MeshLLM: Empowering Large Language Models to Progressively Understand and Generate 3D Mesh

Supplementary Material

1. Additional Implementation Details

1.1. Construction of Primitive-Mesh

KNN-based. We begin by densely sampling point clouds from the mesh and then apply farthest point sampling (FPS) and KNN to identify central points and point clusters. For mesh-derived dense point clouds, FPS begins with a random point and iteratively chooses the farthest point to yield N center points. These N points serve as centroids for KNN clustering. The face category is determined through a voting process based on the categories of these sampled points. The number of categories is set as the integer value of the total number of faces divided by 200, with a maximum limit of 10. This strategy is highly efficient, requiring only 0.2 seconds to segment a 3D mesh, enabling the rapid generation of large-scale results. As shown in Fig. 1, the constructed Primitive-Mesh maintains well-preserved local 3D structural information, while each cluster patch contains a limited number of faces. This ensures compliance with the token length constraints of large language models, effectively expanding the scale of trainable data.

Semantic-based. We further employ the zero-shot 3D part segmentation method, SamPart3D [7], to construct the Semantic-based Primitive-Mesh dataset. SamPart3D is pretrained on Objaverse [2] with a 3D backbone network designed to extract visual features. It then utilizes lightweight MLPs to refine 2D segmentation masks into scale-conditioned groups for point cloud clustering (we set the scaling factor to 1.2), enabling effective 3D data segmentation. We perform SamPart3D on more than 25k high-quality meshes that have undergone aesthetic evaluation [5]. To obtain semantic labels for each part, we render multi-view images and annotate the corresponding 2D regions for each segmented 3D component. We then query GPT-4o using these images for semantic labels. This strategy provides more accurate semantic information for mesh parts but is time-consuming and incurs API query costs. We utilize 128 A800 GPUs and spent over three days constructing this dataset. Fig. 1 presents examples of Semantic-based Primitive-Mesh, demonstrating that the resulting parts contain meaningful local semantic structures. By integrating these segments with their corresponding textual labels, our proposed MeshLLM significantly enhances performance.

1.2. Metric Details

The evaluation of the generation of 3D mesh can be challenging due to the lack of direct correspondence with ground truth data. Given a set of generated meshes S_g and a set of reference meshes S_r, we follow prior works [1, 3, 4, 6] and define the following metrics:

$$\text{MMD}(S_g, S_r) = \frac{1}{|S_r|} \sum_{Y \in S_r} \min_{X \in S_g} D(X, Y),$$

$$\text{COV}(S_g, S_r) = \frac{|\{\arg\min_{Y \in S_r} D(X, Y) | X \in S_g\}|}{|S_r|},$$

$$\text{1-NNA}(S_g, S_r) = \frac{\sum_{X \in S_g} \mathbb{1}[N_X \in S_g] + \sum_{Y \in S_r} \mathbb{1}[N_Y \in S_r]}{|S_g| + |S_r|},$$

where $D(X, Y)$ is a Chamfer Distance (CD) distance between two meshes X and Y. $\mathbb{1}[N_X \in S_g]$ is a indicator function that returns 1 if N_X belongs to S_g, otherwise 0 and $\mathbb{1}[N_Y \in S_r]$ is a ndicator function that returns 1 if N_Y belongs to S_r, otherwise 0. And N_X in the 1-NNA metric is a point cloud that is closest to X in both the generated and reference dataset, i.e.,

$$N_X = \arg\min_{K \in S_r \cup S_g} D(X, K)$$

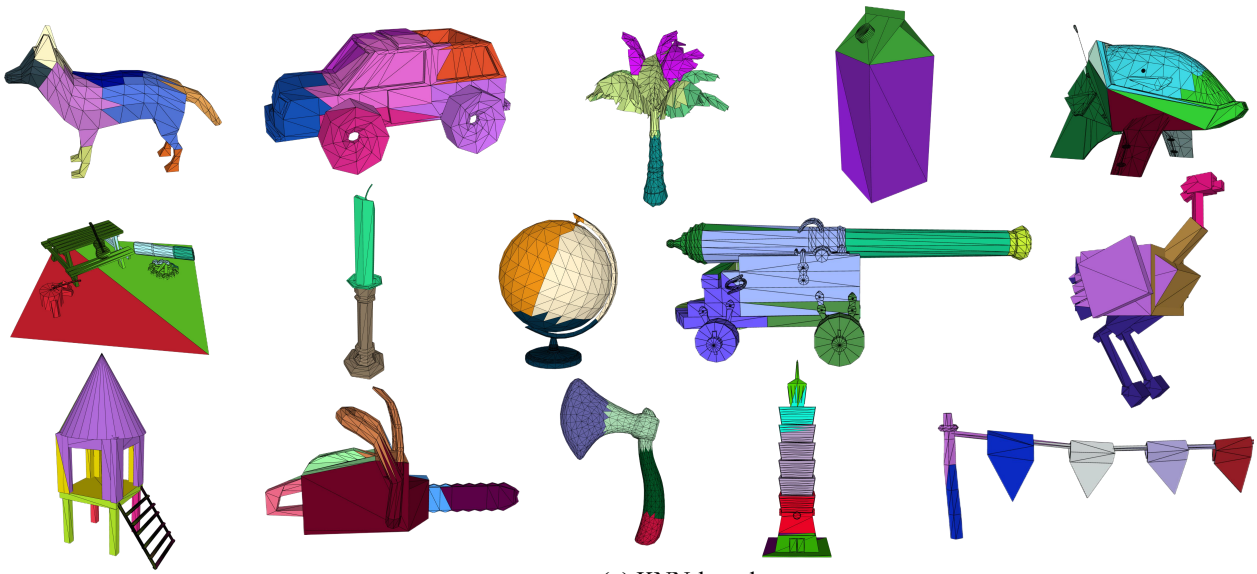
To evaluate point-based measures, we sample 2048 points randomly from all baseline results.

MMD measures the closeness between generated and real meshes by computing the minimum distance from each reference mesh to the generated set. Lower MMD values indicate better shape generation quality as the generated meshes are closer to the real ones. COV measures how well the generated meshes cover the reference set. Higher COV values indicate better diversity in the generated meshes, as more real meshes are matched. 1-NNA evaluates whether the generated and real meshes are evenly distributed. If 1-NNA is close to 50%, the generated meshes are well-mixed with real meshes, indicating good quality.

2. Additional Results

2.1. Shape Novelty Analysis

We compute the Chamfer Distance between samples to identify the three most similar training meshes to the generated meshes for comparison. As shown in Fig. 2, while the overall structure of the generated meshes may resemble examples from the training set, the local details exhibit significant differences. This demonstrates that our model possesses generalization ability and creativity rather than merely replicating training samples.



(a) KNN-based

(b) Semantic-based

Figure 1. **Examples of the constructed Primitive-Mesh**. (a) The KNN-based method is simple and efficient, enabling the rapid construction of large-scale trainable mesh parts while preserving meaningful spatial structures. (b) The Semantic-based method generates mesh parts at the semantic level and includes corresponding textual annotations, which better aid LLMs in accurately understanding and generating meshes.

2.2. Training Strategy Analysis

In MeshLLM, we introduce a progressive training strategy that begins with KNN-based Primitive-Mesh samples, followed by Semantic-based Primitive-Mesh samples, and concludes with training on specific mesh generation and understanding tasks. We further investigate the impact of training order for Primitive-Mesh data. Specifically, we first train MeshLLM on Semantic-based Primitive-Mesh samples and then on KNN-based Primitive-Mesh samples. As shown in Table 1, training on semantic Primitive-Mesh samples later yields better results.

Generally, KNN-based Primitive-Mesh rely primarily on

Figure 2. **Shape novelty**. We compute the Chamfer Distance between the generated meshes and those in the training set, selecting the three closest matches. The notable differences observed among them indicate that MeshLLM exhibits creativity.

Table 1. **Effect of the training order**. MeshLLM$_R$ refers to the reversed training order, where the Semantic-based Primitive-Mesh is trained first, followed by the KNN-based Primitive-Mesh. Pretraining on large-scale data first, followed by fine-tuning on high-quality data, leads to improved model performance.

	COV↑	MMD↓	1-NNA	FID↓	KID↓
MeshLLM$_R$	45.48	**5.64**	63.36	45.77	3.31
MeshLLM	**47.33**	5.72	**60.82**	**42.39**	**2.25**
	BLEU-1↑	CIDEr↑	Meteor↑	ROUGE↑	CLIP↑
MeshLLM$_R$	0.734	1.303	0.435	0.638	0.372
MeshLLM	**0.763**	**1.753**	**0.445**	**0.702**	**0.391**

geometric information for local segmentation. This large-scale preliminary training helps the model learn general geometric features of meshes. Subsequently, finetuning on the more semantically informative Primitive-Mesh samples allows the LLM to refine its understanding and capture detailed semantic distinctions. Conversely, reversing the training order introduces challenges. First, it increases the initial learning difficulty, as the LLM needs to grasp both mesh topology and local semantics simultaneously. Second, when later exposed to 15 times more KNN-based samples, the model may struggle to retain the semantic knowledge learned earlier, which is crucial for downstream mesh generation and understanding, ultimately harming overall performance. These findings suggest that following the typical LLM training paradigm leads to better results for 3D mesh learning, starting with large-scale, diverse data before integrating specialized, high-quality samples. This approach fosters the development of a robust and adaptable model.

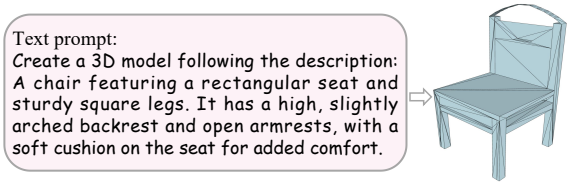


Figure 3. Failure case. The limited semantic dataset size reduces text-geometry alignment for more fine-grained generations.

2.3. Failure Case Analysis

We show a failure case in Fig. 3. Compared to existing language task corpora, mesh datasets remain relatively scarce, resulting in imprecise alignment between textual descriptions and geometric structures, which limits the capability for fine-grained mesh generation. Future work could explore incorporating other modalities (e.g., images) to encode more information and embed it into LLMs, thereby improving performance in detailed mesh generation.

References

[1] Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Billzb Wang, Jingyi Yu, Gang Yu, et al. Meshxl: Neural coordinate field for generative 3d foundation models. *Advances in Neural Information Processing Systems*, 37:97141–97166, 2025. 1

[2] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36:35799–35813, 2023. 1

[3] Ziya Erkoç, Fangchang Ma, Qi Shan, Matthias Nießner, and Angela Dai. Hyperdiffusion: Generating implicit neural fields with weight-space diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 14300–14310, 2023. 1

[4] Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. Polygen: An autoregressive generative model of 3d meshes. In *International Conference on Machine Learning*, pages 7220–7229. PMLR, 2020. 1

[5] Christoph Schuhmann. Improved aesthetic predictor. https://github.com/christophschuhmann/improved-aesthetic-predictor, 2022. 1

[6] 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*, pages 19615–19625, 2024. 1

[7] Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184*, 2024. 1