OCTREES AS AN EXPLICIT 3D LANGUAGE

Anonymous authors
Paper under double-blind review



Figure 1: **OctLLM unifies 3D generation and 3D understanding in a single LLM.** Left: 3D assets generated by OctLLM from text and image conditions. Right: natural-language descriptions it produces for 3D inputs. Bottom right: a shape written as an S-Octree, the coarse-to-fine occupancy sequence OctLLM reads and writes, shown at octree depths 3 to 6. Generation and understanding share this explicit 3D language. Zoom in for a better view.

ABSTRACT

Existing 3D large language models (LLMs) compromise on two fronts: they compress shapes into latent codebook indices or coordinate text, which removes spatial structure from what the model observes, and they acquire the 3D modality by fine-tuning the backbone, which overwrites its general language ability. We present OctLLM, which addresses both limitations. Geometry enters as an explicit 3D sequence of octree occupancy tokens. However, full octree sequences grow rapidly with depth; OctLLM therefore randomly empties penultimate-level nodes and omits descendants while preserving shape, yielding a shorter coordinate- and depth-anchored *Sparse Octree* (S-Octree) for position-aware mask-modeling generation and 3D understanding. On the other front, existing methods introduce a new modality with full fine-tuning or LoRA, but full fine-tuning is costly, LoRA limits 3D capacity, and both modify the language pathway. OctLLM instead adds 3D capacity in parameters separate from the pretrained ones: mesh tokens are routed through independent trainable branches in a subset of blocks while text and image tokens retain the frozen vision-language pathway, and the two streams interact through shared self-attention. It trains far fewer parameters than full fine-tuning, yet sets a new state of the art among unified multimodal LLMs, lowering image-to-3D FID by 17.4% and raising render-grounded captioning by 28.7 points over ShapeLLM-Omni, while matching the backbone on general language benchmarks. Our source code and model weights are available on the project page.

1 INTRODUCTION

Large language models (LLMs) have advanced rapidly, with recent systems showing strong general-purpose reasoning and instruction-following ability across both proprietary and open settings (OpenAI, 2023; Meta AI, 2023; Team GLM, 2024; DeepSeek-AI, 2024). Multimodal extensions have carried these capabilities from text to images and video, and recent systems such as GPT-5 and Qwen3.5-Omni now handle several modalities natively inside a single model (Li et al., 2023; Liu et al., 2023a;

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Wang et al., 2024a; Bai et al., 2025; Li et al., 2024b; Lin et al., 2024; OpenAI, 2025; Qwen Team, 2026b). Extending the same autoregressive paradigm to 3D content, however, remains far less mature. We argue that a 3D LLM must satisfy two requirements at once: 3D geometry should enter the model in an explicit form that preserves its spatial structure, and acquiring this new modality should not erode the general language ability of the underlying model.

Existing 3D LLMs fall short on one or both requirements. The first problem lies in how 3D geometry enters the model. ShapeLLM-Omni (Ye et al., 2025) compresses each shape into VQVAE codebook indices, so the LLM observes only latent codes from which fine geometric detail has already been lost and whose spatial organization is no longer explicit, while LLaMA-Mesh (Wang et al., 2024c) serializes meshes as OBJ text, treating vertex coordinates as ordinary character strings and hiding the locality and hierarchy of the shape; related interfaces built on point-cloud encoders or mesh-derived tokens inherit similar trade-offs (Hong et al., 2023; Xu et al., 2024; Fang et al., 2025). The second problem lies in how the backbone absorbs the new modality. These systems typically adapt the LLM through full-model fine-tuning, which overwrites the pretrained text and vision-language behaviors: as we show in Section 4, both ShapeLLM-Omni and LLaMA-Mesh lose a substantial fraction of their general language ability after 3D training. Existing methods therefore trade away either the spatial structure of the 3D input or the pretrained ability of the language model.

In this paper, we propose OctLLM, a 3D-native multimodal LLM built around two insights. First, an octree can serve as an explicit spatial language for 3D geometry, preserving locality and hierarchy hidden by latent codes or raw mesh text. To handle sequence growth with depth, OctLLM exploits redundant fine-level occupancy by randomly emptying a fraction of second-to-last-level nodes and omitting their finest-level descendants, producing a *Sparse Octree* (S-Octree). Each occupancy token remains anchored to its 3D coordinates and depth, so the LLM can directly model spatial structure when generating and interpreting shapes. With this representation, OctLLM better preserves part structure in generation and produces more accurate 3D descriptions than prior unified multimodal LLMs, including ShapeLLM-Omni (Tables 1 and 2a). Sparsification also reduces training memory and time without sacrificing generation quality (Section D).

Second, introducing 3D demands new capacity without overwriting language ability. Full-parameter fine-tuning is expressive but expensive and exposes every pretrained weight to 3D gradients. LoRA is cheaper but confines 3D learning to a low-rank update that, when merged, still changes the language pathway (Hu et al., 2022). OctLLM instead routes mesh and position-query tokens through full-rank 3D branches at a subset of blocks, training far fewer parameters than full fine-tuning, while text and image tokens stay on the frozen vision-language pathway. Shared self-attention connects the streams; separate mesh embeddings and output head keep occupancy prediction from competing with the language vocabulary. OctLLM reproduces the backbone exactly on MMLU and HellaSwag and stays within one point on GSM8K, whereas ShapeLLM-Omni still loses 34.5 and 40.7 points on the first two despite additional text training data (Table 2b). Our contributions are as follows:

- We introduce OctLLM, which reads and writes S-Octrees as an explicit 3D language, using position-aware masks for generation and direct S-Octree input for understanding.
- We develop a token-routed dual-stream transformer with full-rank 3D branches, mesh embeddings, and a mesh head beside a frozen text-image pathway.
- OctLLM achieves state-of-the-art 3D generation and understanding among unified multimodal LLMs while preserving the backbone's language ability.

2 Related Work

Unified Multimodal Generation and Understanding. Casting generation and understanding as next-token prediction over one discrete sequence works well in 2D (van den Oord et al., 2017; Esser et al., 2021; Sun et al., 2024; Tian et al., 2024; Li et al., 2024a), where unified systems read and write several modalities with one backbone (Chameleon Team, 2024; Wang et al., 2024b; OpenAI, 2024; Xie et al., 2025; Zhou et al., 2025; Liu et al., 2025). Moving to 3D is harder because geometry must first become tokens (Ma et al., 2024), and existing 3D-native systems convert it in ways that discard structure: VQVAE codebook indices (Yin et al., 2023; Chen et al., 2025c; Ye et al., 2025), continuous point features (Qi et al., 2024b), or meshes serialized as coordinate text (Wang et al., 2024c; Fang et al., 2025), with related efforts adding part-level interfaces, reward-driven refinement,

and expert-decoupled designs (Wang et al., 2025; Tang et al., 2025b; Lu et al., 2025; Huang et al., 2026; Yang et al., 2026). None of them lets the LLM operate on an explicit hierarchical 3D structure, and most adapt the backbone by full fine-tuning; OctLLM addresses both limitations.

Autoregressive 3D Generation and Octree Representations. Autoregressive shape generators differ mainly in how they serialize geometry. One line models the mesh itself, from PolyGen (Nash et al., 2020) and MeshGPT (Siddiqui et al., 2024) onward (Chen et al., 2025a;b; 2024; Tang et al., 2025a; Hao et al., 2024; Zhao et al., 2025; Weng et al., 2025; Zhang et al., 2025; Weng et al., 2024): surface fidelity is high, but sequences are long and rely on lossy VQ codebooks. A second line discretizes space hierarchically with octrees (Meagher, 1982), which underlie efficient high-resolution 3D learning (Riegler et al., 2017; Wang et al., 2017; Wang, 2023; Xiong et al., 2025; Ren et al., 2024) and generators such as OctGPT (Wei et al., 2025) and Uni-3DAR (Lu et al., 2025), together with adaptive and cross-scale tokenizations (Deng et al., 2025; Zhang et al., 2024a;b; Qian et al., 2024), where coarse structure and fine detail share one short coarse-to-fine sequence. Latent-diffusion, score-distillation, and feed-forward systems (Zhang et al., 2024c; Xiang et al., 2025; Zeng et al., 2022; Zhang et al., 2023; Liu et al., 2023b; Nichol et al., 2022; Jun & Nichol, 2023; Poole et al., 2023; Lin et al., 2023; Wang et al., 2023; Hong et al., 2024; Tang et al., 2024) produce high-quality assets, but their continuous latents cannot serve as a language that an LLM reads and writes. OctLLM extends this line to a general-purpose LLM, keeping each token's coordinate and depth explicit.

3D Understanding and Language-Preserving Adaptation. Work on 3D understanding progressed from joint shape-text embeddings (Han et al., 2019) to transferring 2D vision-language priors (Radford et al., 2021) onto point clouds (Zhang et al., 2022; Zhu et al., 2023; Xue et al., 2023) with stronger point encoders (Yu et al., 2022; Pang et al., 2022; Guo et al., 2021; Zhao et al., 2021), and 3D MLLMs now connect such encoders, or raw point tokens, to an LLM for dialogue, grounding, and part-aware reasoning (Hong et al., 2023; Xu et al., 2024; Qi et al., 2024a; Yin et al., 2023; Wang et al., 2024c; Fang et al., 2025; Ye et al., 2025; Wang et al., 2025; Paul et al., 2026). How the backbone is adapted has received considerably less attention than what it perceives. Existing 3D LLMs acquire the new modality by updating the pretrained weights themselves, either by full fine-tuning or, where its cost is prohibitive, by low-rank adaptation (Hu et al., 2022). Either way the weights that language tokens depend on are modified, and the pretrained text and vision-language behavior degrades, as Section 4 confirms for existing 3D LLMs. In 2D, modality-decoupled architectures (Liang et al., 2025; Mo et al., 2025) instead route each modality through its own weights while sharing global self-attention; OctLLM brings this principle to 3D with full-rank branches that own their embeddings and output head, so 3D capability is acquired without degrading language ability.

3 METHOD

3.1 OCTREE-BASED 3D TOKENIZATION

OctLLM's pipeline, summarized in Fig. 2, begins by converting a mesh into language-model tokens using an octree. Following OctGPT (Wei et al., 2025), we encode the input mesh as an octree whose finest level matches the target voxel resolution and serialize the occupied 3D space with a Z-order traversal, so nearby 3D nodes tend to remain nearby in the one-dimensional sequence. We then pack every 8 consecutive binary occupancy bits into one byte-level mesh token whose value lies in $[0, 255]$. This reduces the sequence length by a factor of 8 compared with a raw binary sequence, while keeping the representation discrete and faithful to the voxelized octree structure. It also avoids the reconstruction artifacts that can appear when geometry is compressed into learned latent codes.

Although octrees already avoid subdividing empty space, serializing all occupied branches to a fine depth still yields a long sequence, making deeper octrees costly in training time and memory. We further observe that the fine levels are highly redundant: many neighboring nodes have similar occupancy patterns, so randomly omitting a subset leaves the overall 3D structure nearly intact. OctLLM exploits this redundancy by randomly setting a fixed fraction of nodes at the second-to-last level to empty, which also omits their finest-level descendants. We call this additionally thinned representation a *Sparse Octree* (S-Octree), distinguishing it from the octree's inherent empty-space sparsity (See Fig. S6). The removed fraction trades sequence length against the occupancy to be recovered. A completion network later restores the omitted detail, as shown in Fig. 2, so the LLM

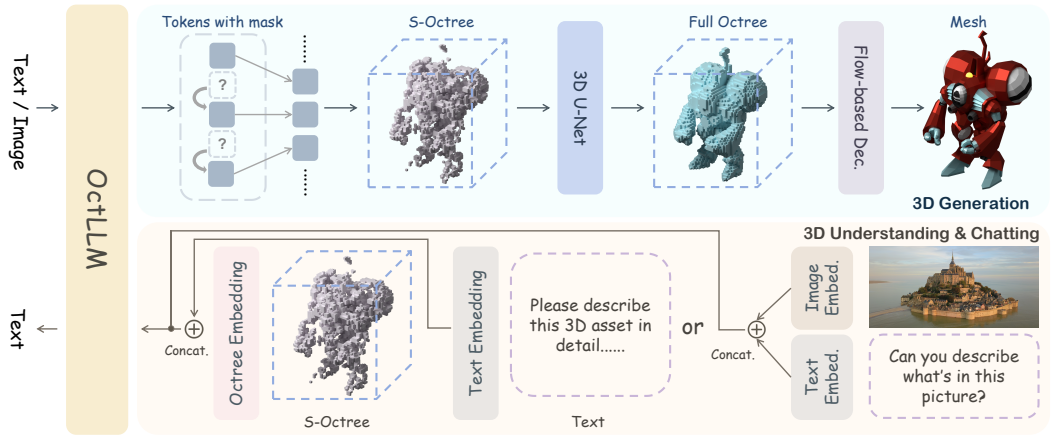


Figure 2: Overview of OctLLM. For generation, OctLLM autoregressively predicts a position-aware S-Octree sequence from a text or image condition, which a 3D U-Net completes into dense occupancy before a flow-based decoder, conditioned on the occupied voxels, produces the mesh. For 3D understanding, the same S-Octree is read with a text prompt as the 3D input language.

focuses on global 3D layout and nontrivial shape parts while a geometry network handles dense local completion. Because the thinned levels grow fastest with depth, the S-Octree shortens the sequence substantially while keeping the coarse structure that determines the object's identity, reducing memory and training time without sacrificing generation quality (Section D).

3.2 3D-Aware Dual-Stream Language Model

Adapting a pretrained LLM to 3D creates a capacity–preservation trade-off. Full-parameter fine-tuning is expressive but costly and exposes pretrained transformations to 3D gradients. LoRA is cheaper but restricts 3D learning to low-rank updates that, when merged, still alter the language pathway (Hu et al., 2022). OctLLM instead routes octree bytes and predictive masks, with their 3D positions and depths, through full-rank trainable branches while text and image tokens remain on the unchanged pathway. Mesh tokens receive independent input and output spaces, and shared self-attention connects the two streams, as shown in Fig. 3.

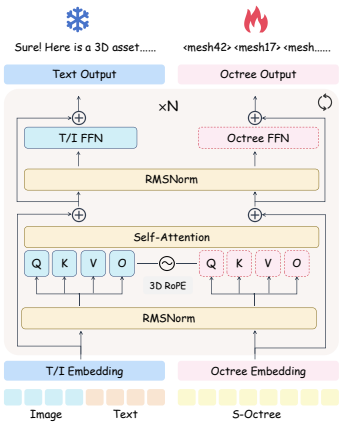
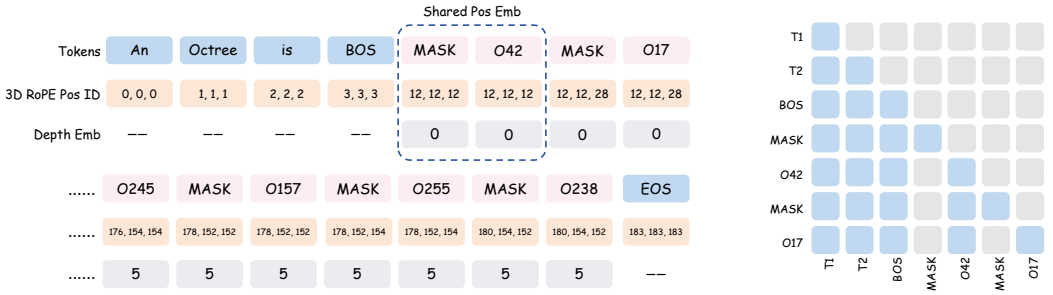


Figure 3: Token-routed dual-stream block. Text and image tokens retain the frozen modules, while S-Octree bytes and predictive <MASK> tokens use trainable 3D embeddings, projections, feed-forward branches, and a dedicated mesh head. Both streams interact through shared causal self-attention, so language, images, and geometry can exchange information.

3D RoPE and Octree Depth Embedding. To let the language model see the spatial layout encoded in the octree sequence, we give every token a three-channel position identifier and apply an independent rotary encoding along each channel, so that a position is a point in a three-dimensional index space rather than a scalar offset. Text and image tokens keep the sequential and raster assignments used by multimodal rotary embeddings (Wang et al., 2024a), but an octree token is not placed by its rank in the sequence: its three channels carry the 3D center coordinates of the node it encodes, rescaled per octree level as in OctGPT (Wei et al., 2025) so that all depths share one coordinate frame. The model therefore sees both the serialized history and the spatial location of each occupancy token, which matters because tokens far apart in the sequence may still be adjacent in 3D space. Coordinates alone, however, do not fully describe an octree token: a shallow node covers a large coarse region while a deep node covers a local surface detail. Following OctGPT (Wei et al., 2025), we therefore add a learnable depth embedding that tells the model the scale at which a token should be interpreted, so the same spatial neighborhood



(a) 3D position and depth assignment.

(b) Attention mask.

Figure 4: Position-aware mask modeling. Each inserted <MASK> token shares the 3D rotary coordinates and depth embedding of the mesh-byte token that follows it, so the hidden state at the mask queries a known 3D location before predicting occupancy. Historical masks are suppressed as attention keys; text, boundary, and previous mesh-byte tokens stay visible.

can be read differently at coarse and fine levels. The left panel of Fig. 4 illustrates how text positions, octree-node coordinates, mask tokens, and mesh-byte tokens are aligned in the expanded sequence.

Predictive Mask Tokens and Mask Suppression. During autoregressive octree generation, the next mesh byte has two meanings: where it is in 3D space and what occupancy pattern it contains, and asking the model to infer both at once makes the sequence harder to learn. Inspired by Uni-3DAR (Lu et al., 2025), OctLLM separates these two questions with a predictive mask token: before predicting a mesh byte, the model receives a <MASK> token carrying the target 3D position and octree depth, whose hidden state acts as a query asking what occupancy should appear at that location given the condition and the geometry generated so far. Because these tokens are position queries rather than geometry observations, historical mask tokens are suppressed in attention while text, image, and previous mesh-byte tokens remain visible under the causal mask, as shown in the right panel of Fig. 4. This keeps the model's context focused on the actual conditioning information and the generated geometry. Details of training and inference with <MASK> token is given in Sections A.1 and A.2.

Token-Routed 3D Transformer Branches. The distribution of mesh bytes is tied to local geometry and has substantially higher entropy than that of natural-language tokens. OctLLM therefore gives mesh-related tokens their own transformer transformations while keeping the pretrained pathway for text and image tokens, as shown in Fig. 3. The two streams still share causal self-attention, so language, images, and geometry can exchange information; only the token-local transformations are routed by token type. Let $r_i \in \{0, 1\}$ denote the route indicator for token i, where $r_i = 1$ for mesh byte tokens and inserted <MASK> tokens, and $r_i = 0$ for text, image/video, <mesh_bos>, and <mesh_eos> tokens. For a routed block, the feed-forward output is computed by token-wise branch selection:

$$\mathrm{FFN}_\ell^{\mathrm{route}}(h_i) = (1 - r_i)\,\mathrm{FFN}_\ell^{\mathrm{base}}(h_i) + r_i\,\mathrm{FFN}_\ell^{\mathrm{3D}}(h_i), \tag{1}$$

where the base FFN is frozen and the 3D FFN is trainable.

The same idea applies to all four attention projections, with Q, K, and V routed on the block input h_i and the output projection routed on the attention result a_i at the same token:

$$\begin{aligned} Q_i &= (1 - r_i)\,W_Q h_i + r_i\,W_Q^{\mathrm{3D}} h_i, & K_i &= (1 - r_i)\,W_K h_i + r_i\,W_K^{\mathrm{3D}} h_i, \\ V_i &= (1 - r_i)\,W_V h_i + r_i\,W_V^{\mathrm{3D}} h_i, & O_i &= (1 - r_i)\,W_O a_i + r_i\,W_O^{\mathrm{3D}} a_i, \end{aligned} \tag{2}$$

where the pretrained projections W_Q, W_K, W_V, W_O are frozen and their 3D counterparts are trainable. The attention operation itself remains a single causal self-attention over the mixed sequence, so only the token-local projection and FFN transformations are separated. The number of routed blocks trades 3D capacity against newly trained parameters, whereas their placement is not arbitrary: the placement study in Section D shows that the 3D stream needs to reach early, middle, and late representations, and that deeper blocks contribute more per branch. We therefore spread the routed blocks uniformly over the backbone and, as the budget grows, spend the additional branches in its second half.

Independent New-Token Embeddings and Output Head. OctLLM adds a small mesh vocabulary for octree bytes, mesh boundaries, and position-query masks. These tokens have a different role from natural-language words, so they receive independent input embeddings and an independent output head while the original vocabulary keeps its pretrained embedding and language head, so 3D occupancy prediction does not compete with ordinary word prediction. A remaining question is how the model should initiate mesh generation, since the pretrained language pathway has no prior ability to emit the newly introduced `<mesh_bos>` token. During instruction tuning we therefore format 3D-generation responses with short natural-language preambles (See the system prompt in Section A.2) that consistently lead into `<mesh_bos>`, so the pretrained stream produces a familiar response pattern while the new head learns the high-probability transition from that pattern into the mesh sequence.

3.3 Sparse Voxel Completion Network

The S-Octree predicted by OctLLM is compact but not yet a complete dense shape, so we decode it into a sparse occupancy grid at the finest octree resolution and use a 3D U-Net to infer the missing fine occupancy from the coarse, partial structure the LLM produced. The encoder downsamples the grid by a factor of eight through four 3D convolutional layers of width 64, 128, 256, and 256, and the decoder mirrors it with four upsampling layers; the network is trained with binary cross-entropy over voxel occupancy. The completed grid then supplies the occupied coordinates that a pretrained flow-based generator takes as its structural condition, together with the original text or image as its semantic condition, to produce the final textured mesh.

3.4 Training Objectives

OctLLM is trained on both 3D generation and 3D understanding data: for generation the model predicts mesh bytes from position-aware mask queries, and for understanding the S-Octree is part of the input and the supervision is the natural-language response. Only content tokens are prediction targets, since mask tokens provide where to predict while mesh bytes or words provide what to predict. Mask tokens are consequently inserted only in generation samples; in understanding samples the S-Octree is already given, so no location needs to be queried and the mesh sequence is fed without them, which also avoids doubling the length of the 3D input.

The loss reflects the same separation between words and mesh bytes: text targets use the language vocabulary distribution and mesh-byte targets use the mesh vocabulary distribution, so occupancy prediction is not diluted by irrelevant word logits. Let $\mathcal{T}_{\text{text}}$ denote supervised text-token positions and $\mathcal{T}_{\text{mesh}}$ denote supervised mesh-byte positions. The training objective is

$$\mathcal{L}_{\text{LLM}} = -\frac{1}{|\mathcal{T}_{\text{text}}| + |\mathcal{T}_{\text{mesh}}|} \left(\sum_{t \in \mathcal{T}_{\text{text}}} \log P_{\text{vocab}}(x_t \mid x_{<t}) + \sum_{t \in \mathcal{T}_{\text{mesh}}} \log P_{\text{mesh}}(x_t \mid x_{<t}) \right), \quad (3)$$

where P_{vocab} is the language vocabulary distribution and P_{mesh} is the mesh-byte distribution. Both generation and understanding data are mixed during training, so the 3D branches learn from mesh-byte prediction while the shared attention pathway also supports 3D-conditioned language responses.

4 Experiments

4.1 Implementation Details

Dataset. We train OctLLM on the union of the Sketchfab subset of Objaverse-XL (Deitke et al., 2023), HSSD (Khanna et al., 2024), ABO (Collins et al., 2022), and ShapeNet (Chang et al., 2015), processed following TRELLIS (Xiang et al., 2025): each asset is rendered from 24 Blender views, one of which becomes its image condition, and four fixed views are captioned by Qwen3.5-9B (Qwen Team, 2026a). Each caption serves as both the text-to-3D prompt and the 3D-to-text reference, so both directions of the 3D language are supervised against the same description. Excluding assets in the PointLLM-200 test set, this gives 195K assets and 585K instructions; the completion network is trained separately on 175K sparse–complete occupancy pairs from the same Objaverse-XL subset.



Figure 5: **Image-to-3D generation.** Each asset is shown from two viewpoints. OctLLM follows the structure and part layout of the conditioning image, while ShapeLLM-Omni and SAR3D produce geometry of lower quality.

Training Details. The S-Octree uses six octree levels, so its finest grid is 64^3, and is sparsified by dropping second-to-last-level nodes with probability 0.5; TRELLIS (Xiang et al., 2025) serves off the shelf as the flow-based decoder of Section 3.3. The text-image pathway is Qwen2.5-VL-7B-Instruct (Bai et al., 2025), whose multimodal rotary embedding (Wang et al., 2024a) already gives every token three position channels, which our 3D coordinates reuse. We route 3D branches through 12 of its 28 decoder blocks, at the 0-indexed layers $\{0, 4, 8, 12, 14, 16, 18, 20, 22, 24, 26, 27\}$; this adds about 2.80B trainable parameters, roughly a quarter of the total and about a third of what full fine-tuning updates. Section D motivates this layout and the 8-branch ablation variant. Each branch is initialized from the block it accompanies, which stabilizes early training. We freeze the language backbone and the vision encoder and train only the routed 3D branches, the mesh embeddings and mesh head, and the octree depth encoding; the mesh vocabulary adds 256 occupancy-byte tokens plus `<mesh_bos>`, `<mesh_eos>`, and `<MASK>` in their own embedding space. Training runs for 69K steps on 16 H20 GPUs and the completion network for 155K steps on 8 RTX 5090 GPUs, both with AdamW at an effective batch size of 64; Section B lists the remaining optimization settings.

4.2 Quantitative Comparison

Evaluation Protocol. Generation is scored on 1K held-out Toys4K assets (Stojanov et al., 2021) by render-based FID and KID with Inception-V3 (Szegedy et al., 2016) and DINOv2 (Oquab et al., 2024) features plus CLIP (Radford et al., 2021) condition alignment, with untextured baseline outputs textured by the same tool. Understanding is evaluated on the PointLLM-200 captioning benchmark (Xu et al., 2024). Following PointLLM, we report Sentence-BERT and SimCSE similarity plus two Qwen3.8-Max judges (Qwen Team, 2026c): GPT-ref compares a caption against the human reference, and GPT-img withholds it, scoring the caption against four renders of the object. n-gram metrics are reported only in Section B, since PointLLM shows them to be unreliable for 3D captioning. All baselines are evaluated with their publicly released model weights and code, and every test asset is converted into the 3D input format that the corresponding method expects. Throughout we keep two families of baselines apart: *task-specific* models trained for one task with one model per modality, and *multimodal LLMs* carrying generation, understanding, and language in one backbone.

3D Generation Quality. As reported in Table 1, OctLLM outperforms all multimodal LLMs on image-conditioned generation, reducing Inception FID by 17.4% and KID by 45% relative to ShapeLLM-Omni. Under text conditioning, OctLLM leads the same family on Inception FID and both KID scores, where it also surpasses the task-specific octree generator OctGPT, and performs comparably to ShapeLLM-Omni on DINOv2 FID and CLIP. TRELLIS attains the best scores overall,

Table 1: **Image-to-3D and text-to-3D generation on Toys4K.** FID/KID are computed on rendered views, with KID scaled by $\times 100$, and CLIP measures condition alignment. Each modality block lists only the methods supporting that conditioning, split into task-specific 3D generators and multimodal LLMs; within a block, the best and second-best result per column are highlighted.

Method	FID (Inc.) ↓	KID (Inc.) ↓	FID (DINOv2) ↓	KID (DINOv2) ↓	CLIP ↑
Image-to-3D					
Task-specific 3D generators					
TRELLIS (Xiang et al., 2025)	18.15	0.13	48.47	2.06	92.86
SAR3D (Chen et al., 2025c)	67.42	3.16	206.79	20.56	84.46
Multimodal LLMs					
ShapeLLM-Omni (Ye et al., 2025)	38.21	1.02	145.14	9.98	86.75
OctLLM (Ours)	31.58	0.56	135.58	7.94	87.31
Text-to-3D					
Task-specific 3D generators					
TRELLIS (Xiang et al., 2025)	30.90	0.34	113.16	3.54	24.71
SAR3D (Chen et al., 2025c)	86.87	4.51	296.74	29.26	18.70
OctGPT (Wei et al., 2025)	46.40	1.13	228.81	15.92	19.70
Multimodal LLMs					
ShapeLLM-Omni (Ye et al., 2025)	52.36	1.38	169.28	10.34	22.46
LLaMA-Mesh (Wang et al., 2024c)	66.68	2.78	275.94	23.62	17.90
AR3D-R1 (Tang et al., 2025b)	57.65	1.78	187.33	12.95	21.89
OctLLM (Ours)	45.61	0.98	175.58	9.01	22.30

Table 2: **3D understanding and language ability.** (a) Captioning on the 200-object PointLLM benchmark, judged by Qwen3.8-Max; the rule separates task-specific understanding models from multimodal LLMs, and the best and second-best result per column are highlighted; Table S2 adds PointLLM-13B and n-gram metrics. (b) General language benchmarks under greedy decoding, with the backbone in gray as a reference rather than a competitor and the best result per column highlighted. Higher is better throughout.

(a) 3D understanding on PointLLM-200.

Method	S-BERT ↑	SimCSE ↑	GPT-ref ↑	GPT-img ↑
PointLLM-7B	48.05	48.63	46.06	45.63
ShapeLLM-7B	37.27	37.10	15.92	30.23
LLaMA-Mesh	30.44	31.08	15.60	17.16
ShapeLLM-Omni	36.08	34.28	22.27	18.14
OctLLM (Ours)	38.80	38.48	26.61	46.88

(b) Language ability preservation.

Method	MMLU ↑	HellaSwag ↑	GSM8K ↑	IFEval ↑
Qwen2.5-VL-7B	64.65	67.44	83.54	75.41
ShapeLLM-Omni	30.17	26.74	27.45	58.04
LLaMA-Mesh	51.84	34.03	52.69	66.54
OctLLM (Ours)	64.65	67.44	83.01	69.87

and OctLLM ranks second on most of the metrics. This gap is expected: TRELLIS is a dedicated 3D generator, whereas OctLLM produces geometry from a frozen-backbone LLM that also describes shapes and retains its language ability. Notably, ShapeLLM-Omni and AR3D-R1 likewise use TRELLIS as the flow-based mesh decoder, so the gain of OctLLM over them comes from our new design of 3D representation and pipeline, which produces higher-quality voxels for TRELLIS.

3D Understanding Quality. Among multimodal LLMs, OctLLM leads on every metric of Table 2a, by more than 28 points under the render-grounded judge, since it reads the coordinate-anchored occupancy sequence which carries explicit 3D information. Across all methods, OctLLM obtains the best GPT-img score and ranks second on the other three metrics, outperforming ShapeLLM-7B throughout and trailing only PointLLM-7B under GPT-ref and the embedding scores. The remaining gap reflects what each input carries: PointLLM reads colored point clouds and can therefore report material and color, whereas the S-Octree encodes geometry alone, so OctLLM describes shape and part structure and says nothing about appearance.

Language Ability Preservation. OctLLM reproduces the backbone exactly on both multiple-choice benchmarks and differs by less than one point on GSM8K, whereas the baselines lose between a third and a half of its accuracy (Table 2b). This follows from the architecture rather than from data balancing: the pretrained parameters are frozen and text tokens never enter a 3D branch, so a text-only conversation reproduces the backbone's computation exactly. The residual drop on IFEval



Figure 6: **Text-to-3D generation.** Prompts are truncated for space and describe how parts are composed; Section C.1 provides the complete conditions. "N/A" marks a prompt for which LLaMA-Mesh returned no valid mesh.

is caused by the mesh head occasionally opening a 3D sequence in the middle of an answer, and is eliminated by masking the mesh vocabulary at inference time.

4.3 Qualitative Comparison

Image-to-3D Generation. Figure 5 compares OctLLM with ShapeLLM-Omni, TRELLIS, and SAR3D on four held-out assets, each rendered from two viewpoints. SAR3D recovers only a coarse silhouette, and its noisy surfaces lose the finer parts of each object. ShapeLLM-Omni produces cleaner surfaces but still does not preserve part-level structure. OctLLM reproduces both the global layout and the individual parts of the conditioning image, and is visually closest to TRELLIS, which also serves as the structure-conditioned decoder of our pipeline.

Text-to-3D Generation. Figure 6 compares OctLLM with ShapeLLM-Omni, AR3D-R1, and LLaMA-Mesh on prompts that name parts and state how they are composed. LLaMA-Mesh returns no valid mesh for the stringed instrument, and ShapeLLM-Omni and AR3D-R1 recover a recognizable object category but follow the part composition stated in the prompt less closely than OctLLM. OctLLM matches these constraints most closely, producing two stacked wings joined by struts, four legs with a long tail, and a rectangular basin on a cylindrical column. With geometry as an explicit 3D sequence, OctLLM controls finer structure and better matches the text prompt.

3D Understanding. Figure S2 in the appendix shows that OctLLM describes 3D inputs in more detail and with fewer errors than the compared methods.

5 Conclusion

We presented OctLLM, a multimodal LLM that treats a sparse octree as an explicit 3D language: geometry enters and leaves the model as occupancy bytes anchored to 3D coordinates and octree depths, routed through trainable branches placed beside a frozen text-image pathway. Our experiments support both design choices. Keeping 3D structure explicit rather than compressed yields the best generation and 3D understanding among unified multimodal LLMs, while adding 3D capacity in separate parameters reproduces the backbone exactly on general language benchmarks, where 3D LLMs that update the pretrained weights lose a third to a half of their accuracy.

AI USE STATEMENT

In this work, we used GPT-5.6 Sol to assist with drafting and language editing, to identify and summarize relevant literature for the related-work section, and to implement portions of the codebase. We used locally deployed Qwen3.5-9B to generate captions that form part of the training data, as described in Section 4 and Section B.1, and Qwen3.8-Max through Alibaba's official API as an automated evaluation judge (Section B). We did not use generative AI to formulate the core research idea or mathematical claims, design or provide feedback on the methodology or experiments, or interpret the results and draw conclusions; proof-related tasks are not applicable to this work. The authors reviewed all AI-assisted text, literature references, code, and generated data, and checked the code and data for correctness and suitability for the reported experiments. We take full responsibility for all content of this work, including AI-assisted text, claims, code, and data.

REPRODUCIBILITY STATEMENT

The main paper and appendix provide the information needed to reproduce our results. Section 3 specifies the representation, architecture, and training objectives, while Section 4 describes the datasets, training setup, and evaluation protocol. Section A and Section B further detail the end-to-end training and inference procedures, data preprocessing and caption annotation, optimization settings, rendering protocol, and metric computation. We have released the source code and model weights in anonymized form for the review process. The release includes all scripts needed to reproduce the reported experiments and results, together with instructions for using the code and carrying out the complete reproduction workflow. Both are accessible from our anonymous project page: `https://anonymous-octllm.github.io/OctLLM-project-page/`.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025. URL `https://arxiv.org/abs/2502.13923`.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024. URL `https://arxiv.org/abs/2405.09818`.

Angel X. Chang, Thomas Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015. URL `https://arxiv.org/abs/1512.03012`.

Sijin Chen, Xin Chen, Anqi Pang, Xianfang Zeng, Wei Cheng, Yijun Fu, Fukun Yin, Zhibin Wang, Jingyi Yu, Gang Yu, Bin Fu, and Tao Chen. MeshXL: Neural coordinate field for generative 3D foundation models. In *NeurIPS*, 2024.

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, Guosheng Lin, and Chi Zhang. MeshAnything: Artist-created mesh generation with autoregressive transformers. In *ICLR*, 2025a.

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. In *ICCV*, 2025b.

Yongwei Chen, Yushi Lan, Shangchen Zhou, Tengfei Wang, and Xingang Pan. SAR3D: Autoregressive 3D object generation and understanding via multi-scale 3D VQVAE. In *CVPR*, pp. 28371–28382, 2025c. doi: 10.1109/CVPR52734.2025.02642.

Jasmine Collins, Shubham Goel, Kenan Deng, Achleshwar Luthra, Leon Xu, Erhan Gundogdu, Xi Zhang, Tomas F. Yago Vicente, Thomas Dideriksen, Himanshu Arora, Matthieu Guillaumin, and Jitendra Malik. ABO: Dataset and benchmarks for real-world 3D object understanding. In *CVPR*, 2022.

DeepSeek-AI. DeepSeek-V3 technical report. *arXiv preprint arXiv:2412.19437*, 2024. URL https://arxiv.org/abs/2412.19437.

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. In *NeurIPS*, 2023.

Kangle Deng, Hsueh-Ti Derek Liu, Yiheng Zhu, Xiaoxia Sun, Chong Shang, Kiran S. Bhat, Deva Ramanan, Jun-Yan Zhu, Maneesh Agrawala, and Tinghui Zhou. Efficient autoregressive shape generation via octree-based adaptive tokenization. In *ICCV*, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *EMNLP*, 2023.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021.

Shuangkang Fang, I-Chao Shen, Yufeng Wang, Yi-Hsuan Tsai, Yi Yang, Shuchang Zhou, Wenrui Ding, Takeo Igarashi, and Ming-Hsuan Yang. MeshLLM: Empowering large language models to progressively understand and generate 3D mesh. In *ICCV*, pp. 14061–14072, 2025.

Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. PCT: Point cloud transformer. *Comput. Vis. Media*, 7(2), 2021.

Zhizhong Han, Mingyang Shang, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Y^2Seq2Seq: Cross-modal representation learning for 3D shape and text by joint reconstruction and prediction of view and word sequences. In *AAAI*, 2019.

Zekun Hao, David W. Romero, Tsung-Yi Lin, and Ming-Yu Liu. Meshtron: High-fidelity, artist-like 3D mesh generation at scale. *arXiv preprint arXiv:2412.09548*, 2024. URL https://arxiv.org/abs/2412.09548.

Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large reconstruction model for single image to 3D. In *ICLR*, 2024.

Yining Hong, Haoyu Zhen, Peihao Chen, Shuhong Zheng, Yilun Du, Zhenfang Chen, and Chuang Gan. 3D-LLM: Injecting the 3D world into large language models. In *NeurIPS*, 2023.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Peng Huang, Yifeng Chen, Zeyu Zhang, and Hao Tang. UniMesh: Unifying 3D mesh understanding and generation. *arXiv preprint arXiv:2604.17472*, 2026. URL https://arxiv.org/abs/2604.17472.

Heewoo Jun and Alex Nichol. Shap-E: Generating conditional 3D implicit functions. *arXiv preprint arXiv:2305.02463*, 2023. URL https://arxiv.org/abs/2305.02463.

Mukul Khanna, Yongsen Mao, Hanxiao Jiang, Sanjay Haresh, Brennan Shacklett, Dhruv Batra, Alexander Clegg, Eric Undersander, Angel X. Chang, and Manolis Savva. Habitat synthetic scenes dataset (HSSD-200): An analysis of 3D scene scale and realism tradeoffs for ObjectGoal navigation. In *CVPR*, 2024.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, 2023.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *NeurIPS*, 2024a.

Yanwei Li, Chengyao Wang, and Jiaya Jia. LLaMA-VID: An image is worth 2 tokens in large language models. In *ECCV*, 2024b.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, and Xi Victoria Lin. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *Transactions on Machine Learning Research*, 2025.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-LLaVA: Learning united visual representation by alignment before projection. In *EMNLP*, 2024.

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*, 2023.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. World model on million-length video and language with blockwise RingAttention. In *ICLR*, 2025.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023a.

Zhen Liu, Yao Feng, Michael J. Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. MeshDiffusion: Score-based generative 3D mesh modeling. In *ICLR*, 2023b.

Shuqi Lu, Haowei Lin, Lin Yao, Zhifeng Gao, Xiaohong Ji, Yitao Liang, Weinan E, Linfeng Zhang, and Guolin Ke. Unified cross-scale 3D generation and understanding via autoregressive modeling. *arXiv preprint arXiv:2503.16278*, 2025. URL https://arxiv.org/abs/2503.16278.

Xianzheng Ma, Brandon Smart, Yash Bhalgat, Shuai Chen, Xinghui Li, Jian Ding, Jindong Gu, Dave Zhenyu Chen, Songyou Peng, Jia-Wang Bian, Philip H. Torr, Marc Pollefeys, Matthias Nießner, Ian D. Reid, Angel X. Chang, Iro Laina, and Victor Adrian Prisacariu. When LLMs step into the 3D world: A survey and meta-analysis of 3D tasks via multi-modal large language models. *arXiv preprint arXiv:2405.10255*, 2024. URL https://arxiv.org/abs/2405.10255.

Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982. doi: 10.1016/0146-664X(82)90104-6.

Meta AI. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. URL https://arxiv.org/abs/2307.09288.

Sicheng Mo, Thao Nguyen, Xun Huang, Siddharth Srinivasan Iyer, Yijun Li, Yuchen Liu, Abhishek Tandon, Eli Shechtman, Krishna Kumar Singh, Yong Jae Lee, Bolei Zhou, and Yuheng Li. X-Fusion: Introducing new modality to frozen large language models. In *ICCV*, pp. 228–238, 2025.

Charlie Nash, Yaroslav Ganin, S. M. Ali Eslami, and Peter Battaglia. PolyGen: An autoregressive generative model of 3D meshes. In *ICML*, 2020.

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. URL https://arxiv.org/abs/2212.08751.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. URL https://arxiv.org/abs/2303.08774.

OpenAI. GPT-4o system card. *arXiv preprint arXiv:2410.21276*, 2024. URL https://arxiv.org/abs/2410.21276.

OpenAI. OpenAI GPT-5 system card. *arXiv preprint arXiv:2601.03267*, 2025. URL https://arxiv.org/abs/2601.03267.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024.

Yatian Pang, Wenxiao Wang, Francis E. H. Tay, Wei Liu, Yonghong Tian, and Li Yuan. Masked autoencoders for point cloud self-supervised learning. In *ECCV*, 2022.

Sneha Paul, Zachary Patterson, and Nizar Bouguila. Point cloud as a foreign language for multi-modal large language model. In *CVPR*, 2026.

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *ICLR*, 2023.

Zekun Qi, Runpei Dong, Shaochen Zhang, Haoran Geng, Chunrui Han, Zheng Ge, Li Yi, and Kaisheng Ma. ShapeLLM: Universal 3D object understanding for embodied interaction. In *ECCV*, 2024a.

Zhangyang Qi, Ye Fang, Zeyi Sun, Xiaoyang Wu, Tong Wu, Jiaqi Wang, Dahua Lin, and Hengshuang Zhao. GPT4Point: A unified framework for point-language understanding and generation. In *CVPR*, 2024b.

Xuelin Qian, Yu Wang, Simian Luo, Yinda Zhang, Ying Tai, Zhenyu Zhang, Chengjie Wang, Xiangyang Xue, Bo Zhao, Tiejun Huang, Yunsheng Wu, and Yanwei Fu. Pushing auto-regressive models for 3D shape generation at capacity and scalability. *arXiv preprint arXiv:2402.12225*, 2024. URL https://arxiv.org/abs/2402.12225.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026a. URL https://qwen.ai/blog?id=qwen3.5.

Qwen Team. Qwen3.5-Omni technical report. *arXiv preprint arXiv:2604.15804*, 2026b. URL https://arxiv.org/abs/2604.15804.

Qwen Team. Qwen3.8-Max. https://chat.qwen.ai/legal-agreement/models, 2026c. Official model description. Accessed September 9, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

Xuanchi Ren, Jiahui Huang, Xiaohui Zeng, Ken Museth, Sanja Fidler, and Francis Williams. XCube: Large-scale 3D generative modeling using sparse voxel hierarchies. In *CVPR*, 2024.

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *CVPR*, 2017.

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. MeshGPT: Generating triangle meshes with decoder-only transformers. In *CVPR*, 2024.

Stefan Stojanov, Anh Thai, and James M. Rehg. Using shape to categorize: Low-shot learning with an explicit shape bias. In *CVPR*, pp. 1798–1808, 2021. doi: 10.1109/CVPR46437.2021.00184.

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. URL https://arxiv.org/abs/2406.06525.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

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. In *ECCV*, 2024.

Jiaxiang Tang, Zhaoshuo Li, Zekun Hao, Xian Liu, Gang Zeng, Ming-Yu Liu, and Qinsheng Zhang. EdgeRunner: Auto-regressive auto-encoder for artistic mesh generation. In *ICLR*, 2025a.

Yiwen Tang, Zoey Guo, Kaixin Zhu, Ray Zhang, Qizhi Chen, Dongzhi Jiang, Junli Liu, Bohan Zeng, Haoming Song, Delin Qu, Tianyi Bai, Dan Xu, Wentao Zhang, and Bin Zhao. Are we ready for RL in text-to-3D generation? a progressive investigation. *arXiv preprint arXiv:2512.10949*, 2025b. URL https://arxiv.org/abs/2512.10949.

Team GLM. ChatGLM: A family of large language models from GLM-130B to GLM-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024. URL https://arxiv.org/abs/2406.12793.

Tencent Hunyuan3D Team. Hunyuan3D 2.1: From images to high-fidelity 3D assets with production-ready PBR material. *arXiv preprint arXiv:2506.15442*, 2025. URL https://arxiv.org/abs/2506.15442.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *NeurIPS*, 2024.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *NeurIPS*, 2017.

Chunshi Wang, Junliang Ye, Yunhan Yang, Yang Li, Zizhuo Lin, Jun Zhu, Zhuo Chen, Yawei Luo, and Chunchao Guo. Part-X-MLLM: Part-aware 3D multimodal large language model. *arXiv preprint arXiv:2511.13647*, 2025. URL https://arxiv.org/abs/2511.13647.

Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024a. URL https://arxiv.org/abs/2409.12191.

Peng-Shuai Wang. OctFormer: Octree-based transformers for 3D point clouds. *ACM Trans. Graph. (SIGGRAPH)*, 42(4), 2023. doi: 10.1145/3592131.

Peng-Shuai Wang, Yang Liu, Yu-Xiao Guo, Chun-Yu Sun, and Xin Tong. O-CNN: Octree-based convolutional neural networks for 3D shape analysis. *ACM Trans. Graph. (SIGGRAPH)*, 36(4), 2017. doi: 10.1145/3072959.3073608.

Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, Yingli Zhao, Yulong Ao, Xuebin Min, Tao Li, Boya Wu, Bo Zhao, Bowen Zhang, Liangdong Wang, Guang Liu, Zheqi He, Xi Yang, Jingjing Liu, Yonghua Lin, Tiejun Huang, and Zhongyuan Wang. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024b. URL https://arxiv.org/abs/2409.18869.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. ProlificDreamer: High-fidelity and diverse text-to-3D generation with variational score distillation. In *NeurIPS*, 2023.

Zhengyi Wang, Jonathan Lorraine, Yikai Wang, Hang Su, Jun Zhu, Sanja Fidler, and Xiaohui Zeng. LLaMA-Mesh: Unifying 3D mesh generation with language models. *arXiv preprint arXiv:2411.09595*, 2024c. URL https://arxiv.org/abs/2411.09595.

Si-Tong Wei, Rui-Huan Wang, Chuan-Zhi Zhou, Baoquan Chen, and Peng-Shuai Wang. OctGPT: Octree-based multiscale autoregressive models for 3D shape generation. In *ACM SIGGRAPH 2025 Conference Papers*, 2025. doi: 10.1145/3721238.3730601.

Haohan Weng, Zibo Zhao, Biwen Lei, Xianghui Yang, Jian Liu, Zeqiang Lai, Zhuo Chen, Yuhong Liu, Jie Jiang, Chunchao Guo, Tong Zhang, Shenghua Gao, and C. L. Philip Chen. Scaling mesh generation via compressive tokenization. *arXiv preprint arXiv:2411.07025*, 2024. URL https://arxiv.org/abs/2411.07025.

Haohan Weng, Yikai Wang, Tong Zhang, C. L. Philip Chen, and Jun Zhu. PivotMesh: Generic 3D mesh generation via pivot vertices guidance. In *ICLR*, 2025.

Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *CVPR*, 2025.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *ICLR*, 2025.

Bojun Xiong, Si-Tong Wei, Xin-Yang Zheng, Yan-Pei Cao, Zhouhui Lian, and Peng-Shuai Wang. OctFusion: Octree-based diffusion models for 3D shape generation. *Comput. Graph. Forum*, 2025. doi: 10.1111/cgf.70198.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. PointLLM: Empowering large language models to understand point clouds. In *ECCV*, 2024.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. ULIP: Learning a unified representation of language, images, and point clouds for 3D understanding. In *CVPR*, 2023.

Zongyuan Yang, Mingjing Yi, Wanli Ma, Chenzhuo Fan, Bocheng Li, Baolin Liu, Yuke Lou, Yingde Song, Yongping Xiong, Zhengdong Guo, and Shimu Wang. EVA01: Unified native 3D understanding and generation via mixture-of-transformers. *arXiv preprint arXiv:2605.16745*, 2026. URL https://arxiv.org/abs/2605.16745.

Junliang Ye, Zhengyi Wang, Ruowen Zhao, Shenghao Xie, and Jun Zhu. ShapeLLM-Omni: A native multimodal LLM for 3D generation and understanding. In *NeurIPS*, 2025.

Fukun Yin, Xin Chen, Chi Zhang, Biao Jiang, Zibo Zhao, Jiayuan Fan, Gang Yu, Taihao Li, and Tao Chen. ShapeGPT: 3D shape generation with a unified multi-modal language model. *arXiv preprint arXiv:2311.17618*, 2023. URL https://arxiv.org/abs/2311.17618.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In *CVPR*, 2022.

Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent point diffusion models for 3D shape generation. In *NeurIPS*, 2022.

Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Trans. Graph. (SIGGRAPH)*, 2023.

Jinzhi Zhang, Feng Xiong, and Mu Xu. G3PT: Unleash the power of autoregressive modeling in 3D generation via cross-scale querying transformer. *arXiv preprint arXiv:2409.06322*, 2024a. URL https://arxiv.org/abs/2409.06322.

Jinzhi Zhang, Feng Xiong, and Mu Xu. 3D representation in 512-byte: Variational tokenizer is the key for autoregressive 3D generation. *arXiv preprint arXiv:2412.02202*, 2024b. URL https://arxiv.org/abs/2412.02202.

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. *ACM Trans. Graph. (SIGGRAPH)*, 43(4), 2024c.

Renrui Zhang, Ziyu Guo, Wei Zhang, Kunchang Li, Xupeng Miao, Bin Cui, Yu Qiao, Peng Gao, and Hongsheng Li. PointCLIP: Point cloud understanding by CLIP. In *CVPR*, 2022.

Xiang Zhang, Yawar Siddiqui, Armen Avetisyan, Chris Xie, Jakob Engel, and Henry Howard-Jenkins. VertexRegen: Mesh generation with continuous level of detail. In *ICCV*, 2025.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021.

Ruowen Zhao, Junliang Ye, Zhengyi Wang, Guangce Liu, Yiwen Chen, Yikai Wang, and Jun Zhu. DeepMesh: Auto-regressive artist-mesh creation with reinforcement learning. In *ICCV*, 2025.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *ICLR*, 2025.

Xiangyang Zhu, Renrui Zhang, Bowei He, Ziyu Guo, Ziyao Zeng, Zipeng Qin, Shanghang Zhang, and Peng Gao. PointCLIP V2: Prompting CLIP and GPT for powerful 3D open-world learning. In *ICCV*, 2023.

APPENDIX – OCTREES AS AN EXPLICIT 3D LANGUAGE

This supplement is organized as follows. Section A details the training and inference procedures, Section B the experimental setup and evaluation protocol, and Section C additional qualitative results, the judge prompts, and an inference-efficiency comparison. Section D isolates the contribution of each design choice, and Section E discusses the remaining limitations and future directions.

A TRAINING AND INFERENCE PROCEDURES

This section makes the end-to-end procedures behind Sections 3.3 and 3.4 explicit. We use c for a text or image condition, M for a mesh, S for its serialized S-Octree, and G and G_S for the complete and sparse occupancy grids, respectively. The language model and the completion network are optimized independently: the former learns the shared 3D language used for generation and understanding, whereas the latter only learns to restore occupancy removed by S-Octree sparsification.

A.1 TRAINING PROCEDURE

For a generation sample, we construct the target S-Octree by voxelizing the mesh, pruning nodes at the designated fine level, traversing the remaining octree in Z-order, and packing every eight occupancy bits into one mesh-byte token. This sparsification is performed once, when the instruction-tuning dataset is built: the random node emptying is drawn a single time per asset and the resulting S-Octree is stored, so each mesh has one fixed byte sequence throughout training and Eq. (3) is applied to a deterministic target. Teacher forcing inserts a <MASK> query immediately before every target mesh byte. The query is assigned the position and depth of that byte, and historical mask queries are suppressed as attention keys. Supervision is applied to the mesh-byte targets through the mesh head. For an understanding sample, the S-Octree is instead part of the input, without inserted mask queries, and only the natural-language answer is supervised through the language head. Both sample types follow the objective in Eq. (3); Algorithm S1 summarizes this procedure together with the separate training of the completion network on paired sparse and complete occupancy grids.

Algorithm S1 Training OctLLM and the sparse voxel completion network.

Require: Pretrained text–image parameters θ_0; trainable 3D parameters θ_{3D}; completion network U_ϕ; generation and understanding datasets $\mathcal{D}_{\text{gen}}$ and $\mathcal{D}_{\text{und}}$
Ensure: Trained θ_{3D} and ϕ
Precompute $S \leftarrow \text{Serialize}(\text{Sparsify}(\text{Octree}(M)))$ for every mesh M ▷ once, before training
Freeze θ_0
for each mixed minibatch $B \subset \mathcal{D}_{\text{gen}} \cup \mathcal{D}_{\text{und}}$ **do**
  for each sample in B with its stored S-Octree S **do**
    if the sample is $(c, S) \in \mathcal{D}_{\text{gen}}$ **then**
      Insert a position-aware <MASK> before every byte in S
      Supervise the S-Octree bytes conditioned on c
    else
      Provide S without mask queries and supervise the text answer
    end if
  end for
  Update θ_{3D} with $\mathcal{L}_{\text{LLM}}$ in Eq. (3); keep θ_0 fixed
end for
for each mesh M in the completion dataset **do**
  $G \leftarrow \text{Voxelize}(\text{Octree}(M))$
  $G_S \leftarrow \text{Voxelize}(\text{Sparsify}(\text{Octree}(M)))$
  Update ϕ with $\mathcal{L}_{\text{comp}} = \text{BCE}(U_\phi(G_S), G)$
end for

A.2 GENERATION PROCEDURE

System prompt. OctLLM uses the following system prompt for both generation and understanding:

```
You are a helpful assistant specialized in 3D asset generation and
understanding, image understanding and chatting. When you are asked
to generate a 3D asset from an image, you should reply in formats
like: "I've produced a 3D model based on the image: ", "Based on the
image, here's the 3D mesh asset I've created: ", etc. And if you are
asked to generate a 3D asset from a text description, you should
reply in formats like: "I've produced a 3D mesh asset based on your
description: ", "Of course! I've generated a 3D mesh based on your
text prompt: ", etc. If you are asked to describe a 3D asset, just
describe it in detail. The text can be varied, but the colon ":"
must be included.
```

The prompt establishes a single assistant role spanning 3D generation, 3D understanding, image understanding, and ordinary dialogue. For generation, it also standardizes the natural-language preamble to end in a colon, matching the instruction-tuning format and creating a reliable transition to <mesh_bos>, which starts constrained octree decoding. Without this task and output-format contract, the backbone may answer conversationally or terminate before emitting the mesh boundary, leaving no dependable handoff to the structured generation loop.

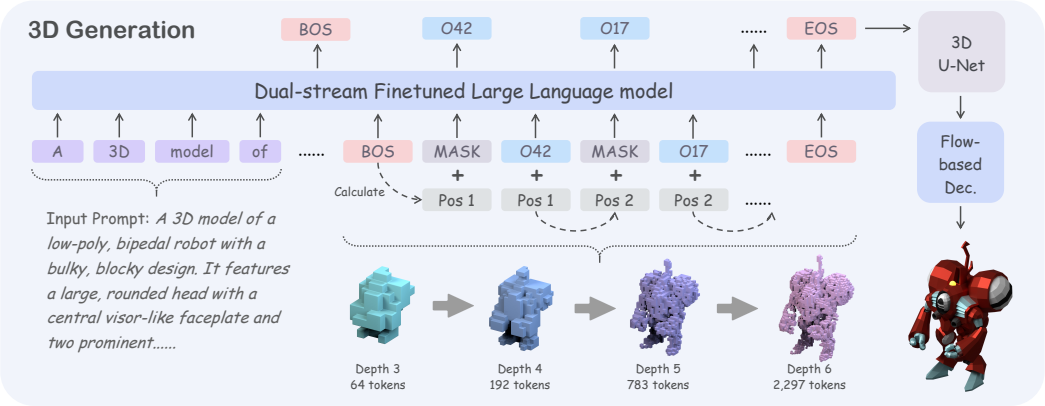


Figure S1: Text-conditioned generation at inference time. The prompt is answered with a short natural-language preamble that leads into <mesh_bos>, after which each mesh byte is produced by first attaching the next node's 3D position to a transient <MASK> query and then sampling only the occupancy value at that position. The partial octree grows coarse to fine, and the completed grid is passed to the completion network and the flow-based decoder.

Figure S1 illustrates the loop described below. At inference time, the partially generated octree uniquely determines the next node in the coarse-to-fine Z-order traversal. OctLLM therefore does not need to generate a coordinate token: it attaches the next node's 3D position and depth to a transient <MASK> query and samples only its occupancy byte. Sampling is logit-masked to the 256 valid occupancy bytes, so the model cannot emit <mesh_eos>, <MASK>, or a word token in the middle of the traversal. Once the byte is sampled, the key–value entries of the mask query are physically removed from the KV cache; this realizes the mask suppression of Section 3.2 at inference time, and the cached context grows only by the sampled byte at each step. The sampled byte is unpacked into eight split decisions and immediately updates the partial octree, which in turn determines the following query. Generation stops when the target octree depth is complete, rather than relying on the language model to infer structural termination.

The completed byte sequence is decoded into the sparse occupancy grid G_S and passed through the 3D U-Net. After binarizing its occupancy prediction at a threshold of 0.5, the occupied voxels are converted to the sparse coordinates expected by TRELLIS. TRELLIS uses these coordinates as the structural condition and the original text or image as the semantic condition, samples its structured latent representation, and decodes the final mesh. Algorithm S2 summarizes this complete path.

Algorithm S2 OctLLM generation from a text or image condition.

Require: Condition c; OctLLM $F_{\theta_0,\theta_{3D}}$; completion network U_ϕ; target octree depth D; TRELLIS pipeline T

Ensure: S-Octree sequence S and decoded mesh $\widehat{M}$

Generate the response prefix and <mesh_bos> from c
Initialize an empty S-Octree sequence S and partial octree O
while O is not complete through depth D **do**
    $(\mathbf{p}, d) \leftarrow \textsc{NextNode}(O)$
    $q \leftarrow$ <MASK>$(\mathbf{p}, d)$
    $b \sim P_{\text{mesh}}(\cdot \mid c, S, q)$ ▷ logits masked to the 256 occupancy bytes
    Remove the key–value entries of q from the KV cache
    Append mesh byte b to S and update O with $\textsc{Unpack8}(b)$
end while
Append <mesh_eos> and decode S into sparse occupancy G_S
$\widehat{G} \leftarrow \mathbf{1}[U_\phi(G_S) > 0.5]$ ▷ binarize predicted occupancy probability
$\widehat{M} \leftarrow T\big(c, \textsc{OccupiedCoordinates}(\widehat{G})\big)$
return $S, \widehat{M}$

A.3 Serialized Output

Rendered meshes conceal what the language model itself produces. Below we therefore give one truncated image-to-3D response from OctLLM. It begins with a generated natural-language preamble and then contains 1,023 occupancy-byte tokens delimited by <mesh_bos> and <mesh_eos>. Whitespace is inserted only at token boundaries for typesetting. Note that the mesh tokens appear here in their literal vocabulary form, <mesh0> through <mesh255>, whereas the figures in this paper abbreviate the same tokens as O0 through O255 to keep the illustrations compact: <mesh238> in the output below and O238 in Fig. 4 denote the identical occupancy byte.

```
I've produced a 3D mesh asset based on the image:
<mesh_bos> <mesh0> <mesh0> <mesh0> <mesh0> <mesh0> <mesh95> <mesh0>
<mesh95> <mesh0>
<mesh0> <mesh0> <mesh0> <mesh175> <mesh0> <mesh175> <mesh0> <mesh0>
<mesh0> <mesh0>
<mesh0> <mesh0> <mesh95> <mesh0> <mesh77> <mesh0> <mesh0> <mesh0>
<mesh0> <mesh175>
<mesh0> <mesh142> <mesh0> <mesh0> <mesh245> <mesh0> <mesh245> <mesh0>
<mesh0> <mesh0>

......

<mesh240> <mesh15> <mesh95> <mesh15> <mesh95> <mesh240> <mesh95>
<mesh240> <mesh23> <mesh238>
<mesh23> <mesh250> <mesh31> <mesh128> <mesh254> <mesh128> <mesh85>
<mesh234> <mesh1> <mesh87>
<mesh136> <mesh126> <mesh95> <mesh160> <mesh240> <mesh128> <mesh160>
<mesh240> <mesh240> <mesh15>
<mesh95> <mesh15> <mesh126> <mesh248> <mesh_eos>
```

B Experimental Setup and Evaluation Protocol

B.1 Additional Dataset Details

Geometry preprocessing. The ShapeNet portion contains the airplane, car, chair, rifle, and table categories. Each mesh in the training dataset is centered and isotropically scaled to $[-1, 1]^3$, and 100K surface points with face normals are used to build a depth-6 O-CNN (Wang et al., 2017) octree that is complete through depth 3. We discard assets that fail preprocessing or yield fewer than 100 or more than 4,000 mesh-byte tokens.

Caption annotation. We caption render indices 014–017 with a locally deployed Qwen3.5-9B (Qwen Team, 2026a). Because the S-Octree contains geometry but no color or material channels, we exclude appearance attributes that cannot be inferred from the 3D input. The prompt is:

```
Describe this 3D object in English using only its geometry and shape.
Ignore color, material, texture, lighting, and background.
Focus on overall structure, major parts, proportions, and spatial
arrangement. Keep the description moderately detailed and concise.
Directly output the description without any other text. For example:
"A spider with multiple legs and a segmented body."
The description should not be longer than 100 words.
```

Caption decoding uses temperature 0.2 and at most 200 new tokens, with thinking disabled. A fixed seed of 42 controls instruction-template and image-condition sampling; missing captions, renders, or mesh sequences are excluded.

Instruction data. We construct instruction-tuning conversations from 25 dialogue templates, following a protocol similar to ShapeLLM-Omni (Ye et al., 2025). For each sample we draw one template at random and fill it with the caption and the serialized S-Octree.

B.2 Optimization Details

Table S1: Optimization settings not specified in Section 4.

Setting	Value
Input limits	8,192 tokens and 262,144 image pixels
OctLLM schedule	AdamW, peak learning rate 1×10^{-4}, cosine decay, and 10% linear warmup
Batch and precision	2 samples/GPU, 2 gradient-accumulation steps, and bfloat16
Validation	0.1% of the training mixture
Completion schedule	AdamW with cosine decay from 1×10^{-4} to 1×10^{-6}

B.3 Rendering Protocol

Meshes are centered and scaled to a unit bounding box, then rendered at 1024×1024 with Blender 4.0 and EEVEE. The 24 camera directions follow a spherical Hammersley sequence; camera radii range from 1.51 to 9.94 and horizontal fields of view from 7.5° to 52.5°. We use identical lighting and camera sampling for all methods. For baselines without native textures, Hunyuan3D-2.1 (Tencent Hunyuan3D Team, 2025) generates the material before rendering, and the resulting GLB materials are preserved.

B.4 Generation Metrics

Distributional metrics. We use render indices 014–017, yielding 4K images per complete distribution. For features X_r and X_g with empirical means μ_r, μ_g and covariances Σ_r, Σ_g, FID is

$$\mathrm{FID} = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\Big(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\Big). \tag{S1}$$

KID is the unbiased squared MMD,

$$\mathrm{KID} = \frac{\sum_{i\neq j} k(x_i, x_j)}{m(m-1)} + \frac{\sum_{i\neq j} k(y_i, y_j)}{n(n-1)} - \frac{2\sum_{i,j} k(x_i, y_j)}{mn}, \quad k(x,y) = \left(\frac{x^\top y}{d} + 1\right)^3. \tag{S2}$$

We compute both metrics with Inception-V3 (Szegedy et al., 2016) and DINOv2 ViT-L/14 register-token features (Oquab et al., 2024). DINOv2 KID averages 100 subsets of at most 1,000 samples, drawn without replacement with seed 42. KID values are scaled by 100 in Table 1.

Condition alignment. CLIP uses ViT-L/14 embeddings (Radford et al., 2021) and reports $100 \max(s, 0)$ for cosine similarity s. Text-to-3D averages the caption similarity over views 014–017 for each asset. For image-to-3D, we compute for each asset the maximum similarity between the condition image and its 24 generated views, and then average this per-asset maximum over all assets. The maximum is used deliberately rather than a mean over views: the condition image shows the object from one unknown viewpoint, so only the rendered view whose pose best matches it provides a meaningful comparison, whereas averaging over all 24 views would penalize every method for viewpoints the condition never shows rather than for structural misalignment. The same protocol is applied to all methods.

B.5 Understanding Evaluation

Each method receives its native representation: point clouds for PointLLM and ShapeLLM, OBJ text for LLaMA-Mesh, VQVAE indices for ShapeLLM-Omni, and S-Octrees for OctLLM. BLEU uses sentence-level method-1 smoothing, ROUGE reports F1, and Sentence-BERT and SimCSE use `all-mpnet-base-v2` and `princeton-nlp/sup-simcse-roberta-large`, respectively. Metrics are macro-averaged over matched assets and scaled by 100. Judge scores are averaged over valid outputs: 200/200 for all entries except LLaMA-Mesh (198/200 for both judges) and PointLLM-7B under GPT-img (199/200).

Table S2 adds PointLLM-13B and lexical metrics. Consistent with PointLLM (Xu et al., 2024), lexical overlap can favor short generic captions; our conclusions therefore rely on semantic similarity and the two judges.

Table S2: Complete PointLLM-200 metrics. All values are scaled by 100; the best result within each group is highlighted.

	n-gram overlap								Semantic		Judge	
Method	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	S-BERT	SimCSE	GPT-ref	GPT-img
Task-specific 3D understanding models												
PointLLM-7B (Xu et al., 2024)	5.43	1.59	0.74	0.48	10.46	0.89	7.67	13.47	48.05	48.63	46.06	45.63
PointLLM-13B (Xu et al., 2024)	5.45	1.63	0.75	0.48	10.46	0.96	7.64	13.61	48.85	49.78	48.09	48.06
ShapeLLM-7B (Qi et al., 2024a)	9.29	2.86	1.57	1.15	12.83	1.04	11.09	11.71	37.27	37.10	15.92	30.23
Multimodal LLMs												
LLaMA-Mesh (Wang et al., 2024c)	14.69	5.25	3.03	2.31	19.27	2.51	17.29	13.13	30.44	31.08	15.60	17.16
ShapeLLM-Omni (Ye et al., 2025)	4.57	1.24	0.58	0.41	8.30	0.56	6.41	11.53	36.08	34.28	22.27	18.14
OctLLM (Ours)	4.77	1.48	0.65	0.43	8.63	0.73	7.01	11.24	38.80	38.48	26.61	46.88

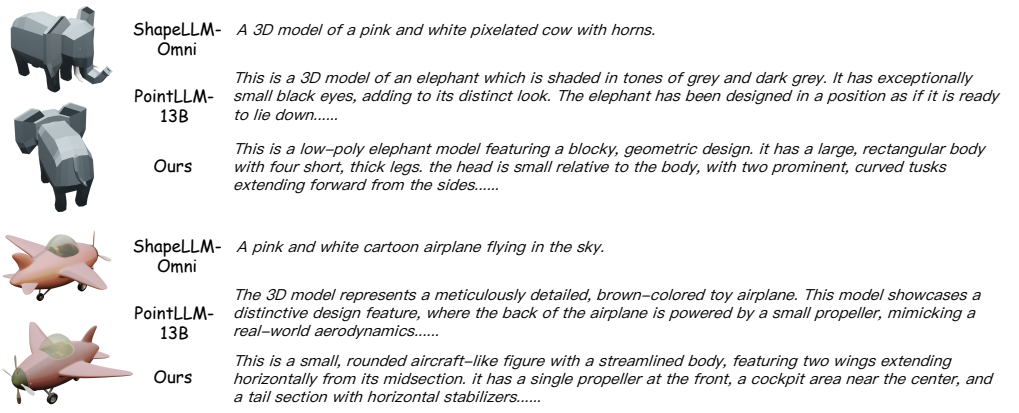


Figure S2: **3D understanding.** Descriptions of two held-out assets, each shown from two viewpoints and truncated for space. ShapeLLM-Omni produces terse descriptions and sometimes misses the object category, while PointLLM-13B emphasizes appearance over structure. OctLLM gives more complete descriptions of the visible parts and their spatial arrangement; Section C.1 reports its complete outputs.

C Additional Results and Evaluation Details

We provide additional qualitative results for OctLLM under image and text conditions. Figures S3 and S4 show the generated meshes from multiple viewpoints.

C.1 Complete Qualitative Conditions and Outputs

Text-to-3D conditions. At inference, each condition in Fig. 6 follows the fixed instruction, "Generate a 3D mesh based on the following text description:"; below we reproduce the complete condition strings in the same top-to-bottom order.

1. *Biplane.* A biplane featuring two sets of wings stacked vertically, connected by struts and wires. The fuselage is elongated with a rounded nose housing a propeller. Tail assembly includes a vertical stabilizer and horizontal stabilizers. Landing gear consists of two wheels positioned under the front section. Overall design emphasizes symmetry and balanced proportions typical of early aviation aircraft.
2. *Stringed instrument.* A stringed musical instrument featuring a hollow, curved body with a central sound hole. It has a long, slender neck extending upwards from the body, terminating in a headstock equipped with tuning pegs. Six strings run along the length of the neck and across the body, anchored at the bridge. The overall form is symmetrical along its vertical axis, with smooth, rounded contours defining the body's shape.
3. *Quadrupedal dinosaur.* A quadrupedal dinosaur model featuring a robust body, four sturdy legs, and a long tail. The head is adorned with prominent horns and a frill, while the limbs are positioned to support its weight. The overall structure suggests a powerful stance, typical of herbivorous dinosaurs.
4. *Pedestal sink.* A pedestal sink featuring a rectangular basin with rounded corners, supported by a cylindrical column. The faucet is centrally mounted on the back edge of the basin, extending forward with a curved spout. The overall design is compact and functional, with smooth surfaces and clean lines.

3D-understanding outputs. The two OctLLM descriptions in Fig. S2 were shortened only in the figure. Their complete, unedited outputs are:

Elephant (top). This is a low-poly elephant model featuring a blocky, geometric design. it has a large, rectangular body with four short, thick legs. the head is small relative to the body, with two prominent, curved tusks extending forward from the sides. large, fan-shaped ears are positioned on either side of the head. the trunk hangs down between the legs. the overall shape is angular and simplified, lacking smooth curves

Aircraft (bottom). This is a small, rounded aircraft-like figure with a streamlined body, featuring two wings extending horizontally from its midsection. it has a single propeller at the front, a cockpit area near the center, and a tail section with horizontal stabilizers. the design is compact and aerodynamic, with smooth curves and a simple, geometric form

C.2 Judge Prompts

The reference-based judge receives the human caption and the model caption and returns a score with a short justification:

```
Evaluate a model-generated caption against a human-generated caption
(ground truth) for a 3D model. Identify the aspects mentioned in the
human caption and calculate the percentage of these aspects correctly
mentioned or partially matched in the model caption. Score from 0 to
100, where each aspect contributes equally to the score. Consider
similar concepts for partial score.

Provide your score (0-100) and a short justification (less than 15
words) in the format of 'score#reason'

Example:
```

```
Human: A white brown skeleton
Model: This is a 3D model of a small, cartoon-like robot. It has a
spherical body and is covered in a layer of white dust.
Output: 50#mention white; skeleton and robot have similar appearance.

Now score the following:
Human: {ground_truth}
Model: {model_output}
Output:
```

The render-grounded judge receives four views of the asset and the model caption, but not the reference:

```
Render-based GPT judge for PointLLM-200 mesh captioning.

Inputs:
- Four RGB renders of the same 3D object: front, right, back, and left.
- One model-generated caption.
- The human caption is intentionally not provided.

Goal:
Score how faithfully the caption describes the visible 3D object in the
renders. The score is a scalar integer from 0 to 100, where higher is
better.

Rubric:
1. Core object identity and function (0-35): award high credit when the
   caption names the correct object category or a close synonym.
2. Geometry, structure, and parts (0-25): reward correct major visible
   components, shape, attachments, symmetry, and distinctive geometry.
3. Color, material, and texture (0-15): reward correct visible colors,
   material cues, texture patterns, and surface finish.
4. Fine-grained attributes and style (0-15): reward accurate style,
   decorative motifs, proportions, pose, orientation, and special
   identifying features.
5. Caption quality and specificity (0-10): reward concise but
   informative captions and penalize vague or repetitive captions.

Penalties and caps:
- Severe hallucination should reduce the score.
- Empty, non-natural-language, or token-like captions should score
0-10.
- Captions for a completely different object should score 0-20.
- Broadly correct category with many wrong details should usually score
  40-70.

Output strict JSON only:
{"score": <integer 0-100>, "reason": "<one short sentence>",
 "matched": ["..."], "errors": ["..."]}
```

C.3 AUTOREGRESSIVE INFERENCE EFFICIENCY

We compare text-to-3D token counts and autoregressive latency on a fixed set of 200 held-out Toys4K prompts. Each model runs sequentially at batch size one on a single NVIDIA GeForce RTX 5090, with one complete warm-up generation and its recommended decoding settings. We report means over all 200 outputs. Latency is synchronized CUDA-event time over each method's autoregressive generation region; input processing, downstream 3D reconstruction, and rendering are excluded. For OctLLM, the timed region includes the complete constrained octree loop and the per-step structural checks needed to select the next query.



Figure S3: **Additional image-to-3D generation results.** Each row shows one image condition followed by the generated mesh from six viewpoints.

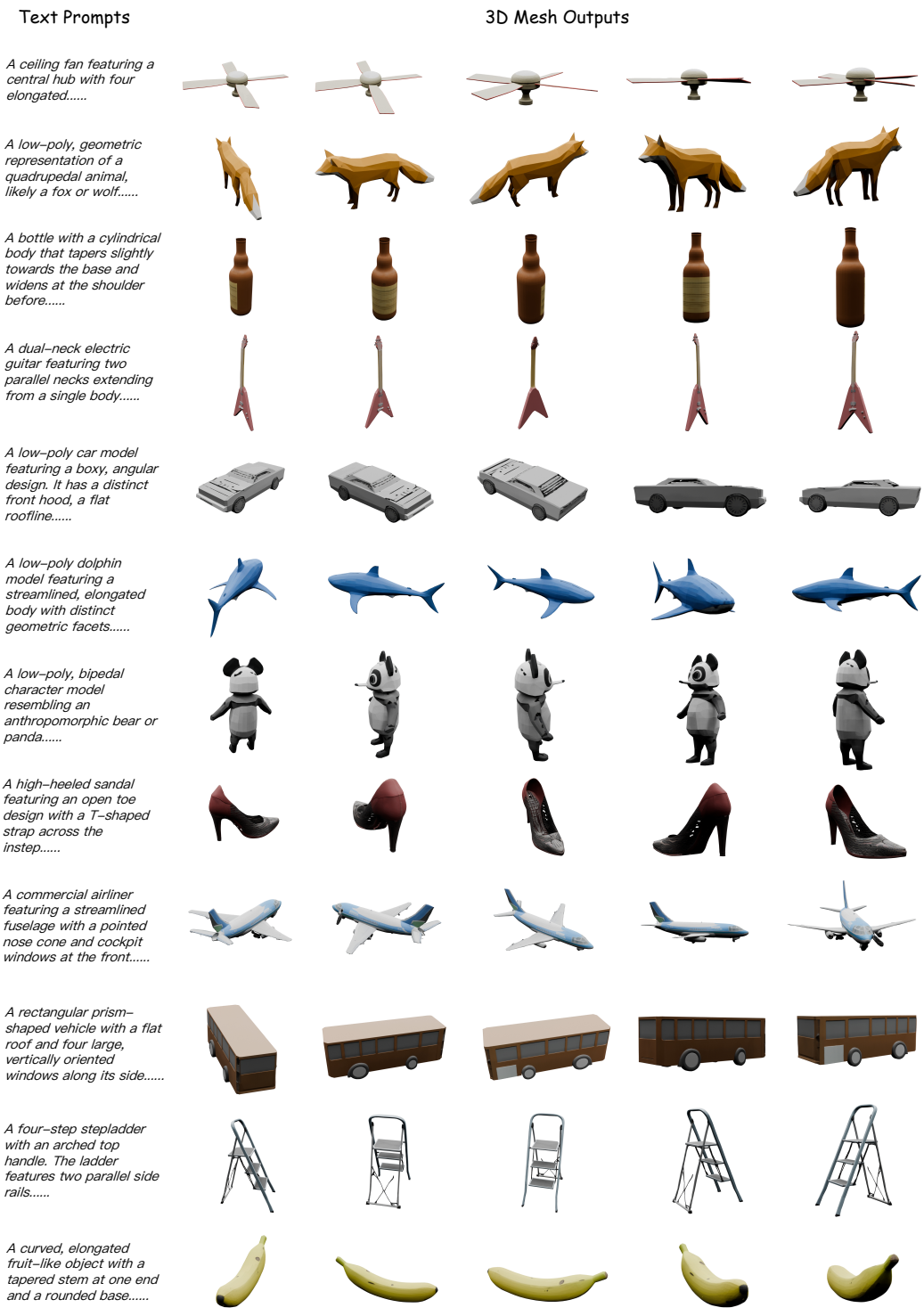


Figure S4: **Additional text-to-3D generation results.** Each row pairs a text condition, truncated in the figure, with five viewpoints of the generated mesh.

Table S3: Text-to-3D token count and autoregressive latency on a fixed 200-prompt Toys4K subset. Values are means on one RTX 5090. Token units are method-native and differ across representations.

Method	Output tokens	Structure tokens	AR latency (s) ↓
ShapeLLM-Omni (Ye et al., 2025)	1,026.00	1,024.00	19.92
AR3D-R1 (Tang et al., 2025b)	1,026.00	1,024.00	23.50
LLaMA-Mesh (Wang et al., 2024c)	4,443.50	4,442.53	85.30
OctGPT (Wei et al., 2025)	10,033.44	2,757.48	52.53
OctLLM (Ours)	1,277.06	1,262.06	37.59

Output tokens follow each method's native output unit, whereas structure tokens report its representation-specific structural sequence: VQ indices for ShapeLLM-Omni and AR3D-R1, OBJ BPE tokens for LLaMA-Mesh, split-node positions for OctGPT, and occupancy-byte tokens for OctLLM. These counts characterize representation length rather than a shared vocabulary. OctLLM averages 1,277.06 output tokens, only modestly more than the fixed-token ShapeLLM-Omni and AR3D-R1 representations and substantially fewer than LLaMA-Mesh and OctGPT. Its optimized constrained decoder takes 37.59 seconds on average, making its latency competitive among the compared methods and faster than OctGPT and LLaMA-Mesh.

D Ablation Study

All variants are trained on ShapeNet airplanes (Chang et al., 2015) under a fixed data and optimization protocol, using four 80 GB A100 GPUs. Training lasts at most 15 epochs, and we report the checkpoint with the best performance on a 5% validation split. Unless stated otherwise, they route 8 branches through layers $\{0, 4, 8, 12, 16, 20, 24, 27\}$ for training efficiency. We compute Inception-V3 FID/KID on rendered normal maps and Sentence-BERT/SimCSE similarity for understanding; KID is scaled by 100. These subset results are not directly comparable with the full-model Toys4K evaluation.

Explicit 3D Language versus Latent Codes. We replace the S-Octree with the 1,024 VQVAE tokens used by ShapeLLM-Omni (Ye et al., 2025), keeping all other settings fixed. In Table S4, the S-Octree reduces FID by 20.5% and improves both understanding metrics at the same KID. Its coordinate-anchored occupancy tokens therefore benefit both generation and understanding. Because the representation is the only changed component, this result directly supports our central premise: exposing spatial coordinates and hierarchy to the LLM is more effective than asking it to operate on opaque latent indices.

Table S4: Ablation on the 3D representation. All settings except the representation are fixed; KID is scaled by 100. Generation metrics are computed on the image-to-3D task.

	Generation		Understanding	
Representation	FID ↓	KID ↓	Sentence-BERT ↑	SimCSE ↑
VQVAE tokens (1,024)	33.10	0.80	67.15	68.87
S-Octree (Ours)	26.31	0.80	69.47	70.28

Preserving Language Ability. We compare OctLLM with a rank-64 LoRA control (Hu et al., 2022) that also mixes UltraChat (Ding et al., 2023) into 3D training, following ShapeLLM-Omni (Ye et al., 2025). Despite the additional text data, LoRA loses 5.1 MMLU and 18.1 HellaSwag points relative to the backbone (Table S5). OctLLM matches the backbone on MMLU, HellaSwag, and IFEval and remains within 0.5 points on GSM8K because 3D gradients update only the routed branches, not the shared language pathway. Together with the representation ablation, this isolates the two complementary insights of OctLLM: explicit spatial tokens improve 3D modeling, while parameter-isolated routing adds this capability without overwriting the pretrained language pathway.

Table S5: Language preservation after ShapeNet-airplane training. The LoRA control includes UltraChat; the frozen backbone is shown as a gray reference.

Method	MMLU ↑	HellaSwag ↑	GSM8K ↑	IFEval ↑
Frozen Qwen2.5-VL-7B-Instruct (reference)	64.65	67.44	83.54	75.41
LoRA + UltraChat	59.58	49.34	69.97	59.33
OctLLM (Ours)	64.65	67.44	83.09	75.41

Components of the S-Octree Pathway. Table S6 isolates the three S-Octree components. Removing predictive mask tokens causes the largest degradation; 3D position embeddings and mask suppression provide additional gains. This pattern supports their complementary roles: mask queries specify where occupancy is predicted, position embeddings ground each query in 3D, and mask suppression keeps previous query placeholders from contaminating later context. The full model consequently combines spatially grounded prediction with a clean autoregressive history.

Table S6: S-Octree component ablations for image-to-3D generation. KID is scaled by 100.

Method	FID ↓	KID ↓
w/o 3D Position Embedding	32.84	1.00
w/o Mask Token	61.08	3.87
w/o Mask Suppression	28.03	0.94
Full model	26.31	0.80

Placement of Routed 3D Branches. With eight routed branches, uniform placement gives the best generation scores under both conditions (Table S7), while the first- and last-layer placements favor understanding. The last eight layers also outperform the first eight on every generation metric. Although uniform placement does not maximize understanding, it remains within 4.2 points of the best targeted placement on both semantic metrics, while reducing image- and text-conditioned FID by 22.7% and 17.8% relative to the next-best placement. We therefore select uniform depth coverage as the stronger joint-task trade-off and assign the full model's four additional branches to later layers.

Table S7: Placement of eight routed 3D branches. FID/KID (scaled by 100) evaluate generation; Sentence-BERT and SimCSE evaluate understanding.

	Generation				Understanding	
Placement	FID (image) ↓	KID (image) ↓	FID (text) ↓	KID (text) ↓	Sentence-BERT ↑	SimCSE ↑
First 8/28 layers	36.63	1.70	38.97	1.62	72.26	74.37
Last 8/28 layers	34.03	1.31	37.09	1.49	73.65	73.71
Uniform 8/28 layers (Ours)	26.31	0.80	30.48	1.07	69.47	70.28

Scaling Routed 3D Capacity. The 4-, 6-, and 8-branch variants use layers $\{0, 8, 16, 24\}$, $\{0, 8, 12, 16, 24, 27\}$, and $\{0, 4, 8, 12, 16, 20, 24, 27\}$, respectively, corresponding to 0.93B, 1.40B, and 1.86B trainable parameters. In Fig. S5, FID changes little from four to six branches but falls from 37.63 to 26.31 with eight; KID likewise improves to 0.80. The gain is therefore not proportional to the budget at the smallest sizes, but the controlled eight-branch result shows that allocating sufficient parameters across the full depth substantially improves generation quality. This supports scaling the isolated 3D pathway rather than modifying the shared backbone when more 3D capacity is needed.

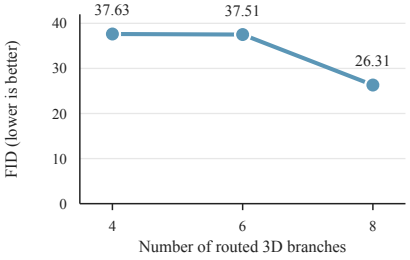


Figure S5: **Scaling routed 3D capacity.** Image-to-3D FID on ShapeNet airplanes. Only the branch count changes.

Full-Rank 3D Branches versus Token-Routed LoRA. The rank-64 LoRA control in Table S5 tests language forgetting, but leaves open whether token-routed LoRA can preserve language while providing sufficient 3D capacity. We therefore replace the default eight full-rank 3D branches with rank-256 LoRA adapters in all 28 layers and route only 3D tokens through them, while keeping the ShapeNet-airplane dataset protocol fixed. The added 3D vocabulary requires trainable embedding and language-head rows, so a row-wise stop-gradient restricts updates to the new token rows. This control matches the backbone on MMLU and HellaSwag, as does the dual-stream model, but the full-rank branches perform better on three of four generation metrics (Table S8): image-conditioned FID falls from 32.44 to 26.31 and KID from 1.06 to 0.80, while text-conditioned FID improves from 30.93 to 30.48. On the full training data we further observed that the full-rank branches train stably, whereas high-rank LoRA suffers gradient explosions: a low-rank residual does not give 3D occupancy enough expressive room, and enlarging the rank makes large-scale training unstable. Token-routed LoRA therefore preserves language ability, but does not match the 3D capacity or optimization stability of dedicated full-rank branches.

Table S8: Full-rank dual-stream branches versus token-routed LoRA on ShapeNet airplanes. FID/KID evaluate image- and text-conditioned 3D generation (KID scaled by 100); language scores are reference values because both variants match the frozen backbone.

	3D Generation				Language Ability (reference)	
Method	FID (image) ↓	KID (image) ↓	FID (text) ↓	KID (text) ↓	MMLU ↑	HellaSwag ↑
Dual-stream branches (Ours)	26.31	0.80	30.48	1.07	64.65	67.44
Token-routed LoRA (rank 256)	32.44	1.06	30.93	0.98	64.65	67.44

S-Octree Efficiency. For the example in Fig. S6, sparsification reduces the sequence from 3,751 to 2,297 mesh tokens while preserving the coarse structure. Figure S7 evaluates the same effect over five ShapeNet categories. Full-octree sequences extend to nearly 6K tokens. Measured on the same four 80 GB A100 GPUs for both settings, S-Octree reduces peak memory from 58 GB to 44 GB and epoch time from 2.17 to 1.37 hours ($1.6\times$). The memory saving matters in practice, since it brings training within the 48 GB budget of widely available accelerators such as the L40 instead of requiring 80 GB cards. The completion network restores the pruned fine-level occupancy.

To assess generation quality, we additionally train on complete octree sequences without random emptying, using the same hardware, ShapeNet-airplane protocol, and eight-branch architecture. Under image conditioning, this model achieves FID 27.54 and KID 0.83 (scaled by 100), compared with 26.31 and 0.80 for S-Octree. The scores are close, with the full octree slightly worse. Thus, in this ablation, random emptying followed by completion preserves generation quality while reducing training time, memory usage, and the autoregressive decoding workload.

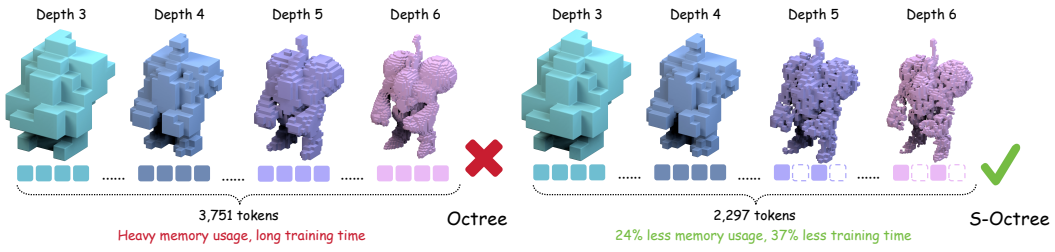
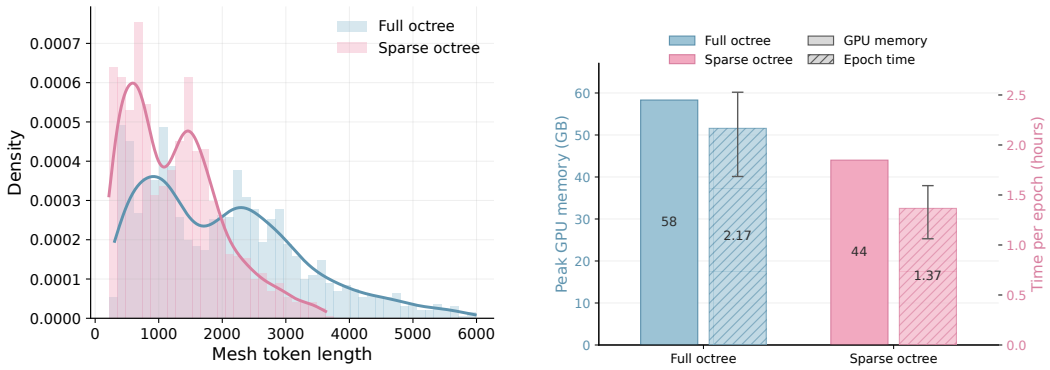


Figure S6: Full octree versus S-Octree on the same asset, shown at depths 3 to 6 with the corresponding mesh-token sequence beneath each level. Pruning part of the second-to-last level, and with it the finest-level descendants, shortens the sequence the LLM must write without altering the coarse structure it predicts.



(a) Mesh-token length distribution.

(b) Training resource cost.

Figure S7: Practical benefits of the S-Octree representation on five selected ShapeNet categories. Compared with full octrees, S-Octrees shift the mesh-token length distribution toward shorter sequences and reduce both peak GPU memory and training time per epoch.

Sparse Voxel Completion Design. We ablate the loss and architecture of the 3D U-Net that completes the S-Octree occupancy before TRELLIS decoding. In Table S9, the default bottlenecked U-Net without cross-scale skip connections achieves the best FID and KID with binary cross-entropy. Replacing BCE with Dice loss slightly degrades both metrics, while adding skip connections or aligning incomplete-shape latents to a pretrained encoder causes substantially larger degradation. These results support binary occupancy supervision through a compact bottleneck for recovering missing fine structure, motivating the completion design used in OctLLM.

Table S9: Ablation on the sparse voxel completion network for image-to-3D generation. KID is scaled by 100.

Completion design	FID ↓	KID ↓
Encoder alignment + MSE	58.93	3.49
3D U-Net + skip connections + BCE	52.14	2.92
3D U-Net + Dice	38.89	1.87
3D U-Net + BCE (Ours)	36.86	1.60

E LIMITATIONS

OctLLM has several limitations. First, its variable-length S-Octree allocates fewer tokens to simple shapes and more to complex ones, yet averages 1,793 tokens across our training set, compared with ShapeLLM-Omni's fixed 1,024-token representation. The longer sequences increase both training time and GPU memory usage. Spatially adaptive octree refinement may retain this flexibility with shorter sequences. Second, geometry passes through two learned predictions—S-Octree generation and 3D U-Net completion—so upstream occupancy errors can propagate or be amplified, producing holes and other artifacts that lower generation quality. Third, OctLLM does not support 3D understanding of appearance or 3D editing. Future work will add native color and material modeling, so OctLLM no longer depends on an external 3D generator and can describe appearance as well as geometry. We also plan to introduce 3D editing in subsequent versions of OctLLM.