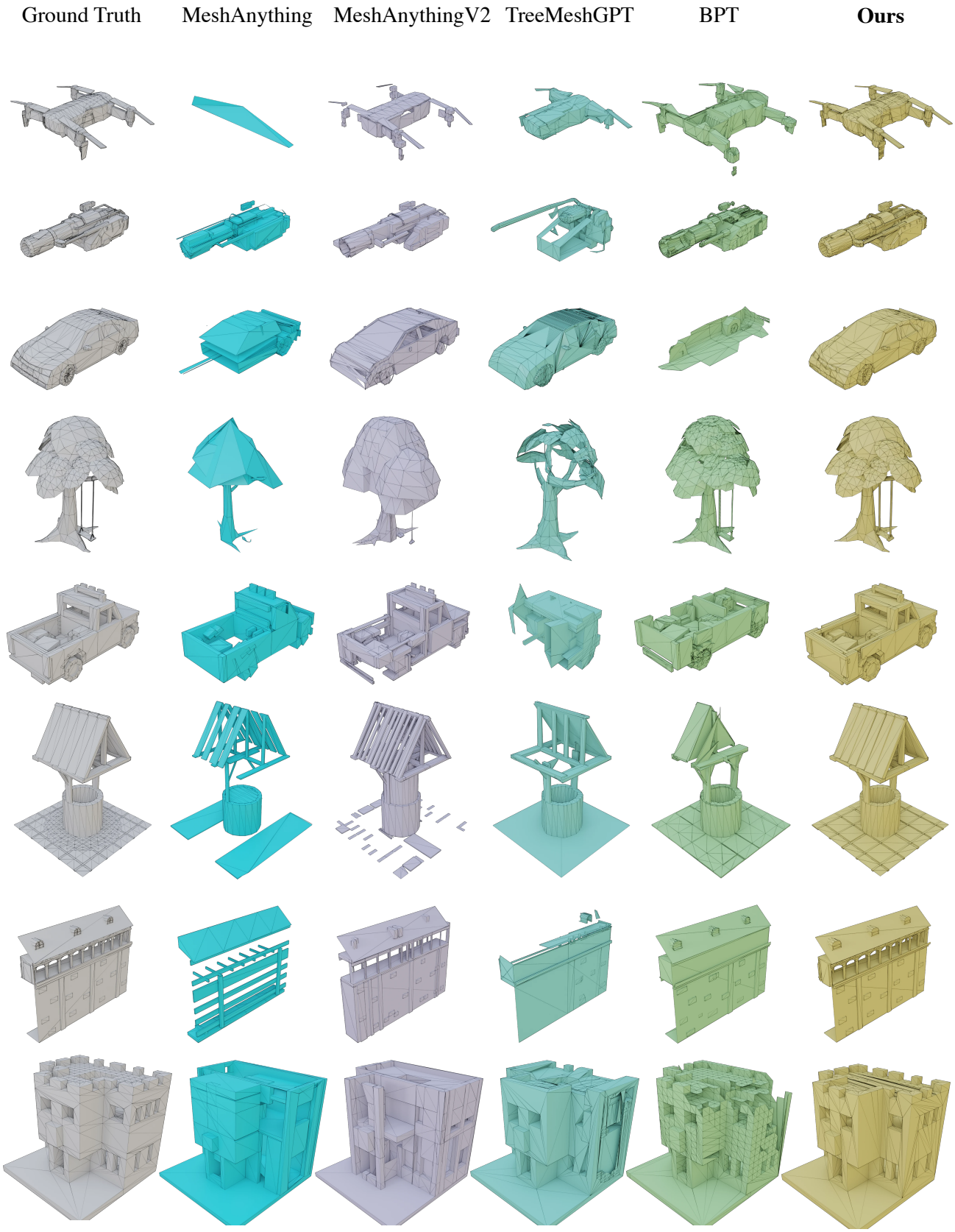


Figure 3. Qualitative results on the Objaverse dataset.