A. Appendix of MeshMosaic

A.1. Data Preprocessing

Training datasets are drawn from Objaverse-XL [10] and other licensed datasets. To enhance data quality, we implemented several filtering procedures. Only meshes with $[500, 32000]$ faces are retained, excluding those with excessively low or high token lengths. Meshes are subsequently cleaned and optimized using PyMeshlab [33]: duplicate vertices/faces removed, closely spaced or overlapping vertices merged, non-manifold elements and edges eliminated.

And we computed a point-to-face ratio for each model:

$$\mathbf{\Phi_{p/f}} = \frac{\mathcal{N}_p}{\mathcal{N}_f} \tag{1}$$

Meshes with $\mathbf{\Phi_{p/f}} > 0.8$ are filtered out to exclude objects with too many open boundaries. For robustness, we augment data via random rotations with three axes and uniform scaling within $[0.9, 1.0]$.

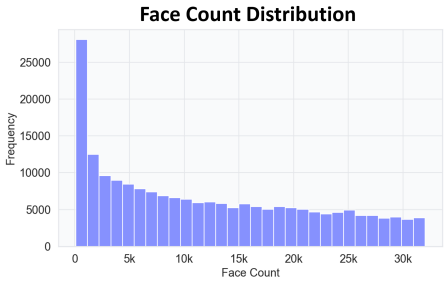

Figure 11. Distribution of face count in our dataset.

Fig. 11 and Fig. 12 provides a comprehensive analysis of our dataset. Moving from left to right, the first graph illustrates the distribution of the number of patches generated through random segmentation. Most simple shapes are divided into fewer than ten patches, whereas a small number of highly complex cases yield over 60 patches. Although our training set contains no more than sixty splits per instance, our method can handle inference tasks involving hundreds of patches during testing. This highlights the strong generalization capability of our method (as shown in Fig. 1).

Next, we report statistics on the number of connected components for samples with native splits as previously noted. Most samples containing fewer than ten patches, similar to the distribution observed from random splits.

Facilitated by our local-to-global architecture, the required token length for each training or inference phase is significantly diminished. We present the distribution of token lengths for all split patches. The vast majority contain fewer than 6,000 tokens, with the longest sequence not exceeding 20,000 tokens. This approach allows us to break down challenging problems into several manageable subproblems, each of which can be solved independently. Lastly, we report that boundary condition tokens are much shorter than full tokens, with all lengths falling below 2,000 tokens.

A.2. Discussion and Ablation

Ablation Study. We perform an extensive ablation study to systematically examine the roles of boundary conditions and global point cloud features within our mesh generation architecture. This analysis provides critical insights into how each conditioning mechanism contributes to the fidelity and coherence of generated meshes.

As illustrated in Fig. 13, we analyze three distinct ablated configurations: (1) **Ours w/o GPC**, in which the global point cloud conditioning feature is entirely removed; (2) **Ours w/o BD**, where the GRU network responsible for boundary condition encoding is omitted; and (3) **Ours w/o SA**, which disables the concatenation of boundary tokens for self-attention within the network.

To ensure a thorough assessment, ablation experiments are conducted under two regimes. The first regime (top row in Fig. 13) involves a controlled overfitting scenario, where the network is trained exclusively on a single airplane mesh for 20 epochs with a batch size of 8; segmentation boundaries are randomized at each iteration to probe the model's adaptability and generalization. The second regime (bottom row) evaluates the network after comprehensive training on our entire dataset, thereby measuring its capability across diverse object geometries.

For consistent comparison and clear visualization, all results in Fig. 13 utilize an identical segmentation scheme, indicated by the purple dividing line, which partitions each shape into three patches at inference time.

When global point cloud information is omitted (**Ours w/o GPC**), the network demonstrates reasonable performance in the overfitted regime, as it only needs to reconstruct a single shape. However, in the full dataset setting, the absence of global context leads to significant errors—most notably, the right portion of the mesh exhibits pronounced deformation and collapse, revealing the necessity of global information for guiding overall shape reconstruction.

When the GRU-based boundary encoding is eliminated (**Ours w/o BD**), visible cracks emerge along the seams in both regimes. In addition, the absence of boundary communication induces substantial mesh density asymmetry in the full dataset setting, with adjacent patches developing inconsistencies. This reflects the model's inability to properly propagate local information between neighboring patches.

Disabling the concatenation of boundary tokens for self-attention (**Ours w/o SA**) again results in prominent seam artifacts, and in the full dataset scenario, produces overlapping, self-intersecting patches. The lack of explicit constraints leads to independent patch generation, which ultimately causes geometric inconsistencies and structural artifacts.

In contrast, the full model employing both boundary and

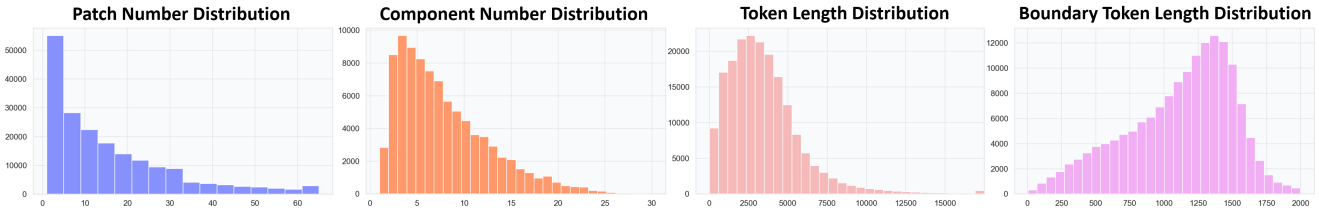

Figure 12. Dataset statistics: from left to right (1) distribution of number of patches per mesh; (2) number of connected components for partially connected components; (3) token length per training patch; (4) token length of boundary condition sequences.

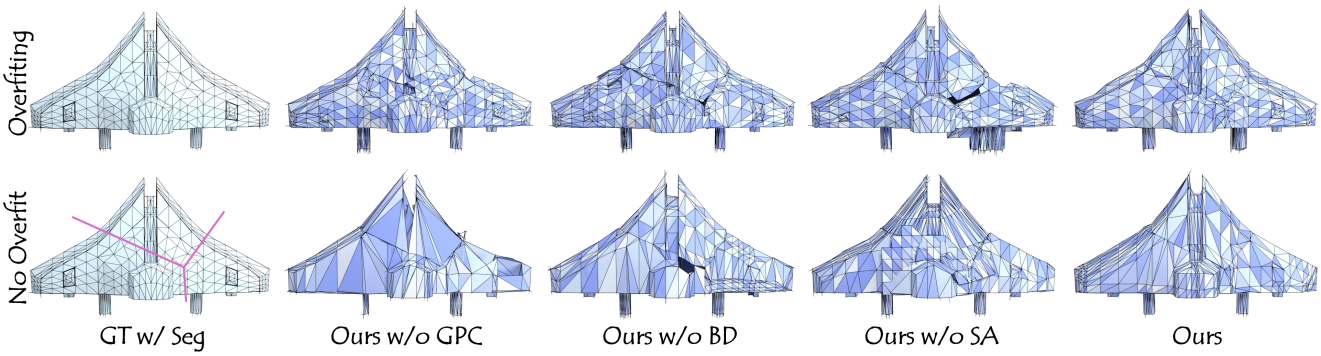

Figure 13. Ablation for boundary condition and global point cloud.

global conditioning produces meshes that are complete, uniform, and visually coherent, with mesh density and topology smoothly balanced across all patches. This clearly demonstrates the effectiveness of our proposed integration of local and global context, and highlights the importance of both conditional mechanisms for high-fidelity mesh generation. It's worth noting that we did not use PartField [26] for semantic decomposition in this example. Instead, we applied random segmentation to divide the aircraft into three parts. This was done to better visualize the connections along the seams and intentionally make them more noticeable. Our inference with semantic segmentation would not produce seams that are clearly misaligned with the object's principal orientation.

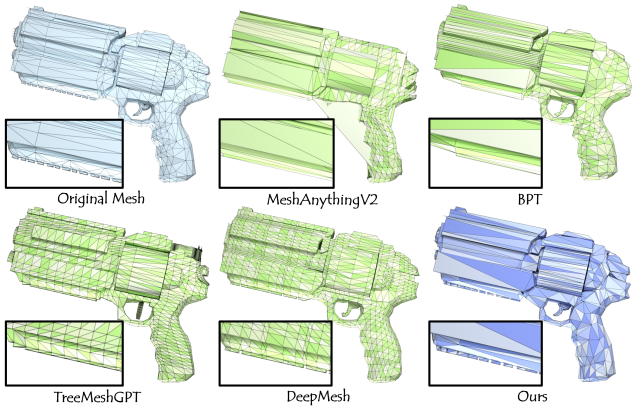

Figure 14. Detail recovery comparison.

Detail Recovery. Thanks to our local-to-global sequential mesh generation strategy, our method significantly surpasses previous approaches in detail preservation. Unlike other methods that rely on a single quantized resolution for the entire model, our approach assigns an independent 512^3 resolution to each patch. As illustrated in Fig. 14, our method is uniquely capable of recovering the original edge details of the pistol, whereas competing methods either fail to capture these features or merge them into indistinct blocks.

Segmentation Input. Although our approach is primarily designed to operate under a segmented training and inference regime, it nevertheless retains the flexibility to infer simple shapes without explicit segmentation. As demonstrated in Fig. 15, both our method and DeepMesh [49] are capable of reconstructing a torus model in the absence of any segmentation.

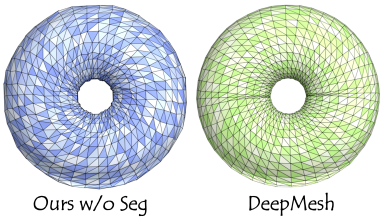

Figure 15. Inference without segmentation.

Further analysis of segmentation strategies is shown in Fig. 16. In the middle example, reconstruction is performed using random segmentation. While the overall shape and fine details can still be recovered, the absence of semantic

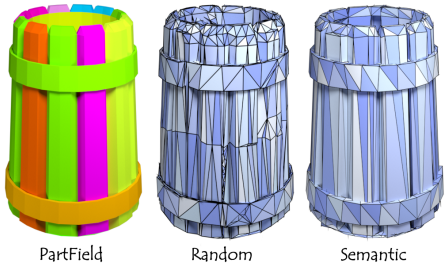

Figure 16. Comparison of random and semantic segmentation.


(a) PartField (b) Random (c) z-axis
Figure 17. Our results with different segmentation.


Figure 18. Segmentation in Fig. 9.

segmentation often results in patch boundaries that traverse flat or non-essential regions, introducing visual clutter and irregularity into the mesh appearance. By employing PartField [26] for semantic guidance, our method achieves noticeably cleaner and more coherent mesh boundaries, significantly enhancing the aesthetic quality without compromising reconstruction fidelity.

We also evaluate the robustness to segmentation inputs in Fig. 17 on the same shape using: (a) PartField semantic parts, (b) random segmentation, and (c) naive z-axis slicing into 10 bands. Although (b,c) yield non-semantic and often jagged boundaries, our method still produces a complete, connected mesh in our tested cases. Without semantic parts, we may observe slightly less regular edge flow near artificial seams, but the global structure remains stable. Fig. 17 also shows our method remains coherent even under "failed" segmentations, suggesting segmentation mainly affects seam placement. Then for a manifold input mesh, any face-label partition yields patch submeshes that are manifold: removing faces can create boundaries but cannot increase an edge's incident-face count beyond 2. Thus, edge-nonmanifoldness is not introduced by partitioning itself. Finally, in Fig. 18, we provide the segmentation result used in Fig. 9.

Comparison with DeepMesh. Directly scaling DeepMesh [49] to our local-to-global setting is non-trivial. To further demonstrate the benefits of our local-to-global framework and the importance of our boundary condition method, we perform an ablation study that directly compares it with DeepMesh [49]. As depicted in Fig. 19, we assess two distinct inference settings for DeepMesh: the first utilizes the entire shape without segmentation, while the second processes each segmented patch individually and subsequently assembles them to form the complete object.

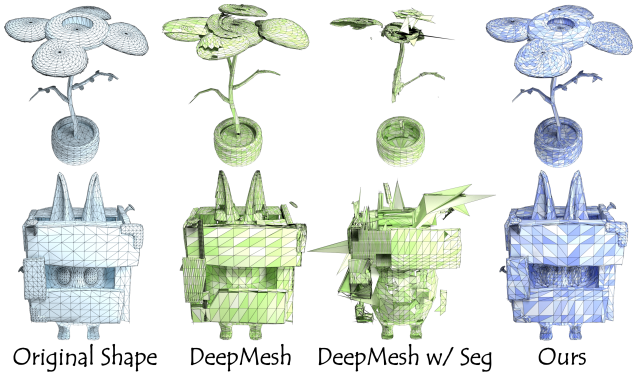

Figure 19. Our method without segmentation.

In the first scenario, where DeepMesh generates the mesh from an unsegmented input, it succeeds in producing reasonable global geometry. However, the quality of reconstructed fine details—such as the eye region in the second example is noticeably lacking. This demonstrates DeepMesh's limitations when handling intricate local features under a global, one-shot autoregressive scheme.

In the second scenario, we input our segmented data into DeepMesh, allowing it to process each patch independently. Local mesh resolution is indeed improved due to smaller region-specific quantization. Nonetheless, the absence of key contextual mechanisms: explicit boundary conditions and global shape information, leads to significant artifacts. The resulting meshes exhibit poor coherence across patch boundaries, with misaligned regions and inconsistent topology.

By contrast, our local-to-global strategy explicitly conditions each segment on both boundary and global cues, enabling seamless integration and faithful reconstruction of complex features throughout the mesh. This comparative analysis clearly highlights the expressive superiority and practical robustness of our method, especially in scenarios that demand high-resolution details and structurally consistent results.

Boundary Encoding. To determine a reasonable boundary budget, we perform an empirical analysis using **10,000** random cuts on **100** Objaverse shapes. The average num-

Figure 20. Results generated by *MeshMosaic* using text prompts (left) or image inputs (right). Initial 3D shapes are created using CLAY [48] and enhanced by our approach.

ber of intersected faces is approximately **288**, while cases exceeding **512** boundary triangles are relatively uncommon, accounting for only about **4%** of all samples. These statistics suggest that a budget of 512 is sufficient for the large majority of cases.

This choice is also motivated by the token budget of the model. Under our tokenization scheme, **512** boundary triangles already expand to nearly **2K** tokens within a **9K** token truncation window. Further increasing the boundary budget would consume more of the available context and leave less room for regular mesh tokens, which may in turn degrade generation quality. In principle, this limit can be relaxed when using a base model with a larger context window.

Imperfect previous patches. To evaluate robustness to imperfect upstream predictions, we further conduct a stress test under the random setting in Fig. 17. Specifically, we manually corrupt previously generated patches by truncating their geometry and inserting noisy elongated triangles in the warp figure, shown in red, before generating subsequent patches conditioned on these imperfect contexts. We observe that the following patches remain largely stable, and the final mesh is still coherent in most cases. This result suggests that our method is reasonably robust to moderate errors in earlier patches and does not exhibit catastrophic failure under such perturbations.

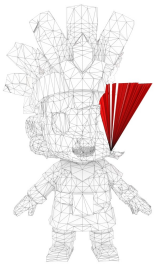

Table 3. Comparison of runtime performance between DeepMesh and our method variants. The table reports the training time per window (9K tokens) and the inference time per token in seconds.

	DeepMesh	Ours w/o BD	Ours w/o GPC	Ours
Train	0.451	0.531	0.558	0.633
Infer	0.025	0.024	0.024	0.024

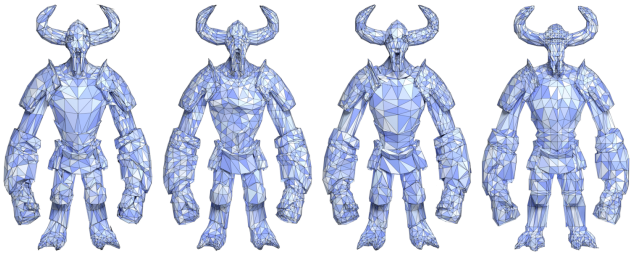

Figure 21. Diversity of our generation results.

Runtime. We developed our method on the DeepMesh [49] codebase, thereby ensuring a comparable runtime environment and a rigorous basis for performance assessment. Tab. 3 details the training and inference efficiency of DeepMesh versus our proposed framework, including variants with specific ablations such as the boundary condition encoding (**BD**) and global point cloud encoding (**GPC**).

By incorporating GRU based boundary condition encoding and a global point cloud conditioning module into our pipeline, we necessarily introduce additional computational operations during the training stage. This enhancement results in a moderate increase in training time relative to the original DeepMesh [49] implementation.

However, our approach leverages the KV cache technique

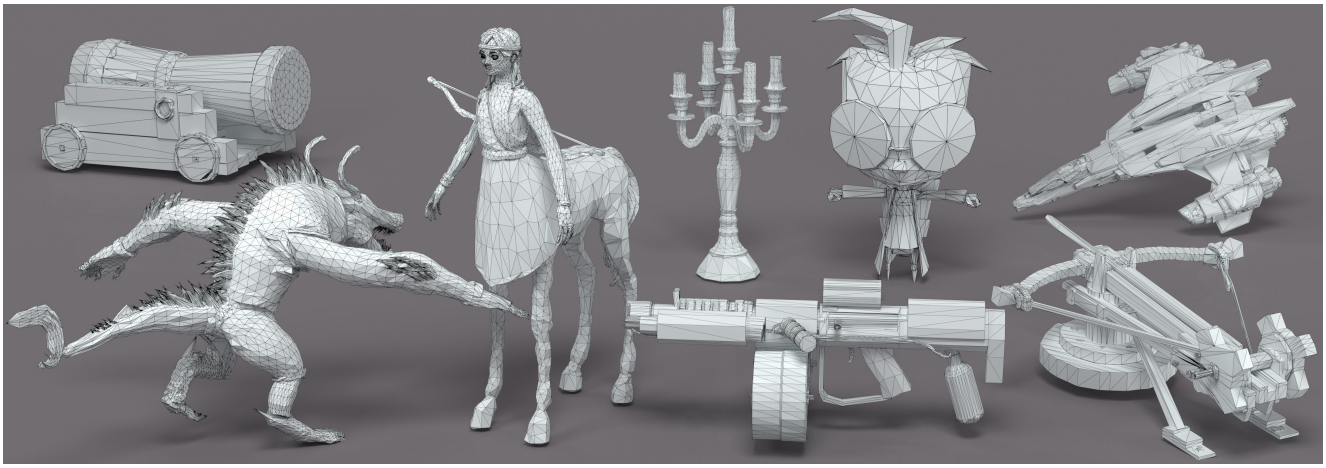

Figure 22. Gallery of our artist mesh generation results.

to substantially accelerate inference. All conditional features from global and boundary sources are preprocessed once at the beginning of inference and then cached for subsequent decoding steps. As a result, our method maintains an average per token inference time that is nearly equivalent to DeepMesh across different ablation settings, thereby preserving strong deployment efficiency and scalability. This is further confirmed by a full pipeline comparison on the mesh in Fig. 17(a). Tab. 4 shows that our method generates 30 patches with a total of **22,736** tokens in 402.26 s on a single A800 GPU, corresponding to **56.52 tok/s**, while DeepMesh generates the whole mesh with **15,782** tokens in 277.06 s, corresponding to **56.96 tok/s**. The peak memory usage increases only slightly from 9.74 GB to 9.92 GB.

It is important to note that the overall inference time depends linearly on the total number of generated tokens. When the token count is matched, our method achieves inference performance on par with DeepMesh. At the same time, the local to global generation strategy enables our method to reconstruct meshes with substantially more polygons, thereby supporting finer geometric detail and more complex structures. This stronger expressive capability naturally leads to longer token sequences and thus higher absolute inference time for rich meshes, while the underlying per token efficiency remains high. Nevertheless, inference on very complex meshes can still be time consuming. For example, as shown by the mesh in the middle of Fig. 1, when the number of faces exceeds 100K, inference typically requires several hours to complete. This remains far from meeting the efficiency demands of industrial applications.

Text and Image-Conditioned Generation. Generating 3D shapes from text or image inputs has become a prominent direction in computer graphics and generative modeling, with recent advances delivering impressive results in open-domain shape synthesis. However, many contemporary techniques, particularly those relying on Signed Distance Functions (SDF), produce meshes by converting dense volumetric grids via algorithms like marching cubes. This process often results in excessive and redundant triangles, leading to overly complex meshes that are inefficient for practical applications in animation, rendering, or interactive editing.

Table 4. Inference efficiency comparison on the mesh in Fig. 17(a). Although our full pipeline processes more total tokens due to patch based generation, it achieves nearly identical per token throughput to DeepMesh with only a marginal increase in peak memory usage.

Method	#Patches	Total Tokens	Time (s)	Tok/s	Peak Memory (GB)
DeepMesh	1	15,782	277.06	56.96	9.74
Ours	30	22,736	402.26	56.52	9.92

In Fig. 20, we showcase examples where state-of-the-art SDF-based methods, such as CLAY [48], generate initial 3D geometry from either textual prompts or image inputs. We then refine these preliminary outputs using *MeshMosaic*, producing artist-quality meshes that retain rich geometric details while optimizing triangle utilization. Compared to the raw outputs from CLAY, meshes processed by our framework exhibit cleaner topology, enhanced visual fidelity, and improved efficiency, making them far better suited for real-world downstream tasks. These results highlight the effectiveness of our method for transforming dense generative outputs into structured, high-quality assets tailored for professional use.

Diversity Generation. We further illustrate the versatility and diversity of mesh outputs produced by *MeshMosaic*. As depicted in Fig. 21, our framework is capable of generating a broad spectrum of meshes even when provided with an identical point cloud input. This demonstrates the network's intrinsic capacity for structural variation and contextual adaptation. For example, in the minotaur warrior scenario, our method synthesizes markedly distinct mesh representations

for different anatomical and accessory regions—including chest armor, shoulder plates, arms, and head. Each of these regions features unique geometric patterns and connectivity details, clearly reflecting localized artistic interpretation.

Importantly, despite considerable variations in mesh density, topology, and local connectivity, all generated results exhibit strong global coherence and visual consistency. There are no conspicuous artifacts or discontinuities between regions, confirming that our local-to-global generation strategy supports both creative flexibility and structural integrity across the mesh. This capability enables downstream tasks such as animation, editing, and customization by supporting the generation of diverse, high-quality assets from a unified geometric representation.

More Results Finally, we present more of our results in Fig. 22 to demonstrate the powerful capabilities of *MeshMosaic*.