3D-PLOT-LLM: Part-Level Object Tokens for 3D Large Language Models

Jintang Xue
University of Southern California
Los Angeles, California, USA
jintangx@usc.edu

Xinyu Wang
University of Southern California
Los Angeles, California, USA
xwang350@usc.edu

Yixing Wu
University of Southern California
Los Angeles, California, USA
yixingwu@usc.edu

Jingwen Chen
Ohio State University
Columbus, Ohio, USA
chen.15073@buckeyemail.osu.edu

C.-C. Jay Kuo
University of Southern California
Los Angeles, California, USA
jckuo@usc.edu

Abstract

3D multimodal large language models (3D MLLMs) describe a 3D object as a whole but cannot address, name, or reason about its parts. Prior part-aware attempts add segmentation decoders, heavier 3D encoders, or bounding-box grammars at substantial parameter cost. We take a fundamentally different path: we reorganize the input token stream so that parts become directly addressable through the LLM's own vocabulary. Our model, **3D-PLOT-LLM**, partitions the frozen point encoder's patches into K locally coherent regions and inserts, before each region's patch tokens, a learnable per-region marker and a reserved vocabulary token `<part_k>`; a Marker-Space Refinement (MSR) module then conditions each marker on its region's spatial statistics and adjacency neighbors. The model thus cites parts in its output and follows prompts that refer to parts by token, a capability absent from prior object-level 3D MLLMs. To probe this interface, we construct **PartVerse-QA**, a vocabulary-level part-QA benchmark adapted from PartVerse mesh annotations (77K training pairs and 588 held-out queries on disjoint object splits), on which 3D-PLOT-LLM reaches caption-to-slots Jaccard 0.459 and Exact-match 13.78% ($+64\%$ relative over the strongest non-MSR variant), with a slot-to-caption GPT-4o judge of 44.68. On the text-description track of the 3DCoMPaT-GrIn part-aware grounded description benchmark, 3D-PLOT-LLM outperforms PointLLM, Kestrel, PARIS3D, and SegPoint on every reported text metric, and ShapeLLM on 3 of 4, with up to $+3.03$ GPT-4o judge over PointLLM. On Objaverse whole-object captioning, adding PartVerse-QA at Stage 2 yields $+0.65$ SBERT and $+1.85$ GPT-4o over PointLLM, and tops PointLLM-PiSA on 4 of 5 traditional metrics (SBERT, SimCSE, BLEU-1, METEOR) despite targeting a different (part-grounded) objective. All with under 1M new trainable parameters on a frozen point encoder, an order of magnitude below prior part-aware 3D MLLMs, and no learned part detector, segmentation decoder, or bounding-box head.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

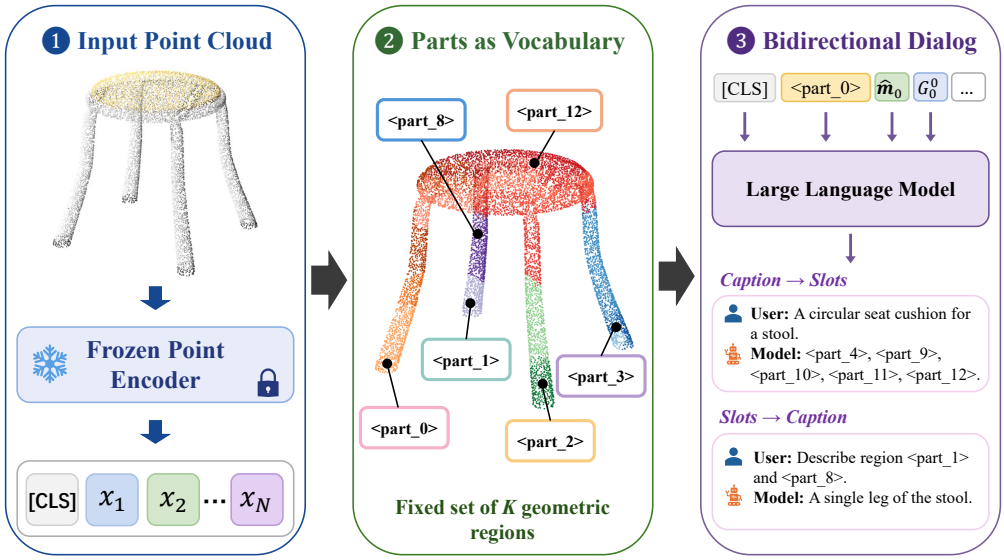


Figure 1: **3D-PLOT-LLM** treats each geometric region of a 3D object as a first-class addressable token in the LLM's vocabulary. A frozen point encoder's patch tokens are partitioned into a fixed set of K geometric regions (center panel), and a reserved vocabulary slot `<part_k>` makes each region readable and writable through the same interface the LLM uses for text (right panel). No segmentation decoder is used; the interface addresses coarse geometric regions, not point-wise masks.

1 Introduction

3D multimodal large language models (3D MLLMs) align point clouds with language for free-form object-level understanding (e.g., captioning, classification, QA [14, 15, 32, 42]), with a parallel branch lifting MLLMs to whole-scene grounding [6, 17, 18, 29, 40, 48]. At the object level, however, 3D MLLMs still present each object as a flat sequence of patch tokens, with no compositional units to refer to, name, or aggregate over at the part level: PointLLM [42] can caption a chair but has no token denoting its back or legs. Prior part-aware object 3D MLLMs pair the LLM with dense-prediction machinery (Kestrel [2]: a segmentation decoder driven by LISA/GLaMM [21, 34] `[SEG]` tokens; Part-X-MLLM [41]: a heavier dual-pathway encoder with a bounding-box grammar), inflating the non-LLM parameter budget by roughly an order of magnitude. More fundamentally, neither changes the LLM's input vocabulary, so parts remain outputs of an external module rather than entries the LLM can read, emit, and reason over in its own token stream; their dense predictions are strong, but that axis is orthogonal to the interface question we address. Our model, **3D-PLOT-LLM**, closes this gap by making parts addressable tokens in the LLM's vocabulary (Figure 1).

A natural reflex would be to define a canonical part decomposition and reserve a vocabulary slot per semantic part. We argue against this on three grounds: (i) dense part annotation does not scale, since existing human annotations use category-specific schemas and reach at most tens of thousands of objects, far below the $\sim$ 660K-object alignment regime of 3D MLLMs [42]; relying on other pretrained modules just relocates this cost upstream. (ii) Part granularity is intrinsically ambiguous, since a chair admits many valid hierarchies (seat+back; +armrests; +four legs), each reasonable. (iii) Captioning does not require a semantic-category label; "a wooden cylindrical pole at the bottom" is a useful answer whether the part is labeled "stool leg" or "support post".

We instead position parts as emerging from input-side token organization rather than architectural decomposition. We partition each object's patch tokens into a fixed set of K geometric regions via a deterministic, training-free procedure that grows regions over patch spatial coordinates and frozen point encoder features (App. A), and reserve a per-object vocabulary `<part_0>`,...,`<part_K-1>` in the LLM's tokenizer. Part identity emerges from data as a learned composition of these region tokens; a nameable part ("the shovel handle", "the boat hull") is expressed as a set of `<part_k>` tokens, learned end-to-end from PartVerse-QA. Each region also carries a learnable marker $\mathbf{m}_k \in \mathbb{R}^{384}$, refined by a lightweight Marker-Space Refinement (MSR) module conditioning the marker on the

region's spatial statistics and adjacency. With under 1M new trainable parameters on a frozen 22M-parameter point encoder, this part-level interface (absent from prior object-level 3D MLLMs) yields consistent gains across PartVerse-QA, the text-description metrics of 3DCoMPaT-GrIn, and Objaverse captioning (§4). Our contributions:[1]

- **Part-aware addressing without canonical decomposition.** 3D-PLOT-LLM reserves per-object vocabulary tokens <part_0>…<part_K-1> tied to a deterministic, training-free spatially coherent partition of frozen-encoder patches; the LLM learns part identity as compositions of these tokens and uses them as a bidirectional interface it both reads and emits, without a learned part detector or segmentation decoder for pre-segmented objects.
- **Marker-Space Refinement (MSR).** A lightweight residual module conditioning per-region markers ($\mathbb{R}^{384}$) on region statistics and inter-region adjacency, supplying the structural context that binds <part_k> embeddings to region content.
- **PartVerse-QA: a bidirectional caption-to-slots (C2S) and slot-to-caption (S2C) part-addressing benchmark.** Adapted from PartVerse [10] mesh annotations: 77K training pairs + 588 held-out queries (392 C2S, 196 S2C) on object-disjoint splits, aligned to Objaverse 8192-point clouds and quality-filtered (App. E).
- **Empirical evaluation across three benchmarks.** PartVerse-QA; the part-aware point grounded description task (PaPGD) of 3DCoMPaT-GrIn [2, 38], compared with PointLLM, ShapeLLM [32], Kestrel [2], PARIS3D [20], and SegPoint [16]; and Objaverse captioning, compared with PointLLM, ShapeLLM, ENEL [39], and PiSA [14] (re-evaluated and cited rows are marked in the tables).

2 Related work

Object-level 3D MLLMs. Object-level 3D MLLMs share a recipe: project patch tokens from a point encoder into the LLM input space and train on large-scale point-text supervision. PointLLM [42] exemplifies this with a Point-BERT [47] encoder aligned to Vicuna [7] via a two-stage protocol (660K Cap3D [26] captions of Objaverse [8] for alignment, then 70K GPT-generated complex instructions for tuning); PiSA [14] extends it with a self-augmented data engine and PiSA-Bench. ENEL [39] removes the point encoder altogether and trains on the same PointLLM corpus. Other systems vary specific ingredients: GPT4Point [33] substitutes a Q-Former projector and scales to ∼1M Objaverse-XL captions; ShapeLLM [32] swaps in a ReCon++ encoder for embodied interaction; Point-Bind [15] aligns points to ImageBind [13] space. A parallel foundation-encoder line, notably Uni3D [49] (billion-parameter 2D-initialized ViT aligned with image-text features), supplies backbones to several of these MLLMs. Across the landscape, no object MLLM exposes an addressable part vocabulary; each treats the object as a flat token stream and cannot cite specific parts in its answer.

Part-aware object 3D MLLMs. Prior systems fall along two design lines. Mask-based methods inherit LISA [21]/GLaMM [34]-style [SEG] tokens: Kestrel [2] wraps part references in … /[SEG] and attaches a segmentation decoder with query refinement to a Uni3D-g [49] backbone, trained on the 3DCoMPaT-GrIn split it constructs on 3DCoMPaT++ [38]; PARIS3D [20] renders to multi-view 2D for a ViT-H (SAM) backbone and routes [SEG] through a custom mask decoder, releasing RPSeg3D; SegPoint [16] adopts the same LISA-style recipe for unified scene-level 3D segmentation and serves as a part-task baseline in Ahmed et al. [2]. Box-based Part-X-MLLM [41] pairs a dual-pathway encoder (XYZ+normals; RGB) with a bounding-box grammar (⟨boxs⟩ … ⟨boxe⟩ plus <adds>/<dels>/<mods> edits) emitting parts as autoregressive box tokens, with part ambiguity resolved by post-hoc clustering of boxes through text semantics. All inflate the non-LLM parameter budget by roughly an order of magnitude, and in none does the LLM read or write a part as a first-class vocabulary entry: Kestrel's [SEG] is a single anchor slot re-used across all parts and decoded by an external head, and Part-X-MLLM's box tokens address coordinate bins rather than per-part identifiers. 3D-PLOT-LLM removes the downstream machinery entirely and instead reserves per-slot vocabulary tokens <part_0>…<part_K-1>, addressable uniformly in prompts and outputs and anchored to a fixed coarse spatial partition, so the LLM learns slot-to-caption bindings end-to-end without a segmentation or box head.

[1] Code, weights, and PartVerse-QA: https://github.com/jintangxue/3D-PLOT-LLM.

Scene-level 3D MLLMs and identifier-token precedent. A parallel line targets 3D MLLMs at the room-scene level (3D-LLM [17], LL3DA [5], LLaVA-3D [50], 3D-LLaVA [9], Reason3D [19], SpatialLM [29], Pts3D-LLM [40]). A sub-thread within it introduces learnable identifier tokens as scene-level addressing: Chat-Scene [18] inserts per-object `<OBJ_k>` into the tokenizer; Descrip3D [43] adds relational descriptions on top; 3DGraphLLM [48] enriches each `<OBJ_k>` with a kNN subgraph; Grounded 3D-LLM [6] uses a single `<ref>` delimited by `<p>...</p>`, trained with a phrase-level contrastive CLASP objective. In 2D, Groma [27] and Set-of-Mark prompting [45] bind identifiers to regions from a region proposer or an off-the-shelf segmenter. These works establish vocabulary-level addressing as an effective scene-level interface where an external proposer (e.g. Mask3D [37]) first carves the scene into discrete objects; in every case the identifier is bound to a detected proposal. Applying this interface to parts within a single object is qualitatively different because neither enabling assumption holds: (i) no external proposer produces canonical part boundaries within an object (motivating our deterministic, training-free partition, §3.1); (ii) while spatial disambiguation is a shared requirement at both scene and part levels, parts within one object are often semantically similar (e.g., four chair legs) and share the same object-level bounding box, so the per-token global cues that scene-level methods like Chat-Scene inject collapse to a constant at the sub-object level (motivating our per-region statistics within the object frame and inter-region adjacency in §3.3). MSR (§3.3) is what lets the LLM ground language on regions that no detector has labeled.

Complementary part-level work. A separate thread targets part generation: PartCrafter [24], OmniPart [46], MMPart [4] produce decomposed part meshes via image-conditioned generation. Outside the MLLM paradigm, PartNet [30] provides a hierarchical 3D part benchmark across 24 categories; PartSLIP [25] and SATR [1] transfer 2D vision-language knowledge from GLIP [22] to 3D part segmentation via multi-view rendering; PartVerse [10] provides mesh-level part annotations on Objaverse, which we adapt for Stage-2 PartVerse-QA supervision. These works produce geometry, masks, or annotations rather than language-level part reasoning, and complement ours.

3 Method

Overview. 3D-PLOT-LLM represents a 3D object as a structured token sequence in which parts are first-class, addressable entries in the LLM's vocabulary (Figure 2). A frozen point encoder tokenizes $8{,}192$ points into 512 patch tokens $\{x_i\}$ with centers $\{c_i\}$ and a global `[CLS]`. Three components then operate before the LLM: (i) a deterministic, training-free region partition (§3.1) groups the patches into K spatially coherent regions and exposes per-region statistics and inter-region adjacency; (ii) Marker-Space Refinement (MSR, §3.3) conditions each region's learnable marker on those statistics and adjacency; and (iii) per-region token assembly (§3.2) inserts, before each region's patches, the refined marker and a reserved vocabulary token `<part_k>` addressable in both prompts and responses. No segmentation decoder, bounding-box head, or point-wise mask supervision is used.

3.1 Geometric region partition

We organize the 512 patch tokens into K spatially coherent regions via a deterministic, training-free partition of the frozen encoder's patches (full algorithm in App. A). The partition is not purely spatial: regions grow under a mixed cost of patch distance and feature dissimilarity, with weight $\lambda{=}0.8$ on distance (App. A). The partition is structural substrate, not a semantic claim; it provides K stable addressable slots so that the per-region markers (§3.2), reserved vocabulary tokens `<part_k>`, and MSR (§3.3) all index into a consistent structural level. Two properties are required. **Determinism**: because the encoder is frozen, the same input always yields the same partition, so region index k has a stable meaning across inference runs and the LLM can learn slot-to-content bindings. **Approximate size balance**: no single region should carry a disproportionate share of the object's content, so per-region markers' representational load remains roughly comparable across k. Any spatially coherent partition with these two properties would serve. For each region k we record its inter-region adjacency E (induced by the patch connectivity graph; App. A) and a 7-dim statistics vector $s_k \in \mathbb{R}^7$ (region centroid xyz, relative size, xyz span); both feed into MSR (§3.3).

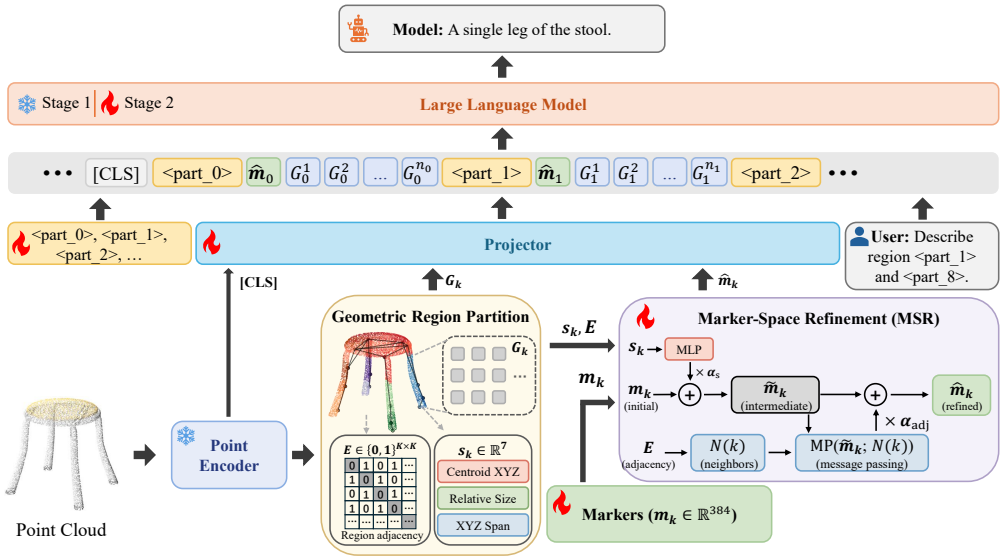


Figure 2: **3D-PLOT-LLM pipeline.** A frozen point encoder produces 512 patch tokens that flow through three components: (i) **Geometric Region Partition** (§3.1) groups patches into K regions $\{G_k\}$ and exposes per-region statistics $s_k \in \mathbb{R}^7$ and inter-region adjacency E; (ii) **Marker-Space Refinement** (§3.3) updates each learnable marker $\mathbf{m}_k \to \hat{\mathbf{m}}_k$ via stats and graph residuals; (iii) **Token Assembly** (§3.2) interleaves `[CLS]`, reserved `<part_k>` tokens, $\hat{\mathbf{m}}_k$, and G_k before the LLM (frozen Stage 1, tuned Stage 2). `<part_k>` is a vocabulary token and bypasses the projector; continuous tokens are projected. No segmentation decoder, bounding-box head, or point-wise mask supervision is used.

3.2 Token assembly

For each region k, the assembly places a reserved label token `<part_k>` and a learnable per-region marker $\mathbf{m}_k \in \mathbb{R}^{384}$ immediately before that region's patch tokens:

$$\underbrace{\texttt{[CLS]}}_{\text{global}}, \underbrace{\texttt{<part_0>}, \mathbf{m}_0, G_0}_{\text{region 0}}, \underbrace{\texttt{<part_1>}, \mathbf{m}_1, G_1}_{\text{region 1}}, \ldots, \underbrace{\texttt{<part_K-1>}, \mathbf{m}_{K-1}, G_{K-1}}_{\text{region } K-1}$$

where G_k is the set of patch tokens in region k, serialized in their original encoder patch index order inside the LLM input. The total LLM-input length is $1{+}2K{+}\sum_k |G_k| = 545$ tokens at $K{=}16$. The tokens in this assembly play two roles. The patch set G_k and the per-region marker $\mathbf{m}_k$ (a learnable 384-d vector in the projector's input space; the projector is a multi-layer MLP mapping backbone features to the LLM's 4096-d embedding) form the **content** of each region. The reserved token `<part_k>` (a 4096-d vocabulary entry the LLM both reads and emits) acts as the **label**; it carries no intrinsic geometric content, with its meaning arising entirely from the learned binding to its region's content tokens. We therefore call `<part_k>` a region pointer: it names a spatial region, not a category; the same index refers to different content on different objects; and unlike a closed-set semantic part vocabulary it composes with free-form language in both directions.

3.3 Structural refinement

The per-region markers $\{\mathbf{m}_k\}$ introduced in §3.2 are unaware of their region-level spatial context: marker $\mathbf{m}_k$ does not see its region's centroid or size, nor that it is adjacent to region k'. Marker-Space Refinement (MSR) supplies this context through a lightweight two-stage residual update on the markers.

Let $s_k \in \mathbb{R}^7$ be region k's statistics vector (3-dim centroid xyz in the object frame, 1-dim relative size $|G_k|/N$ where $N{=}512$ is the patch budget, and 3-dim axis-aligned xyz span), and $\mathcal{N}(k) = \{k' : (k, k') \in E\}$ the set of regions adjacent to k (§3.1). MSR updates each marker $\mathbf{m}_k \in \mathbb{R}^{384}$ through

two residuals:

$$\tilde{\mathbf{m}}_k = \mathbf{m}_k + \alpha_{\text{s}} \cdot \text{MLP}(s_k), \tag{1}$$

$$\hat{\mathbf{m}}_k = \tilde{\mathbf{m}}_k + \alpha_{\text{adj}} \cdot \text{MP}(\tilde{\mathbf{m}}_k; \mathcal{N}(k)), \tag{2}$$

with $\alpha_{\text{s}}, \alpha_{\text{adj}}$ learnable residual scales. MLP : $\mathbb{R}^7 \to \mathbb{R}^{384}$ has two linear layers with hidden width 384 and GELU activation in between. MP is a message-passing layer in the MPNN sense [12] with mean aggregation followed by a two-layer update MLP ϕ:

$$\text{MP}(\tilde{\mathbf{m}}_k; \mathcal{N}(k)) = \phi\left(W_c \tilde{\mathbf{m}}_k + \frac{1}{|\mathcal{N}(k)|} \sum_{k' \in \mathcal{N}(k)} W_n \tilde{\mathbf{m}}_{k'} \right), \tag{3}$$

where $W_c, W_n \in \mathbb{R}^{384 \times 384}$ project the central and neighbor markers respectively. The residual form, combined with $\alpha_{\text{s}}, \alpha_{\text{adj}}$ initialized at 0.05 and a zero-initialized output layer in ϕ, makes MSR start from identity ($\hat{\mathbf{m}}_k \approx \mathbf{m}_k$) at the beginning of training. Branch contributions are quantified by a stats-vs.-adjacency ablation in App. G.

The refined marker $\hat{\mathbf{m}}_k$ enters the projector in place of $\mathbf{m}_k$. Because $\hat{\mathbf{m}}_k$ sits immediately before G_k in the assembly, the region-specific structural context it carries directly conditions the LLM's early attention over G_k; the patch tokens themselves remain frozen encoder outputs.

3.4 Training data and two-stage protocol

Training is alignment followed by instruction tuning, matching the standard 3D MLLM schedule [32, 42]. **Stage 1 (alignment)** trains the projector, per-region markers, and MSR on 660K Cap3D captions of Objaverse objects for 3 epochs, with the LLM and point encoder frozen. `<part_k>` tokens are deferred to Stage 2 because (i) Cap3D captions are whole-object descriptions with no part-level signal to learn their embeddings from, and (ii) the frozen Stage-1 LLM does not update its input-embedding matrix in any case. **Stage 2 (instruction tuning)** adds the `<part_k>` tokens to the tokenizer and trains the projector, per-region markers, MSR, and the LLM on the union of PointLLM's 70K complex instructions and 77K PartVerse-QA pairs we derive from PartVerse mesh annotations (App. E) for 3 epochs; the encoder remains frozen. The PartVerse-QA pairs cover two directions of the `<part_k>` interface: **caption-to-slots** (C2S; given a free-text part caption, predict the `<part_k>` set covering it) and **slot-to-caption** (S2C; given a `<part_k>` set, generate a part caption). Together they supply the part-grounded supervision under which `<part_k>` embeddings acquire bindings to region content. In matched-data experiments without PartVerse-QA pairs (no-PV), the `<part_k>` embeddings receive no binding signal and the C2S/S2C interface is non-functional; whole-object and grounded-description tasks still benefit (Tables 1, 3). Full hyperparameters in App. B.

Implementation. We instantiate the point encoder as Point-BERT [47] pretrained via ULIP-2 [44], and the LLM as Vicuna-v1.5-7B [7], matching PointLLM [42]'s released setup. Both encoder and LLM are held identical across our variants and the PointLLM baseline, so any gain is attributable to our architectural contributions rather than encoder/LLM strength. We use $K{=}16$ regions by default, set by the 512-patch budget (32 patches per region) and motivated by the PartVerse part-count distribution (App. E); App. K studies the sensitivity to K with a full $K{=}8$ retrain.

4 Experiments

4.1 Evaluation protocols

Language metrics. We report semantic metrics (SBERT [36], SimCSE [11], GPT-4o LLM-as-judge) and lexical-overlap metrics (BLEU-1 [31], ROUGE-L [23], METEOR [3]), matching PointLLM [42]; both families are reported because lexical overlap alone underestimates open-vocabulary captioning quality. Three splits are evaluated: the 200 Objaverse held-out objects of the PointLLM captioning benchmark [42]; our held-out PartVerse-QA split, which we construct on top of PartVerse [10] mesh annotations (App. E); and 3DCoMPaT-GrIn PaPGD's public test split [2] (6770 multi-part queries). The PartVerse-QA benchmark (§3.4) is scored as: **C2S** by Jaccard and Exact-match over 392 queries; **S2C** by captioning metrics and a GPT-4o judge over 196 queries. Objaverse traditional metrics and PartVerse-QA S2C metrics (traditional and GPT-4o judge) are all reported as

Table 1: **Objaverse captioning.** 3D-PLOT-LLM (full) leads SBERT, SimCSE, GPT-4o; the matched-data no-PV variant outperforms PointLLM on all six metrics at the same 70K Stage-2 budget, isolating architectural from data-volume contribution. Cells are 5-run means; per-cell ±std and significance in App. C. Stage-2 reports total instruction-tuning samples. **Best** and 2nd per column.

Method	Stage-2	SBERT↑	SimCSE↑	GPT-4o↑	BLEU-1↑	ROUGE-L↑	METEOR↑
PointLLM-PiSA	132K	48.21	48.38	–	3.81	7.29	12.32
ShapeLLM (re-eval)	75K	35.49	36.13	12.86	6.02	8.00	12.33
ENEL (re-eval)	70K	47.70	48.22	37.02	**15.18**	**19.28**	**17.89**
PointLLM (re-eval)	70K	47.66	48.16	39.08	3.80	7.02	12.19
3D-PLOT-LLM (no-PV)	70K	47.94	48.44	40.56	3.84	7.10	12.27
3D-PLOT-LLM (full)	147K	**48.31**	**48.67**	**40.93**	3.94	7.22	12.59

5-run means under matched protocols; per-run standard deviations for the main metrics are listed in App. C. C2S Jaccard and Exact-match use deterministic decoding (single run); 3DCoMPaT-GrIn (Table 3) follows cited-baseline reporting.

4.2 Main results

Objaverse captioning. We report two model variants in Table 1 to disentangle architectural and data contributions: a matched-data variant (no-PV, trained on the same 660K Cap3D + 70K complex data as PointLLM) and our full model (adding 77K of our PartVerse-QA pairs at Stage 2). All six metrics (SBERT, SimCSE, GPT-4o judge, BLEU-1, ROUGE-L, METEOR) are reported as 5-run means; per-cell std in Table 5 (App. C). Under matched data, 3D-PLOT-LLM no-PV outperforms PointLLM on all six metrics (up to $+1.48$ GPT-4o judge), an architectural-level lift on whole-object captioning. With PartVerse-QA at Stage 2, the gain extends to $+0.65$ SBERT and $+1.85$ GPT-4o judge over PointLLM, leading PointLLM on all six metrics; METEOR ($+0.40$) reaches significance ($p \approx 0.04$). The other two re-evaluated baselines, ShapeLLM and ENEL [39], score higher than PointLLM on the lexical-overlap metrics but not on the semantic ones: ShapeLLM is far below on SBERT, SimCSE, and the GPT-4o judge, and ENEL is within noise of PointLLM on SBERT and SimCSE and below it on the judge (37.02 vs 39.08).

We further benchmark against PointLLM-PiSA [14], which adds 62K self-augmented captions (132K total) targeting captioning quality. Our 147K-total mix targets part grounding instead and leads PiSA on 4 of 5 traditional metrics (SBERT, SimCSE, BLEU-1, METEOR; PiSA's GPT-4o cell is "–" due to its closed-source judge pipeline). Part-grounded supervision is therefore a viable scaling axis complementary to whole-object caption augmentation.

PartVerse-QA. PartVerse-QA probes a capability no prior part-aware 3D MLLM natively supports (§2): bidirectional `<part_k>`-token addressing in a single LLM token stream. Prior methods emit part references as decoder masks or boxes but accept no region encoding as model input, so direct cross-method comparison is structurally constrained (App. E, "Cross-method evaluation"); the architectural ranking is independently validated against partition-agnostic mesh GT in App. F. Table 2 compares our own architectural progression under identical PartVerse-QA supervision. 3D-PLOT-LLM reaches Jaccard 0.459 and Exact-match 13.78% on the held-out split ($\sim$77$\times$ random chance at the median query cardinality of 3 over K=16 slots; App. E). Vocabulary addressing alone (vocab-only) already establishes a non-trivial 0.413 Jaccard / 8.42% Exact-match baseline; MSR then lifts both metrics ($+0.046$ Jaccard; $+5.36$ percentage points / $+64\%$ relative on Exact-match). The intermediate variant with raw markers but no MSR (row 2) regresses below vocab-only (0.263 Jaccard, 3.06% Exact-match): un-conditioned markers carry no region-level structural signal (§3.3), distracting early attention without informing it; only MSR's conditioning on per-region statistics and adjacency makes the marker substrate functional. Our model also leads every S2C metric (SBERT 62.04, GPT-4o 44.68, METEOR 34.21), with the GPT-4o judge gap significant ($+2.75$, $p \approx 0.012$; App. C). A label-permutation probe (App. H) collapses Jaccard from 0.459 to 0.209 and Exact-match from 13.78% to 1.28% when `<part_k>` identifiers are shuffled at inference but markers and patches are unchanged, evidence that the LLM has learned a genuine token-region binding rather than treating `<part_k>` as a cosmetic label. App. L projects an open-vocabulary part segmenter, Find3D [28], onto the slot grid under the identical protocol (the part-aware 3D MLLMs of §2 either have no public

Table 2: **Architectural progression on PartVerse-QA.** Vocab-only is the no-marker reference; raw markers without MSR (row 2) lack region-level context and regress below vocab-only (§3.3). MSR reaches Jaccard 0.459 and Exact-match 13.78% ($+64\%$ relative over vocab-only) at 0.7M extra parameters. C2S uses deterministic decoding; S2C metrics are 5-run means (App. C). Alternative refinement designs in App. I.

	Caption-to-slots		Slot-to-caption			
Method	Jaccard↑	Exact-match↑	SBERT↑	GPT-4o↑	METEOR↑	Word-F1↑
Vocab-only	<u>0.413</u>	<u>8.42%</u>	61.23	<u>43.98</u>	33.72	0.416
+ Markers (no MSR)	0.263	3.06%	<u>61.57</u>	41.93	<u>33.99</u>	<u>0.420</u>
+ MSR (ours)	**0.459**	**13.78%**	**62.04**	**44.68**	**34.21**	**0.421**

Table 3: **3DCoMPaT-GrIn PaPGD, multi-part split, text-description metrics.** 3D-PLOT-LLM outperforms PointLLM, Kestrel, PARIS3D, and SegPoint on every reported text metric, and ShapeLLM on 3 of 4. Cited baseline rows are from Kestrel's Table 1; our row excludes PartVerse-QA from Stage-2 to ensure fair comparison.

Method	BLEU-4↑	METEOR↑	SBERT↑	SimCSE↑	GPT-4o↑
ShapeLLM	8.01	30.05	**84.74**	<u>85.38</u>	–
PARIS3D	8.45	31.62	82.08	83.86	–
SegPoint	7.24	27.62	78.50	81.47	–
Kestrel	8.55	32.88	82.19	84.13	–
PointLLM (our FT)	<u>8.85</u>	<u>33.19</u>	83.43	85.33	<u>40.31</u>
3D-PLOT-LLM (ours)	**9.32**	**33.93**	<u>84.24</u>	**86.23**	**43.34**

checkpoints or take rendered views or scene-level inputs): it reaches Jaccard 0.304 against our 0.459 and trails in every mapping-quality stratum.

3DCoMPaT-GrIn part-grounded description, text track. 3DCoMPaT-GrIn PaPGD (Table 3) is the part-aware point grounded description split of 3DCoMPaT++ [2, 38]. We fine-tune both PointLLM ("our FT" row) and 3D-PLOT-LLM on the 3DCoMPaT-GrIn training split for this evaluation; our Stage-2 here excludes PartVerse-QA, adding no benchmark-related supervision over the cited baselines, so the gap reflects the architectural contribution. The benchmark is originally a joint segmentation+text task; as 3D-PLOT-LLM is decoder-free, we evaluate the text-description track here (App. D (f)) and the grounding track separately below. Only the PointLLM and 3D-PLOT-LLM rows are run by us (hence their GPT-4o cells); the other rows are cited. 3D-PLOT-LLM outperforms PointLLM on all five reported text metrics ($+0.47$ BLEU-4, $+0.74$ METEOR, $+0.81$ SBERT, $+0.90$ SimCSE, $+3.03$ GPT-4o judge). It also leads PARIS3D, SegPoint, and Kestrel on every reported metric, and ShapeLLM on three of four ($+1.31$ BLEU-4, $+3.88$ METEOR, $+0.85$ SimCSE). The semantic GPT-4o judge widens the gap relative to lexical and embedding metrics, mirroring the amplification on PartVerse-QA S2C.

3DCoMPaT-GrIn part-grounded description, grounding track. Each slot is the union of its patches' points, so the interface also induces point masks. We issue every GrIn part name as a caption-to-slots query to the model trained on PartVerse-QA, which receives no GrIn training, and score the induced mask by mIoU against per-point fine part labels. It reaches 0.221 / 0.220 / 0.242 on the single-part / multiple-parts / embodied splits, against oracle ceilings of 0.448 / 0.441 / 0.488 for the best slot set on the $K{=}16$ grid (App. M). This is about half of what the interface can express without a decoder; slots are addressable coarse regions, not boundary-accurate masks.

Parameter efficiency. 3D-PLOT-LLM adds $\approx$ 0.8M architectural parameters (markers 6K, `<part_k>` embeddings 66K, MSR ≈ 0.74M) on top of the shared frozen 22M encoder and fine-tuned LLM, an order of magnitude below the segmentation-decoder and box-grammar machinery of prior part-aware 3D MLLMs.

4.3 Ablations

Component ablation. Table 4 isolates the four architectural pieces: K-slot grouping, per-region markers, reserved `<part_k>` tokens (training-coupled with PartVerse-QA), and MSR. Input to-

Table 4: **Component ablation.** MSR (row 6, ours) leads on every PartVerse-QA metric (C2S, S2C-SB, S2C-GPT) and on Objaverse SBERT. Grp: grouping; Mkr: markers; L+PV: reserved <part_k>+PartVerse-QA; C2S: caption-to-slots Jaccard; S2C-SB / S2C-GPT: slot-to-caption SBERT / GPT-4o judge.

	Components				Objaverse		PartVerse-QA		
Method	Grp	Mkr	L+PV	MSR	SBERT↑	GPT-4o↑	C2S↑	S2C-SB↑	S2C-GPT↑
PointLLM	✗	✗	✗	✗	47.66	39.08	–	–	–
+ Grouping	✓	✗	✗	✗	48.06	40.68	–	–	–
+ Markers	✓	✓	✗	✗	<u>48.21</u>	40.56	–	–	–
+ L+PV (no Mkr)	✓	✗	✓	✗	47.83	40.77	<u>0.413</u>	61.23	<u>43.98</u>
+ Mkr+L+PV (no MSR)	✓	✓	✓	✗	48.08	**41.36**	0.263	<u>61.57</u>	41.93
+ **MSR (full)**	✓	✓	✓	✓	**48.31**	<u>40.93</u>	**0.459**	**62.04**	**44.68**

ken reorganization alone yields $+0.40$ SBERT and $+1.60$ GPT-4o over the unstructured-sequence PointLLM baseline; adding markers extends the SBERT gain to $+0.55$. Vocab tokens alone (row 4) establish a non-trivial 0.413 C2S Jaccard and 43.98 S2C GPT-4o grounding signal; MSR's conditioning on per-region stats and adjacency makes the marker substrate functional, lifting C2S Jaccard to 0.459, S2C GPT-4o to 44.68, and S2C SBERT to 62.04 (row 6, ours) at only 0.7M extra parameters. Inserting the interface does not degrade whole-object captioning: every interface variant is at or above PointLLM on the Objaverse GPT-4o judge (40.56 to 41.36 vs 39.08; App. C).

Additional ablations. Five supplementary ablations support each design choice: PartVerse-QA data scaling shows near-monotonic, steeply rising C2S and S2C curves (App. J); refinement-location ablation favors the marker substrate over <part_k> embeddings (App. I); the MSR branch split shows stats and adjacency carry complementary signal (0.284 and 0.305 Jaccard alone, 0.459 combined; App. G); and App. K reports a full $K{=}8$ retrain (part description unchanged, S2C SBERT 62.30 vs 62.04; grounding point-IoU $0.460 \rightarrow 0.406$) and an inference-time partition sweep (at most 0.8 SBERT and 0.013 Jaccard of change).

4.4 Qualitative results

Figure 3 illustrates two regimes. **Top**: a single <part_k> vocabulary carries both C2S and S2C across granularities, with slot sets emerging as data-learned compositions that correspond to no predefined taxonomy. **Bottom**: PointLLM drifts to wrong classes (candle holder → bin, bed → sports table, garbage bin → planter) and confabulates parts to fit the misclassification, while 3D-PLOT-LLM stays on-class and recovers multi-part materials/colors, the qualitative signature of the $+3.03$ GPT-4o gap in Table 3.

5 Limitations and future work

3D-PLOT-LLM addresses parts at K-slot granularity rather than point-wise: "where is the handle" returns a slot set of <part_k> tokens, each a patch cluster, which suffices for language-level part QA but is not a substitute for segmentation-decoder methods such as Kestrel [2] when point-wise masks are needed. Repeated parts are spatially distinct by construction (each wheel has its own slot; a collective reference grounds to their union), but a singular reference to one of several identical parts cannot pick an instance (Fig. 13c), while S2C captions a supplied wheel slot cleanly (Fig. 8a): the ambiguity is on the query side. PartVerse-QA is built around the fixed $K{=}16$ interface, and its union-IoU filter keeps only the representable pairs (73% of part-annotated points; on the unfiltered strata the model still stays 2.0 to $2.6\times$ above chance, App. E). Direct comparisons are limited to baselines with public checkpoints. The fixed slot budget $K{=}16$ accommodates 94.5% of objects in the aligned PartVerse pool by part count (App. E); a learned variable-K or hierarchical partition is left to future work, alongside swapping in a stronger backbone (ReCon++ [32], Uni3D [49]) without architectural change. For full scenes, our part tokens can be nested under object identifiers from scene-level machinery, since the partition needs only an object's patch tokens. C2S and S2C also compose without curated slot sets: in interactive use, the slot set given to an S2C query comes from the model's own C2S answer or from a point the user clicks, which maps deterministically to its slot. Composed part QA, such as asking for the material of the handle, and point-referred questions need

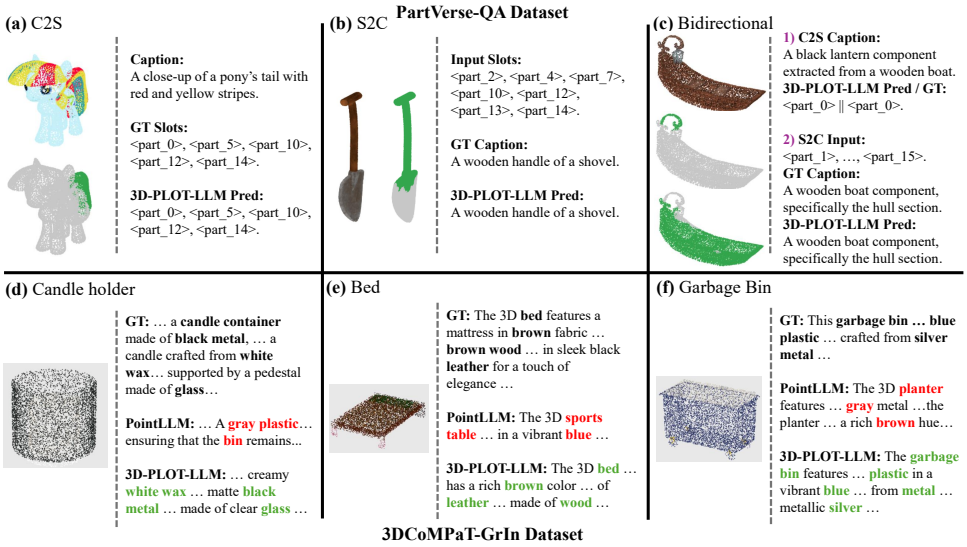


Figure 3: **Qualitative examples of part-aware tokenization. Top row (PartVerse-QA, vocabulary-level part addressing).** 3D-PLOT-LLM (a) maps a free-form caption to the exact `<part_k>` set covering the described region (C2S, pony tail, 5 slots, Jaccard 1.00); (b) describes a multi-slot input in one sentence (S2C, shovel handle, 7 slots $\rightarrow$ "a wooden handle of a shovel", Word-F1 1.00); (c) handles both directions on the same object across granularities ranging from a single-slot lantern to a near-whole-object 10-slot hull. **Bottom row (3DCoMPaT-GrIn, part-aware grounded description).** PointLLM [42] drifts to a wrong object class and confabulates parts to fit the misclassification ("bin", "sports table", "planter"; red); 3D-PLOT-LLM stays anchored to the correct class (candle holder, bed, garbage bin) and recovers fine-grained material/color attributes (green: matches GT). Full dialogues, additional examples, and failure modes in App. N.

instruction data that PartVerse's caption-style annotations do not supply; building such QA pairs is the next step. Code, checkpoints, the benchmark, and the partition packs are publicly available (link in §1).

6 Conclusion

We showed that an object-level 3D MLLM can be made part-aware without a segmentation decoder or bounding-box grammar, by turning every geometric region into a first-class addressable token refined by Marker-Space Refinement (MSR). 3D-PLOT-LLM introduces vocabulary-level part addressing on PartVerse-QA, leads PointLLM, Kestrel, PARIS3D, and SegPoint on every text-description metric of 3DCoMPaT-GrIn PaPGD, and lifts Objaverse captioning above PointLLM, all with under 1M additional parameters on a frozen point encoder. A label-permutation probe (App. H) confirms that the LLM learns a genuine token-region binding rather than treating `<part_k>` as a cosmetic label. Reframing addressable parts as a vocabulary problem opens a lightweight, decoder-free route to part-aware 3D language models, with part-grounded supervision a viable scaling axis complementary to whole-object caption augmentation.

Acknowledgments and Disclosure of Funding

The authors acknowledge the Center for Advanced Research Computing (CARC) at the University of Southern California for providing computing resources that have contributed to the research results reported within this publication (`https://carc.usc.edu`). This work received no external funding. The authors declare no competing interests.

References

[1] Ahmed Abdelreheem, Ivan Skorokhodov, Maks Ovsjanikov, and Peter Wonka. Satr: Zero-shot semantic segmentation of 3d shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15166–15179, 2023.

[2] Mahmoud Ahmed, Junjie Fei, Jian Ding, Eslam Mohamed Bakr, and Mohamed Elhoseiny. Kestrel: 3d multimodal llm for part-aware grounded description. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8973–8983, 2025.

[3] Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72, 2005.

[4] Omid Bonakdar and Nasser Mozayani. Mmpart: Harnessing multi-modal large language models for part-aware 3d generation. *arXiv preprint arXiv:2509.16768*, 2025.

[5] Sijin Chen, Xin Chen, Chi Zhang, Mingsheng Li, Gang Yu, Hao Fei, Hongyuan Zhu, Jiayuan Fan, and Tao Chen. Ll3da: Visual interactive instruction tuning for omni-3d understanding reasoning and planning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26428–26438, 2024.

[6] Yilun Chen, Shuai Yang, Haifeng Huang, Tai Wang, Runsen Xu, Ruiyuan Lyu, Dahua Lin, and Jiangmiao Pang. Grounded 3d-llm with referent tokens. *arXiv preprint arXiv:2405.10370*, 2024.

[7] Wei-Lin Chiang, Zhuohan Li, Ziqing Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3):6, 2023.

[8] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 13142–13153, 2023.

[9] Jiajun Deng, Tianyu He, Li Jiang, Tianyu Wang, Feras Dayoub, and Ian Reid. 3d-llava: Towards generalist 3d lmms with omni superpoint transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 3772–3782, 2025.

[10] Shaocong Dong, Lihe Ding, Xiao Chen, Yaokun Li, Yuxin Wang, Yucheng Wang, Qi Wang, Jaehyeok Kim, Chenjian Gao, Zhanpeng Huang, et al. From one to more: Contextual part latents for 3d generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8230–8240, 2025.

[11] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910, 2021.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pages 1263–1272. Pmlr, 2017.

[13] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15180–15190, 2023.

[14] Zilu Guo, Hongbin Lin, Zhihao Yuan, Chaoda Zheng, Pengshuo Qiu, Dongzhi Jiang, Renrui Zhang, Chun-Mei Feng, and Zhen Li. Pisa: A self-augmented data engine and training strategy for 3d understanding with large models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1273–1283, 2026.

[15] Ziyu Guo, Renrui Zhang, Xiangyang Zhu, Yiwen Tang, Xianzheng Ma, Jiaming Han, Kexin Chen, Peng Gao, Xianzhi Li, Hongsheng Li, et al. Point-bind & point-llm: Aligning point cloud with multi-modality for 3d understanding, generation, and instruction following. *arXiv preprint arXiv:2309.00615*, 2023.

[16] Shuting He, Henghui Ding, Xudong Jiang, and Bihan Wen. Segpoint: Segment any point cloud via large language model. In *European Conference on Computer Vision*, pages 349–367. Springer, 2024.

[17] 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. *Advances in Neural Information Processing Systems*, 36:20482–20494, 2023.

[18] Haifeng Huang, Yilun Chen, Zehan Wang, Rongjie Huang, Runsen Xu, Tai Wang, Luping Liu, Xize Cheng, Yang Zhao, Jiangmiao Pang, et al. Chat-scene: Bridging 3d scene and large language models with object identifiers. *Advances in Neural Information Processing Systems*, 37:113991–114017, 2024.

[19] Kuan-Chih Huang, Xiangtai Li, Lu Qi, Shuicheng Yan, and Ming-Hsuan Yang. Reason3d: Searching and reasoning 3d segmentation via large language model. In *2025 International Conference on 3D Vision (3DV)*, pages 1177–1186. IEEE, 2025.

[20] Amrin Kareem, Jean Lahoud, and Hisham Cholakkal. Paris3d: Reasoning-based 3d part segmentation using large multimodal model. In *European Conference on Computer Vision*, pages 466–482. Springer, 2024.

[21] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9579–9589, 2024.

[22] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10965–10975, 2022.

[23] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[24] Yuchen Lin, Chenguo Lin, Panwang Pan, Honglei Yan, Yiqiang Feng, Yadong Mu, and Katerina Fragkiadaki. Partcrafter: Structured 3d mesh generation via compositional latent diffusion transformers. *arXiv preprint arXiv:2506.05573*, 2025.

[25] Minghua Liu, Yinhao Zhu, Hong Cai, Shizhong Han, Zhan Ling, Fatih Porikli, and Hao Su. Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21736–21746, 2023.

[26] Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. Scalable 3d captioning with pretrained models. *Advances in Neural Information Processing Systems*, 36:75307–75337, 2023.

[27] Chuofan Ma, Yi Jiang, Jiannan Wu, Zehuan Yuan, and Xiaojuan Qi. Groma: Localized visual tokenization for grounding multimodal large language models. In *European Conference on Computer Vision*, pages 417–435. Springer, 2024.

[28] Ziqi Ma, Yisong Yue, and Georgia Gkioxari. Find any part in 3d. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1–10. IEEE, 2025.

[29] Yongsen Mao, Junhao Zhong, Chuan Fang, Jia Zheng, Rui Tang, Hao Zhu, Ping Tan, and Zihan Zhou. Spatiallm: Training large language models for structured indoor modeling. *arXiv preprint arXiv:2506.07491*, 2025.

[30] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2019.

[31] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[32] 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 *European Conference on Computer Vision*, pages 214–238. Springer, 2024.

[33] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26417–26427, 2024.

[34] Hanoona Rasheed, Muhammad Maaz, Sahal Shaji, Abdelrahman Shaker, Salman Khan, Hisham Cholakkal, Rao M Anwer, Eric Xing, Ming-Hsuan Yang, and Fahad S Khan. Glamm: Pixel grounding large multimodal model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13009–13018, 2024.

[35] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 3505–3506, 2020.

[36] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

[37] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3d: Mask transformer for 3d semantic instance segmentation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8216–8223. IEEE, 2023.

[38] Habib Slim, Xiang Li, Yuchen Li, Mahmoud Ahmed, Mohamed Ayman, Ujjwal Upadhyay, Ahmed Abdelreheem, Arpit Prajapati, Suhail Pothigara, Peter Wonka, et al. 3dcompat++: An improved large-scale 3d vision dataset for compositional recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[39] Yiwen Tang, Ziyu Guo, Zhuhao Wang, Renrui Zhang, Qizhi Chen, Junli Liu, Delin Qu, Dong Wang, Bin Zhao, and Xuelong Li. Exploring the potential of encoder-free architectures in 3d lmms. In *International Conference on Learning Representations*, volume 2026, pages 150063–150084, 2026.

[40] Hugues Thomas, Chen Chen, and Jian Zhang. Pts3d-llm: Studying the impact of token structure for 3d scene understanding with large language models. *arXiv preprint arXiv:2506.05689*, 2025.

[41] 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.

[42] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *European Conference on Computer Vision*, pages 131–147. Springer, 2024.

[43] Jintang Xue, Ganning Zhao, Jie-En Yao, Hong-En Chen, Yue Hu, Meida Chen, Suya You, and C-C Jay Kuo. Descrip3d: Enhancing large language model-based 3d scene understanding with object-level text descriptions. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1746–1756, 2026.

[44] Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 27091–27101, 2024.

[45] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. Set-of-mark prompting unleashes extraordinary visual grounding in gpt-4v. *arXiv preprint arXiv:2310.11441*, 2023.

[46] Yunhan Yang, Yufan Zhou, Yuan-Chen Guo, Zi-Xin Zou, Yukun Huang, Ying-Tian Liu, Hao Xu, Ding Liang, Yan-Pei Cao, and Xihui Liu. Omnipart: Part-aware 3d generation with semantic decoupling and structural cohesion. In *Proceedings of the SIGGRAPH Asia 2025 Conference Papers*, pages 1–12, 2025.

[47] 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19313–19322, 2022.

[48] Tatiana Zemskova and Dmitry Yudin. 3dgraphllm: Combining semantic graphs and large language models for 3d scene understanding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8885–8895, 2025.

[49] Junsheng Zhou, Jinsheng Wang, Baorui Ma, Yu-Shen Liu, Tiejun Huang, and Xinlong Wang. Uni3d: Exploring unified 3d representation at scale. *arXiv preprint arXiv:2310.06773*, 2023.

[50] Chenming Zhu, Tai Wang, Wenwei Zhang, Jiangmiao Pang, and Xihui Liu. Llava-3d: A simple yet effective pathway to empowering lmms with 3d-awareness. *arXiv preprint arXiv:2409.18125*, 2024.

A Geometric region partition: algorithm

This appendix expands §3.1. The partition procedure takes the $N{=}512$ patch centers $\{c_i\}$ and patch features $\{f_i\}$ from a frozen Point-BERT and returns a grouping into $K{=}16$ regions. This specific procedure is implementation, not contribution: any spatially coherent deterministic partition with approximate size balance would serve the architectural argument equally well.

The algorithm has four stages.

(S0) Patch connectivity graph. A k-nearest-neighbor graph ($k{=}12$) over the 512 patch centers, with edges exceeding the 75th percentile of kNN lengths gated to prevent thin regions from short-circuiting across empty space. The gating can leave the graph disconnected; for isolated regions (empty $\mathcal{N}(k)$), the neighbor sum in Eq. 3 is dropped and MP reduces to $\phi(W_c\tilde{\mathbf{m}}_k)$.

(S1) Superpixel growth. Farthest-point seeds expand by adding neighbors in order of a mixed cost $c_{uv} = \lambda \left\|c_u - c_v\right\|_2 + (1-\lambda)(1-\cos(f_u, f_v))$ with $\lambda{=}0.8$. Each superpixel caps at $\sim$3 patches, yielding 150–200 superpixels.

(S2) Balanced merge to K. On the superpixel adjacency graph, we pick $K{=}16$ geodesic farthest-point seeds and grow each by priority-queue expansion under the same cost c_{uv} (lowest-cost adjacent unassigned superpixel first), capped at $\lceil 512/K \rceil = 32$ patches per region. Disconnected components seed and expand independently, with seeds allocated in proportion to component size. The result is regions of $\sim$32 patches on average.

(S3) Boundary smoothing and connectivity cleanup. Four iterations reassign each boundary patch to the region with lower mean c_{uv} to its in-region neighbors (ties broken by region index). A final connectivity pass splits non-contiguous regions and absorbs fragments smaller than four patches into the lowest-cost neighbor; any excess regions are merged by centroid distance until exactly $K{=}16$ regions remain. All 512 patches end up assigned; final region sizes may deviate from the S2 cap of 32 due to boundary reassignment and fragment absorption.

The partition is fully deterministic given the frozen Point-BERT output (the random seed is derived from the patch centers), so identical centers produce identical partitions; we extract it once per object and cache it offline. The choice of $K{=}16$ is motivated in App. E.

B Hyperparameters and compute

All runs use $2\times$A100 80GB GPUs with bf16 mixed precision. **Stage 1 (alignment):** LLM and Point-BERT frozen; projector, per-region markers, and MSR trainable. Data: 660K Cap3D captions of Objaverse, 3 epochs, LR 2×10^{-3} with cosine schedule and 3% warm-up; effective batch 48; $\sim$48 hours per run. **Stage 2 (instruction tuning):** full LLM, projector, markers, and MSR all trainable. `<part_k>` tokens are added to the tokenizer at this stage, initialized to the mean of the existing vocabulary embeddings, and trained jointly. Data: PointLLM's 70K complex instructions + 77K PartVerse-QA (§3.4), 3 epochs, LR 2×10^{-5} with cosine schedule and 3% warm-up; effective batch 16 under DeepSpeed ZeRO-3 [35]; $\sim$40 hours per run. **MSR:** both MLPs in MSR (the statistics MLP $\mathrm{MLP}_s : \mathbb{R}^7 \to \mathbb{R}^{384}$ in Eq. 1 and the message MLP ϕ inside $\mathrm{MP}(\cdot;\mathcal{N}(k))$ in Eqs. 2–3) use GELU activations and no bias on either of their two linear layers. Residual scales $\alpha_{\mathrm{s}}, \alpha_{\mathrm{adj}}$ of Eqs. 1–2 are initialized to 0.05. The output layer of ϕ in Eq. 3 (weight and bias) is zero-initialized so that $\hat{\mathbf{m}}_k \approx \mathbf{m}_k$ at the start of training; MLP_s uses Xavier initialization on its two linear layers. **Evaluation:** sampling-based inference scores are averaged over 5 seeds; see §4.1 for per-split details.

C Per-run variance and significance tests

This appendix backs the main-paper Tables 1 (Objaverse) and 2 (PartVerse-QA) with 5-run mean $\pm$ std and pairwise Welch's two-sample t-tests ($n_1{=}n_2{=}5$, two-sided). Each seed independently re-runs sampling-based inference on the same trained ckpt. PointLLM, ShapeLLM, and ENEL are re-evaluated by us under the matched protocol, so their 5-run std are reported in Table 5 alongside our variants; PiSA (Table 1) is cited from its paper and the ShapeLLM, PARIS3D, SegPoint, and Kestrel rows of Table 3 from Kestrel's Table 1; these sources give only published means or single-run numbers without per-run variance estimates.

Excluded from std reporting: the cited PiSA row (no per-run variance released); PartVerse-QA C2S Jaccard and C2S Exact-match (deterministic decoding by design, reported in main Table 2 without std); 3DCoMPaT-GrIn (no per-run variance reported by cited baselines). Table 5 unifies the variance panels: the four semantic / lightweight-lexical Objaverse metrics (SBERT, SimCSE, GPT-4o, METEOR) and the two PartVerse-QA S2C metrics with non-trivial 5-run variance (SimCSE and the GPT-4o judge); the latter two columns are blank for rows whose model is not trained against the `<part_k>` part-token interface (ShapeLLM, ENEL, PointLLM, no-PV).

Table 5: **Per-row mean $\pm$ std (Objaverse + PartVerse-QA S2C).** All cells are 5-run mean $\pm$ std (Objaverse on val_200, PartVerse-QA S2C on the 196-query held-out split). "—": PartVerse-QA S2C metric not applicable to that row (ShapeLLM, ENEL, and PointLLM have no `<part_k>` interface at all; no-PV has the interface architecturally but is not trained on PartVerse-QA, so its `<part_k>` embeddings remain unbound). PartVerse-QA C2S Jaccard and Exact-match are deterministic (no std applicable) and are reported in main Table 2. Pairwise Welch's t-test results are summarized in the "Pairwise significance tests" paragraph below. **Best** and <u>2nd</u> per column among rows with applicable data.

	Objaverse				PartVerse-QA S2C	
Method	SBERT	SimCSE	GPT-4o	METEOR	SimCSE	GPT-4o
ShapeLLM (re-eval)	35.49±0.37	36.13±0.49	12.86±1.35	12.33±0.09	—	—
ENEL (re-eval)	47.70±0.69	48.22±0.70	37.02±1.29	**17.89±0.51**	—	—
PointLLM (re-eval)	47.66±0.58	48.16±0.75	39.08±1.83	12.19±0.30	—	—
3D-PLOT-LLM (no-PV)	47.94±0.52	48.44±0.50	40.56±1.55	12.27±0.14	—	—
Vocab-only	47.83±0.33	48.56±0.29	40.77±1.06	12.42±0.52	<u>62.24±0.80</u>	<u>43.98±1.25</u>
+ Markers (no MSR)	<u>48.08±0.49</u>	**48.92±0.54**	**41.36±1.26**	12.50±0.17	62.23±0.92	41.93±1.56
3D-PLOT-LLM (ours)	**48.31±0.40**	<u>48.67±0.35</u>	<u>40.93±1.02</u>	<u>12.59±0.13</u>	**63.08±1.04**	**44.68±0.96**

Pairwise significance tests. **(i) Objaverse, matched-data architectural contribution (no-PV vs. PointLLM)**: no-PV outperforms PointLLM on all six metrics on means (Table 1, up to $+1.48$ GPT-4o judge); Welch's t-tests on the four metrics tabulated above give $p \geq 0.21$ (within run-to-run noise), consistent with the architectural-level lift reported in §4.2. **(ii) Objaverse, full model with PartVerse-QA (full vs. PointLLM)**: METEOR is significant ($p \approx 0.04$); GPT-4o ($p \approx 0.09$) and SBERT ($p \approx 0.08$) are marginal; SimCSE is non-significant ($p > 0.15$). **(iii) PartVerse-QA architectural progression (ours vs. +Markers, no MSR)**: this isolates MSR's specific contribution

beyond raw markers. Deterministic $+0.196$ C2S Jaccard and $+10.7$ percentage points C2S Exact-match (no std applicable) plus the significant $+2.75$ S2C GPT-4o judge ($p \approx 0.012$).

D Evaluation prompt templates

We list the prompt strings used for training supervision and evaluation. Chat templating matches PointLLM [42]; 3DCoMPaT-GrIn formatting matches Kestrel [2]'s release.

Chat template. All training and inference use the Vicuna v1.1 [7] chat format with a fixed system prompt and `USER`/`ASSISTANT` roles separated by a single space, with `</s>` terminating each `ASSISTANT` turn. The `<point>` placeholder appears at the start of the first `USER` turn followed by a newline, and is replaced at runtime by `<point_start>`, the projected point-cloud token block, and `<point_end>`; subsequent `USER` turns in multi-round conversations omit `<point>`:

```
A chat between a curious user and an artificial intelligence
assistant.  The assistant gives helpful, detailed, and polite
answers to the user's questions.  USER: <point>\n{instruction}
ASSISTANT: {response}</s>USER: {follow-up} ASSISTANT:
{response}</s>...
```

The prompt strings below populate `{instruction}` for each task.

(a) Objaverse brief-description alignment (Stage 1). We use PointLLM's 660K Cap3D-aligned brief-description set [42]. Each sample draws uniformly from the released pool of 30 paraphrases of "describe this object briefly"; five representative examples:

- "Offer a clear and concise description of this point cloud object."
- "How would you interpret this 3D point cloud?"
- "What kind of object is illustrated by this collection of points?"
- "Convey a summary of the 3D structure represented in this point cloud."
- "Describe the object that this point cloud forms."

The reference response is the Cap3D [26] brief caption for that object.

(b) Objaverse complex instructions (Stage 2). We use PointLLM's 70K complex-instruction set [42], split into detailed-description ($\sim$15K), single-round ($\sim$40K), and multi-round ($\sim$15K) GPT-4-generated dialogues conditioned on each object's Cap3D caption. A representative detailed-description sample:

```
USER: <point>\nProvide a meticulous explanation of what these points
represent.  ASSISTANT: This three-dimensional model represents a
large aircraft with a dominating white color scheme....</s>
```

(c) PartVerse-QA caption-to-slots (training and evaluation). Given a part caption, the model returns matching `<part_k>` tokens. The instruction template is:

```
<point>\nText:  ''{caption}'' Output only matching <part_n> tokens,
comma-separated.
```

The reference response is comma-separated `<part_k>` tokens without spaces (e.g. `<part_1>,<part_4>,<part_7>`). The literal string `<part_n>` appearing in the instruction is a generic placeholder in the natural-language prompt; the model emits actual `<part_k>` vocabulary tokens in its response.

(d) PartVerse-QA slot-to-caption (training and evaluation). Given a slot set, the model returns a one-sentence caption. The instruction template is:

```
<point>\nDescribe region <part_k1>, <part_k2>, ..., and <part_kn> in
one short sentence.
```

The slot list is comma-separated with "and" before the final token (e.g. `<part_0>, <part_5>, and <part_13>`).

(e) Objaverse captioning evaluation. The model is prompted with the single instruction "Caption this 3D model in detail.", following PointLLM [42].

(f) 3DCoMPaT-GrIn (training and evaluation). We fine-tune on the full 3DCoMPaT-GrIn train split (111,514 samples) released with Kestrel [2], which combines three subsets: Part-Aware Point Grounded Description (PaPGD, 80,760), Direct Segmentation (8,076), and Reasoning Segmentation (22,678). Each subset has its own prompt format; the PaPGD subset uses one of 30 paraphrases requesting a part-grounded description with in-line segmentation masks. Three representative PaPGD paraphrases:

- "Kindly give me a detailed description of the 3D model. Please incorporate interleaved segmentation masks for the corresponding components in your answer."
- "Can you offer a comprehensive analysis of the 3D model? Please include interleaved segmentation masks for the relevant parts in your response."
- "Please provide an exhaustive overview of the 3D model. Include segmentation masks for each distinct component in your answer."

Because 3D-PLOT-LLM is decoder-free, the mask-request phrasing is treated as text instruction and the model is supervised only on the language portion of the ground-truth caption. Evaluation (Table 3) is on the PaPGD multi-part split as defined by Kestrel (text track; the grounding-track protocol is in App. M); fine-tuning uses the Stage-2 schedule (App. B) for 3 epochs.

(g) GPT-4o captioning judge. Objaverse and 3DCoMPaT-GrIn PaPGD GPT-judge scoring use the prompt of Xu et al. [42] verbatim, with judge model `gpt-4o-2024-08-06`:

> "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'."

Cited GPT-4o cells in Tables 1 and 3 are blank because GPT-4o snapshots (and, for the cited Kestrel rows, the judge prompt itself) differ across papers, so their reported numbers are not directly comparable to ours under a fixed `gpt-4o-2024-08-06` judge.

E PartVerse-QA benchmark: construction and statistics

Purpose. Existing 3D object captioning benchmarks (Objaverse val [42], PiSA-Bench [14]) evaluate whole-object descriptions but contain no part-referring queries; part-aware benchmarks such as 3DCoMPaT-GrIn [2, 38] target segmentation grounding through pointwise masks, which is orthogonal to the token-level interface 3D-PLOT-LLM exposes. To evaluate a 3D MLLM's ability to name parts in its output vocabulary and to resolve part captions to a finite set of addressable slots, we construct **PartVerse-QA**, a new benchmark we build on top of the part annotations of the PartVerse mesh dataset [10] and align to the Objaverse point-cloud representation consumed by PointLLM [42] and 3D-PLOT-LLM. The aligned benchmark (query JSONs, alignment and part-to-slot mapping tables, and partition packs) is released under the MIT license of PartVerse at `https://huggingface.co/datasets/jintangx/PartVerse-QA`; the evaluation scripts are in the code repository.

Source data. All semantic part annotations used by PartVerse-QA come from the public PartVerse mesh dataset [10]; we introduce no part labels of our own. PartVerse provides meshes with per-triangle part IDs for a subset of Objaverse-style objects. Because PointLLM and 3D-PLOT-LLM consume a fixed 8192-point sample, we re-project the mesh-level labels onto the point cloud and then onto our $K=16$ region slots. The pipeline below is deterministic given the PartVerse meshes and the Objaverse point clouds; no additional human labels are introduced.

Construction pipeline.

1. **Point re-projection.** For each object in PartVerse, we align its mesh to the 8192-point Objaverse sample consumed by PointLLM using an iterative ICP step followed by Umeyama rigid alignment, then transfer per-triangle part IDs to each of the 8192 points via nearest-neighbor on the aligned mesh.
2. **Semantic-to-slot mapping.** For each semantic part s on an object, we locate the minimum set of region slots whose union covers the point support of s, and record the union-IoU between the slot union and the semantic part.
3. **IoU filtering (quality control).** We keep a (semantic part, slot set) pair only if its union-IoU is at least 0.5. Below this threshold the $K{=}16$ slot decomposition is judged too coarse or too ambiguous to supervise as a part-grounded description; such pairs are discarded. The retained pool has median union-IoU 0.713 (mean 0.725), well above the 0.5 floor.
4. **Query instantiation.** Each retained (semantic part, slot set) pair is instantiated into caption-to-slots (C2S) and slot-to-caption (S2C) queries (§3.4); exact prompt templates are listed in Appendix D. C2S is scored by set-level Jaccard and Exact-match. S2C is scored by a set of standard captioning metrics: Word-F1 (token-level overlap), Sentence-BERT (semantic similarity), BLEU-1, METEOR, and ROUGE-L, plus a GPT-4o judge for free-form semantic agreement; the same metric set is used for Objaverse captioning in Table 1.

Train / evaluation split. The aligned pool contains $10{,}200$ PartVerse objects with $67{,}521$ raw (semantic part, object) annotations; the IoU ≥ 0.5 filter (Step 3) retains $26{,}065$ unique (semantic part, slot set) pairs. After an object-level split, the benchmark yields $77{,}607$ training queries and 588 evaluation queries (392 C2S, 196 S2C) on an 83-object held-out set. Crucially, the held-out objects are disjoint not only from our PartVerse-QA training pool but also from PointLLM's Stage-1 alignment data (660K Cap3D-captioned Objaverse objects) and Stage-2 instruction set (70K complex-instruction objects), so the model never sees any held-out object during pre-training, alignment, or instruction tuning. Statistical comparisons are reported at the query level (392 C2S, 196 S2C): each query is an independent (semantic part, slot set) pair, and the 5-run inference variance and Welch's t-tests (App. C) operate over query-level scores.

Filtering sensitivity and interface coverage. The union-IoU filter of Step 3 is computed from the partition and the mesh annotations alone, with no language model in the loop, and it removes pairs that the $K{=}16$ grid represents poorly. It keeps 38.6% of raw part instances but 73.3% of all part-annotated points: the median removed part spans 186 points, about a third of one 512-point slot, against $1{,}447$ points for kept parts. To show what the filter changes, Table 6 evaluates the unchanged model on every stratum of ground-truth mapping quality, including evaluation sets rebuilt from the raw annotations for all cases the filter removes (456 queries across $[0, 0.5)$; parts mapping to zero slots are excluded because no answer key exists), each against a cardinality-matched chance baseline (uniform random slot subsets of the model's predicted cardinality, $1{,}000$ draws per query). Absolute scores decline smoothly as the answer key degrades, while the model-to-chance ratio stays between $2.0\times$ and $2.6\times$ throughout: the threshold changes absolute numbers, not the capability conclusion. The benchmark therefore evaluates addressable coarse grounding on the representable subset, covering about three quarters of the annotated part surface.

Table 6: **Caption-to-slots as a function of ground-truth mapping quality.** The unchanged model on all strata of the union-IoU between a part and its gold slot set; the three strata at or above 0.5 make up the 392-query benchmark. Chance is a cardinality-matched random baseline.

Union-IoU stratum	n	Jaccard↑	Chance	Ratio
≥ 0.7	156	0.552	0.244	2.3×
$[0.6, 0.7)$	114	0.471	0.201	2.3×
$[0.5, 0.6)$	122	0.330	0.137	2.4×
$[0.4, 0.5)$	102	0.318	0.120	2.6×
$[0.2, 0.4)$	244	0.176	0.086	2.0×
$[0.0, 0.2)$	110	0.147	0.068	2.2×

Parts per object (pre-filter, 10,200 objects). Figure 4 plots the number of PartVerse semantic-part annotations per aligned object, before any IoU filtering. The distribution is concentrated in the low single digits: median 6, p_{90}=13, p_{95}=17; 83% of objects carry at most 10 parts and 94.5% carry at most 16. This grounds our K=16 default. A smaller K (e.g., K=8) would leave a substantial fraction of objects' parts overflowing the slot budget and force semantic parts to share slots, degrading the part-token interface (App. K: C2S point-IoU 0.460 → 0.406 at K=8); a larger K (e.g., K=32) would double the `<part_k>` vocabulary and dilute the average per-token training signal, with diminishing coverage gains beyond the 95th percentile and finer slots that fragment beyond the training-free partition's effective resolution against semantic parts. A direct comparison across K is confounded by the filter, which selects different (semantic part, slot set) pairs at different K (the K=8 held-out set built by the same pipeline shares 69 objects with the K=16 set). App. K therefore reports a full K=8 retrain scored with point-IoU, which is defined in point space and comparable across K; the architectural argument itself is K-agnostic (any spatially coherent partition with the two properties of §3.1 would serve).

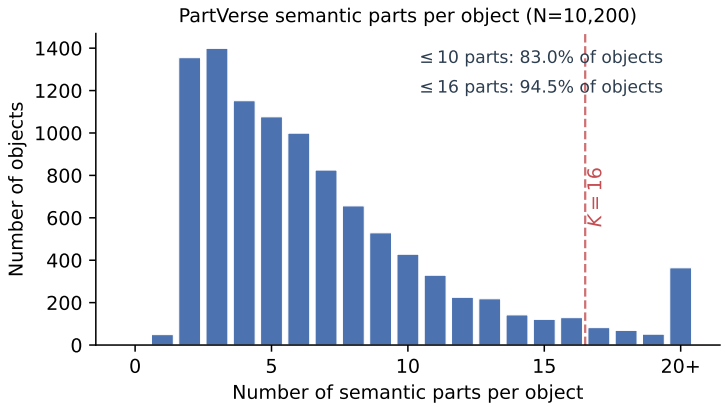


Figure 4: **Semantic parts per object in our aligned PartVerse pool (N=10,200).** Median 6, p_{95}=17. The dashed line marks our slot budget K=16, which covers 94.5% of objects.

Slots per query (post-filter, 77,607 queries). Figure 5 plots the slot-set size per training query, after the IoU ≥ 0.5 filter and query expansion. The benchmark covers a wide granularity range: 24% of queries target a single slot (fine-grained part references), the median query has 3 slots, and the 95th percentile reaches 11 slots (near-whole-object regions). C2S and S2C share the same cardinality distribution by construction.

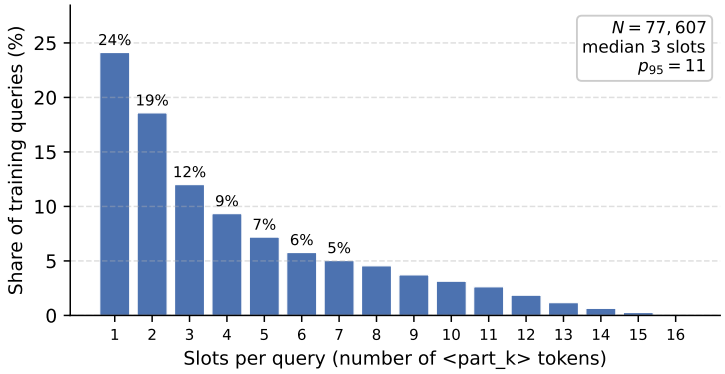


Figure 5: **Slots per query across 77,607 training queries.** The benchmark spans single-slot fine-grained parts (24%) through near-whole-object regions (11 slots at p_{95}, 16 slots at the tail). Eval splits (392 C2S + 196 S2C) follow the same distribution.

Faithfulness to PartVerse annotations. We do not modify PartVerse's raw mesh annotations; our additions (re-projection onto the 8192-point Objaverse sample, slot-union mapping onto the $K{=}16$ grid, the IoU ≥ 0.5 filter, and query-template instantiation) operate deterministically on the public PartVerse meshes and Objaverse point clouds.

Cross-method evaluation. This subsection deepens the structural argument summarized in §4.2 (PartVerse-QA paragraph) and details why prior part-aware 3D MLLMs are not retrofitted onto either direction of the benchmark. Prior part-aware 3D MLLMs treat part references as decoder outputs only: segmentation-based methods (Kestrel [2], PARIS3D [20], SegPoint [16]) emit point masks via a `[SEG]` token decoded by an external head; box-based Part-X-MLLM [41] emits autoregressive box tokens addressing coordinate bins; none read or write `<part_k>` tokens, and none accept a region encoding (mask, box, or token set) as model input. **The S2C direction (slot set as input)** is therefore not retrofittable: there is no input pathway in any prior method through which a region specification can reach the LLM, and any natural-language translation of the slot set would leak the target into the prompt and trivialize the task. **The C2S direction (slot set as output)** is in principle retrofittable by projecting a method's predicted masks or boxes onto our $K{=}16$ slot grid. Kestrel and Part-X-MLLM have no public checkpoints, and PARIS3D and SegPoint take rendered views or scene-level inputs that would need per-method adaptation, an engineering layer between the model and the metric; App. L projects the open-vocabulary part segmenter Find3D instead. Since the projection adds its own error, App. L also reports Jaccard separately for parts that the slot grid covers well and poorly. App. F scores our own variants against mesh-level masks without any slot projection. The slot-mapping tables are part of the released benchmark, so future work can build unified cross-method evaluations on top.

F Model-level partition-agnostic mesh mIoU

This appendix complements Appendix E, which establishes that the $K{=}16$ partition's median union-IoU against PartVerse mesh-level semantic parts is 0.713. We extend that partition-level statistic to a model-level mIoU computed against mesh-derived per-point masks rather than slot-set ground truth, providing a partition-agnostic view of the architectural progression in Table 4.

Method. For each held-out C2S query we lift the model's predicted `<part_k>` set to the 8192-point Objaverse cloud via the PointBERT-Group rule used at training, and build the mesh-level ground-truth mask by ICP+Umeyama alignment of the cached PartVerse mesh-sampled cloud onto the same point cloud; alignments are pre-computed at PartVerse-QA construction (Appendix E) and reused here. 390 of 392 queries pass; two are dropped at fewer than 50 mesh points after alignment.

Table 7: **Mesh mIoU on PartVerse-QA C2S ($N{=}390$).** Set Jaccard from Tables 2 and 4 for reference. Mesh GT comes from PartVerse mesh annotations ICP-aligned to the 8192-point cloud, with no slot-set reprojection. Rows span the variants of Tables 4 and 8.

Variant	Set Jaccard↑	Mesh mIoU mean↑	Mesh mIoU median↑
Vocab-only (no markers)	0.413	0.374	0.374
+ Markers (no MSR)	0.263	0.257	0.192
+ MSR stats only ($\alpha_{\text{adj}}{=}0$)	0.284	0.287	0.239
+ MSR adjacency only ($\alpha_s{=}0$)	0.305	0.303	0.254
+ MSR full (ours)	**0.459**	**0.399**	**0.426**
Gold slot set (slot-set ceiling)	1.000	0.688	0.669
Partition oracle ($K{=}16$ ceiling)	—	0.711	0.705

Findings. Three observations across Table 7. (i) The architectural ranking is preserved under the partition-agnostic GT (MSR full $>$ Vocab-only $>$ MSR adjacency-only $>$ MSR stats-only $>$ Markers without MSR), including the markers-no-MSR regression below vocab-only. (ii) The partition oracle's mean mesh mIoU 0.711 matches the median union-IoU 0.713 from Appendix E, confirming pipeline consistency. (iii) Dense-decoder methods (PARIS3D, SegPoint, Kestrel) are not directly comparable on this metric: their structural ceiling exceeds any $K{=}16$-based ceiling (e.g., the 0.711 partition oracle here), the explicit design tradeoff of this work (§2, Appendix E "Cross-method evaluation").

G MSR branch ablation: stats vs. adjacency

MSR has two residual branches (Eqs. 1–2): a stats branch driven by per-region statistics $s_k \in \mathbb{R}^7$ via MLP(s_k), and an adjacency branch driven by message passing over $\mathcal{N}(k)$ via MP($\tilde{\mathbf{m}}_k; \mathcal{N}(k)$). Table 8 ablates each branch in isolation against the full two-branch MSR.

Table 8: **MSR branch ablation.** Stats-only and adjacency-only each achieve a substantial fraction of the full MSR's gains, but neither alone matches the full configuration on the part-grounded objective. The two branches are individually informative and complementary: combining them lifts PartVerse-QA Jaccard by $+0.154$ over adjacency-only and $+0.175$ over stats-only.

	Objaverse		PartVerse-QA	
MSR branch active	SBERT↑	GPT-4o↑	C2S Jac↑	S2C GPT-4o↑
Stats only (α_{adj}=0)	<u>47.88</u>	<u>40.26</u>	0.284	<u>43.93</u>
Adjacency only (α_{s}=0)	47.31	39.21	<u>0.305</u>	43.69
Both (ours)	**48.31**	**40.93**	**0.459**	**44.68**

Branches are complementary, not redundant. Each isolated branch reaches a substantial but incomplete fraction of the full-MSR PartVerse-QA Jaccard, and combining them exceeds the sum of individual-branch improvements over the no-MSR baseline (0.263, Table 4): the combined gain is $+0.196$, versus $+0.021$ (stats-only) and $+0.042$ (adjacency-only) summing to $+0.063$. The asymmetry across metrics is interpretable: stats-only leads on Objaverse captioning (per-region geometry signals whole-object structure), and adjacency-only leads on PartVerse-QA Jaccard (inter-region adjacency binds part references to slot sets), with the two branches complementing in the full model.

H Probing token-region binding

To check whether `<part_k>` functions as a real addressing handle rather than a cosmetic label, we permute the `<part_k>` identifiers by a fixed random σ at inference while keeping the partition, markers, and patches unchanged (no retraining). If the binding is real, the C2S predictions should shift off the original label space and Jaccard against the original GT should collapse. Table 9 confirms this: Jaccard drops from 0.459 to 0.209 and Exact-match from 13.78% to 1.28%, well below single-slot chance ($1/16$=6.25%), which is evidence that the LLM has learned a genuine token-region binding.

Table 9: **Label-permutation probe on PartVerse-QA C2S (**392 **held-out queries).** Partition, markers, and patches unchanged; `<part_k>` labels permuted at inference.

Setting	Jaccard↑	Exact-match↑
Baseline (ours)	**0.459**	**13.78**%
Label permutation (σ random)	0.209	1.28%

I Design space of refinement module

This appendix expands the single-line mention in §4.3: we ablate where the stats + adjacency conditioning module is applied: to the per-region marker (our MSR), to the vocabulary token `<part_k>` ("vocab refinement"), or to both. The only varying axis across rows is the channel that receives the conditioning.

Findings. Three observations are consistent across metrics in Table 10:

(i) Marker refinement dominates. Applying the conditioning module to the marker (MSR) lifts C2S Jaccard from 0.263 to 0.459 ($+0.196$); applying the same module to the vocab token instead (vocab refinement) lifts Jaccard only to 0.278 ($+0.015$). The asymmetry is $\sim 13\times$ on Jaccard. On S2C GPT-4o, marker refinement also leads: $+2.75$ over no refinement and $+2.10$ over vocab refinement.

(ii) Marker location is more parameter-efficient. MSR achieves the best result with 0.7M extra parameters; the vocab-side variant requires 3.2M ($\sim 4.5\times$ more) because each `<part_k>` embedding

Table 10: **Design space of refinement module: where to apply structural conditioning.** Rows share the "+ Markers (no MSR)" assembly of Table 2 and differ only in the conditioning target: nowhere; the 4096-d <part_k> vocabulary embedding; the 384-d marker $\mathbf{m}_k$ (ours, MSR); or both. C2S Jaccard is deterministic; S2C GPT-4o is the 5-run mean on the PartVerse-QA 196-query held-out split (per-row $\pm$std and Welch's t-tests in prose below).

Refinement location	C2S Jaccard↑	S2C GPT-4o↑	Params↓
No refinement	0.263	41.93	**0**
Vocab refinement	0.278	42.58	3.2M
Marker + vocab refinement	<u>0.442</u>	<u>43.39</u>	3.9M
Marker refinement (ours, MSR)	**0.459**	**44.68**	<u>0.7</u>M

is 4096-d vs the marker's 384-d. The same conditioning loss thus provides denser per-parameter gradient on the marker.

(iii) Combining is non-additive. Refining both locations reaches 0.442, -0.017 below MSR alone, suggesting the two channels encode mutually interfering inductive biases when stacked rather than additive structural signal. We therefore use single-location MSR.

J PartVerse-QA data scaling

Figure 6 and Table 11 report 3D-PLOT-LLM trained on $0/16/30/50/75/100\%$ of the 77,607 PartVerse-QA pairs at Stage 2, with the Markers baseline of Table 2 as the no-refinement reference. Three observations stand out:

(i) **Sample efficiency.** Ours with just 16% PartVerse-QA data already matches the no-refinement Markers baseline trained on the full pool (caption-to-slots Jaccard 0.268 vs. 0.263, Table 2): a small slice of part-grounded supervision is enough to reach a baseline-equivalent grounding signal.

(ii) **Steep high-data scaling.** Ours rises to 0.459 Jaccard at 100%, with the steepest segment in $75 \rightarrow 100\%$ ($+0.128$ Jaccard, $+2.66$ slot-to-caption GPT-4o judge), and ultimately a $+0.196$ Jaccard margin over the no-refinement Markers baseline (Table 2) at full data.

(iii) **Objaverse is flat.** Whole-object Objaverse captioning varies little across fractions (47.6–48.3 SBERT, 38.8–40.9 GPT-4o), consistent with Stage-2 captioning loss being saturated by the shared 70K complex-instruction pool that every variant trains on.

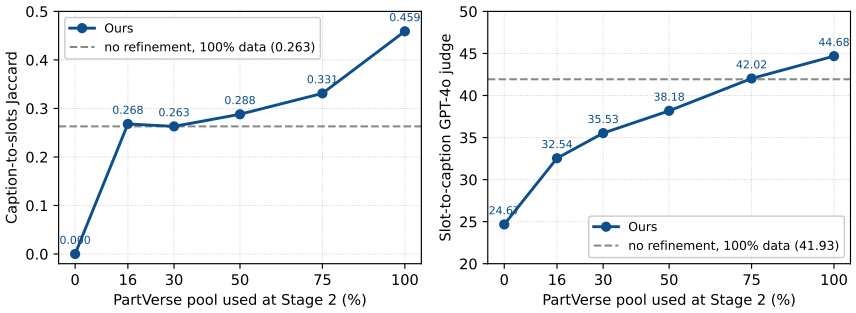


Figure 6: **PartVerse-QA supervision scaling.** Caption-to-slots Jaccard (left) and slot-to-caption GPT-4o judge (right) on the held-out PartVerse-QA split as a function of the fraction of the 77,607-pair pool used at Stage 2 (six points: $0/16/30/50/75/100\%$). The dashed horizontal references mark the no-refinement comparator at 100% data; ours meets the no-refinement Jaccard reference using only 16% of the pool, and the no-refinement slot-to-caption GPT-4o judge by 75%.

K Sensitivity to the slot count and to the partition

Slot count. We trained a complete $K{=}8$ counterpart of the full model (same recipe, both stages) and evaluated it under the identical protocol on the $K{=}8$ version of PartVerse-QA built by the same

Table 11: **PartVerse-QA supervision scaling.** Ours with only 16% of the PartVerse-QA pool already matches the no-refinement baseline (Markers row in Table 2, Jaccard 0.263) trained on the full pool. Fractions: share of the $77{,}607$-pair PartVerse-QA pool used at Stage 2.

Fraction	PV-Jac↑	Obj-SBERT↑	Obj-GPT-4o↑	S2C-GPT-4o↑
0%	0.000	47.94	40.56	24.67
16%	0.268	47.65	<u>40.77</u>	32.54
30%	0.263	47.74	38.83	35.53
50%	0.288	<u>48.00</u>	40.75	38.18
75%	<u>0.331</u>	47.85	40.69	<u>42.02</u>
100%	**0.459**	**48.31**	**40.93**	**44.68**

pipeline (258 C2S and 129 S2C held-out queries on the 69 objects shared with the $K{=}16$ split). Across K we compare point-IoU for caption-to-slots, which is defined in point space and comparable across slot granularities, unlike slot-set Jaccard. Table 12 shows three consistent effects: whole-object captioning at $K{=}8$ falls back to the PointLLM level (47.05 ± 0.21 vs 47.66 ± 0.58), part description is unchanged (S2C SBERT 62.30 vs 62.04), and part grounding degrades moderately (point-IoU $0.460 \rightarrow 0.406$), since eight slots are too coarse to isolate typical PartVerse parts (the mean part occupies 4.5 of 16 slots). The dependence on K is smooth rather than knife-edge; $K{=}16$ is set by the 512-patch budget and was not tuned on benchmark scores.

Table 12: **Slot-count sensitivity.** Full retrain at $K{=}8$ vs the $K{=}16$ model, same recipe and protocol; 5-run mean $\pm$ std where applicable.

	$K{=}8$	$K{=}16$ (paper)
Objaverse SBERT	47.05 ± 0.21	48.31 ± 0.40
PartVerse-QA S2C SBERT	62.30 ± 0.92	62.04 ± 1.21
PartVerse-QA C2S point-IoU	0.406	0.460

Partition hyperparameters and partition family. The partition of App. A has two graph hyperparameters, the kNN degree ($k{=}12$) and the long-edge pruning percentile (75th). We regenerated the partition at several settings with the original pipeline and evaluated the unchanged $K{=}16$ model with each partition swapped in at inference (no retraining). The regenerated default reproduces the native partition on 96.6% of patch labels, whereas the variants genuinely alter it (55 to 58% agreement). Across a $4\times$ range of k and the whole pruning range the spread is 0.8 SBERT, comparable to 5-run noise, and no setting falls below the PointLLM baseline beyond noise. Absolute levels in Table 13 are not directly comparable with Table 1, because the swap pathway rebuilds slot indices, statistics, and adjacency at load time; rows within Table 13 share this pathway and are comparable with one another. Swapping to a k-means partition of the patch coordinates (a different family, purely spatial) gives 47.74 ± 0.26 on captioning and, with the ground truth remapped to that partition by the identical mapping pipeline, 0.446 vs 0.459 Jaccard on caption-to-slots; the two answer keys are equally faithful (mean gold union-IoU 0.677 k-means vs 0.686 native). The graph hyperparameters are therefore not load-bearing, and the feature term of the merge cost is not either; we keep the kNN construction for determinism and local coherence.

Table 13: **Partition sensitivity at inference.** The unchanged $K{=}16$ model with regenerated partitions swapped in; Objaverse SBERT, 5-run mean $\pm$ std, all rows through the same swap pathway.

Partition at inference	SBERT↑
kNN $k{=}12$, prune at 75th pct (default, regenerated)	47.55 ± 0.41
kNN $k{=}6$ (halved)	48.07 ± 0.33
kNN $k{=}24$ (doubled)	48.24 ± 0.25
Prune at 50th pct (tighter)	47.71 ± 0.26
No pruning	47.42 ± 0.44
k-means (16) on patch coordinates (different family)	47.74 ± 0.26

L Projected open-vocabulary baseline on caption-to-slots

This appendix gives the protocol behind the Find3D comparison in §4.2. Among part-aware 3D MLLMs, Kestrel and Part-X-MLLM have no public checkpoints at the time of writing, and PARIS3D and SegPoint take rendered views or scene-level inputs that would need per-method adaptation. The strongest reproducible representative of the open-vocabulary text-to-part-mask family that is point-cloud native and takes free-text queries is Find3D [28]; it is trained on Objaverse, so it is in-domain for the held-out set. For each of the 392 C2S queries, Find3D receives the caption as its text query; its per-point similarity is converted to a query probability (softmax over the query logit and an unlabeled logit) and mean-pooled per slot. A slot set is obtained by (a) a probability threshold swept over $\{0.3, 0.4, 0.5, 0.6, 0.7\}$, reporting the best (0.5), and (b) an oracle-cardinality rule that takes the top-m slots with m equal to the gold cardinality, a generous upper bound. Scoring uses the identical slot-to-point mapping and metrics as our model (gold-vs-gold IoU 1.0 under the shared mapping). Find3D's text encoder truncates long captions at 64 tokens, a property of that baseline. Table 14 reports the results overall and per mapping-quality stratum.

Table 14: **Projected open-vocabulary baseline on PartVerse-QA C2S (392 queries).** Find3D masks projected onto the $K{=}16$ slot grid; strata by gold union-IoU as in Table 6.

	All 392 queries		Jaccard by gold union-IoU		
	Jaccard↑	Point-IoU↑	$[0.5, 0.6)$	$[0.6, 0.7)$	≥ 0.7
Find3D, best threshold (0.5)	0.304	0.311	0.227	0.315	0.356
Find3D, oracle cardinality	0.294	0.294	0.169	0.300	0.388
3D-PLOT-LLM (ours)	**0.459**	**0.460**	**0.330**	**0.471**	**0.552**

M 3DCoMPaT-GrIn grounding track

This appendix gives the protocol behind the grounding-track paragraph of §4.2. Ground-truth masks come from the official fine-grained per-point part labels of the 3DCoMPaT++ validation and test splits, transferred by nearest neighbor to the $8{,}192$-point clouds our models take as input; labels are transferred for all $7{,}443$ GrIn validation objects, with a mean transfer distance of 0.024. The queries are the part names of the GrIn validation annotations: 677 single-part, $2{,}000$ multiple-parts, and $2{,}002$ embodied queries. One query each in the multiple-parts and embodied splits is dropped because its ground-truth mask is empty. The PartVerse-QA-trained $K{=}16$ model answers each query zero-shot with greedy decoding. The text-track model of Table 3 is not used here, because it is trained without PartVerse-QA and never learns to answer with slots. The instruction is:

```
<point>\nThe following text describes ONE component of this 3D
object:\n‘‘{part}’’\nWhich pseudo-part tokens (<part_0> through
<part_15>) best correspond to this component?  Answer with
comma-separated tokens only (e.g.  ‘‘<part_0>,<part_15>’’), with no
other text.
```

The predicted slot set is turned into a point mask by taking the union of its patches' points, the rule used in App. F, and scored against the ground-truth mask by mIoU, acc@25, and acc@50, where acc@25 and acc@50 are the fractions of queries with IoU of at least 0.25 and 0.50. The oracle ceiling is the mean over queries of the best IoU that any slot set on the $K{=}16$ grid can reach; it measures the limit set by the grid, not the model. The model emits a slot set for every query and reaches 49 to 50% of the ceiling on all three splits (Table 15).

Table 15: **Zero-shot grounding on 3DCoMPaT-GrIn.** Slot-induced point masks of the PartVerse-QA-trained model against per-point fine part labels; the oracle ceiling is the best mIoU any slot set on the $K{=}16$ grid can reach.

Split	n	mIoU↑	acc@25↑	acc@50↑	Oracle ceiling
single_part	677	0.221	0.332	0.158	0.448
multiple_parts	1,999	0.220	0.326	0.147	0.441
embodied	2,001	0.242	0.379	0.176	0.488

N Qualitative examples and failure modes

This appendix supplements §4.4 and Figure 3 with untruncated outputs across all three benchmarks, plus a discussion of representative failure modes. App. N.1 expands the PartVerse-QA bidirectional dialogues across the slot-cardinality range; App. N.2 reproduces the full grounded descriptions of Figure 3(d-f) and adds three more 3DCoMPaT-GrIn cases at full length; App. N.3 provides Objaverse whole-object side-by-side captions against PointLLM and ShapeLLM, the qualitative companion to Table 1; App. N.4 delineates three failure modes that characterize where the $K{=}16$ vocabulary-level addressing interface breaks down.

N.1 PartVerse-QA: extended bidirectional dialogues

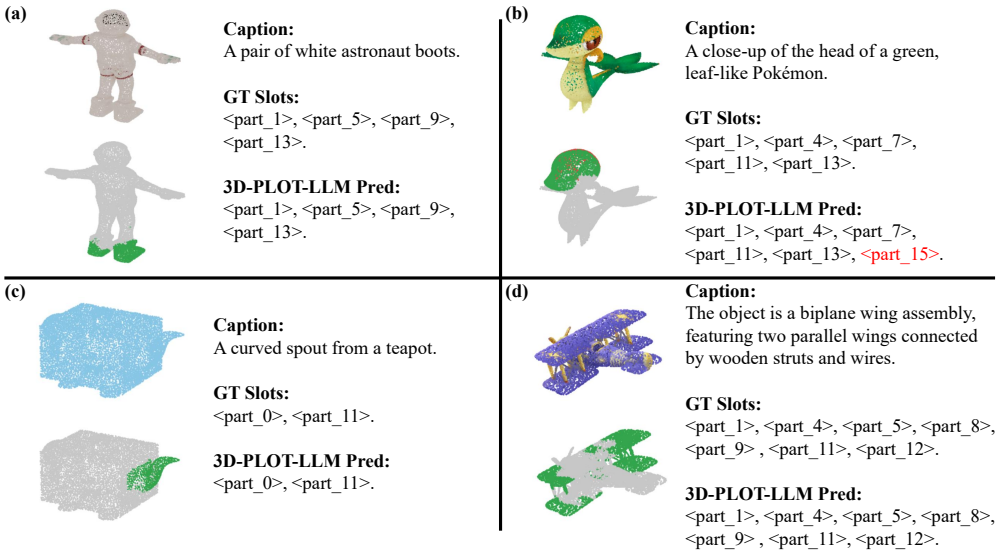


Figure 7: **PartVerse-QA C2S dialogues (extended).** Each panel shows the RGB input (top) and the $K{=}16$ partition with target slots highlighted in green (bottom; in panel (b), the additional slot the model emits is shown in red), alongside the caption and predicted slot set. Panels (a), (c), and (d) are exact set matches at 4, 2, and 7 slots respectively; panel (b) is a near-miss at 5 slots where the extra red slot `<part_15>` is itself inside the head region. Panel (d) also shows that token-level set agreement is attainable even when the green partition highlight does not visually cover the full upstream PartVerse part annotation: token-Jaccard measures slot-token agreement, which is independent of how the partition aligns with the PartVerse annotation.

Figure 7 extends the Caption-to-Slots (C2S) interface beyond the single Jaccard $= 1.00$ panel in the main figure. Four panels span slot cardinality 2 to 7 out of $K{=}16$: two clean exact matches (a teapot spout at 2 slots, a pair of astronaut boots at 4 slots), one near-miss on a Pokémon-style head (5 slots, Jaccard 0.83) where the model emits one extra slot that also lies inside the head region, and a biplane wing (7 slots) at token-level exact match (Jaccard $= 1.00$) where the green partition highlight does not cover the full PartVerse "wing assembly" annotation, illustrating that token-Jaccard measures slot-token agreement and is independent of how the partition aligns with the PartVerse annotation.

Figure 8 extends the Slots-to-Caption (S2C) direction in the same way, with slot cardinality 1 to 6 out of $K{=}16$. Panels include a near-perfect single-slot car wheel ("…view of a car wheel" $\rightarrow$ "…of a car wheel"), an exact-match 3-slot apple leaf, an attribute-drop 5-slot pair of green headphones (the color adjective "green" and the disambiguating "component" are dropped from the predicted caption), and a 6-slot stringed-instrument base where the prediction is more descriptive than the GT ("base component" $\rightarrow$ "the body of a traditional Chinese stringed instrument, likely a pipa or similar"), a case in which Word-F1 penalizes plausibly added specificity. All eight panels are from the held-out PartVerse-QA split with object-disjoint training.

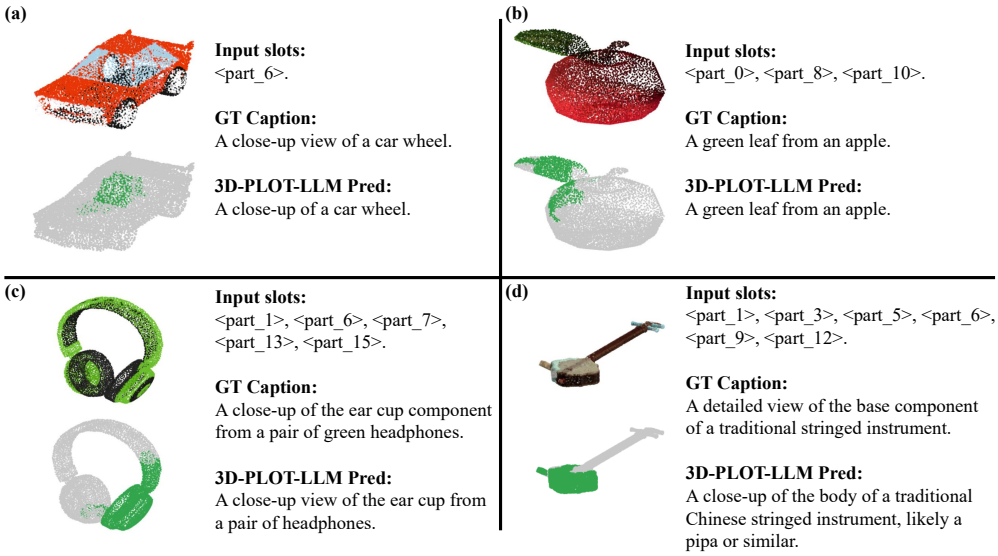


Figure 8: **PartVerse-QA S2C dialogues (extended).** Each panel shows the RGB input (top) and the $K=16$ partition with input slots highlighted (bottom, in green), alongside the GT caption and 3D-PLOT-LLM prediction. Panel (a) is a single-word omission (1 slot, "view" dropped); panel (b) is an exact 3-slot match; panel (c) drops a color adjective and a part-label modifier ("green"/"component"); panel (d) shows a case where the prediction is more descriptive than the GT ("body" as a more anatomically precise term, plus the cultural qualifier "Chinese" and a hedged species guess "likely a pipa or similar"), illustrating that Word-F1 penalizes additional specificity even when the additions are plausible.

N.2 3DCoMPaT-GrIn: full grounded descriptions

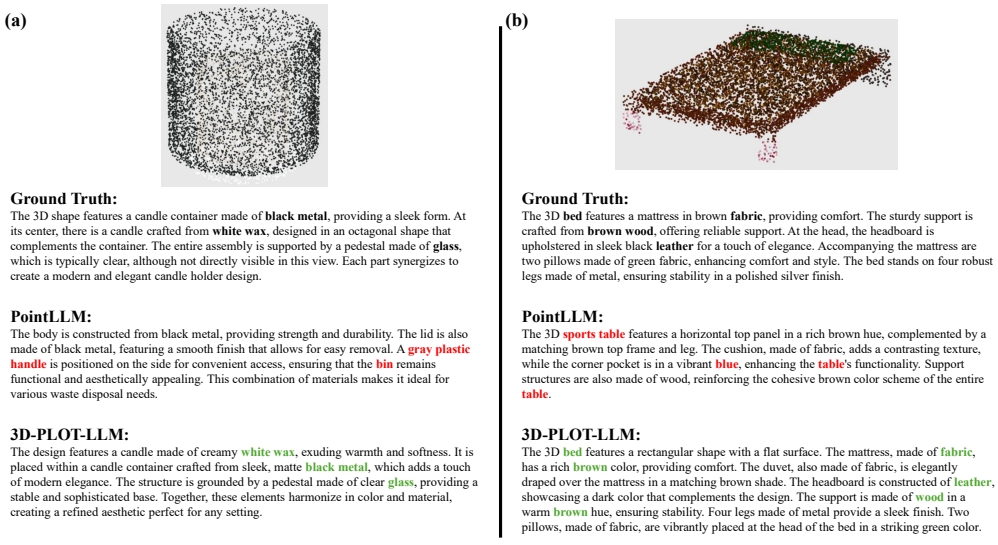
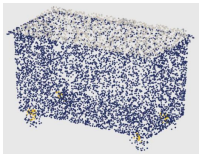


Figure 9: **3DCoMPaT-GrIn full grounded descriptions: Figures 3(d-e) at full length.** Verbatim ground-truth, PointLLM, and 3D-PLOT-LLM descriptions for the candle holder (a) and bed (b) examples shown abridged in the main paper. PointLLM misclassifies both (bin, sports table) and the part-level materials/colors follow the misclassification (red); 3D-PLOT-LLM recovers the correct class and the multi-material attributes (green: matches GT).

(a)

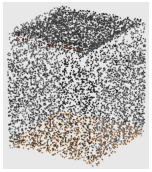


Ground Truth:
This **garbage bin** features a robust body made of **blue plastic**, providing ample storage. The top is covered by a shiny lid, crafted from **silver metal**, ensuring durability and a secure closure. It is supported by two wheels, also made of **blue** plastic, allowing for easy movement. The lid connector is constructed of metal, securely attaching the lid to the body, while the wheel connector, also metal, reinforces the stability of the wheels.

PointLLM:
The 3D **planter** features a sturdy foot made of **gray** metal, ensuring stability. The container is crafted from vibrant blue metal, adding visual appeal. At the top, the **planter** is filled with soil, presenting a rich **brown** hue that enhances its natural look. This combination of materials results in a functional and attractive **planter** suitable for various environments.

3D-PLOT-LLM:
The **garbage bin** features a body made of **plastic** in a vibrant **blue**, showcasing vertical grooves for added design appeal. It is topped with a lid, crafted from **metal** and finished in a sleek metallic **silver**, providing easy access to the contents. The bin is equipped with wheels, made of **plastic** and matching the **blue** color of the body, ensuring mobility and functionality. This combination of materials and colors enhances both durability and aesthetic charm.

(b)



Ground Truth:
The 3D shape features a **barbecue grill** with various distinct parts. The **bbq cooking grate** is made of black **metal**, providing a durable cooking surface. The vertical front panel is crafted from plastic, designed in **dark gray** for a lightweight yet sturdy structure. The vertical side panel and vertical back panel are composed of metal, providing essential support and stability. The handle is made of brown wood for a comfortable grip, while the horizontal bottom panel is also wood, contributing to the grill's overall robust design.

PointLLM:
The 3D **cabinet** showcases a structured design with distinct materials. The **horizontal top panel** and **horizontal bottom panel** are made of wood, featuring a dark brown color. The vertical back panel is also wood, exhibiting a rich, warm wood tone. In contrast, the vertical side panel is crafted from metal, presenting a sleek silver finish. The drawer, made of metal, aligns with the silver hue of the side panels, while the handle, constructed from wood, enhances the cabinet's aesthetic with its dark brown color.

3D-PLOT-LLM:
The 3D **bbq grill** comprises several key components, each contributing to its design and function. The **bbq cooking grate** is crafted from **metal** and is colored **black**, ensuring optimal heat distribution. The vertical back panel, vertical front panel, and vertical side panel are all made of **metal** and are colored **dark grey**, providing strength and stability. The horizontal bottom panel is made of wood and features a rich brown color, while the handle, constructed from wood, is also brown, allowing for easy transport.

Figure 10: **3DCoMPaT-GrIn full grounded descriptions: Figure 3(f) at full length plus an additional drift case.** (a) garbage bin (the third abridged main-paper panel) at full length, with PointLLM drifting to "planter". (b) a barbecue grill on which PointLLM drifts to "cabinet", confabulating horizontal top and bottom panels and the materials a cabinet would have. 3D-PLOT-LLM stays on-class in both panels and recovers the multi-material attributes the ground truth describes.

(a)



Ground Truth:
The **sink** is cylindrical with a smooth curved top, made of granite in a **brown** color. The faucet spout, made of **metal**, features a gold color that elegantly contrasts with the **sink**. The faucet lever, also made of **metal**, has a matching gold color for a cohesive design. The sink stopper, composed of **metal**, is in a black color, providing a functional yet stylish element to the overall appearance of the sink.

PointLLM:
The 3D shape is a cylindrical **basket** consisting of three main parts: body, lid, and handle. The body is made of **wood** with a natural **wood** color, while the lid is crafted from metal in a shiny **bronze** color, adding a decorative touch. The handle, also made of metal, matches the shiny **bronze** color of the lid, creating a cohesive and appealing design. This blend of materials and colors enhances the **basket**'s functionality and aesthetic charm.

3D-PLOT-LLM:
The 3D shape is a cylindrical **sink** made of warm **brown** marble, showcasing a smooth surface. The faucet body, faucet lever, and faucet spout are all crafted from shiny metallic **metal**, enhancing the modern aesthetic. A metallic **sink** stopper fits securely in the center drain hole, while the surrounding plate is also made of **metal**, adding to the elegant design. This combination beautifully balances functionality and style.

(b)



Ground Truth:
The 3D shape consists of several components: the body, made of metal and typically colored chrome, serves as the main structure. The spout is also made of **metal**, featuring a shiny finish. The lever, constructed from **metal**, is designed for easy operation. The aerator, made of plastic, usually appears in a contrasting color. Finally, the plate, which is **metal**, provides stability and completes the **faucet** assembly.

PointLLM:
The 3D **parasol** features a handle made of black **plastic**, a **canopy** crafted from grey **fabric** providing shade, a matching grey **fabric** tie wrap that secures the **canopy** when closed, and a sleek grey **plastic** pole that supports the entire structure. This combination of materials and colors results in a functional and stylish **outdoor accessory**.

3D-PLOT-LLM:
The 3D shape features a **perforated metallic** structure. The **handle** is **metallic** and colored **silver**, providing a sturdy grip. The body, also **metallic** and **silver**, forms the main part of the design. The spout extends outward, crafted from **silver metal** to direct water flow. Finally, the aerator, located at the end of the spout, is made of bright yellow plastic, ensuring a functional outlet for water. Each component harmonizes for both utility and aesthetic appeal.

Figure 11: **3DCoMPaT-GrIn full grounded descriptions: two additional cases with maximally different drift targets.** (a) a cylindrical sink with attached metal faucet (spout, lever, stopper), misread by PointLLM as a wooden basket, a same-domain shape-driven drift; 3D-PLOT-LLM recovers the brown body and the metal faucet attachments. (b) a standalone metal faucet whose body, spout, and aerator are correctly described by 3D-PLOT-LLM in the metallic domain, while PointLLM hallucinates a parasol with fabric canopy and plastic pole. This faucet drift flips both material (metal $\rightarrow$ fabric) and use context (plumbing fixture $\rightarrow$ outdoor accessory).

Figures 9, 10, and 11 reproduce the three abridged 3DCoMPaT-GrIn panels of Figure 3(d-f) at full length and add three additional 3DCoMPaT-GrIn cases (a barbecue grill, a sink, and a faucet) covering different drift targets. Each panel pairs the input point cloud with the ground-truth, PointLLM, and 3D-PLOT-LLM grounded descriptions verbatim; green highlights mark words that match the ground-truth class or material/color attributes, and red highlights mark hallucinated words that conflict with the ground truth. All six cases share the same script: PointLLM, treating the object as a flat token stream, drifts to a wrong object class and confabulates parts that fit the misclassification (bin, sports table, planter, cabinet, basket, parasol); 3D-PLOT-LLM, with structurally refined per-region markers, recovers both the correct class and the material/color attributes the ground truth describes.

N.3 Objaverse: whole-object captioning side-by-side

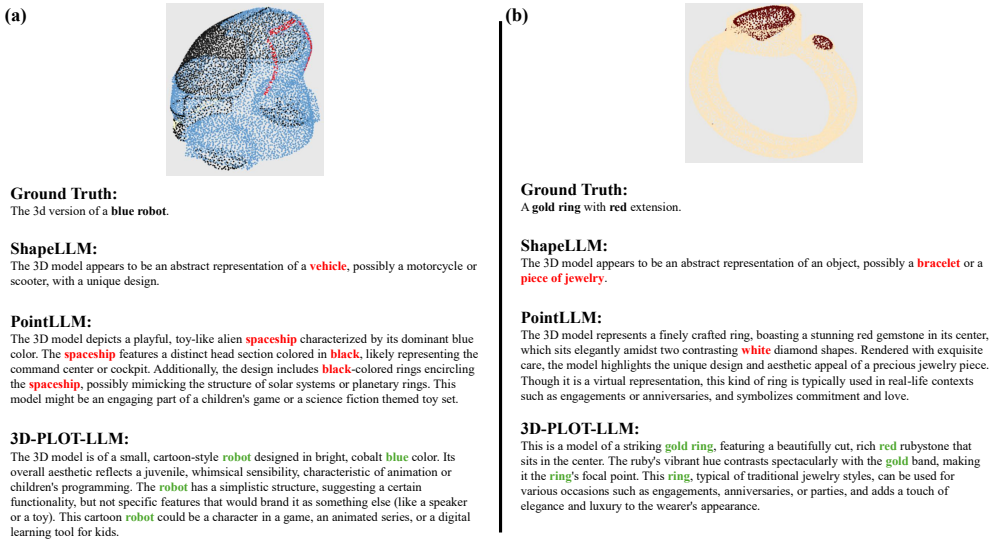


Figure 12: **Objaverse whole-object captioning: PointLLM failure modes.** Outputs from 3D-PLOT-LLM, PointLLM, and ShapeLLM on two held-out objects: (a) a blue robot mis-classed by PointLLM as "spaceship" (class drift); (b) a gold ring with a red gemstone, correctly classed by PointLLM but hallucinated as "boasting two contrasting white diamond shapes" that are absent from the point cloud (structural hallucination). 3D-PLOT-LLM stays anchored to both class and attributes in each case (green: matches GT; red: PointLLM hallucinations).

Figure 12 shows the ground-truth, 3D-PLOT-LLM, PointLLM, and ShapeLLM outputs side by side, surfacing two distinct PointLLM failure modes: class drift (blue robot $\rightarrow$ "spaceship") and structural hallucination (a gold ring described as "boasting two contrasting white diamond shapes" that are absent from the point cloud), with 3D-PLOT-LLM staying anchored to both class and attributes in each case. App. N.4 item (3) gives an Objaverse failure mode where both models miss a 2D-surface attribute the point cloud does not carry.

N.4 Failure modes

We catalog three representative failure modes (visualized in Fig. 13) and attribute each to its primary cause: query-side language ambiguity, inherent benchmark difficulty, and the frozen point encoder shared with the PointLLM baseline.

(1) Symmetric repeated parts. On a PartVerse-QA C2S car (Fig. 13c), the $K{=}16$ partition places different wheels in different slots, but near-synonymous captions targeting different wheels have no basis to pick one over another and the model commits to a single slot regardless. The reverse S2C direction (Fig. 8, panel a) recovers a clean caption when one wheel slot is supplied, isolating the failure to query-side ambiguity rather than slot binding.

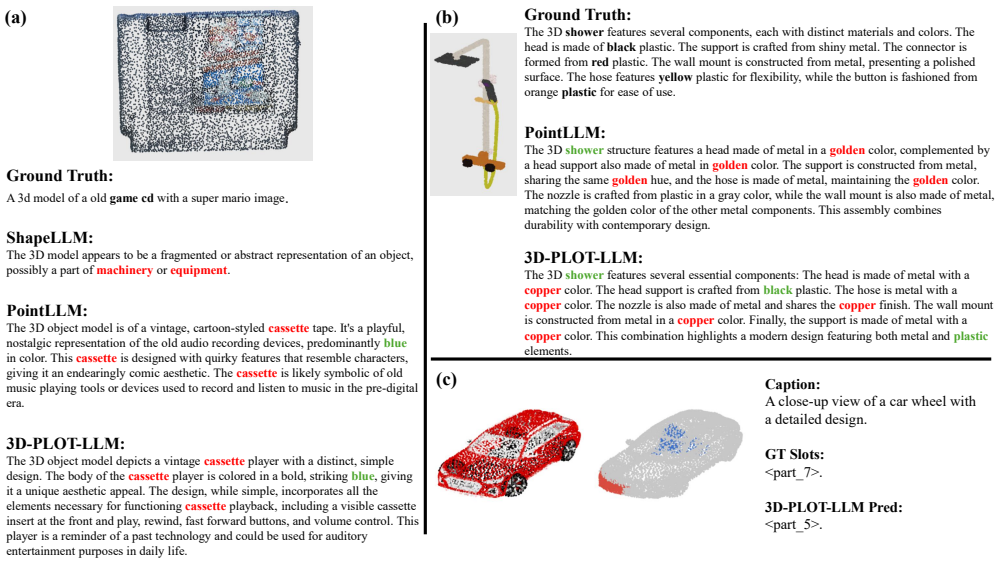


Figure 13: **Failure modes visualized across the three benchmarks.** (a) Mode (3), Objaverse: both models call a Mario-image CD a cassette, recovering only the shared blue body color. (b) Mode (2), 3DCoMPaT-GrIn shower: both collapse the multi-color GT toward a single metal color (golden for PointLLM, copper for ours); 3D-PLOT-LLM still recovers black plastic on the head support that PointLLM omits. (c) Mode (1), PartVerse-QA C2S: a caption "…a car wheel with a detailed design" has GT `<part_7>` but the model commits to `<part_5>`; the partition places different wheels in different slots.

(2) Both models degrade together. On a 3DCoMPaT-GrIn shower (Fig. 13b), both 3D-PLOT-LLM and PointLLM collapse the GT's six distinct material/color combinations toward a single metal color (copper for ours, golden for PointLLM). 3D-PLOT-LLM still recovers one non-metal element (black plastic on the head support) that PointLLM omits, but neither model recovers the full multi-color spec. Two different architectures failing in the same direction suggests the multi-color rendering is hard for the benchmark itself, not a weakness specific to either model.

(3) 2D-surface content not carried by the point cloud. Objaverse Mario-image CD (Fig. 13a): both models see "a blue rectangular vintage media object" and converge on cassette, missing the 2D Mario texture. The bound is at the encoder level (frozen Point-BERT, shared with PointLLM) and orthogonal to the part-vocabulary contribution.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Contributions stated in the abstract and §1 are validated by experiments in §4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in §5 and App. N.4.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Architecture, training, evaluation, hyperparameters, prompts, and the partition algorithm are specified in §3–§4.1 and Apps. A, B, D; code, checkpoints, and the benchmark are public (see the next item).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Code and all training and evaluation scripts are at `https://github.com/jintangxue/3D-PLOT-LLM`, the released checkpoint at `https://huggingface.co/jintangx/3D-PLOT-LLM-7B`, and the PartVerse-QA benchmark with its construction tools, alignment tables, and partition packs at `https://huggingface.co/datasets/jintangx/PartVerse-QA`. Methods, hyperparameters, and the partition algorithm are detailed in §3 and Apps. A, B, D.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Training and evaluation details are in §3.4, §4.1, and Apps. B, D, E.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Objaverse and PartVerse-QA S2C cells are 5-run mean $\pm$ std (1-sigma, across 5 decoder seeds on a fixed checkpoint), with per-cell std and Welch's two-sample t-test ($n_1{=}n_2{=}5$, two-sided) p-values reported in App. C. PartVerse-QA C2S uses deterministic decoding (single eval) and 3DCoMPaT-GrIn follows Kestrel's single-run protocol [2] for parity with cited baselines.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: GPU type and per-stage training compute are specified in App. B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work conforms with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

Justification: This is foundational research with no direct deployment path; downstream impacts mirror those of the upstream PointLLM and Vicuna source models without introducing new categories of risk.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released artifacts pose no high-risk misuse profile beyond that of the source models and datasets.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All baseline models and datasets are cited at first use and used under their public research licenses: PointLLM code (CC BY-NC-SA 4.0) and weights (CC BY-NC 4.0, Llama 2 community license), PartVerse (MIT), Objaverse (ODC-By 1.0), 3DCoMPaT++ and the GrIn split (their published terms), and the public ENEL and Find3D checkpoints for re-evaluation. Our code is released under CC BY-NC-SA 4.0 as a derivative of PointLLM,

our weights under the same non-commercial terms as PointLLM's, and PartVerse-QA under the MIT license of PartVerse.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The aligned PartVerse-QA benchmark is documented in App. E and released with a dataset card at `https://huggingface.co/datasets/jintangx/PartVerse-QA` under the MIT license of PartVerse; the model is released with a model card at `https://huggingface.co/jintangx/3D-PLOT-LLM-7B`.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: 3D-PLOT-LLM augments a Vicuna-7B language backbone with reserved `<part_k>` vocabulary tokens and per-region structural markers (§3); GPT-4o is used as a caption-level evaluation judge (App. D).

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.